PYTHON

BASIC TO ADVANCED

Deep Dive into Python To Build Intelligent Systems

ယခု စာအုပ်သည် သင်ထောက်ကူ အနေဖြင့် အသုံးပြုရန် Python Programming အကြောင်းအား အသေးစိတ် ရေးသားထားသော စာအုပ် ဖြစ်ပါတယ်။

ယခု စာအုပ်အား ပြန်လည် ရောင်းချခြင်း တစ်စိတ်တစ်ပိုင်းအား ပြန်လည် ဖော်ပြခြင်း တစ်နည်းနည်းဖြင့် အသုံးပြုခြင်းများ ခွင့်မပြုပါ။ ယခု စာအုပ်အား တစ်နည်းနည်းဖြင့် အသုံးပြုလိုလျှင် စာရေးသူ Win Htut ရဲ့ ခွင့်ပြုချက် ရယူရန် လိုအပ်ပါသည်။

စာရေးသူအား စာအုပ်ရေးရန် တိုက်တွန်းပေးခဲ့တော့ GreenHackers Founder Dr.Aung Win Htut နှင့် စာရေးသူအား ရေးသားရာတွင် ကူညီ ပေးခဲ့ကြတော့ National Cyber City members များအားလုံးကို ကျေးဇူး တင်ရှိပါတယ်။ ထို့ပြင် စာရေးသူရဲ့ မိဘ နှစ်ပါး အားလည်း အထူး ကျေးဇူး တင်ရှိပါသည်။

ယခု Deep Dive Into Python ရဲ့ Second Edition အားလည်း ဆက်လက် ရေးသားနေဆဲ ဖြစ်ပြီး ထို Second Edition ထဲတွင်တော့ projects များစွာ ပါဝင်မှာ ဖြစ်ပါတယ်။

Let Deep Dive into Python!

1 st Ninja Edition

Table of Contents

Installation 1
Starting “ Hello World ” 2
Using Visual Studio Code 2
Start Python in Visual Studio Code 5
Python Standard Input and Output 6
Standard Input in Python Programming 8
Standard Data Type 8
Numbers in Python 9
Complex numbers 10
Booleans 10
Type Conversion 11
Implicit Type Conversion or coercion 11
Explicit Type Conversion 12
Data type conversion with string 13
String in Python 14
Accessing the Character using Slice operator 16
upper() 19
lower() 20
isupper() 21
islower() 21
count() 22
find() 23
replace() 24
Operators 24
Arithmetic Operators 25
Relational Operator 27
Logical Operators 28

And Logical Operator 28
Or Logical Operator 28
Not Logical Operator 29
Bitwise Operators 29
Bitwise or (|) 31
Bitwise 1's complement (~) or Bitwise Not 31
Bitwise XOR (^) 33
Bitwise right shift (>>) 34
Bitwise Left (<<) 35
Assignment Operators 37
Special Operators 37
Identity Operators 38
is not identity operator 39
Dive to String 40
Membership Operators 41
Python Control Structure 41
Indentation Error - Sample Program (54) 42
elif 42
else statement 43
Loops 44
While loop 44
while loop with the break statement 44
For loop 45
For loops through a String 45
For loops with break statement 46
For loops with continue statement 46
Pass 47
List in Python 48
Declaring a List 48
Creating a list with Data 48
Creating a list with Multiple Data 49
Accessing data 49

Accessing Data from Multidimensional list 50
Remove Method - Removing element from list 51
pop() Method - Popping elements from list 52
index() 53
Error 54
append( ) method 54
extend() method 56
Using operator for extending list 57
insert() method 57
count() method 58
reverse() method 59
sort() method 59
copy() method 60
clear() method 61
Dive To List 61
Namespaces 62
Introduction to Function 64
The range() function (Standard Library Function) 64
Programmer Defined Function 65
Python Function with Parameters 66
Parameter Passing 67
Avoidable Error 69
List passing to a function 69
Pass by reference in python 70
What is a reference? 71
Everything is an Object 73
Returning function from a function 76
Function Passing to a Function 78
Avoidable Error 79
Positional and Keyword Arguments 79
Unpacking 80
Unpacking list to tuple 81

Unpacking String, Set, Dictionary 82
Extended Unpacking (Python 3.5 ထိ အနိမ့်ဆုံး လိုအပ်ပါတယ်) 85
List extended unpacking 86
String extended unpacking 86
Tuple extending unpacking 87
list extended unpacking using tuple 87
Arbitrary Arguments 90
Positional arguments and Arbitrary arguments 91
List Pass to Function 93
Keyword Argument a Deeper Look 94
End of Positional Arguments (*) 95
**kwargs 96
End of positional argument and keyword argument 96
Simple Function Timer 97
Default Values 99
Parameter Defaults problem 101
Docstrings 105
Function Annotations 107
Lambda Expressions/function 109
Declaration of lambda function 110
More Example For lambda with two parameters 110
Higher-Order Function 111
Lambda Function with *args and **kwargs 112
Lambda, args and sum() function 113
Lambda with Sorted () 113
Ascending dictionary with Lambda and sorted() 116
Ascending last elements 117
Random Module 117
Uniform from random 118
Randint from random 118
Randrange from random 118
Choice from random 119

Random, sorted and lambda 119
Function Introspection 120
__name__,__default__,__kwdefaults__ 122
__code__ attribute 123
Co_varnames 123
Co_argcount 124
Inspect module 124
Different between Function and Method 124
Getsource 125
Getmodule 126
Getcomments 126
Callable 126
Different Types of Callable 127
Some Object are not callable 129
Iterable 130
map() 131
Map with Lambda expression 133
Generator 133
Zip() Function 135
Filter () 136
Reduce () function 137
Warning Our Reduce Function 140
Partial () Function 140
Operator Module 143
Comparison operator from operator module 144
Scope of Variable 145
Nonlocal Variables 147
Closure 149
__closure__ Attribute 152
Decorator 153
Counter Application with Decorator 155
@Property 156

Wraps 157
Logger Application 158
Timer Application Using Decorator 160
Cache Fibonacci using Decorator 161
Fibonacci Using Recursive 162
Factorial with Decorator 166
@lru_cache 166
Modules 168
Globals Function 170
Delete Module 171
Object and Class 172
Adding attributes 173
Adding Behaviors 174
Self 175
Update Attribute 177
Class Attribute 179
Class Function 180
Inheritance 182
Multiple Inheritance 184
Multi-level Inheritance 185
Issubclass () method 186
isinstance(obj , class) method 187
Encapsulation 188
Private 189
Polymorphism 190
Polymorphism with class method 191
Operator Overloading 192
Method Overloading 194
Method Overriding 196
Abstraction 197
Abstraction For What? 198
Abstract Method and Class 198

Super() 199
File Handling 200
Reading From File 202
Readlines and Readline 203
Writing the File 204
Creating a File 205
File Accessing Mode 205
With Statement with File 206
File Pointer 206
seek() Function 207
Exception 210
Handling Exception 211
Try , except , finally 213
Raise 214
Creating Own Exception 215
JSON (JavaScript Object Notation) 216
JSON Values To Python 218
Serialization JSON 220
Dump and Dumps 221
Deserializing JSON 222
Load and Loads 222
Project Using Weather API 224
Database With MySQL 231
Connecting to MySQL Database 243
Creating Database 244
Creating a Table 245
Adding More column 247
Adding Data to the Table 248
Insert Many Data to the Table 249
Fetching 250
Fetching only one 251
Format 251

Using where 251
Order 253
Update 254
Delete 255
Join Table 255
MySQL in CLOUD 257
Connect To Cloud Database With Python 261
Creating a Database Table in Cloud 262
MongoDB with Python 263
Data Structure Between MySQL and MongoDB 264
Connection MongoDB with Python 270
MongoDB Primary Key 272
Distinct 274
insert_many() 274
Properties Zero 276
Properties One 277
MongoDB Query With Operator 278
MongoDB REGEX 281
Ascending and Descending Data 282
Delete 282
Drop collection 284
MongoDB CSV file 284
Update 287
Update Many 289
limit() 290
Atlas Cloud 291
Django(Web Development) 298
First Project With Django 300
First Step 304
Second Step 305
Third Step 305
Django Template Language 306

MongoDB and Django 309
Template Inheritance 312
Django Form Action 315
GET 318
POST 319
Application With PostgreSQL 320
Migration 323
Adding template in Application 331
Application Design 333
Adding Data To PostgreSQL 337
Fetching Data From PostgreSQL 339
Delete Items 343
Practical RESTful API 344
Django REST Framework 346
Migration 348
Serializers 349
Viewsets 350
Routers and DefaultRouter 350
Postman 353
Django Form Login/out 358
Django Admin 364
LoginView 365
is_authenticated 370
NumPy 373
NumPy Ndarray 374
Creating a ndarray function and object 374
Dimensions 375
Empty 375
C - Contiguous layout 376
Jupyter Notebook 379
Cell Types 383
Code 384

Markdown Cells 385
LaTeX equations 386
Adding File or Image 386
Raw NBconvert 388
Nbextensions 389
Collapsible Heading 391
Autopep8 392
Hinterland 393
NumPy.Zeros 393
NumPy.ones 393
Create NumPy Array With data 394
NumPy.frombuffer 395
NumPy.fromiter 396
Incremental Sequences 397
Logarithmic Sequences 397
Meshgrid Arrays 398
Properties of other Arrays 400
Full_like 400
Matrix Arrays 401
Indexing and Slicing 403
Reshaping and Resizing 405
flatten() 405
Newaxis 405
Expand_dims 406
Vstack , hstack 407
Concatenate 408
Arithmetic Operations 409
Element Wise Elementary Mathematical Functions 409
Trigonometric Functions 410
Unit Circle in Trigonometric 411
Sine , Cosine , Tangent 414
Rounding Functions 420

Quadrants of the coordinate plane 421
Two - variable linear equation introduction 423
Intercepts 426
Intro to slope 428
Negative Slope 428
Linear Regression Using Least Squares Method (Best Fit Line) 431
Correlation Coefficient ® 437
Python Program (LSR) 439
Data Science Project Using Linear Regression 440
Data Gathering and Cleaning 441
Exploratory and Visualise the Data 446
Evaluation 449
Matplotlib For Visualization 452
Packages Management 454
Matplotlib Example Project 458
Character Description 460
Character Colour 461
Bar Charts 462
Bar Charts and CSV 465
Pie Chart 469
Matplotlib Circle (Donut Chart) 472
Matplotlib Real Time Data 476
Wave Form (FuncAnimation) 478
3D Plot 480
Image Processing Project 485
Why Gray? 493
MedianBlur (Median Filtering) 494
Median mask 496
Simple Thresholding 499
Binary Inverted Thresholding 501
Truncate Thresholding 501
ToZero Thresholding 502

ToZero Invert Thresholding **502**

Adaptive Thresholding **503**

Gaussian Kernel and Filtering **505**

CV2.bitwise_and() **510**

Installation

Python အားအသုံးပြုနိုင်ရန်အတွက် Python ကို download ဆွဲပေးဖို့ လိုအပ်ပါတယ်။

https://www.python.org/downloads/ ယခု link သို့သွားပါ။ မိမိအသုံးပြုနေသော OS ရွေးချယ် ပြီး download ဆွဲပါ။ exe file ကို install ပြုလုပ်ချိန်တွင် သတိပြုရန်မှာ ပထမဆုံးပေါ်လာသော window form တွင် Add path ကို အမှန်ခြစ် ပေးခဲ့ပါ။ ပြီးလျင် Install Now ကို နှိပ်ပါ။ ပုံတွင် ကြည့်နိုင်ပါသည်။

Python ကို install လုပ်ပြီးလျှင် စစ်ဆေးရန်အတွက် window key + r ကိုနှိပ်ပါ။ ထို့နောက် cmd ဟုရေးသားပြီး ok ကိုနှိပ်ပါ။ ထိုကဲ့သို့ ပြုလုပ်ပြီးနောက် ပေါ်လာသော console တွင် python ဟုရေးသားပါ။ ပြီးလျှင် enter ကိုနှိပ်ပါ။

ယခုပုံအတိုင်း ပေါ်လာလျှင် python ကို အောင်မြင်စွာ install လုပ်ပြီး ဖြစ်သလို python ကို စတင်

အသုံးပြုနိုင်ပြီဖြစ်ပါတယ်။

Starting " Hello World "

Hello World program တစ်ပုဒ်ကို စတင်ရေးသားပါမည်။ အထက်တွင် ပြထားသည့် console တွင် print(" Hello world ") ဟုရေးသားပြီး enter ကိုနှိပ်ပါ။

ယခုပြထားသည့် ပုံအတိုင်း Hello World ဆိုသည့် output ပေါ်လာလျှင် " Hello World" program ကို အောင်မြင်စွာရေးသားနိုင်ပါပြီ။

Using Visual Studio Code

Console ကိုလည်းကောင်း visual studio code ကိုလည်းကောင်း အသုံးပြုသွားမည် ဖြစ်ပါတယ်။ Visual Studio Code ကို download ပြုလုပ်နိုင်ရန်အတွက် https://code.visualstudio.com/download link ကိုသွားပါ။ မိမိ OS အလိုက် download ဆွဲပါ။ Install လုပ်ပြီးနောက် visual studio code ကိုဖွင့်ပါ။ ထို့နောက် file ထဲကိုသွားပါ။ ပြီးလျှင် open folder ကိုနှိပ်ပါ။

ပေါ်လာသော window form နေရာတွင် မိမိ ကြိုက်နှစ်သက်ရာ နေရာ တစ်ခုသို့ သွားပြီး folder တစ်ခုကို တည်ဆောက် ပါ။

ပြီးလျှင် ထို folder ကို တစ်ချက်နှိပ်ပြီး select ကို နှိပ် လိုက်လျှင် visual studio code ထဲ တွင် မိမိတို့ ဆောက်ထားသော folde ပေါ်လာသည်ကို မြင်ရမည် ဖြစ်ပါတယ်။ ထို folder ထဲတွင် program များကို ရေးသားထားမည် ဖြစ်ပါတယ်။ အောက်ပါ ပုံများကို ကြည့်ပါ။

စာရေးသူအနေဖြင့် Book ဆိုသည့် folder တစ်ခုကို တည်ဆောက်ထားသည့်အတွက် Book folder ကိုမြင်ရတာ ဖြစ်ပါတယ်။ ထို folder ထဲတွင် python program file များကို ထည့်သွင်းရေး သားသွားမည်ဖြစ်ပါတယ်။ ညာဘက်ခြမ်းက Welcome page ကိုပိတ်လိုက်ပါ။ ထို့နောက် Book folder အောက်တွင် app.py ဆိုသည့် python file တစ်ခုကို တည်ဆောက်ပါမည်။

ယခုပုံတွင် ပြထားသည့်အတိုင်း Book folder ဘေးတွင်ရှိသော new file icon ကို နှိပ်လိုက် သည်နှင့် တစ်ပြိုင်နက် text box တစ်ခု ပေါ်လာမည် ဖြစ်ပါတယ်။ ၎င်း box ထဲတွင် app.py ဆို သည့် စာသားကို ရေးသားပြီး enter ကိုနှိပ်ရမည် ဖြစ်ပါတယ်။ ထို့နောက် အောက်ပါပုံတွင် ပြထား သည့်အတိုင်း Book folder ထဲတွင် app.py ဆိုသည့် python file တစ်ခုကို မြင်တွေ့ရမည် ဖြစ်ပါတယ်။ ထိုကဲ့သို့ python file ကို create လုပ်လိုက်သည်နှင့် အောက်တွင်ပြသထားသော ပုံအတိုင်း notification များတက်လာမည် ဖြစ်ပါတယ်။

install ကိုသာနှိပ်ပါ။ ထို့နောက် အောက်တွင် ပြသထားသောပုံအတိုင်း extension ကိုသွားပါ။

အဝါရောင် မြှားဖြင့်ပြထားသော အကွက်သည် extension များကိုထည့်သွင်းနိုင်သော နေရာ ဖြစ်သည်။ ၎င်းအကွက်ကို နှိပ်ပါ။ ပြီးလျှင် အနီရောင်မြှားဖြင့်ပြသထားသော search extension….. bar တွင် python ဟုရေးသားပြီး enter ကိုနှိပ်ပါ။ ပထမဆုံးနေရာတွင် ပေါ်လာသည့် star ပြထား သည့် python ကို နှိပ်ပါ။ ညာဘက်ပိုင်းတွင် မြင်ရသည့်အတိုင်း ပေါ်လာမည် ဖြစ်ပြီး install မလုပ်ရ သေးလျှင် install ကိုနှိပ်ပါ။

Start Python in Visual Studio Code

Python ကို visual studio code ဖြင့် စတင်ရေးသားမည် ဖြစ်ပါတယ်။

ယခုပုံတွင် ပြသထားသည့်အတိုင်း app.py ထဲတွင် program တစ်ကြောင်းကို ရေးသားထား မည် ဖြစ်ပါတယ်။ အထက်ပါပုံအတိုင်းရေးသားပြီးလျှင် program ကို save ရန်အတွက် file ထဲက save(Ctrl+S) ကိုနှိပ်ပါ။

ယခုပုံတွင် ပြထားသည့် အတိုင်း app.py ပေါ်တွင် right click ကိုနှိပ်၍ open in integrated terminal ဆိုတာကို နှိပ်ပါ။

အောက်ပါပုံတွင်ပြထားသည့်အတိုင်း terminal တစ်ခုပွင့်လာသည်ကို မြင်တွေ့ရမှာ ဖြစ်ပါတယ်။

ထို terminal သည် မိမိ တို့၏ python program များကို run ရန်အတွက် နေရာဖြစ်ပါတယ်။ python program ကို run ရန်အတွက် ပုံတွင် ပြသထားသည့်အတိုင်း Python app.py ဟုရေး သားကာ run ရမည်။ ကျွန်တော်တို့ရေးသားထားသော program သည် app.py ထဲတွင်ရှိပြီး app.py file ကို run ချင်တာ ဖြစ်သည့်အတွက် python app.py ဟုရေးသားရခြင်း ဖြစ်ပါတယ်။ ထိုသို့ရေးသားပြီးလျှင် enter ကိုနှိပ်လိုက်သည် နှင့်တစ်ပြိုင်နက် ကျွန်တော်တို့ရေးသားထားသည့် hello world ဆိုသည့် output ကိုမြင်တွေ့ရမှာ ဖြစ်ပါတယ်။

Python Standard Input and Output

Python programming တွင် built-in functions များစွာပါဝင်ပါတယ်။ standard input အနေဖြင့် input( ) function ကိုအသုံးပြုပြီး standard output အနေဖြင့် print( ) ကို အသုံးပြုပါတယ်။

format function ကိုအသုံးပြုပြီးတော့လည်း format (ပုံစံတကျ) ဖြင့် print ထုတ်နိုင်ပါ တယ်။ format function ကိုသုံးရာတွင် data များကိုဖော်ပြဖို့အတွက် curly bracket { } ကိုအသုံးပြုပါတယ်။ format function ကိုသုံးရာတွင် format ရှေ့ ၌ dot (.) ကိုရေးပေးရမည်ဖြစ် ပြီး format function ထဲမှာလည်း မိမိဖော်ပြချင်သော data များကို အစဉ် လိုက် comma (,) ခြား ပြီး ရေးသားပေးရမည် ဖြစ်ပါတယ်။ app.py ထဲတွင်အောက်ပါ program ကိုရေးသားထားမည်ဖြစ် ပါသည်။

Sample Program (1)

```
a = 10; b = 5;

print('The value of a is {} and b is {}'.format(a,b))
#output

#The value of a is 10 and b is 5
```

အထက်ပါ program (1) ကိုကြည့်ပါ။ .format( ) ထဲတွင် a နှင့် b ကိုရေးသားထားပြီး ၎င်း a နှင့် b ၏ တန်ဖိုးများကို လိုချင်တယ်ဆိုရင် curly bracket { } ဖြင့်ထုတ်ပေးရပါတယ်။ ကျွန်တော်တို့ ရေးသားထားသော program မှာဆိုရင် curly bracket { } နှစ်ခုပါဝင်ပြီး first curly bracket { } သည် .format( ) function အတွင်းရှိ first element ဖြစ်သော a ကိုရည်ညွှန်းပြီး second curly bracket {} သည် second element ဖြစ်သော b ကို ရည်ညွှန်းပါတယ်။ program ကို run လျှင် output အနေဖြင့် The value of a is 10 and b is 5

ဆိုတာကို မြင်တွေ့ရမှာ ဖြစ်ပါတယ်။ C programming တွင်မူ integer များအတွက် ထိုကဲ့သို့သော စာသားများကြားတွင် ဖော်ပြလိုသည့်အခါမျိုးဌ %d ကို အသုံးပြုပါတယ်။

Format function ကိုအခြားသော နည်းလမ်းဖြင့်လည်း ထပ်မံအသုံးပြုနိုင်ပါသေးတယ်။ .format( ) function အတွင်းရေးသားထားသော element များသည် index0 မှစတင်ပြီး အစဉ်လိုက်

နေရာယူပါတယ်။ ထိုသို့ အစဉ်လိုက်နေရာယူထားသော element များကို print ထုတ်သောအခါတွင် မိမိလိုသလို index number များကိုအသုံးပြု၍ element များကို ရှေ့နောက် ပြောင်းလဲနိုင်ပါတယ်။ ရှေ့ဆုံး element ကိုပေါ်စေချင်သော အခါမျိုးတွင် curly bracket{ } အတွင်းဌ { 0 } ကိုအသုံးပြုပါတယ်။ ထို့နည်းတူ ဒုတိယ element ကိုပေါ်စေချင်သော အခါမျိုးဌလည်း { 1 } ကိုအသုံးပြုပါတယ်။ အောက်တွင် sample program (2) ကို output နှင့်တကွ ဖော်ပြထားပါတယ်။

Sample Program (2)

app.py

```
print('hello {2} {3} from {0} {1} '.format('green','hackers','win','htut'));
#output

#hello win htut from green hackers
```

ထို့ပြင် format function ကိုသုံးပြီး string များကို print ထုတ်ရာတွင်လည်း keyword arguments များကိုလည်းအသုံးပြုနိုင်ပါသေးတယ်။ sample program (3) ကို output နှင့်တကွ အောက်တွင် ဖော်ပြထားပါတယ်။

Sample Program (3)

```
print('Hello {name} team {gh}'.format(name='Win Htut', gh='Green Hackers'))

#output

#Hello Win Htut team Green Hackers
```

Python တွင် ဒဿမကိန်းများကို print ထုတ်ရာဌ % operator ကိုသုံးပြီး မိမိတို့ လိုအပ်သလို control လုပ်နိုင်ပါသေးတယ်။ ဒဿမနှစ်နေရာထိ print လုပ်ချင်သောအခါတွင် %.2f ဟုရေးသားရပြီး 3 နေရာထိ print ထုတ်ချင်သောအခါတွင်မူ %.3f ဟုရေးသားရပါတယ်။ sample program (4) ကို output နှင့်တကွ အောက်တွင် ဖော်ပြထားပါတယ်။

Sample Program (4)

```
a = 10.12345

print('the value of a is %.3f '%a)
#output
#the value of a is 10.123
print('the value of a is %.2f '%a)
```

```
#output

#the value of a is 10.12
```

အထက်ပါ program (4) တွင် a အတွင်းရှိ value ကို % operator ဖြင့် control လုပ်မည့် ဖြစ်သည့်အတွက် print( ) function အတွင်းရှိ single quote အပိတ်၏ နောက်တွင် % sign ဖြင့် a ကို ရေးသားပေးရမည်ဖြစ်ပြီး စာသားအတွင်း ပေါ်ချင်သည့်နေရာတွင် %.3f သို့မဟုတ် %.2f အစရှိ သဖြင့် ရေးသားပေးရမည် ဖြစ်တယ်။

Standard Input in Python Programming

Standard input ဆိုတာ user ဆီမှ ထည့်ပေးလိုက်သော သို့မဟုတ် program ထဲသို့ ဝင်လာ သော data or elements များကို input လို့ခေါ်ပါတယ်။ keyboard and mouse တို့သည် input များဖြစ်ပြီး monitor, soundbox and etc တို့သည် output များဖြစ်ကြပါတယ်။ Python programming တွင် user ဆီမှ input များကို ဖတ်ရန် input( ) function ကိုအသုံးပြုနိုင်ပါတယ်။ Python programming သည် အရမ်းရိုးရှင်းသကဲ့သို့ input များကို ဖတ်ရာတွင်လည်း အရမ်းကို လွယ်ကူပါတယ်။ input sample program ကို output နှင့်တကွ အောက်တွင် ဖော်ပြထားပါတယ်။

Sample program (5)

```
data = input('enter some data :')
print('user input data is :',data)
#output
#enter some data :hello
#user input data is : hello
```

Standard Data Type

Python programming မှာဆိုရင် standard data type အနေနဲ့ 6 မျိုးရှိပါတယ်။

1. Numbers
2. String
3. List
4. Tuple
5. Dictionary
6. Set

Numbers in Python

Number data type တွေဆိုတာ numeric values တွေကို သိုလှောင်ဖို့အတွက် သုံးပါတယ်။ value တစ်ခု assign သတ်မှတ်ပေးလိုက်တာနဲ့ number objects ကို ပြုလုပ်နိုင်ပါတယ်။

For example:

```
var1 = 1
```

var2 = 2

del ဆိုတဲ့ statement ကိုအသုံးပြုပြီး number objects တွေကို ဖြတ်ထုတ်နိုင်ပါတယ်။

For example:

del var1

Del ဆိုတာကိုသုံးပြီး single object တစ်ခုတည်းတင်မကပဲ multiple objects တွေကိုပါ ဖြတ်ထုတ်နိုင်ပါသေးတယ်။

For example:

del var1[var2,var3,var4]

Python programming language မှာ number data type ကို numerical type အနေနဲ့ သုံးမျိုး support ပေးထားပါတယ်။

- int (single integer)
- float (floating point real values)
- complex (complex)
- booleans

int မှာဆိုရင် long integers အနေနဲ့ ကိုယ်စားပြုပါတယ်။ ဆိုလိုချင်တာက python မှာ integer တွေအားလုံးသည် long တွေဖြစ်ပါတယ်။ Float ဆိုတာကတော့ ဒဿမ ကိန်းတွေကို ဆိုလိုတာပါ။

Int	float	complex
10	0.0	3.14j
100	15.20	45.j
-786	-21.9	9.322e-36j
80	32.3+e18	.876j
-490	-90.	-.6545+0j
-0*260	-32.54e100	3e+26j
0*69	70.2-E12	4.53e-7j

Complex numbers

Complex numbers တွေဆိုတာ real number နှင့် imaginary number ပါဝင်တာကို ဆိုလိုတာပါ။

for example:

Bj

အထက်ပါ ဖော်ပြချက်မှာဆိုရင် B သည် real number ဖြစ်ပြီး j သည် imaginary number ဖြစ်ပါတယ်။ j သည် square root of -1 နှင့် အတူတူပင် ဖြစ်ပါတယ်။

for example:

3+7j

အထက်ပါ ဖော်ပြချက်မှာဆိုရင် 3 သည် real part ဖြစ်ပြီး 7j သည် imaginary part ဖြစ်ပါသည်။ ထို့ကြောင့် 3+7j ကို complex number ဟုခေါ်ပါတယ်။

Real number + imaginary number = complex number

Booleans

Booleans data type ထဲမှာတော့ True နှင့် False တို့ပါဝင်မှာ ဖြစ်ပါတယ်။ True ကို number အနေဖြင့် ပြောမယ်ဆိုရင် 1 ဖြစ်ပြီး False သည် 0 ဖြစ်ပါတယ်။

```
Python 3.8.3rc1 (tags/v3.8.
Type "help", "copyright", "
>>> bool(1)
True
>>> bool(0)
False
>>> bool(0.0)
False
>>> bool(2.1)
True
>>>
```

bool(1) ဆိုသည့် code ကိုရေးလိုက်ရင် result အနေနဲ့ True ဆိုတာကို ရရှိမှာ ဖြစ်ပါတယ်။ ရှေ့က bool ဆိုတာကတော့ သူ့ရဲ့ data type ဖြစ်ပါတယ်။ bool(0) လို့ ရေးလိုက်ရင် False ဆိုတဲ့ result ကိုရရှိမှာ ဖြစ်ပါတယ်။ ဒဿမ ကိန်းတွေထည့်ရင်လည်း number တစ်ခုခု ဖြစ်နေခဲ့ရင် True ဆိုတဲ့ result ကိုပဲရရှိမှာ ဖြစ်ပါတယ်။ bool(-2) ထိုကဲ့သို့ အနုတ်ကိန်းတွေ ထည့်ခဲ့ရင်လည်း True ကိုပဲ ရရှိမှာ ဖြစ်ပါတယ်။

Type Conversion

Python တွင်မည်သည့် value မှာမဆို data type တွေရှိပါတယ်။ Data Type တွေဆိုတာ data တွေကိုခွဲခြားထားခြင်းဖြစ်ပြီး compiler or interpreter အား data တွေကို ဘယ်လိုပုံစံဖြင့် အသုံးပြုမည်ဖြစ်ကြောင်း ပြောကြားခြင်း ဖြစ်ပါတယ်။ Type Conversion ဆိုသည်မှာ data type တစ်ခုကနေ အခြား data type တစ်ခုသို့ ပြောင်းလဲခြင်းကို ဆိုလိုပါတယ်။ Dataတွေကို ကိုင်တွယ်တဲ့နေရာမှာ type တွေကို လိုသလိုပြောင်းလဲခြင်းဖြင့် ပိုမိုအသုံးပြုရ လွယ်ကူစေပါတယ်။ Type conversion မှာဆိုရင် implicit and explicit ဆိုပြီး နှစ်မျိုးရှိပါတယ်။

Implicit Type Conversion or coercion

Implicit Type Conversion သည် runtime မှာ python ကနေပြီး data တွေကို တိုက်ရိုက် conversion ပြုလုပ်ခြင်း ဖြစ်ပါတယ်။ user ကနေပြုလုပ်ပေးစရာ မလိုပါဘူး။ data များအား မည်သည့် data type ဖြစ်သည်ကို သိလိုပါက type() function

ကိုအသုံးပြုနိုင်ပါတယ်။ အောက်ပါ program တွင် c_sum ရဲ့ data type ကိုသိနိုင်ရန် type() function ကိုသုံးထားပါတယ်။

Sample Program(6)

```
a_int = 1

b_float = 1.0
c_sum = a_int + b_float
print(c_sum)   #2.0

print(type(c_sum))     #<class 'float'>
```

အထက်ပါ program (6) တွင် int and float တို့ကို ပေါင်းရာ၌ int သို့ပြောင်းလဲမသွားဘဲ output တွင် float data type သို့ပြောင်းလဲသွားပါတယ်။ python တွင် int data size ထက် float data size က ပိုများပါတယ်။ float မှ int သို့ပြောင်းလျှင် float တွင်ဝင်နေသော ဒဿမ ကိန်းများ ပျောက်ဆုံးသွားနိုင်ပါတယ်။

Explicit Type Conversion

Explicit Type Conversion သည် user မှ data များကို လိုသလိုပြောင်းလဲခြင်း ဖြစ်ပါတယ်။ python တွင်မူ int() , float( ) , str( ) စသည့် function များကို explicit type conversion တွင်အသုံးပြုနိုင်ပါတယ်။ Explicit Type Conversion ကို type casting လို့လည်း ခေါ်ဆိုပါသည်။

Sample Program (7)

```
a_int = 1

b_float = 1.0
c_sum = a_int + b_float
#in this case the type is float
print(type(c_sum))
#converting float to int using int()
new_int=int(c_sum)
print(type(new_int))

#output
#<class 'float'>

#<class 'int'>
```

အထက်ပါ Sample Program (7) run ကြည့်လျှင် output အနေဖြင့် data type နှစ်ခု ထွက်လာပါမည်။ ပထမတစ်ခုသည် python မှ handle လုပ်ထားသော implicit type ဖြစ်ပြီး ဒုတိယတစ်ခုဖြစ်သည့် int သည် user မှ handle လုပ်ထားသော explicit type ဖြစ်သည်။ သတိပြုရန် အချက်များမှာ

- Implicit type တွင် data loss ဖြစ်ခြင်းကို ကာကွယ်နိုင်ရန် python interpreter သည် data size သေးရာမှာ ကြီးရာသို့ ပြောင်းပေးသည်။
- Explicit type သည် data loss ဖြစ်နိုင်ပါသည်။ အဘယ်ကြောင့်ဆိုသော် user မှ predefined function များကိုသုံးပြီး data type များအား လိုသလို ပြောင်းလဲခြင်း ကြောင့် ဖြစ်သည်။

Data type conversion with string

```
Python 3.8.3rc1 (tags
Type "help", "copyrig
>>> print("hello")
hello
>>> print('hello')
hello
>>>
```

String ဆိုတာ character တစ်လုံးတည်း သို့မဟုတ် တစ်လုံးထက်ပိုပြီး စုပေါင်းထားခြင်းကို string ဟုခေါ်ပါသည်။ string တွင် letters, numbers and symbols တို့ ပါဝင်ပါတယ်။

Sample Program (8)

(integer type ဖြင့် နောက်ဆုံး စာကြောင်းတွင် printing လုပ်ထားပါသည်)

```
int_one = 15
int_two = 10
total = int_one + int_two
#printing like integer type
print('The total is', total)
```

အထက်ပါ Sample Program (8) run ကြည့်လျှင် output 25 ကို ရပါမည် ။ program error တက်မည် မဟုတ်ပါ။ ယခုရေးထားသော integer type အား string type ဖြင့် အောက်ပါအတိုင်း ပြန်လည် ရေးသားပါမည်။

Sample Program (9)

```
int_one = 15

int_two = 10
total = int_one + int_two
#printing like integer type

print('The total is'+total)
```

အထက်ပါ Sample Program (9) run ကြည့်လျှင် အောက်ပါအတိုင်း Type Error ရပါမည်။ ဆိုလိုသည်မှာ implicit နည်းလမ်းနဲ့ integer မှတစ်ဆင့် string သို့ မပြောင်းလဲနိုင်ကြောင်း ဖော်ပြထားခြင်းဖြစ်ပါတယ်။

```
#output
```

```
# Traceback (most recent call last):
#   File ".\app.py", line 5, in <module>
#     print('The total is'+total)

# TypeError: can only concatenate str (not "int") to str
```

Explicit နည်းလမ်းကိုသုံးပြီး int မှ string သို့ type casting ပြုလုပ်ပါမည်။ အောက်ပါ program ကိုကြည့်လျှင် + operator နှင့် , တို့ကွာခြားသွားသည်ကို မြင်နိုင်ပါသည်။ အောက်တွင် ဖော်ပြထားသော Sample Program(10) run ကြည့်လျှင် output အနေဖြင့် 25 ကိုရပါမည်။

Sample Program (10)

```
int_one=15
int_two=10
total=int_one+int_two
#printing like integer type
print('The total is '+str(total))
#output
#The total is 25
```

String in Python

Python programming မှာဆိုရင် string ကို single quotes (‘some thing’)or double quotes (“ some thing ”)ထဲမှာထည့်ပြီး တစ်စုတစ်စည်းထဲ သတ်မှတ်ထားခြင်းကို ဆိုလိုပါတယ်။ စာသားတွေကို single quote ထဲမှာ ရေးရင်လည်းရသလို double quote ထဲမှာရေးရင်လည်း ရပါတယ်။ သို့သော် မိမိရေးသားဖော်ပြလိုသော စာသားထဲမှာ single quote ပါနေခဲ့မယ်ဆိုရင် double quote ကိုသုံးသင့်ပါတယ်။ print(“hello Myanmar’s people”) ထိုကဲ့သို့ဖြစ်ပါတယ်။ သို့မဟုတ် Triple quote ကိုလည်း အသုံးပြုလို့ရပါသေးတယ်။

```
Python 3.8.3rc1 (tags/v3.8.3rc1:802e
Type "help", "copyright", "credits"
>>> print("""'hello world'""")
'hello world'
>>>
```

မိမိဖော်ပြလိုသော စာသားကို variable တစ်ခုကို အသုံးပြုပြီးလည်း ဖော်ပြလို့ရပါသေးတယ် ။

ပထမ code စာကြောင်းတွင် Hello world ဆိုသည့် စာသားကို mgmg ဆိုသည့် variable ထဲသို့ = (equal to) ဆိုသည့် assignment operator ကိုအသုံးပြုပြီး ထည့်လိုက်သည်။ ထို့ကြောင့် mgmg ဆိုသည့် variable ကို program ထဲမှာ မြင်တိုင်း Hello world ဆိုသည့် စာကြောင်းကိုပဲ ထုတ်ပေးမည်ဖြစ်သည်။ ဒုတိယစာကြောင်းတွင် mgmg ဆိုသည့် variable ကို ခေါ်သောကြောင့် result အနေနှင့် Hello world ဆိုသည့် စာသားကို ရရှိခြင်းဖြစ်သည်။ Len ကိုအသုံးပြုပြီး စာသားတွေရဲ့ အရေအတွက်ကို ထုတ်ပါမယ်။

Sample Program (11)

```
mgmg = ("Hello world")

print(len(mgmg))
```

len(length) သည် မိမိထုတ်လိုက်တဲ့ variable ထဲမှာရှိသော စာသားရဲ့ အလုံးအရေအတွက်ကို ပြပေးပါတယ်။ space ကိုပါ character တစ်ခုအနေနဲ့ ထည့်သွင်းရေတွက်ပါတယ်။ အထက်ပါ program ကို run လိုက်ရင် output အနေနဲ့ 11 ကိုရရှိမှာဖြစ်ပါတယ်။

Example:

```
str= "Hello World" (or)

str= 'Hello World'
```

Python မှာဆိုရင် slice operator တွေကိုသုံးပြီး စာသားတွေကို တစ်လုံးချင်းစီ ဖော်ပြခိုင်းလို့ရပါတယ်။ သူ့ရဲ့ index တွေကို zero ကနေပဲစပြီး ရေတွက်ပါတယ်။

Accessing the Character using Slice operator

[ ] = တစ်လုံးခြင်းစီကို သီးခြားဖော်ပြလိုတဲ့အခါမှာ သုံးပါတယ်။

```
print (str[0])
```

```
0 1 2 3 4 5 6 7 8 9 10
mgmg = ("Hello world")
print(len(mgmg[0]))
```

Example :

ယခု program ကိုရေးကြည့် လိုက်မယ်ဆိုရင် result အနေနဲ့ 1 ကိုရှိမှာဖြစ်ပါတယ်။

Sample Program (12)

```
str = ("Hello world")

print(str[10])
```

အထက်ပါ Sample Program (12) run ကြည့်မယ်ဆိုရင်လည်း result အနေနဲ့ character d ကို ရရှိမှာ ဖြစ်ပါတယ်။

Sample Program (13)

```
str = ("Hello world")
print(len(str[11]))
#output
# Traceback (most recent call last):
#   File ".\app.py", line 4, in <module>
#     print(len(str[11]))
# IndexError: string index out of range
```

အထက်ပါ Sample Program (13) run ကြည့်မယ်ဆိုရင် out of range ဆိုတဲ့ error ကို မြင်တွေ့ရမှာပါ။ ဘာကြောင့်လဲဆိုတော့ စုစုပေါင်း character 11 လုံးသာပါဝင်ပြီး index 10 သည် အများဆုံးဖြစ်နေတဲ့အတွက်ပါ။

Str = စာသားတွေအားလုံးကို ဖော်ပြလိုတဲ့အခါမှာ သုံးပါတယ်။

Sample Program (14)

```
print(str)

str = ("Hello world")

print(str)
```

အထက်ပါ Sample Program (14) run လိုက်ရင် result အနေနဲ့ Hello world ဆိုတဲ့ စာသား အပြည့်အစုံကို ရရှိမှာပါ။print(str[0:5]) ရှေ့က 0 သည် စမဲ့ index number ဖြစ်ပြီး 5 သည် ဆုံးမဲ့ index number ဖြစ်ပါတယ်။ ဥပမာ ကို Sample Program (15)တွင်ကြည့်ပါ။

Sample Program (15)

```
str = ("Hello world")
print(str[0:5])
#output

#Hello
```

ယခု Sample Program (15) run ကြည့်မယ်ဆိုရင် result အနေနဲ့ Hello ဆိုတာကို ရရှိမှာဖြစ်ပါတယ်။ index အစကနေအဆုံးထိ character တွေကိုထုတ်ပေးပါတယ်။ ပိုပြီးနားလည်သွားအောင် အောက်က program လေးတွေကို ဆက်ရေးကြည့်နိုင်ပါတယ်။

[2:5] = စာသားတွေကို မိမိလိုချင်သလောက်ပဲ ဖော်ပြလိုသောအခါမျိုးမှာ သုံးပါတယ်။

Sample Program (16)

```
str = ("Hello world")

print(str[2:5])

#llo
```

အထက်ပါ Sample Program (16) run လိုက်ရင် result အနေနဲ့ llo ဆိုတဲ့ character သုံးလုံးကို ရရှိမှာဖြစ်ပါတယ်။ ရှေ့က 2 သည်စမဲ့ index number ဖြစ်ပြီး နောက်က 5 သည် ဆုံးမဲ့ index number ဖြစ်ပါတယ်။ [2:] = သတ်မှတ်ထားတဲ့ index number ကနေ ကျန်တဲ့အဆုံးထိ ဖော်ပြလိုသော အခါမျိုးမှာ သုံးပါတယ်။

Sample Program (17)

```
str = ("Hello world")

print(str[2:])
#output
#llo world
```

အထက်ပါ Sample Program (17) run လိုက်ရင် result အနေနဲ့ llo world ဆိုတာကို ရရှိမှာဖြစ်ပါတယ်။ ထည့်ပေးလိုက်တဲ့ index number ကနေစပြီး နောက်ပိုင်းမှာရှိတဲ့ စာလုံးတွေ အားလုံးကို ဖော်ပြပေးမှာ ဖြစ်ပါတယ်။ [:5] = ထည့်ပေးလိုက်တဲ့ index number ရဲ့ အရှေ့ထိ စာသားအားလုံးကို ဖော်ပြပေးမှာဖြစ်ပါတယ်။ နောက်ပိုင်းစာလုံးတွေကိုတော့ ဖော်ပြပေးမှာ မဟုတ်ပါဘူး။

Sample Program (18)

```
str = ("Hello world")
print(str[:5])
#output
#Hello
```

အထက်ပါ Sample Program (18) run လိုက်မယ်ဆိုရင် result အနေနဲ့Hello ကိုရရှိမှာဖြစ်ပါတယ်။

```
0 1 2 3 4 5 6 7 8 9 10
mgmg = ("Hello world")
print(len(mgmg[0]))
```

ယခုပုံလေးကို ပြန်လည် ဖော်ပြပေးထားပါတယ်။ ပိုပြီး နားလည်သွားမှာပါ။ print(str * 2) = စာသားတွေကို နှစ်ကြိမ်ဖော်ပြလိုသောအခါမျိုးမှာ သုံးပါတယ်။

Sample Program (19)

```
str = ("Hello Green Hackers")

print(str*2)
#Output

#Hello Green HackersHello Green Hackers
```

အထက်ပါ Sample Program (19) run ကြည့်ရင် result အနေနဲ့ စာသားကို နှစ်ကြိမ် ဖော်ပြထားတာကို မြင်ရမှာဖြစ်ပါတယ်။

print (str + "WinHtut") = စာသားတွေကို တစ်ခုနဲ့တစ်ခုပေါင်းပြီး ဖော်ပြလိုတဲ့ အခါမှာသုံးပါတယ်။

Sample Program (20)

```
str = ("Hello Green Hackers")

print(str+"Win Htut")
#Output

#Hello Green HackersWin Htut
```

အထက်ပါ Sample Program (20) တွင် result အနေနဲ့ စာသားနှစ်ခုလုံးကို ဖော်ပြပေးထားတာကို မြင်ရမှာပါ။ String Handling တွေထဲက built–in method တွေဖြစ်တဲ့ upper and lower အကြောင်းကို ဆက်သွားပါမယ်။

upper()

Python programming မှာဆိုရင် upper() သည် built-in method ဖြစ်ပြီး string တွေကို လိုအပ်သလို handle လုပ်ဖို့ အသုံးပြုပါတယ်။ lowercase နဲ့ရေးထားတဲ့ characters တွေအားလုံးကို uppercase (အကြီး)အဖြစ် ပြောင်းလဲပေးလိုက်ပါတယ်။ တကယ်လို့ စာသားတွေထဲမှာ uppercase (အကြီး)တွေ ပါနေခဲ့မယ်ဆိုရင် မူလအတိုင်း အကြီးတွေကိုပဲ ပြန်လည်ဖော်ပြပေးမှာပါ။

Syntax:

String.upper()

Parameter:

upper(မည်သည့် parameter မှ မပါဝင်ပါ)

Return:

စာလုံးအသေးတွေကို အကြီးအဖြစ် ပြောင်းလဲပေးပါတယ်။

Sample Program (21)

```
str = ("Hello Green Hackers")
print(str.upper())
#output
#HELLO GREEN HACKERS
```

အထက်ပါ Sample Program (21) အရ စာလုံးအကြီးတွေကို ပြန်ဖော်ပြပေးတာ မြင်ရမှာဖြစ်ပါတယ်။

lower()

Python programming မှာဆိုရင် lower() သည် built-in method ဖြစ်ပြီး string တွေကို လိုအပ်သလို handle လုပ်ဖို့ အသုံးပြုပါတယ်။ uppercase နဲ့ရေးထားတဲ့ characters တွေအားလုံးကို lowercase (အသေး)အဖြစ် ပြောင်းလဲပေးလိုက်ပါတယ်။ တကယ်လို့ စာသားတွေထဲမှာ lowercase (အသေး)တွေ ပါနေခဲ့မယ်ဆိုရင် မူလအတိုင်း အသေးတွေကိုပဲ ပြန်လည်ဖော်ပြပေးမှာပါ။

Syntax:

String.lower()

Parameter:

lower (မည်သည့် parameter မှ မပါဝင်ပါ)

Return:

စာလုံးအကြီးတွေကို အသေးတွေအဖြစ် ပြောင်းပေးပါတယ်။

Sample Program (22)

```
str = ("Hello Green Hackers")
print(str.lower())
#output

#hello green hackers
```

အထက်ပါ Sample Program (22) အရ result အနေနဲ့ characters တွေအားလုံးကို lowercase (စာလုံးအသေး) တွေနဲ့ ဖော်ပြထားတာကို မြင်ရမှာပါ။

Sample Program (23)

```
str = ("Hello Green Hackers")
print(str.lower(1))
#output
#Traceback (most recent call last):
#   File ".\app.py", line 5, in <module>
#     print(str.lower(1))

# TypeError: lower() takes no arguments (1 given)
```

Upper or lower ထဲမှာ အထက်ပါအတိုင်း argument တစ်ခုခု ထည့်လိုက်မယ်ဆိုရင် အထက်ပါ Sample Program (24) တွင်ပြထားသည့်အတိုင်း error message ကို မြင်တွေ့ရမှာဖြစ်ပါတယ်။

isupper()

isupper() သည် python ရဲ့ built-in method ဖြစ်ပြီး string တွေကို handle လုပ်ဖို့အတွက် အသုံးဝင်ပါတယ်။ တကယ်လို့ characters တွေအားလုံးဟာ အကြီးတွေချည်းပဲ ဖြစ်နေခဲ့မယ်ဆိုရင် True ဆိုတဲ့ return value ကိုပြန်ရမှာဖြစ်ပြီး characters တွေထဲမှာ အကြီးတွေရော အသေးတွေရာ ပါဝင်နေခဲ့မယ်ဆိုရင်တော့ return value အနေနဲ့ False ကို ပြန်ရမှာဖြစ်ပါတယ်။

Syntax:

```
string.isupper()
```

Parameter:

isupper သည် မည်သည့် parameter မှ မပါဝင်ပါ။

Returns:

Characters အားလုံးသည် အကြီးဖြစ်နေရင် return value အနေနဲ့ True ကိုရမှာဖြစ်ပြီး အသေးတွေပါဝင်နေခဲ့ရင်တော့ False ကိုရရှိမှာဖြစ်ပါတယ်။

Sample Program (25)

```
str = ("HELLOMYANMAR")
print(str.isupper())
#output
#True
```

အထက်ပါ Sample Program (25) run ကြည့်ရင် result အနေနဲ့ true ကိုရရှိမှာဖြစ်ပါတယ်။ characters တွေ အားလုံးအကြီးတွေ ဖြစ်နေလို့ပါ။

islower()

islower() သည် python ရဲ့ built-in method ဖြစ်ပြီး string တွေကို handle လုပ်ဖို့အတွက် အသုံးဝင်ပါတယ်။ တကယ်လို့ characters တွေအားလုံးဟာ အသေးတွေချည်းပဲ ဖြစ်နေခဲ့မယ်ဆိုရင် True ဆိုတဲ့ return value ကိုပြန်ရမှာဖြစ်ပြီး characters တွေထဲမှာ အကြီးတွေရော အသေးတွေရာ ပါဝင်နေခဲ့မယ်ဆိုရင်တော့ return value အနေနဲ့ False ကို ပြန်ရမှာဖြစ်ပါတယ်။

Syntax:

```
string.islower()
```

Parameter:

islower သည် မည်သည့် parameter မှ မပါဝင်ပါ။

Returns:

Characters အားလုံးသည် အသေးဖြစ်နေရင် return value အနေနဲ့ True ကိုရမှာဖြစ်ပြီး အသေးတွေပါဝင်နေခဲ့ရင်တော့ False ကိုရရှိမှာဖြစ်ပါတယ်။

Sample Program (26)

```
str = ("hellomyanmar")
print(str.islower())
#output
#True
```

အထက်ပါ Sample Program (26) ကို run ကြည့်ရင် result အနေ နဲ့ True ကိုရရှိမှာဖြစ်ပါတယ်။ ဘာကြောင့်လဲ ဆိုတော့ စာလုံးတွေ အားလုံး အသေးတွေဖြစ် နေလို့ပါ။

Sample Program (27)

```
str = ("helloMyanmar")

print(str.islower())
#output
#False
```

M ဆိုတဲ့ character ကို အကြီးပြောင်း လိုက်တဲ့ အတွက် result မှာ False ဆိုပြီး ရရှိခြင်းဖြစ်ပါတယ်။

count()

count() သည် python programming ၏ built-in function ဖြစ်ပြီး parameter အနေနဲ့ထည့်ပေးလိုက်တဲ့ character ရဲ့ ဘယ်နှစ်လုံးရှိသလဲ ဆိုတဲ့ အရေအတွက်ကို return အနေနဲ့ ပြန်ပေးပါတယ်။

syntax:

string.count("မိမိထည့်လိုသော character ")

parameter:

မိမိ ရှာလိုသော character ကိုထည့်ပေးရပါတယ်။

Sample Program (28)

```
str = ("helloMyanmar")
print(str.count("l"))
#output
#2
```

အထက်ပါ Sample Program (28) မှာဆိုရင် character " l " ရဲ့ ဘယ်နှစ်ကြိမ်ပါလဲ ဆိုတဲ့ အကြိမ် အရေအတွက်ကိုရှာထားပါတယ်။ result အနေနဲ့ 2 ကိုပြန်ပေးပါတယ် ဘာကြောင့်လဲ ဆိုတော့ string ထဲမှာ " ll " ဆိုပြီး character နှစ်လုံးပါနေတဲ့ အတွက်ကြောင့်ဖြစ်ပါတယ်။

Sample Program (29)

```
str = ("helloMyanmar")
print(str.count("ll"))
#output
```

```
#1
```

အထက်ပါ Sample Program (29) ကတော့ character " ll " ဆိုတာကို ဘယ်နှစ်ခါ ပါလဲ စစ်ထားခြင်း ဖြစ်ပါတယ်။ " ll " သည် တစ်ခါတည်းပါသည့် အတွက်ကြောင့် 1 ဆိုပြီး return value ပြန်ရခြင်းဖြစ်ပါတယ်။

Sample Program (30)

```
str = ("helloMyanmar")
print(str.count("m"))
#output
#1
```

အထက်ပါ Sample Program (30) သည် character " m " ဘယ်နှစ်ကြိမ်ပါ သလဲဆို တာကို ရှာထားခြင်းဖြစ်ပါတယ်။

find()

find() သည် python programming ၏ built-in function တစ်ခုဖြစ်ပြီး character တစ်လုံးရဲ့ index တည်နေရာကို ရှာပေးတဲ့ နေရာမှာ အသုံးဝင်လှပါတယ်။

syntax:

string.find(" မိမိ ရှာလိုသော character ")

parameter:

မိမိ ရှာလိုသော character ကိုထည့်ပေးရပါတယ်။

Return:

ထည့်ပေးလိုက်သော character ၏ index နေရာ အတိအကျကို return value အနေနဲ့ ပြန်ပေးပါတယ်။

Sample Program (31)

```
str = ("helloMyanmar")
print(str.find("y"))
#output

#6
```

အထက်ပါ Sample Program (31) တွင် character y ကိုရှာထားပြီး result အနေနဲ့ y ရဲ့ Index တည်နေရာ အတိအကျ ကို ပြန်လည်ဖော်ပြပေးပါတယ်။

replace()

replace() ဆိုတာက python programming language ရဲ့ built-in function တစ်ခုဖြစ်ပါတယ်။ return အနေနဲ့ အစားထိုးလိုက်တဲ့ string တွေကို ပြန် ပေးပါတယ်။

syntax:

string.replace(old,new)

parameters:

old- အစားထိုးချင်တဲ့ စာသားကိုထည့်ပေးရပါတယ်။
new- အသစ်ပေါ်လာစေချင်တဲ့ စာသားကို ထည့်ပေးရပါတယ်။

return value:

return အနေနဲ့ string တွေကို ပြန်ပေးပြီး ဘယ်လို string တွေလည်းဆိုရင် user ကနေ အဟောင်းနေရာမှာ အစားထိုးလိုက်တဲ့ string အသစ်တွေပါ။

Sample Program (32)

```
str = ("helloMyanmar WinHtut")
new_string = str.replace("WinHtut", "GH")
print(new_string)
#output
#helloMyanmar GH
```

ရှေ့က Sample Program (31) ကဲ့သို့string ကို တိုက်ရိုက် print ထုတ်ရင်လည်း ရသလို ယခုကဲ့သို့ variable အသစ်ထဲကို လည်း assign လုပ်ပြီး print ထုတ်လို့ရပါတယ်။

Operators

အခြေခံအားဖြင့် ပေါင်းခြင်း၊ နုတ်ခြင်း၊ စားခြင်း၊ မြှောက်ခြင်း၊ အကြွင်းရှာခြင်း၊ နှိုင်းယှဉ်ခြင်း စတဲ့ mathematical ဆိုင်ရာ လုပ်ဆောင်ချက်တွေကို လုပ်ဆောင်ပေးတဲ့ သင်္ကေတ (symbols) တွေကို operators လို့ခေါ်ပါတယ်။ operator ရဲ့ ဘေးတစ်ဘက်စီ သို့မဟုတ် ဘေးမှာ ရှိသော variable or value တွေကိုတော့ operand လို့ ခေါ်ပါတယ်။

Python programming မှာ operator များကို ယေဘုယျ အားဖြင့် (၈) မျိုးခွဲခြား နိုင်ပါသည်။

1. Arithmetic Operators
2. Relational Operators
3. Logical Operators
4. Bitwise Operators
5. Assignment Operators
6. Special Operators
7. Identity Operators
8. Membership Operators တို့ဖြစ်ပါတယ်။

Arithmetic Operators

Arithmetic Operators တွေကို mathematical operations တွေလုပ်ဆောင်ဖို့အတွက် အသုံးပြုပါတယ်။ ဥပမာ ပေါင်းခြင်း၊ နုတ်ခြင်း၊ စားခြင်း၊ မြှောက်ခြင်း တို့ဖြစ်ပါတယ်။

Python programming မှာ arithmetic operators တွေကို အခြေခံ အားဖြင့် 7 မျိုးခွဲခြားနိုင်ပါတယ်။

1. Add {a+b}
   (ပေါင်းပေးရာတွင် အသုံးပြုပါတယ်။ operands နှစ်ခု ဖြစ်နိုင်သလို unary plus တစ်ခုထဲလည်း ဖြစ်နိုင်ပါတယ်)

2. Subtract {a-b}
   (နုတ်ပေးရာတွင် အသုံးပြုပါတယ်။ operands နှစ်ခုဖြစ်နိုင်သလို unary minus တစ်ခုထဲလည်း ဖြစ်နိုင်ပါတယ်)

3. Multiply {a*b}
   (operands နှစ်ခုကို မြှောက်ရာတွင် အသုံးပြုပါတယ်)

4. Divide {a/b}
   (ဘယ်ဘက်က operand ကို ညာဘက်က operand ကနေစားပါတယ် python programming မှာ float ဒသမ ကိန်းနဲ့ ပြန်လည် ဖော်ပြပေးပါတယ်)

5. Modulus { x % y }
   (အကြွင်းရှာတဲ့ အချိန်မှာ အသုံးပြုပါတယ် ဘယ်ဘက်က operand ကို ညာဘက် operand ကနေ remainder လုပ်ပါတယ်)

6. Floor division { x // y }
   (division လုပ်တာခြင်း တူသော်လည်းပဲ float ဒသမကိန်းဖြင့် ပြန်လည်ဖော်ပြခြင်းမျိုး မဟုတ်ပဲ ကိန်းပြည့်အနေဖြင့်သာ ပြန်လည် ဖော်ပြပေးပါတယ်။)

7. Exponent { x**y }
   (y ကို x ရဲ့ power အနေဖြင့်ထားပြီး မြှောက်တာပါ 2**3 ဆိုလျှင် 2 power 3 ဆိုတာမျိုးပါ)

Sample program ကို output နှင့်တကွ ဖော်ပြထားပါတယ်။

Sample Program (33)

```
a = 13
b = 4
print('a + b =', a+b)
#add  a + b = 17
print('a - b =', a-b)
#subtract  a - b = 9
print('a * b =', a*b)
#Multiply  a * b = 52
print('a / b =', a/b)
#Divide  a / b = 3.25
print('a "%" b =', a%b)
```

```
#Modulus  a "%" b = 1
print('a // b =', a//b)
#floor division  a // b = 3
print('a ** b =', a**b)
#Exponent  a ** b = 28561
```

Relational Operator

Relational Operator များကို အခြေခံအားဖြင့် (၆) မျိုးခွဲခြားနိုင်ပါသည်။ relational operator များကို operand များအား နှိုင်းယှဉ်လိုသည့် အခါမျိုးတွင် အသုံးပြုပါတယ်။

1. Greater than { x > y }
   (left operand ဖြစ်သည့် x သည် right operand ဖြစ်သည့် y ထက် ကြီးလျှင် မှန်သည်)
2. Less than { x < y }
   (left operand ဖြစ်သည့် x သည် right operand ဖြစ်သည့် y ထက် ငယ်လျှင် မှန်သည်)
3. Equal to { x == y }
   (left operand နှင့် right operand သည် နှစ်ခုလုံးဟာ တူညီနေလျှင်မှန်သည်)
4. Not equal to { x != y }
   (left operand နှင့် right operand သည် တစ်ခုနှင့် တစ်ခု မတူလျှင် မှန်သည်)
5. Greater than or equal to { x >= y }
   (left operand သည် right operand ထက် ကြီးလျှင် သို့မဟုတ် left and right operand ညီနေလျှင် မှန်သည်)
6. Less than or equal to { x <= y }
   (left operand သည် right operand ထက် ငယ်နေလျှင် သို့မဟုတ် left and right operand များ ညီနေလျှင် မှန်သည်)

Sample program နှင့် output ကို အောက်တွင် ဖော်ပြထားပါသည်။

Sample Program (34)

```
a = 13
b = 4
print('a > b =', a>b)
#Greater than output a > b = True
print('a < b =', a<b)
#Less than output a < b = False
```

```
print('a == b =', a==b)
#Equal than output a == b = False
print('a != b =', a!=b)
#Not Equal than output a != b = True
print('a >= b =', a>=b)
#Greater than or Equal output a >= b = True
print('a <= b =', a<=b)
#Less than or equal output a <= b = False
```

Logical Operators

Python programming မှာ logical operator သုံးမျိုးရှိပါတယ် ။ သူတို့ သုံးမျိုးစလုံးသည် အခြားသော programming တော်တော်များများတွင် symbols များကို အသုံးပြုကြပြီး python တွင်မူ စာသားများဖြင့် ဖော်ပြပါတယ်။ logical operator သုံးမျိုးမှာ and, or, not တို့ ဖြစ်ပါတယ်။ and သည် left and right operands နှစ်ခုလုံးမှန်နေလျှင် true ဖြစ်ပြီး or သည် left and right operands နှစ်ခုမှ နှစ်ခုလုံး သို့မဟုတ် တစ်ခုမှန်နေလျှင် true ဖြစ်ပါသည်။ not သည် သူ့ဘေးမှ operand မဟုတ်ဘူးဆိုလျှင် true ဖြစ်ပါသည်။

And Logical Operator

Keyword အနေဖြင့် and ကိုအသုံးပြုပြီး အခြားသော programming language တော်တော်များများတွင် & ကို အသုံးပြုကြသည်။ဘေးတစ်ဘက်စီတွင်ရှိသော conditions နှစ်ခုလုံးမှန်မှသာလျှင် output ကို ထုတ်ပေးပါသည်။ အောက်တွင် sample program ကိုဖော်ပြထားပါတယ်။

Sample Program (35)

```
a = 10
b = 9
c = 13
if a > b and c > a:
   print("Both conditions are True")
```

Or Logical Operator

Keyword အနေဖြင့် or ကို အသုံးပြုပြီး အခြားသော programming language တွေမှာဆိုရင် | ကိုသုံးပါသည်။ ဘေး တစ်ဘက်စီတွင် ရှိသော conditions နှစ်ခုထဲမှ တစ်ခု မဟုတ် တစ်ခု မှန်လျှင် အလုပ် လုပ်ပါသည်။

Sample Program (36)

```
a = 10

b = 9

c = 13
if a > b or c > a:

print("At least one of the conditions is True")
```

Not Logical Operator

Keyword အနေဖြင့် not ကို အသုံးပြုပြီး အခြားသော programming language များတွင်မူ symbol ဖြစ်သည့် ! ကို အသုံးပြုပါသည်။ ဆိုလိုသည်မှာ not နောက်မှ condition သည် true condition ဖြစ်နေလျှင် not true မမှန်ဘူးဟု ဆိုလိုခြင်းဖြစ်ပြီး output အနေဖြင့် မမှန်ဘူးဆိုသည့်အတိုင်း false ထွက်လာမှာဖြစ်ပါတယ်။ condition သည် မှားနေလျှင် not false မမှားဘူးဟု ဆိုလိုခြင်းဖြစ်ပြီး output အနေဖြင့် true ထွက်လာမှာ ဖြစ်ပါတယ်။

Sample Program (37)

```
x = True
print('not x is ', not x)
#not logical operator output x is false
#Reverse program
x=False
print('not x is ', not x)

#not logical operator output not x is True
```

Bitwise Operators

Operands တွေနဲ့ bit တစ်ခုခြင်းစီ bit by bit လုပ်ဆောင်တဲ့ operator တွေကို bitwise operators တွေလို့ ခေါ်ဆိုပါတယ်။ Python programming language မှာဆိုရင် bitwise operators (6) မျိုးရှိပါတယ်။

Bitwise AND (&) သည် 0 and 0 ဆိုရင် output အနေဖြင့် 0 ပြန်ပေးပြီး 0 and 1 ဆိုရင်တော့ 0 ပြန်ပေးသလို 1 and 1 ဆိုရင်လည်း output အနေဖြင့် 1 ပြန်ပေးပါတယ်။ အောက်တွင် ဖော်ပြ ထားသောပုံကို ကြည့်ခြင်းအားဖြင့် အလွယ်တကူသဘောပေါက်နိုင်ပါတယ်။

AND

x	*y*	*xy*
0	*0*	*0*
0	*1*	*0*
1	*0*	*0*
1	*1*	*1*

Sample Program (38) for bitwise and (&)

```
a = 60     #60 = 0011 1100
b = 13     #13 = 0000 1101
c = 0
c = a & b  #12 = 0000 1100

print("value of c is " , c)
```

အထက်ပါအတိုင်း program ရေးပြီး run ကြည့်လျှင် output အနေဖြင့် 12 ထွက်လာသည်ကို မြင်ရပါမည်။ ဘာကြောင့် 12 ထွက်လာသနည်း။ a value ဖြစ်သည့် 60 သည် decimal value ဖြစ်ပြီး သူ့ရဲ့ binary value မှာ 8 bits ဖြင့်ကြည့်လျှင် 0011 1100 ဖြစ်သည်။ ထိုနည်းတူ b value ဖြစ်သည့် 13 ကို binary value 8 bits ဖြင့်ကြည့်လျှင် 0000 1101 ဖြစ်သည်။

a	b	output
0	0	0
0	0	0
1	0	0
1	0	0
1	1	1
1	1	1
0	0	0
0	1	0

ဖော်ပြပါ table အားကြည့်လျှင် output အနေဖြင့် 0000 1100 ထွက်လာသည်ကို မြင်ရပါမည်။ ထို binary value များအား decimal value အဖြစ် ပြန်ပြောင်းကြည့်လျှင် 12 ရပါမည်။ ထို့ကြောင့် ဖော်ပြပါ Python program အား run သော အချိန်တွင် output အဖြစ် 12 ကို ပြန်ရရှိခြင်း ဖြစ်ပါသည်။

Bitwise or (|)

bitwise or သည် 0 and 1 ဆိုလျှင် output အနေဖြင့် 1 ပြန်ပေးပြီး 1 and 1 ဆိုလျှင်လည်း 1 ပြန်ပေးသည် ထိုနည်းတူ 0 and 0 ဆိုလျှင် 0 ပြန်ပေးပါသည်။

OR

x	*y*	*x+y*
0	*0*	*0*
0	*1*	*1*
1	*0*	*1*
1	*1*	*1*

Sample Program (39)

```
a = 60     #60 = 0011 1100

b = 13     #13 = 0000 1101
c = 0

c = a | b  #61 = 0011 1101

print("value of c is " , c)
```

အထက်ပါ Sample Program (39) အား run ကြည့်လျှင် output အနေဖြင့် 61 ကို ပြန်ပေးပါမည်။

a	b	output
0	0	0
0	0	0
1	0	1
1	0	1
1	1	1
1	1	1
0	0	0
0	1	1

binary value အနေဖြင့် output တွင် 0011 1101 ကို ပြန်ရပါမည်။ ထို value အား decimal အနေဖြင့် 61 ကို ပြန်ရပါသည်။ ထို့ကြောင့် program အား run သော အချိန်တွင် output အနေဖြင့် 61 ရခြင်း ဖြစ်သည်။

Bitwise 1's complement (~) or Bitwise Not

Bitwise 1's complement or Bitwise Not ကို အရှင်းဆုံး ပုံစံနှင့် ရေးရမည်ဆိုလျှင် အောက်ပါအတိုင်း ဖြစ်သည်။

a=10

b= - (a+1)

b= - (00001010 +1)

b= -11 (in decimal ဖြစ်သည်)

ပိုသိလိုပါက အောက်ပါ တို့ကို ဖတ်နိုင်ပါသည်။

Bitwise 1's complement operator သည် unary operator အမျိုးအစား ဖြစ်သည်။ (~x) unary operator ဆိုသည်မှာ operand တစ်ခုတည်းပါရှိသော operator ကို ဆိုလိုခြင်းဖြစ်သည်။ Bitwise NOT ကို binary 1's complement လုပ်တယ်လို့လည်း ခေါ်ပါတယ်။ 1'complement လုပ်ခြင်းဆိုသည်မှာ binary value များကို ပြောင်းပြန်လှန် invert လုပ်ခြင်းနှင့် တူညီသည်။

x =10 ဟု ထားမည်ဆိုလျှင် 8 bit အနေဖြင့် 00001010 ရရှိမည်။ ထို00001010 အား (ones complement) ပြုလုပ်လျှင်11110101 ပြန်လည်ရရှိပါမည်။

အထက်တွင် ရရှိလာသော 11110101 ကို 2's complement ပြန်လုပ်မှသာ python programming ရဲ့ ~ Bitwise Not operator or complement operator အဖြေကို ရရှိမှာ ဖြစ်ပါတယ်။

1111 0101 (မူရင်း number)

0000 1010 (1's complement လုပ်ပြီး)

+1 (2's complement လုပ်ခြင်း)

0000 1011 (decimal value အနေဖြင့် 11 ကို ရရှိပါသည်။ 2's complement လုပ်ခြင်း ဖြစ်သည့်အတွက် ရှေ့မှ negative ထည့်ပေးလျှင် -11 ကို ရရှိပါသည်။)

2's complement လုပ်လိုလျှင် 1's complement လုပ်ခြင်းမှ ရရှိခဲ့သော binary value ကို 1 ပေါင်းပေးခြင်းဖြင့် ရရှိလာသော value သည် 2's complement ဖြစ်သည်။

အထက်ပါ Complement method ကို binary value များအား positive and negative ပြောင်းလိုသော အခါများတွင် အသုံးပြုပါသည်။

Sample Program (40)

```
a = 10
c = ~a;
print('Value of c is ' , c)

#Value of c is -11
```

Bitwise XOR (^)

Bitwise XOR (^) သည် တူသည့် 1 and 1 ဆိုလျှင် 0 output ပေးပြီး မတူသည့် 0 and 1 ဆိုလျှင် 1 output ပေးပါသည် သို့သော် bitwise xor သည် 0 and 0 ဆိုလျှင်တော့ 0 သာ output ပေးပါသည်။ ဆိုလိုသည်မှာ input value တွေမှာ data မရှိသောအခါတွင် 0 ကို output ပြန်ပေးခြင်းဖြစ်သည်။ ဥပမာ အနေဖြင့် a=5 (0000 0101) နှင့် b=3 (0000 0011) တို့အား XOR လုပ်ကြည့်လျှင် output အနေဖြင့် 0000 0110 ပြန်ရပါမည်။ decimal value အနေဖြင့် 6 ကို ပြန်ရပါမည်။

Input 1	5	00000101
Input 2	3	00000011
(Bitwise XOr)		
Output	6	00000110

Sample Program (41)

```
a = 5
b = 3
c = a ^ b;
print("Value of c is " , c)
#Value of c is 6
```

Bitwise right shift (>>)

Bitwise right shift operator သည် binary value များကို bit အလိုက် ကိုင်တွယ်ရာတွင် အလွန်အသုံးဝင်ပါသည်။ variable တစ်ခုကို x=10 (0000 1010) အဖြစ်ထားပါမည်။ y=x>>2ဟုရေးမည် ဆိုလျှင် y value သည် 2 ရ ရှိပါမည်။

>> 2 right shift 2 ဆိုသည်မှာ ညာဘက်မှ bit 2 လုံး ဖြုတ်လိုက်ခြင်းဖြစ်သည်။ ဆိုလိုသည်မှာ ဘယ်ဘက်မှ 00 နှစ်လုံးဖြင့် တွန်းထုတ်လိုက်ခြင်းဖြစ်သည်။

x = 10 (0000 1010) ညာဘက်မှ bit 2 လုံး ဖြုတ်မှာ ဖြစ်သည့်အတွက် 0000 10 သာ ကျန်မည် 8 bit အနေဖြင့် ကြည့်လျှင် 0000 0010 ဟု သိနိုင်သည်။ ထို့ကြောင့် decimal value ဖြင့် ကြည့်လျှင် 2 ကို ရရှိခြင်းဖြစ်သည်။ အထက်ပါ ဖော်ပြထားချက်များသည် right shift 2 လုပ်ခြင်းဖြစ်သည်။ right shift တစ်လုံး ထုတ်ပုံကို အောက်တွင် ဖော်ပြထားသည်။

Sample Program (42)

```
a = 60
c = a >> 2
```

```
print("Value of c is ", c)

#Value of c is 15
```

အထက်ပါ Sample Program (42) **အား** run **ကြည့်လျှင်** output value **အဖြစ်** decimal 15 **ကို ရရှိပါမည်။ အဘယ်ကြောင့် ဆိုသော်** 60 binary **အနေဖြင့်** 0011 1100 **ဖြစ်သည်။ ထိုထဲမှ** 2 bit **ကို** right shift **လုပ်မည်ဆိုလျှင်** 0011 1100 **နောက်မှ** zero **နှစ်လုံးကို ဖြုတ်ပစ်ရမည် ဖြစ်သည်။ ထို့ကြောင့်** binary value **အနေဖြင့်** 0011 11 decimal **အနေဖြင့်** 15 **ရ ရှိခြင်းဖြစ်သည်။**

Sample Program (43) for 3 bit right shift

```
a = 60

c = a >> 3
print("Value of c is ", c)

#Value of c is 7
```

အထက်ပါ Sample Program (43) **ကို** run **ကြည့်လျှင်လည်း** decimal **အနေဖြင့်** 7 **ကို ရရှိပါမည်။ အဘယ်ကြောင့်ဆိုသော်** 60 **ရဲ့** binary value **သည်** 0011 1100 **ဖြစ်ပြီး ထို** value **အား** 3 bit right shift **လုပ်လျှင်** 0011 1100 **ညာဘက်မှ** bit **သုံးခုကို ဖြုတ်ပစ်ရမည် ဖြစ်သည်။** binary value **အနေဖြင့်** 0011 1 **ကျန်ရှိမည်ဖြစ်ပြီး** decimal **အနေဖြင့်** 7 **ရရှိခြင်းဖြစ်ပါသည်။**

Bitwise Left (<<)

Bitwise left **သည် ဘယ်ဘက်မှ** bit **များအား ဖြုတ်ထုတ်လိုက်ခြင်း ဖြစ်ပြီး တစ်နည်း အားဖြင့် ညာဘက်မှ** bit **များ တွန်းထည့်လိုက်ခြင်း ဖြစ်သည်။** variable **တစ်ခုကို** x=10 (0000 1010) **အဖြစ်ထားပါမည်။** y=x << 2 **ဟုရေးမည် ဆိုလျှင်** y value **သည်** decimal **အားဖြင့်** 40 **ရရှိမည် ဖြစ်ပြီး** binary **အားဖြင့်** 0010 1000 **ဖြစ်သည်။ အဘယ်ကြောင့်ဆိုသော်** 10 **သည်** binary **အားဖြင့်** 0000 1010 **ဖြစ်ပြီး သူ့အား** left shit **ဘယ်ဘက်မှ** << 2 bit 2 **လုံး ဖြုတ်လိုက်မည်ဆိုလျှင်** 0000 1010 output **အားဖြင့်** 0010 10**သာ ကျန်ရှိမည်** 8 bit **အားဖြင့်** 0010 1000 **ကျန်ရှိမည်။** decimal **အနေဖြင့်** 40 **ကျန် ရှိနေခြင်း ဖြစ်သည်။** left shift **သည်ညာဘက်မှ** 2 bit zero 2 **လုံး တွန်းထည့်လိုက်ခြင်းကြောင့် ဘယ်ဘက်မှ** zero **နှစ်လုံး အပြင်ထွက်သွား ခြင်းဖြစ်သည်။**

Bitwise left **တွင် ညာဘက်မှ** zero **တစ်လုံး** 1 bit **ထည့်လိုက်ပြီး ဘယ်ဘက်မှ** zero **တစ်လုံး** 1 bit **ထွက်သွားသောပုံကို အောက်တွင် ပြထားသည်။**

Sample Program (44)

```
a = 60
c = a << 2;
print("Value of c is " , c)

#value of c is 240
```

အထက်ပါ Sample Program (44) ကို run ကြည့်လျှင် output အနေဖြင့် 240 ရရှိမည်။ အဘယ်ကြောင့်ဆိုသော် 60 သည် binary အားဖြင့် 0011 1100 ဖြစ်သည်။ bit နှစ်လုံး ညာဘက်မှ တွန်းထည့်လိုက်မည်ဆို လျှင် ဘယ်ဘက်မှ bit နှစ်လုံး ထွက်သွားမည် 0011 1100 8bit အနေဖြင့် ကြည့်လျှင် 1111 0000 ဖြစ်သည်။ decimal အားဖြင့်240 ဖြစ်သည်။

Sample Program (45) 4 shift to the left

ဘယ်ဘက်သို့ <<4 ရေးပါမည်။ a= 60 (0011 1100) ဖြစ်သည်။

```
a = 60
b = a<<4
print(b)

#960
```

အထက်ပါ Sample Program (45) ကို run ကြည့်လျှင် output အနေဖြင့် 960 ကို ရရှိပါမည်။ အဘယ်ကြောင့်ဆိုသော် python programming တွင် int သည် 4 bytes နေရာ ယူသည်။ ယခင် သင်ခန်းစာများတွင် 8bit နဲ့သာ တွက်ခဲ့ကြသည်။ သို့သော် python သည် 4 bytes=32 bits ဖြစ်တဲ့ အတွက် right shift လုပ်သော အချိန်တွင် မသိသာသော်လည်း left shift လုပ်သော အချိန်တွင်မူ output သည် အရမ်းကို ကွာခြားပါသည်။

32bits=0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0011 1100 0000

ယခု ဖော်ပြထားသော binary value များသည် 60 ကို left shift 4 လုံး လုပ်ထားခြင်း ဖြစ်သည်။ 32 bits အနေနှင့် တွက်ချက်မှသာလျှင် အဖြေမှန်ကို ရရှိမည် ဖြစ်သည်။

Assignment Operators

Assignment operators ဆိုတာကတော့ equation တစ်ကြောင်း သို့မဟုတ် code line တစ်ခုရဲ့ ညာဘက်မှာ ရှိတဲ့ value တွေကို calculate or instruction အတိုင်း လုပ်ဆောင်ပြီး ဘယ်ဘက်မှာရှိသော variable တစ်ခုထဲသို့ assign ထည့်လိုက်ခြင်းပင် ဖြစ်ပါသည်။ a = 5 ဆိုလျှင် 5 ဆိုတဲ့ ညာဘက်က value တစ်ခုကို a ဆိုတဲ့ ဘယ်ဘက်က variable ထဲသို့ ထည့်လိုက်ခြင်းပင် ဖြစ်ပါသည်။

Assignment operator ကိုအခြားသော operator တွေနဲ့ပေါင်းပြီး သုံးလို့လည်း ရပါသေးတယ်။ ၎င်းတို့ကို compound operator လို့ခေါ်ပါတယ်။ a += 5 သည် a = a+5 ကိုပင် ဆိုလိုခြင်းဖြစ်သည်။ ထိုနည်းတူ a -= 5 သည် လည်း a=a-5 ပင်ဖြစ်သည်။ Python programming တွင် compound operator များစွာကို support လုပ်ပေးထားပြီး ၎င်းတို့အား အောက်ပါ ပုံတွင် ဖော်ပြထားပါသည်။

Operators	Example	Equivalent
=	x = 5	x = 5
+=	x += 5	x = x + 5
-=	x -= 5	x = x - 5
*=	x *= 5	x = x * 5
/=	x /= 5	x = x / 5
%=	x %= 5	x = x % 5
//=	x //= 5	x = x // 5
**=	x **= 5	x = x ** 5
&=	x &= 5	x = x & 5
|=	x |= 5	x = x | 5
^=	x ^= 5	x = x ^ 5
>>=	x >>= 5	x = x >> 5
<<=	x <<= 5	x = x << 5

Special Operators

Python programming မှာ တစ်ခြား programming language တွေမှာလုံးဝ နီးပါး မတွေ့ရတဲ့ english letter ရေးသလို operators တွေ ပါ နေပါသေးတယ်။ identity operator and membership operator လို့ ခေါ်ပါတယ်။

Identity Operators

Python programming မှာ is နှင့် is not ဆိုတာတွေက identity operators တွေ ဖြစ်ပါတယ်။ identity operators တွေကို နေရာတိုင်းမှာတော့ မသုံးပါဘူး။ memory allocation လုပ်ခြင်း နေရာတူတဲ့ value or variable and object တွေကို စစ်တဲ့ နေရာတွေမှာသာ သုံးပါတယ်။ memory allocation တူတယ်ဆိုတာက memory ထဲမှာ program memory ဆိုတာ သပ်သပ် ထပ်ခွဲထားပါတယ်။ ထို program memory ထဲမှာမှ variable တွေကို သိမ်းတဲ့ နေရာ ဆိုပြီး သပ်သပ် ထပ်ခွဲ ထားပါသေးတယ်။ a=20 , b=23 ဟု variable နှစ်ခုကို တည်ဆောက်လိုက်ပါသည်။ ထိုကဲ့သို့ variable များ တည်ဆောက်ပြီးသည်နှင့် တစ်ပြိုင်နက် ထို variable မှာ ကိုယ်ပိုင် address များ ပိုင်ဆိုင်သွားပါပြီ။ ဥပမာ a=20 (0x100) , b=23 (0x200) ဟု မှတ်ယူကြည့်ပါ။ identity operators သည် ထို variable value များကို စစ်ခြင်း မဟုတ်ပဲ variable ရဲ့ address များကိုသာ စစ်ခြင်း ဖြစ်ပါသည်။ Python programming တွင် variable များသည် name မတူသော်လည်း value တူပါက address များ အတူတူပင် ဖြစ်သည်။ ဆိုလိုသည်မှာ a =20 , b=20 ထို variable နှစ်လုံးသည် name မတူသော်လည်း သူတို့ value များ တူသည့်အတွက် address အတူတူ ပင်ဖြစ်သည်။ အဘယ်ကြောင့်ဆိုသော် python programming သည် အခြားသော programming များကဲ့သို့ memory management မလုပ်ပါ။ အခြားသော programming language တစ်ခုဖြစ်သည့် C programming တွင် variable များသည် value တူသော်လည်း address မတူပါ။ python programming တွင် a,b,c သုံးခုလုံးရဲ့ value သည် 10 ဖြစ်နေလျှင် ထို a,b,c တို့သည် memory address မှာ အတူတူပင်ဖြစ်သည်။ python တွင် id() ဆိုသည့် method ကိုသုံးပြီး memory address များကိုကြည့်နိုင်သည်။

Sample Program (46)

```
a = 10

b = 10
c = 10

print(id(a),id(b),id(c))
```

အထက်ပါ Sample Program (46) အတိုင်း ရေးပြီး စမ်းကြည့်လျှင် output အနေဖြင့် memory address များ အတူတူ ထွက်လာသည်ကို မြင်ရပါမည်။ Identity operator သည် မှန်လျှင် true ပြန်ပေးပြီး မှားလျှင်တော့ false ပြန်ပေးပါသည်။ a is b ဆိုလျှင် a နှင့် b သည် memory address တူသည့် အတွက် true ပြန်ပေးပါသည် ။ sample program ကို အောက်တွင် ဖော်ပြထားသည် ထို program အား run ကြည့်လျှင် output အနေဖြင့် true ကို ရပါမည်။

Sample Program (47)

```
a = 10
b = 10
c = a is b
print(c)

#True
```

is not identity operator

is not identity operator သည် operand နှစ်ခု မတူလျှင် true ပေးပြီး တူလျှင် false ပြန် ပေးပါသည်။ a is not b ဆို လျှင် a နှင့် b သည် တူနေသည့်အတွက် false ပြန်ပေးပါမည်။ sample program ကို အောက်တွင် ဖော်ပြထားသည်။ ထို program ကို run ကြည့်လျှင် false ရပါမည်။

Sample Program (48)

```
a = 10

b = 10
c = a is not b
print(c)

#False
```

Sample Program (49)

```
a = 10

b = 20
c = a is not b
print(c)

#True
```

Sample Program (49) ကို run ကြည့်လျှင် true ရပါမည်။ အဘယ်ကြောင့်ဆိုသော် a နှင့် b သည် မတူတော့ သောကြောင့် memory address များလည် မတူတော့သည့်အတွက် a နှင့် b သည် မတူဘူးဟု ဆိုလျှင် true ဆိုသည့် output ကို ပြန်ရခြင်းဖြစ်သည်။ ၎င်းတို့ address များအား အောက်ပါ အတိုင်း id() method ကိုသုံးပြီး စစ်ဆေးနိုင်ပါသည်။

Sample Program (50)

```
a = 10

b = 20
c = a is not b
print(c)
print('a =', id(a), 'b = ', id(b))
# True

# a = 140710321199040 b =  140710321199360
```

အထက်ပါ Sample Program (50) ကို run ကြည့်လျှင် output အနေဖြင့် memory address များ မတူ သည်ကို မြင်တွေ့နိုင်ပါသည်။

Dive to String

String များကိုလည်း အောက်ပါအတိုင်း နှိုင်းယှဉ်ကြည့်နိုင်သည်။ ယခု program တွင် relational operator (==) ကိုပါ သုံးပြထားပါသည်။ သတိပြုရန်မှာ relational operator သည် value များကိုသာ နှိုင်းယှဉ်စစ်ဆေးခြင်းဖြစ်ပြီး identity operator ကတော့ memory address များကို နှိုင်းယှဉ်စစ်ဆေးခြင်းဖြစ်သည်။ output သည် True ရပြီး memory address များလည်း တူညီသည်ကိုအောက်ပါ program တွင်တွေ့ရပါမည်။

Sample Program (51)

```
string1='hello'
string2='hello'
print(id(string1),id(string2))
string3=string1 is string2
print(string3)
if string1 == string2:
   print('They are same')

#output
#1671375414320 1671375414320
#True

#They are same
```

Membership Operators

Python programming တွင် membership operators နှစ်မျိုးရှိသည်။ in နှင့် not in ဖြစ်သည်။ membership operators များကို string , list , tuple , set and dictionary တို့ထဲရှိ element များကို စစ်ဆေးရာတွင် အသုံးပြုပါသည်။ sample program အနေဖြင့် a ဆိုသည့် string တစ်ခုနှင့် l ဆိုသည့် list တစ်ခု တည်ဆောက်ပါမည်။ a= 'winhtut' , l = [1,2,3,4,5] a ဆိုသည့် string ထဲတွင် h ဆိုသည့် စာလုံးပါလားစစ်ချင်သော အခါတွင် membership operator ဖြစ်သည့် in ကို သုံးပြီး 'h' in a ဆို ပြီး စစ်ဆေးနိုင်သည်။ a ထဲတွင် h ပါဝင်နေပါက output အနေဖြင့် True ကို ထုတ်ပေးမည် ဖြစ်ပြီး မပါဝင်ပါက False ကို ထုတ်ပေးမည်ဖြစ်သည်။ not in သည်လည်း မိမိရှာ လိုသော sequence ထဲမှာ မရှိလျှင် True ကို ပြန်ထုတ်ပေးပြီး ရှိနေလျှင် false ကို ပြန်ထုတ်ပေး ပါသည်။လေ့ကျင့်ရန် example program အား အောက်တွင် ဖော်ပြထားပြီး output နှင့်ပါ ယှဉ်တွဲ ပြထားပါသည်။

Sample Program (52)

```
string1=[1,2,3,4,5,'w']
string2=[1,2,3,4,5]
print( 1 in string1)
if 'w' in string1 :
   print('w is in string1')
print( 2 not in string2)
if 'w' not in string2 :
   print('w is not in string2')
 #output
#True
#w is in string1
#False

#w is not in string2
```

Python Control Structure

Condition and If Statement

- Equals: a == b a နှင့် b သည် အတူတူပင်ဖြစ်သည်။
- Not Equals: a != b a နှင့် b သည် မတူပါ။
- Less than: a < b a သည် b ထက် ငယ်သည်။
- Less than or equal to: a <= b a သည် b ထက် ငယ်သည် သို့မဟုတ် ညီသည်။
- Greater than: a > b a သည် b ထက် ကြီးသည်။
- Greater than or equal to: a >= b a သည် b ထက် ကြီးသည် သို့မဟုတ် ညီသည်။

Sample Program (53)

```
a = 33

b = 200
if b > a:
```

```
  print("b is greater than a")
```

b သည် a ထက်ကြီးခဲ့ရင် b is greater than a ဆိုသည့် စာသားကို ဖော်ပြပေးပါ ဟု ရေးသား ထားခြင်းဖြစ်ပါသည်။

Indentation Error - Sample Program (54)

```
a = 33
b = 200
if b > a:

print("b is greater than a")
```

အထက်ပါ Sample Program (54) ကို run ကြည့်ရင် indent error ကို ရရှိမှာ ဖြစ်ပါတယ်။ အဘယ်ကြောင့်ဆိုသော် python programming language သည် scope များကို သတ်မှတ်ရန် whitespace များကို အသုံးပြုပါသည် ။ ဥပမာ if statement အောက်မှ အလုပ်လုပ်ရန် ရေးသားသော code များသည် if အောက်တည့်တည့်တွင် ရှိနေလျှင် error တက်မည်ဖြစ်ပါသည်။ ထို if statement အောက်မှ အနည်းဆုံး space တစ်ချက် သို့မဟုတ် tab တစ်ချက် ခြားပေးထားရပါမည်။ အခြားသော programming language တော်တော်များများဖြစ်သည့် c/c++,java စသည် တို့သည် scope များကို curly-brackets များ ဖြင့် သတ်မှတ်ကြပါသည်။

elif

အခြားသော programming language များတွင်တော့ else if လို့သုံးကြပါတယ်။ programming language တော်တော်များများနှင့်ယခု python programming မှာလည်းအသုံးပြုပုံ မှာအတူတူပင်ဖြစ်သည်။elifသည်သူ့ရှေ့မှာရှိသော condition တစ်ခုကိုအရင်စစ်ဆေးပြီးမှထို condition ပြီးမှသာ elif ကိုလာစစ်ပါသည်။ထို့ကြောင့် elif ကို second optionအနေဖြင့် အသုံးပြုပါသည်။

Sample Program (55)

```
a = 33
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
#output

#a and b are equal
```

ယခုကဲ့သို့ condition နှစ်ခုကို တစ်ခုပြီး တစ်ခု စစ်ဆေးလိုသောအခါမျိုးမှာ အသုံးပြုသလို နှစ်ခုထက်များသော condition များကို စစ်ဆေးလိုသော အခါမျိုးများတွင်လည်း အသုံးပြုပါသည်။

အောက်ပါ အတိုင်းလည်း elif ကို နှစ်ခုသုံးပြီး ရေးသားနိုင်ပါသည်။

Sample Program (56)

```
a = 35
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
elif a!=b:
  print('a is less than b')
#output

#a is less than b
```

else statement

else statement သည် သူ့အထက်တွင်ရေးသားထားသော စစ်ဆေးချက်များ တစ်ခုမှ အလုပ် မလုပ်တော့သော အခါတွင် နောက်ဆုံး option တစ်ခု အနေဖြင့် အလုပ် လုပ်ပါသည်။

Sample Program (57)

```
a = 35

a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
#output

#a is greater than b
```

အထက်ပါ Sample Program (57) တွင် if and elif တို့သည် တစ်ခုမှ အလုပ် မလုပ်လျှင် else ဆိုသည့် နောက်ဆုံး statement သည် အလုပ်လုပ် သွားပါမည်။

Loops

Python မှာ loop နှစ်မျိုးရှိပါတယ် for loop and while loop ဖြစ်ပါတယ်။

While loop

While loop သည် သူ့နောက်မှာ ရှိသော condition မှန်နေသမျှ statement များကို အလုပ် လုပ်ပါတယ်။

Sample Program (58)

```
a = 35

i = 1
```

```
while i < 6:
  print(i)
  i += 1
#1 2 3 4 5
```

while loop with the break statement

while loop သည် condition မမှန်တော့သည့် အချိန်မှာလည်း အဆုံးသတ်နိုင်သလို break statement ဖြင့်လည်း အဆုံးသတ်နိုင်ပါသည်။ break statement သည် အတွင်းဆုံး loop တွင် ရှိနေလျှင် ထိုအတွင်းဆုံး loop တစ်ခု တည်းကိုသာ terminate လုပ်ပါသည်။ ထို့နောက် အပြင်ဘက်မှ loop များကို ဆက်လက် အလုပ် လုပ်ပါသည်။

Sample Program (59)

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
  #1 2 3
```

အထက်ပါ program တွင် i တန်ဖိုး 3 သို့ရောက်သောအခါတွင် program အဆုံးသတ်သွားမည် ဖြစ်ပါသည်။ 4 ကြောင်းမြောက်တွင် i တန်ဖိုးဟာ 3 နှင့်တူခဲ့လျှင် break ဆိုပြီး program (while loop)ကို ရပ်မည်ဟု ရေးထားသောကြောင့်ဖြစ်သည်။

For loop

Python ရဲ့ for loop ဟာ အခြား programming language တွေရဲ့ for loop နဲ့ တော်တော်ကွဲပြားပါတယ်။ python မှာ ရေးရတာ အရမ်းရိုးရှင်းပြီး လွယ်ကူပါတယ်။ keyword အနေဖြင့် for ကိုသာ အသုံးပြုပါတယ်။ list, tuple, set တို့ထဲကလည်း data များကို အလွယ်တကူ ထုတ်ယူနိုင်ပါတယ်။

Sample Program (60)

```
fruits = ["apple", "banana", "cherry"]

for x in fruits:
  print(x)
# output
# apple
# banana

# cherry
```

အထက်ပါ Sample Program (60) တွင် fruits ဆိုသည့် list တစ်ခုကို တည်ဆောက်ထားပြီး ထို list ထဲတွင် apple banana cherry စတဲ့ data များကို ထည့်ထားပါတယ်။ ထို list ထဲမှ data များကို ထုတ်ရန် program မှာ for x in fruits: သာ ဖြစ်သည်။

For loops through a String

Sample Program (61)

```
for x in "banana":

    print(x)
# """
# OUTPUT
# b
# a
# n
# a
# n
# a

#  """
```

banana ဆိုတဲ့ string ထဲမှာ ရှိတဲ့ စာလုံးများကို တစ်လုံးခြင်းစီ ထုတ်ခြင်းဖြစ်ပါတယ်။

For loops with break statement

ယခု program တွင် list တစ်ခု တည်ဆောက်မည်ဖြစ်ပြီး ထို list ထဲတွင် မိမိထည့်ချင်သော data များ ထည့်ထားပါမယ်။ ထို list ထဲမှ data များကို print ထုတ်ရန် for loops ကို သုံးပါမည်။ print ထုတ်လို့ ရလာသော data များထဲမှ မိမိ check လုပ်လိုသော စကားလုံးနှင့် တူနေလျှင် program ကို break လုပ်ရန် အတွက် ရေးသားပါမည်။

Sample Program (62)

```
fruits = ["aung", "maung", "winhtut","greenhackers"]

for x in fruits:
  print(x)
  if x == "winhtut":
    break

# OUTPUT
# aung
# maung

# winhtut
```

For loops with continue statement

Sample Program (63)

```
fruits = ["aung", "maung", "winhtut","greenhackers"]
for x in fruits:
  if x == "winhtut":
    continue
  print(x)
# output
# aung
# maung

# greenhackers
```

Continue statement သည် program ကို ရပ်လိုက်ခြင်းမျိုး မဟုတ်ပဲ သူ့အထက်မှ condition မှန်နေလျှင် သူ့အောက်မှ instruction ကို ဆက်မလုပ်တော့ပဲ အစသို့ ပြန်သွားခြင်းဖြစ်သည် ။ ထို့ကြောင့် program ကို run ကြည့်သောအခါ output တွင် winhtut ဆိုသည့် စာသားကို မတွေ့ရခြင်း ဖြစ်သည်။

Pass

Pass statement ကို NOP (no operation) လို့လည်း ခေါ်သလို null statement လို့လည်း ခေါ်ပါတယ်။ null statement လို့ ပြောလိုက်လျှင် comment နှင့် တူသည်ဟု ထင်နိုင်သော်လည်း commnet ကို python interpreter က ကျော်သွားပါသည်။ သို့သော် pass ကိုတော့ ကျော်မသွားပါဘူး။ null statement ဟု ဆိုသော်လည်းပဲ pass ကို program ရေးသားရာမှာ အသုံးပြုပါတယ်။ loop or function တစ်ခုကို ကြေငြာထားပြီး မည်သည့် အလုပ်မှ မလုပ်သေးသောအခါတွင် သူတို့ အထဲ၌ pass ကို ထည့်ထားပါသည်။ အဘယ်ကြောင့်ဆိုသော် loop or function တွေကို အလွတ်ကြီးထားလို့ မရသောကြောင့်ဖြစ်သည်။

```
for char in 'Python':
```

ယခု အတိုင်းပဲ code ရေးပြီး run ကြည့်လျှင် error တက်ပါမည်။ IndentationError: expected an indented block ဟုပြပါမည်။ သို့သော် အောက်ပါအတိုင်း ရေးကြည့်လျှင်တော့ error တက်မည်မဟုတ်ပါ။ မည်သည့် output မျှ ထွက်မလာသော်လည်း program run သွားမည် ဖြစ်သည်။

```
for char in 'Python':
    pass
```

အထက်ပါ အတိုင်း pass ကို သုံးလျှင် error တက်မည် မဟုတ်ပါ။ အောက်ပါအတိုင်း program ကို ထပ်ပြီး implement လုပ်နိုင်သေးသည်။

Sample Program (64)

```
for char in 'Python':
    if char == 'h':
        pass
print("Current character:", char)
```

```
#output
# Current character: n
```

Sample Program (64) ကို runကြည့်လျှင် output ၌ n ကို ရရှိပါမည်။ အဘယ်ကြောင့်ဆိုသော် char ဆိုသည့် variable ထဲတွင် Python မှ နောက်ဆုံးစာလုံး n ရှိနေသေးသောကြောင့်ဖြစ်သည်။

List in Python

Python မှာဆိုရင် list က နေရာတော်တော်များများမှာ သုံးတဲ့ compound data type ဖြစ်ပါသည်။ Python ရဲ့list ဟာ တစ်ခြား programming မှာဆိုရင် array နဲ့ ဆင်တူသော်လည်းပဲ python ရဲ့ list ကတော့ powerful tool တစ်ခုဖြစ်ပါတယ်။ List တစ်ခုတည်းမှာကိုပဲ data type တွေဖြစ်တဲ့ Integers, String, Double, etc စတာတွေ အားလုံးပါဝင်နိုင်ပါတယ်။ Python မှာဆိုရင် list ထဲက data type တွေကို control လုပ်ဖို့ လွယ်ကူပါတယ်။ ထို့ကြောင့် Python မှာ list ကို တစ်ချိန်တည်းမှာ မတူညီတဲ့ data တွေ အများကြီး သိုလှောင်နိုင်သလို ထပ်ပေါင်းတာတွေ ဖယ်ထုတ်တာတွေ access လုပ်တာတွေ စတဲ့ Handling လုပ်ဖို့ လွယ်ကူပါတယ်။

Declaring a List

Syntax:

List= [ ]; ယခုနည်းအတိုင်းပဲ square bracket ကို အသုံးပြုပြီးတော့ list ကိုကြော်ငြာလို့ရပါတယ်။

Sample Program (65)

```
list = []

print(list)
```

အထက်ပါ program ကို run လိုက်တဲ့ အချိန်မှာ result အနေနဲ့ square bracket [ ] ကိုရ ရှိမှာဖြစ်ပါတယ်။

Creating a list with Data

List ထဲသို့ data တစ်ခု ထည့်လိုတဲ့ အခါ အောက်ပါ အတိုင်း ထည့်သွင်းပါတယ်။

Sample Program (66)

```
list = ["Hello world"]

print(list)
#output

#['Hello world']
```

result အနေနဲ့ [‘Hello World’] ဆိုတာကို ရရှိမှာ ဖြစ်ပါတယ်။

Creating a list with Multiple Data

list တစ်ခုထဲသို့ data အများကြီး ထည့်လိုတဲ့အခါ (,) ခြားပြီး ထည့်ရုံပါပဲ။

Sample Program (67)

```
list = ["Hello world","Green", "Hackers"]

print(list)
#output

#['Hello world', 'Green', 'Hackers']
```

Accessing data

Sample Program (68)

```
list = ["Hello world","Green", "Hackers"]

print(list[0])
print(list[1])
print(list[2])
#Output
#Hello world
#Green

#Hackers
```

List တစ်ခုထဲမှာ ရှိတဲ့ data တွေကို access(ရယူ) လိုတဲ့အခါမျိုးမှာဆိုရင် ၎င်းတို့ရဲ့ index number တွေကို သုံးပြီးတော့ ရယူနိုင်ပါတယ်။ index တွေကိုရေတွက်တဲ့အခါမှာ zero(0) ကနေ စတင်ပြီးရေတွက်ရပါတယ်။ အထက်ပါ program မှာဆိုရင် Hello World သည် index zero ဖြစ်ပြီး ကျန်တဲ့ data များသည်လည်း အစဉ်လိုက် ဖြစ်ပါသည်။ ထို့ကြောင့် result အနေနဲ့ Hello World / Green /Hackers တို့ကို ရရှိခြင်းဖြစ်ပါသည်။

Sample Program (69)

```
List=['green','hackers','winhtut']
print(List[-1])
#output

#winhtut
```

List ကို ယခု ပုံစံ အတိုင်းဖြင့်လည်း နောက်ကနေ ပြန်ပြီး access လုပ်နိုင်ပါသေးသည်။ ယခု Program ကို run ကြည့်လျှင် output အနေဖြင့် winhtut ကို ရပါလိမ့်မည်။

Accessing Data from Multidimensional list

Array မှာကဲ့သို့ List ကို multi-dimensional array ပုံစံဖြင့်လည်း အသုံးပြုလို့ရပါသေးသည်။ multi-dimensional ကို နားလည်ဖို့ ဆိုလျှင် ပထမဦးစွာ row and column ကို နားလည်ရန် လိုအပ်ပါသည်။ row and column ကို သေချာ ခွဲခြားနိုင်ရန် သင်ထောက်ကူပုံတစ်ခု အောက်တွင် ဖော်ပြထားပါသည်။

	Column 0	Column 1	Column 2	Column 3	Column 4	Column 5
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]	a[2][4]	a[2][5]

	column 0	column 1
row 0	green	hackers
row 1	winhtut	gh

Sample Program (70)

```
List=[['green','hackers'],['winhtut','gh']]
print(List[0][1])
print(List[1][1])
#output
#hackers

#gh
```

အထက်ပါ Program ကို run ကြည့်လျှင် output အနေဖြင့် hackers and gh ဆိုတာကို ရပါမည်။ multi-dimensional list ကို ကြေငြာသောအခါတွင် [] square bracket များခွဲပြီး ကြေငြာပေးရပါမည်။ ပထမ square bracket သည် row 0 ဖြစ်ပြီး ဒုတိယ bracket သည် row 1 ဖြစ်သည်။ ထို row များ အထဲ၌ ထည့်ထားသော ပထမဆုံး element သည် column 0 ဖြစ်ပြီး ဒုတိယ element သည် column 1 ဖြစ်သည်။ ['green','hackers'] green သည် row 0 , column 0 ဖြစ်ပြီး hackers သည် row 0 , column 1 ဖြစ်သည်။ ထို့ကြောင့် ယခု အတိုင်း print(List[0][1]) ဟု print လုပ်လိုက်သောအချိန်တွင် hackers ဟု Output အနေဖြင့် ထွက်လာခြင်းဖြစ်ပြီး row 0 , column 0 သာ print လုပ်ခဲ့မည် ဆိုလျှင် green ဟုသာ ထွက်လာမည်ဖြစ်သည်။ ထိုနည်းတူပင် print(List[1][1]) row 1 , column 1 ကို print လုပ်သောအခါတွင်လည်း gh ဟု output ထွက်လာခြင်းဖြစ်သည်။ row 1 , column 0 print(List[1][0]) ဟုပြောင်းပြီး print လုပ်ခဲ့မည်ဆိုလျှင် output အနေဖြင့် winhtut ဆိုသည့် စာသားကို မြင်ရမည်ဖြစ်သည်။

Remove Method - Removing element from list

List ထဲမှ element များကို မိမိစိတ်ကြိုက်ထုတ်နိုင်ရန် Python တွင် remove() ဆိုသည့် Method တစ်ခုပါဝင်ပါသည်။ remove() method ကိုသုံးရာတွင် parameter အဖြစ် ဘယ် element ကို ထုတ်မည်ဖြစ်ကြောင်း ထည့်ပေးရပါမည်။ remove() method ကိုသုံးရာတွင် မိမိ ထုတ်ပစ်လိုသော name ကို အတိအကျ ရေးပေးပြီး ထုတ်ပစ်နိုင်ပါသည်။ ဥပမာ list တစ်ခု ထဲတွင် element သုံးခု ရှိခဲ့လျှင် [1,2,3] ထို element သုံးခုထဲမှ ပထမဆုံးတစ်ခု ဖြစ်သည့် 1 ကို ထုတ်ပစ်လိုသော List.remove(1) ဟု ရေးရပါမည်။ နောက်ထပ် ဥပမာ တစ်ခုအနေဖြင့် list တစ်ခု ရှိမည်။ ထို List ထဲတွင် ['win','htut','gh'] စသည်တို့ရှိမည်။ ထို list ထဲမှ gh ဆိုသည့် စာသားကို ထုတ်ပစ်လိုလျှင် list.remove('gh') ဟုရေးပေးရမည်ဖြစ်ပါသည်။ sample program ကို အောက်တွင်ရေးပြထားပါသည်။

Sample Program (71)

```
my_list=[1,2,3,4,5,6,'win','htut',9,10]

# printing all element from list
print("Original list:")
print(my_list)
#removing element from list
my_list.remove(3)
print("\nAfter removing first time from list 3")
print(my_list)
my_list.remove(5)
print("\nAfter removing second time from list 5")
print(my_list)
my_list.remove('win')
print("\nAfter removing third time from list 'win'")
print(my_list)
```

Sample Program (71)ကို run ကြည့်လျှင် အောက်ပါ အတိုင်း မြင်တွေ့ရပါမည်။

```
#Output
Original list:
[1, 2, 3, 4, 5, 6, 'win', 'htut', 9, 10]
After removing first time from list 3
[1, 2, 4, 5, 6, 'win', 'htut', 9, 10]
After removing second time from list 5
[1, 2, 4, 6, 'win', 'htut', 9, 10]
After removing third time from list 'win'

[1, 2, 4, 6, 'htut', 9, 10]
```

ပထမဆုံးအကြိမ်တွင် list ထဲမှ ရှိသော elements များအားလုံးကို print လုပ်လိုက်ပြီး ဒုတိယအကြိမ်တွင် ထို list ထဲမှ 3 ကို ထုတ်လိုက်ပါသည်။ ထို့ကြောင့် ဒုတိယအကြိမ် print လုပ်သောအခါတွင် list ထဲမှ 3 ပျောက်သွားခြင်းဖြစ်သည်။ remove() method သည် index number နှင့် အလုပ်လုပ်ခြင်း မဟုတ်ပဲ parameter အဖြစ် ထည့်ပေးလိုက်သည့်အတိုင်း

အတိအကျ အလုပ်လုပ်ခြင်းဖြစ်သည်။ အောက်ဆုံးတွင် win ဆိုသည့် စာသားကို list ထဲမှ ထုတ်ထားပါသည်။ ထို့ကြောင့် နောက်ဆုံး output တွင် win မပါလာခြင်းဖြစ်သည်။

pop() Method - Popping elements from list

pop method() ကိုသုံးရာတွင် parameter အနေဖြင့် index number ထည့်ပေးရမည် ဖြစ်ပြီး ထို parameter အတိုင်း list ထဲမှ ဖြုတ်ထုတ်ပါသည်။ remove method နှင့် မတူသည်မှာ pop method သည် index number ဖြင့် အလုပ်လုပ်ခြင်းဖြစ်သည်။ ထို့ပြင် pop() method ထဲတွင် မည့်သည့် parameter မှ မထည့်ပေးလိုက်သည့် အချိန်တွင်မူ List ရဲ့ နောက်ဆုံး element ကို ဖြုတ်ထုတ်ပါသည်။

Syntax:

```
list.pop(index_number)
```

Sample Program (72)

```
my_list=[1,2,3,4,5,6,'win','htut',9,10]
# printing all element from list
print("Original list:")
print(my_list)
#popping element from list
my_list.pop(3)
print("\nAfter popping first time from list 3")
print(my_list)
my_list.pop(5)
print("\nAfter popping second time from list 5")
print(my_list)
my_list.pop()
print("\nAfter popping third time from list index -1")

print(my_list)
```

Sample Program (72)ကို run ကြည့်လျှင် output အနေဖြင့် အောက်ပါ အတိုင်းမြင်တွေ့ ရပါမည်။ အထူးသတိပြုရန်မှာ pop သည် index number ဖြင့် အလုပ်လုပ်သောကြောင့် pop တစ်ခါ လုပ်ပြီးတိုင်း index number များပြောင်းလဲသွားမည်ဖြစ်သည်။

```
#output
Original list:
[1, 2, 3, 4, 5, 6, 'win', 'htut', 9, 10]
After popping first time from list 3
[1, 2, 3, 5, 6, 'win', 'htut', 9, 10]

After popping second time from list 5
```

```
[1, 2, 3, 5, 6, 'htut', 9, 10]
After popping third time from list index -1

[1, 2, 3, 5, 6, 'htut', 9]
```

index()

index() method ကို list ထဲမှာ ရှိသော elements များအား ရှာဖွေရာတွင် အသုံးပြုပါသည်။

Syntax:

```
list.index(element)
```

Parameters:

index method ကိုသုံးရာတွင် မိမိရှာလိုသော single argument တစ်ခုကို ထည့်ပေးရပါသည်။

return value:

index method ကို အသုံးပြုရာတွင် return value အနေဖြင့် မိမိထည့်ပေးလိုက်သော argument ရဲ့ index number ကို ပြန်ရပါသည်။

Sample Program (73)

```
my_list=[1,2,3,4,5,6,'win','htut',9,10]
#searching element win
index=my_list.index('win')
#printing the index number
print(index)
#output

#6
```

Sample Program (73)ကို run ကြည့်လျှင် output အနေဖြင့် 6 ကိုရရှိပါမည်။

Error

List ထဲတွင် မပါဝင်သော elements များအား index method ထဲတွင် argument အဖြစ် ထည့်သုံးသော အခါတွင်မူ value error: ကို ရရှိပါမည်။

Sample Program (74)

```
my_list=[1,2,3,4,5,6,'win','htut',9,10]
#searching element win
index=my_list.index('win')
#printing the index number
print(index)

print(index1=my_list.index('gh'))
```

Sample Program (74)ကို run ကြည့်လျှင် output အနေဖြင့် အောက်ပါအတိုင်း value error ကို ရရှိပါမည်။

```
6
```

```
Traceback (most recent call last):
File "app.py", line 7, in <module>
print(index1=my_list.index('gh'))
ValueError: 'gh' is not in list
```

append( ) method

append( ) method ကို list တစ်ခုထဲမှာ elements များထပ်ထည့်ရန်အသုံးပြုပါတယ်။ append နောက်တွင် ထည့်ပေးလိုက်သော element ကို list elements များရဲ့ နောက်ဆုံးနေရာတွင် သွားရောက် ထည့်ပေးပါတယ်။ list ထဲကို elements များသာမက list များပါ ထပ်ထည့်နိုင်ပါသေးတယ်။ append method သည် original list ကိုသာ modifies လုပ်ပေးပြီး မည်သည့် return value မျှ ပြန်ပေးမည် မဟုတ်ပါ။

Sample Program (75)

```
#declaring and initializing a list

my_list=[1,2,3,4,5,6,'win','htut',9,10]
print('Before update elements list\n',my_list)
#appending an element to the list
my_list.append('greenhackers')
#printing updated element list
print('Updated elements list\n',my_list)
# Output
# Before update elements list
# [1, 2, 3, 4, 5, 6, 'win', 'htut', 9, 10]
# Updated elements list

# [1, 2, 3, 4, 5, 6, 'win', 'htut', 9, 10, 'greenhackers']
```

append() method ကိုသုံးပြီး list ထဲသို့ အခြားသော list တစ်ခုလုံး ထပ်ထည့်ပါမည်။ ပထမဆုံးအနေ ဖြင့် list တစ်ခု အားစတင် ကြေငြာပါမည်။ ထို list နာမည်အား my_list ဟု နာမည် ပေးပါမည်။ထို my_list list ထဲတွင် elements အချို့ ထည့်ထားပါမည်။ ထို့နောက် my_list ထဲသို့ ထပ်ထည့်ရန် list တစ်ခု ထပ်ဆောက်ပါမည်။ ထို list နာမည်ကို new_list ဟု နာမည် ပေးလိုက်ပါမည်။ ထို new_list ထဲတွင်လည်း elements အချို့ထည့်ထားပါမည်။ ပြီးလျှင် append method ကိုသုံးပြီး list တစ်ခုမှ အခြား list တစ်ခုသို့ ပေါင်းထည့်ရန် my_list.append(new_list) ဟုရေးပါမည်။

Sample Program (76)

```
#declaring and initializing a list

my_list=[1,2,3,4,5,6]
print('Before modifies elements list\n',my_list)
#declaring and initializing a new list
new_list=['win','htut','greenhackers']
#appending list to a list
my_list.append(new_list)
```

```
#after modifies my_list

print('After modifies my_list ',my_list)
```

Sample Program (76)ကို run ပြီးလျှင် output အနေဖြင့် အောက်ပါအတိုင်း list နှစ်ခု ပေါင်းသွားသည်ကို တွေ့ရပါမယ်။

```
# Before modifies elements list
# [1, 2, 3, 4, 5, 6]

# After modifies my_list  [1, 2, 3, 4, 5, 6, ['win', 'htut', 'greenhackers']]
```

extend() method

အထက်တွင်ရေးပြထားသော append သည် list တစ်ခုထဲသို့ အခြား list တစ်ခု ထပ်ထည့် ခြင်းဖြစ်သည်။ list တစ်ခုအား အခြား List တစ်ခုမှ elements များ ထပ်ထည့်ပြီး ချဲ့လိုပါက extend method ကိုအသုံးပြုနိုင်ပါသည်။ extend method သည် append method နှင့် ရေးသားပုံခြင်း ဆင်တူသော်လည်း output မှာမူ ကွဲပြားပါသည်။ extend method ထဲသို့ single argument တစ်ခု ထည့်ပေးရပါမည်။ extend method သည်လည်း မူရင်းရှိသော list ကို Modifies လုပ်ခြင်းသာဖြစ်ပြီး မည်သည့် return value မှ ပြန်မည်မဟုတ်ပါ။ list များကို extend လုပ်ရာတွင် operator + or += များကိုလည်း အသုံးပြုနိုင်ပါသေးသည်။

Sample Program (77)

```
#declaring and initializing a list

my_list=[1,2,3,4,5,6]
#declaring and initializing a new list
new_list=['win','htut','greenhackers']
#extending list to a list
my_list.extend(new_list)
#after modifies my_list

print('After modifies my_list ',my_list)
```

Sample Program (77)ကို run ကြည့်လျှင် output အနေဖြင့်အောက်ပါ အတိုင်း မြင်တွေ့ရပါမည်။

```
# After modifies my_list  [1, 2, 3, 4, 5, 6, 'win', 'htut', 'greenhackers']
```

မှတ်ချက်။ append method ကို အသုံးပြုခြင်းသည် list တစ်ခု ထဲသို့ အခြား list တစ်ခုသာ ထပ်ပေါင်း ထည့်ခြင်းဖြစ်ပြီး extend method သည် list တစ်ခုထဲသို့ အခြား list တစ်ခုမှ elements များ extending လုပ်ခြင်းဖြစ်သည်။

Using operator for extending list

append() or extend() method များကို မသုံးပဲ operator ကိုသာသုံးပြီး အောက်ပါ အတိုင်း list များကို extending ပြုလုပ်နိုင်ပါသေးသည်။

Sample Program (78)

```
#declaring and initializing a list

my_list=[1,2,3,4,5,6]
#declaring and initializing a new list
new_list=['win','htut','greenhackers']
#extending list to a list
my_list +=new_list
#after modifies my_list

print('After modifies my_list ',my_list)
```

Sample Program (78)ကို run ကြည့်လျှင်လည်း extend() method မှာကဲ့သို့ output တူနေမည် ဖြစ်ပါသည်။

insert() method

list တစ်ခုထဲမှာရှိတဲ့ elements တွေထဲကို အခြားသော elements များ ထပ်ပေါင်း ထည့်လိုသောအခါမျိုးမှာ insert() method ကိုအသုံးပြုပါတယ်။ insert() method တွင် parameter နှစ်ခုပါဝင်ပြီး ပထမ argument သည် index number ဖြစ်ပြီး ဒုတိယ argument သည် မိမိ ထည့်လိုသော element ဖြစ်သည်။ insert() method သည် list ထဲသို့ element ကိုသာ insert လုပ်ခြင်း ဖြစ်ပြီး မည်သည့် return value မျှ ပြန်ပေးမည်မဟုတ်ပါ။ insert သည် index number ဖြင့် အလုပ်လုပ်သောကြောင့် မိမိတို့အနေဖြင့် 5 ခုမြောက်နေရာမှာ element insert လုပ်လိုလျှင် argument အနေဖြင့် 4 ကို ထည့်ပေးရပါမည်။ အဘယ်ကြောင့်ဆိုသော် python index number သည် zero ကနေစပါသည် (အထက်မှာလည်း ဖော်ပြခဲ့ပြီးဖြစ်သည်)။ insert() method ကိုသုံးပြီး list ထဲသို့ element များသာမက list များကိုပါ ထည့်နိုင်ပါသေးသည်။ insert နှင့် append, extend တို့နှင့်မတူသည်မှာ insert သည် မိမိထည့်လိုသော နေရာအတိအကျကို index number သုံးပြီး ထည့်နိုင်သည်။ သို့သော် append and extend method တို့သည် elements များထပ်ထည့်လျှင် list ရဲ့ နောက်ဆုံးနေရာများတွင်သာ အစဉ်လိုက် နေရာယူသည် မိမိ ထည့်လိုသော နေရာသို့ ရောက်မည် မဟုတ်ပါ။

Sample Program (79)

```
#declaring and initializing a list
my_list=[1,2,3,4,5,6]
#printing original list
print('original list',my_list)
#inserting an element to list
my_list.insert(4,'win')
#printing elements after modifies

print('\nAfter modifies',my_list)
```

Sample Program (79)ကို run ကြည့်လျှင် output အနေဖြင့် အောက်ပါ အတိုင်း မြင်ရပါမည်။

```
original list [1, 2, 3, 4, 5, 6]
After modifies [1, 2, 3, 4, 'win', 5, 6]
```

List တစ်ခုထဲသို့ မိမိ ထည့်လိုသော List များ ထပ်ထည့်ပါမည်။အောက်တွင် sample program ကို ရေးပြထားပါသည်။

Sample Program (80)

```
#declaring and initializing a list
my_list=[1,2,3,4,5,6]
#declaring and initializing a new list
new_list=['win','htut','greenhackers']
#inserting a list to a list
my_list.insert(2,new_list)
#after modifies my_list
print('After modifies my_list ',my_list)
```

Sample Program (80)ကို run ကြည့်လျှင် output အနေဖြင့် အောက်ပါ အတိုင်း မြင်ရပါမည်။

```
After modifies my_list  [1, 2, ['win', 'htut', 'greenhackers'], 3, 4, 5, 6]
```

count() method

count method သည် list တစ်ခုထဲမှ elements များ အကြိမ်ရေ မည်မျှပါဝင်သည်ကို count လုပ်ရာတွင် အသုံးပြုပါသည်။ count() method ကိုသုံးရာတွင် list ထဲမှ element အကြိမ်အရေအတွက်ကို သိနိုင်ရန် argument တစ်ခုပါဝင်ပါသည်။ ပထမဆုံးအနေဖြင့် list တစ်ခု တည်ဆောက်ပါမည်။ ထို list ထဲတွင် 1,2,3,4,5,6,1 စသည့် elements များထည့်ထားပါမည်။

Sample Program (81)

```
#declaring and initializing a list
my_list=[1,2,3,4,5,6,1]
times=my_list.count(1)
print(times)
#output
#2
```

အထက်ပါ program တွင် my_list.count(1) သည် my_list ဆိုသည့် list ထဲမှာ 1 ဆိုသည့် element အကြိမ်မည်မျှ ပါဝင်သည်ကို ရေတွက်ရန်ဖြစ်သည်။ အထက်ပါ program ကို run ကြည့်လျှင် output အနေဖြင့် 2 ကို ရရှိမှာဖြစ်ပါတယ် အဘယ်ကြောင့်ဆိုသော် my_list ထဲတွင် 1 နှစ်ခု ပါဝင်နေသောကြောင့်ဖြစ်သည်။

reverse() method

reverse() method သည် list ထဲမှ elements များအား reverse လုပ်ရန် အသုံးပြုပါသည်။ reverse method ကိုအသုံးပြုရာတွင် မည်သည့် argument မှ ပါဝင်ရန် မလိုအပ်သလို မည်သည့် return value မျှလည်း ပြန်ပေးမည်မဟုတ်ပါ။ reverse method သည် elements များအား reverse လုပ်ပေးပြီး original list အား updates လုပ်ပေးလိုက်မည်သာဖြစ်ပါသည်။

Sample Program (82)

```
#declaring and initializing a list
my_list=[1,2,3,4,5,6,1]
my_list.reverse()
```

```
print(my_list)
#output
#[1, 6, 5, 4, 3, 2, 1]
```

Sample Program (82)ကို run ကြည့်လျှင် output အနေဖြင့် နဂိုမူရင်းရှိသော original list ထဲမှ elements များအား ပြောင်းပြန် လုပ်ထားသည်ကို မြင်ရပါမည်။

sort() method

sort() method သည် list ထဲမှာ ရှိသော elements များကို အစဉ်လိုက် ပြန်လည် sorting လုပ်နိုင်ရန် အသုံးပြုပါသည်။ sort() method သည် မည်သည့် return value ကိုမျှပြန်ပေးမည် မဟုတ်ပဲ original list ကိုသာ changes လုပ်မည်ဖြစ်သည်။

Sample Program (83)

```
#declaring and initializing a list
my_list=[1,2,3,4,5,6,1,0]
my_list.sort()
print(my_list)
#output
#[0, 1, 1, 2, 3, 4, 5, 6]
```

Sample Program (83) ကို run ကြည့်လျှင် list ထဲမှ elements များကို အစဉ်လိုက် စဉ်ပေးထားသည်ကို မြင်ရပါမည်။ number များသာမက characters များကိုလည်း sort လုပ်နိုင်ပါသေးသည်။ အောက်တွင် sample program ကို ဖော်ပြထားပြီး ထို program အား run ကြည့်လျှင် output အနေဖြင့် character များအားအစဉ်လိုက် စီစဉ်ထားသည်ကို မြင်ရပါမည်။

Sample Program (84)

```
#declaring and initializing a list
my_list=['u','i','e','a','o']
my_list.sort()
print(my_list)
#output
#['a', 'e', 'i', 'o', 'u']
```

copy() method

copy() method ကို list တစ်ခုမှ အခြားတစ်ခုကို copy ကူးရာတွင် အသုံးပြုပါသည်။ assignment operator ကိုအသုံးပြုပြီးလည်း copy လုပ်နိုင်ပါသည်။ copy() method သည် မည်သည့် parameter မျှမလိုပါ။ copy() method သည် return value အနေဖြင့် list တစ်ခုကို return ပြန်ပေးပြီး original list ကိုတော့ modifies ပြုလုပ်ခြင်း မရှိပါဘူး။

Sample Program (85)

```
#declaring and initializing a list
my_list=['u','i','e','a','o']
new_list=my_list
print(new_list)
new_list.append('vowels')
```

```
print(new_list)
```

Sample Program (85) ကို run ကြည့်လျှင် အောက်ပါအတိုင်း output များကို မြင်တွေ့ရပါမည်။ new_list ထဲသို့ append လုပ်ထားသည့် elements ကိုလည်း မြင်တွေ့ရပါမည်။

```
#output
#['u', 'i', 'e', 'a', 'o']

# ['u', 'i', 'e', 'a', 'o', 'vowels']
```

clear() method

clear() method သည် list ထဲမှာ ရှိသော elements များအားလုံးကို ဖျက်ပစ်ရန် အတွက် အသုံးပြုပါသည်။ clear() method သည် မည်သည့် return value မျှပြန်ပေးမည်မဟုတ်ပါ။ clear() method ကို အသုံးမပြုနိုင်သည့် အချိန်မျိုးတွင်မူ del operator ကိုလည်း အသုံးပြုနိုင်ပါသည်။

Sample Program (86)

```
#declaring and initializing a list
my_list=['u','i','e','a','o']
my_list.clear()

print(my_list)
```

Sample Program (86) ကို run ကြည့်လျှင် output အနေဖြင့် square brackets[] ကိုသာရရှိမည်။ အဘယ်ကြောင့်ဆိုသော် list ထဲရှိ elements များအားလုံးကို clear လုပ်ပစ်သောကြောင့် ဖြစ်သည်။

Dive To List

List များသည် string များနှင့် လုံးဝမတူပါ။ string များသည် value တူလျှင် memory address များ တူညီသည်။ List များသည် value တူညီပါသော်လည်း memory address မတူသော နေရာတွင် သိမ်းဆည်းပါသည်။ ထို့ကြောင့် value တူသော list နှစ်ခုကို is ဖြင့် identity လုပ်ကြည့်လျှင် False ကိုသာ ရပါမည်။ အဘယ်ကြောင့်ဆိုသော် သူတို့သိမ်းဆည်းသော memory address များ မတူညီခြင်းကြောင့်ဖြစ်သည်။ sample program ကို အောက်တွင် identity operator နှင့် relational operator နှစ်ခုလုံးနှင့်ပါ ကွဲပြားစွာ ရေးပြထားပါသည်။ output များကို စစ်ဆေးလျှင် memory address များ မတူညီသည်ကို မြင်ရပါမည်။

Sample Program (87)

```
string1=[1,2,3,4,5]
```

```
string2=[1,2,3,4,5]
print(id(string1),id(string2))
string3=string1 is string2
print(string3)
if string1 == string2:
    print('They are same')
#output
#26691064 26692224
#False
#They are same
```

Namespaces

Namespace ကို သိဖို့ ပထမဦးစွာ name ကို အရင်သိရပါတယ်။ a = 20 တွင် a သည် name or identifier ဖြစ်ပြီး 20 သည် memory ထဲမှာ stored လုပ်ထား်းသည့် object ဖြစ်သည်။ a ဆိုသည့် name နှင့် 20 ဆိုသည့် object တို့အား id() function ကိုသုံးပြီး Print ထုတ်ကြည့်လျှင် same address ဖြစ်နေသည်ကို မြင်နိုင်ပါသည်။ Python programming မှာ အရာအားလုံးနီးပါးဟာ object တွေပါပဲ။ object အကြောင်းကို နောက်ပိုင်းသင်ခန်းစာတွေမှာ အသေးစိတ် ဆွေးနွေးသွားပါမယ်။ strings , lists , functions and etc အားလုံးဟာ object တွေပါ။

Sample Program (88)

```
a =20
print('a address is',id(a))
print('20 address is',id(20))
#output
#a address is 140711874009344
#20 address is 140711874009344
```

Namespaces ဆိုတာ name တွေ အများကြီးပေါင်းထားခြင်းကို ဆိုလိုပါတယ်။ program တစ်ပုဒ်မှာ name တွေကို unique ဖြစ်စေရန်နှင့် name တွေကို အသုံးပြုရာမှာ မမှားစေဖို့(conflict) အတွက် အသုံးပြုသော system တစ်ခုဖြစ်ပါတယ်။ အခြေခံအားဖြင့် name space သုံးမျိုးရှိပါတယ်။

1. Local Namespace :

Local namespace ဆိုတာ function တစ်ခုအတွင်းမှာရှိတဲ့ name တွေကို ဆိုလိုတာပါ။ သူတို့ရှိနေတဲ့ function ကိုအခေါ်ခံလိုက်ရတဲ့အချိန်မှာ namespace တွေကို create လုပ်ပါတယ်။ function ကို နောက်ပိုင်းသင်ခန်းစာများတွင်အသေးစိတ် ဆွေးနွေးသွားပါမည်။

```
name1 = 5 #name1 is global namespace
def outer_fun():
  name2 = 6 #name2 is local namespace
  def inner_fun():
    name3 = 7 #name3 is inner local namespace
```

2. Global Namespace :

module or library တွေကို program ထဲမှာ ခေါ်သုံးခြင်းဖြင့် ပါလာတဲ့ names တွေကို Global Namespace လို့ ခေါ်ပါတယ်။ ဥပမာ စာရေးသူက winhtut ဆိုတဲ့ module or library ကို တည်ဆောက်ထားပြီး ထို library ထဲတွင် ပါရှိလာသော name ကိုဆိုလိုသည်။

```
from math import log2, log10
```

math ဆိုသည့် module မှ log2 နှင့် log10 စသည် name တို့ကို ခေါ်သုံးမည်ဟု ဆိုလိုခြင်းဖြစ်သည်။

3. Built–in Namespace :

ကိုယ်တိုင်ကြေငြာထားသော variable လည်းမဟုတ်သလို module or library များမှ ခေါ်သုံးထားခြင်းမျိုးလည်းမဟုတ်ပဲ built-in အနေဖြင့် ပါဝင်နေသော functon and name များကို ဆိုလိုပါသည်။ print() and id()တို့သည် built – in namespace များ ဖြစ်ကြသည်။

Lifetime of a namespace and scope of object တို့ကို function ခန်းတွင် အသေးစိတ် ဆွေးနွေးသွားပါမည်။

Introduction to Function

Programs တွေဟာကြီးမားလာတာနဲ့အမျှ ရှုပ်ထွေးလာပါတယ်....။ Error တွေကို ပြန်ဖြေရှင်းဖို့လည်းအရမ်းခက်ခဲလာပါတယ်။ ရှုပ်ထွေးတဲ့ program တစ်ခုကို အပိုင်းလေးတွေ ခွဲလိုက်ခြင်းအားဖြင့် နောက်လူတွေ ပြန်ကြည့်ရင် လွယ်သွားသလို programs ကို update လုပ်ဖို့ပဲ ဖြစ်ဖြစ်၊ ပြန်ပြီးတော့ trace လုပ်ဖို့လည်းလွယ်ကူသွားပါတယ်။

Function များတွင် standard library function and programmer defined function ဟူ၍နှစ်မျိုးရှိပါသည်။ Standard library function သည် မူရင်း ပါဝင်သော build-in-function များ သို့မဟုတ် programmer များမှ အလွယ်သုံးနိုင်ရန် ဖန်တီးပေးထားသော function များကို ဆိုလိုခြင်းဖြစ်ပြီး programmer defined function သည် program ထဲတွင် မိမိတို့ ရေးသော function များကို ဆိုလိုခြင်း ဖြစ်သည်။

The range() function (Standard Library Function)

range() function သည် standard library function ဖြစ်ပြီး return value အနေဖြင့် zero မှစပြီး number များကို အစဉ်လိုက်တစ်ပေါင်းပြီး ထုတ်ပေးပါသည်။ ထို function ထဲတွင် arguments အနေဖြင့် ထည့်ပေးလိုက်သော number ကိုမရောက်ခင်အထိ numbers များကို sequence အလိုက်ထုတ်ပေးပါသည်။

Sample Program (89)

```
for x in range(10):
    print(x)
#output
# """
# 0 1 2 3 4 5 6 7 8 9
# """
```

Sample Program (89)ကို run ကြည့်လျှင် 0 မှစပြီး 9 ထိ number များကို အစဉ်လိုက် မြင်တွေ့ ရပါမည်။

Sample Program (90)

```
for x in range(2, 6):
    print(x)
#output
# """
# 2 3 4 5 # """
```

range() function ထဲတွင် အထက်ပါအတိုင်း arguments နှစ်ခုလည်းထည့်နိုင်ပါသည်။ ပထမတစ်ခု သည် starting point ဖြစ်ပြီး နောက်တစ်ခုသည် end လုပ်ရန်ဖြစ်သည်။ program အား run ကြည့်လျှင် 2 မှာ စပြီး 5 ထိ အစဉ်လိုက်ပေါ်နေသည်ကို မြင်တွေ့နိုင်ပါသည်။ second parameter 6 ဖြစ်လျှင် သူ့ထက် တစ် လျော့ပြီး 5 အထိ သာ ဖော်ပြသည် ဆိုတာကို သတိပြုပါ။

Programmer Defined Function

Syntax of Function:

```
def function_name(parameters):
    """docstring"""
statement(s)
```

- def keyword သည် function တစ်ခုအား စတင်ကြေငြာရာတွင် အသုံးပြုပါသည်။ python function များကို ကြေငြာရာတွင် def ကိုသုံးပေးရမည်။
- function_name သည် မိမိ ကြေငြာလိုသော function တစ်ခုရဲ့ name ကို ရေးနိုင်သည်။ အထက်ပါ program တွင် function_name နေရာ၌ fun_name လို့လည်းရေးနိုင်သည်။
- parameters သည် function တစ်ခုအား value များတွက်ချက်လိုသောအခါတွင် အသုံးပြုပါသည်။ print စာသားများ ရေးသားထားသော function တစ်ခုတွင်မူ parameters ကို ထည့်ရန် မလိုအပ်ပါ။
- နောက်ဆုံးမှာ ရေးထား်သော column သည် ထို function ရဲ့ header ပိုင်းဆုံးပြီ ဖြစ်ကြောင်း ကြေငြာပေးခြင်းဖြစ်သည်။
- Docstring သည် ထည့်လည်းရသလို မထည့်လည်းရပါသည်။ documentation string နေရာတွင် ယခု function က ဘယ်လိုအလုပ်တွေလုပ်သလဲ ဆိုတာကို comment အနေဖြင့် ရေးသားထားခြင်းသည် good programming practice ဖြစ်သည်။
- Statements ဆိုသည့် နေရာသည် function body နေရာဖြစ်သည်။ ထိုနေရာတွင် မိမိတို့ အလုပ်လုပ်သော instruction or function များကို ရေးသားရပါမည်။
- Return သည် မိမိတို့ ယခုရေးလိုက်သော function ကိုအခြားသော function တစ်ခုမှ လှမ်းခေါ်သောအခါတွင် value or data တစ်ခုခု ပြန်ပေးလိုသောအခါတွင် အသုံးပြုသည်။ အတွက်အချက်များမပါပဲ စာသားများသာ ဖော်ပြလိုသောအခါတွင်မူ return ကို ထည့်ရေး ရန် မလိုအပ်ပါ။

Sample Program (91)

```
#Writing a first function
def   green():
        print('We are Myanmar')
```

အထက်တွင် green ဆိုသည့် function တစ်ခုအား ကြေငြာထားပြီး function header ပိုင်းတွင် parameters အနေဖြင့် ဘာမှ ထည့်မထားသေးပါ။ function body တွင်မူ We are Myanmar ဆိုသည့် စာသားကို ဖော်ပြရန် instruction တစ်ခု ရေးထားပါသည်။ အထက်ပါ program ကို run ကြည့်လျှင် မည်သည့် output မျှ ထွက်လာမည်မဟုတ်ပါ။ အဘယ်ကြောင့်ဆိုသော် green() ဆိုသည့် function ကိုမည်သည့် function ကမှခေါ်ထားခြင်း မရှိသလို print ထုတ်ထားခြင်းလည်း မရှိပါ။ green() function အားခေါ်ရန် အောက်ပါ အတိုင်းရေးသားပါမည်။

Sample Program (92)

```
def green():
   print('We are Myanmar')
green()
```

Sample Program (92)ကို run ကြည့်လျှင် We are Myanmar ဆိုသည့် စာသားကို မြင်ရပါမည်။

Python Function with Parameters

Python function တစ်ခုအား parameter များထည့် (parameter passing) ပါမည်။

Sample Program(93)

```
def green(gh):
   print('We are Myanmar'+gh)
green('green hackers')
#output

#We are Myanmar green hackers
```

အထက်ပါ program တွင် def green() နောက်တွင် gh ဆိုသည့် variable တစ်ခုကို သုံးလိုက်ပါသည်။ ထို variable နေရာတွင် ၎င်း function အားခေါ်ဆိုသောနေရာမှ ထည့်ပေးလိုက်သော value တစ်ခု ဝင်လာမည်ဖြစ်သည်။ အောက်တွင် function ကို ပြန်ခေါ်သော နေရာ၌ green hackers ဆိုသည့် စာသားကို ထည့်လိုက်ပါသည်။ ထိုစာသားသည် def green(gh) မှ gh ဆိုသည့်နေရာတွင် အစားဝင်သွားပါမည်။ ထို့ကြောင့် print ထုတ်သောအခါတွင် We are Myanmar green hackers ဟု ပေါ်နေခြင်းဖြစ်သည်။ အဘယ်ကြောင့်ဆိုသော် gh နေရာတွင် ထို function ကို ခေါ်သောနေရာက ထည့်ပေးလိုက်သော green hackers ဆိုသည့် စာသားအစား ဝင်လာသောကြောင့်ဖြစ်သည်။

Parameter Passing

စုံ (သို့) မ စစ်ဆေးတဲ့ even_or_odd() ဆိုတဲ့ function တစ်ခု တည်ဆောက်ပြီး ထို function အား မတူညီသော parameters များဖြင့် ခေါ်ပါမည်။

Sample Program (94)

```
#Python function
def even_or_odd(x):
   if(x%2== 0):
         print('even')
   else:
         print('odd')
even_or_odd(2)
```

```
even_or_odd(5)
#output
#even

#odd
```

Sample Program (94)ကို run ကြည့်လျှင် ပထမ output သည် even ရမည်ဖြစ်ပြီး ဒုတိယ Output သည် odd ရပါမည်။ အဘယ်ကြောင့်ဆိုသော် even_or_odd function ထဲတွင် ပထမ တစ်ကြိမ်၌ 2 ကို ထည့်ပေးလိုက်ပြီး ထို 2 သည် x နေရာတွင် အစားထိုးသွားပါသည် even_or_odd(2) ။ ထို့နောက် x အား 2 ဖြင့်စားသောအခါ အကြွင်းသည် zero (0) ဖြစ်နေလျှင် even ဆိုသည့် output ကို ထုတ်ပေးသည်။ ဒုတိယ တစ်ကြိမ် even_or_odd function ထဲတွင် parameter အနေဖြင့် 5 ကိုထည့်ပေးလိုက်ပြီး ထို 5 သည် x နေရာတွင် အစားဝင်လာပါသည်။ ထို x အား 2 ဖြင့် စားသောအခါ စားရှိမပြတ်သောကြောင့် zero (0) နှင့် မညီပါ။ ထို့ကြောင့် output အနေဖြင့် else နောက်မှ odd ကို ထုတ်ပေးခြင်းဖြစ်ပါသည်။ အထက်ပါ program ကို အောက်ပါအတိုင်း implement လုပ်ကြည့်နိုင်ပါသည်။

Quiz

Sample Program (95)

```
#Python function

def even_or_odd(x):
    if(x%2== 0):
        print('even')
    else:
        print('odd')
i=1
while i<5:
    print('{} time calling is'.format(i))
    even_or_odd(i)
    i=i+1
#output
#  1 time calling is
# odd
# 2 time calling is
# even
# 3 time calling is
# odd
# 4 time calling is

# even
```

Sample Program (95)တွင် function အား while loop ကို သုံးပြီး ထပ်ခါ ထပ်ခါ ခေါ်ထားပါသည်။ ထို့ပြင် .format ကိုလည်း ပြန်သုံးထားပါသည်။ အထက်ပါ program ကို

နားလည်သဘောပေါက်အောင် ကြိုးစားပြီး မိမိ စိတ်ကြိုက် program ကို လိုသလိုပြောင်းလဲ ရေးသားခြင်းဖြင့် programming skill တိုးတက်လာမည်ဖြစ်သည်။

Avoidable Error

1-function တစ်ခုကို အခြား function တစ်ခုထဲမှာ သွားရောက်ပြီး ကြေငြာလျှင် syntax error ဖြစ်နိုင်သည်။

2- meaningful ဖြစ်တဲ့ function names တွေ ၊ meaningful ဖြစ်တဲ့ parameter names တွေ ပေးခြင်းအားဖြင့် programs ကို ဖတ်ရလွယ်ကူစေတယ်၊ comments တွေ အများကြီးပေးစရာလည်း မလိုတော့ပါဘူး။

3- small functionsတွေ ရေးသားခြင်းအားဖြင့် programs ပြန်ရေးဖို့၊ အမှားရှာဖွေဖို့၊ ပြန်လည် ပြင်ဆင်ဖို့၊ ပိုကောင်းအောင်ရေးဖို့ လွယ်ကူစေပါတယ်။

List passing to a function

အထက် program များတွင် function တစ်ခုထဲသို့ string and number များ ဖြတ်ခဲ့ပါသည်။ ယခု program တွင်မူ function ထဲသို့ list တစ်ခုလုံးဖြတ်သွားမှာ ဖြစ်ပါတယ်။ ပထမဆုံး အနေဖြင့် listFun ဆိုသည့် function တစ်ခု တည်ဆောက်ထားပြီး ထိုထဲတွင် variable x ကို သုံးပြီး list တစ်ခုကို ကြေငြာထားပါသည်။

Sample Program (96)

```
# pass list to a function
def listFun(x):
  x[0] = 15
#declaring a list
a_list = [10, 11, 12, 13, 14, 15]
listFun(a_list)
print(a_list)
#output

#  [15, 11, 12, 13, 14, 15]
```

Sample Program (96)အား run ကြည့်သောအခါတွင် output အနေဖြင့် a_list ရဲ့ ပထမဆုံး value 10 နေရာတွင် 15 ချိန်းသွားသည်ကို မြင်ရပါမည်။ အဘယ်ကြောင့်ဆိုသော listFun function ထဲတွင် a_list ဆိုသည့် list တစ်ခုလုံးကို parameter အနေဖြင့် ထည့်ပေးလိုက်ပြီး ထို list သည် x နေရာတွင် နေရာယူသွားပါသည်။ x[0]=15 ဟု ရေးထားသောကြောင့် a_list ဆိုသည့် list ရဲ့ ပထမဆုံး index 0 နေရာကို 15 ဟု assign လုပ်လိုက်ပါသည်။ ထို့ကြောင့် output ထုတ်သောအခါတွင် ပထမဆုံး value 10 အစား 15 ထွက်နေခြင်းဖြစ်သည်။

Pass by reference in python

Program memory ထဲ၌ stack and heap ဆိုပြီး နှစ်မျိုးရှိပါတယ် ။ stack ထဲတွင် methods များ functions များ ကို သိမ်းဆည်းထားပြီး heap memory ထဲတွင်မူ value များ ဖြစ်တဲ့ 10,20,30… စတဲ့ objects များကို အကန့်အလိုက် သိမ်းဆည်းပါတယ်။

Python Memory Manager

Win Htut
Green Hackers Team

What is a reference?

Reference ဆိုတာ Memory address ကို ဆိုလိုခြင်းဖြစ်ပါသည်။ variable တစ်ခုကို စတင်ကြေငြာ မယ်ဆိုပါစို့ var_a =10 တွင် var_a သည် 10 ဆိုသည့် object ကို ကိုယ်စားပြုသည်ဟု ထင်ရသော်လည်း var_a သည် reference ဆိုသည့် object ရဲ့ memory တွင် သိမ်းဆည်းထားသော memory address ကိုသာ ကိုယ်စားပြုပါသည်။

Python Memory Manager

ပုံတွင်ပြထားသည့် အတိုင်း var_a သည် 0x1008 ဆိုသည့် memory address ကိုသာ ညွှန်းထားခြင်း (reference) လုပ်ထားခြင်းဖြစ်သည်။ 0x1008 ဆိုသည့် memory address ကသာ object ဖြစ်သည့် 10 နှင့် သက်ဆိုင်ပါသည်။ python programming တွင် id() ဆိုသည့် built-in function ဖြင့် var_a reference လုပ်ထားသော memory address ကိုသိနိုင်သည်။ ထို memory address များကို hexadecimal နှင့်ပြလိုပါကလည်း hex() function ကိုသုံးနိုင်သည်။ print (hex(id (var_a))) ယခု အတိုင်းရေးသားနိုင်ပါသည်။

Sample Program (97)

```
def myFun(x):
    x=[20, 30, 40]
    print(x)
lst =[10,11,12,13,14,15]
myFun(lst)
print(lst)
#output
#[20, 30, 40]

#[10,11,12,13,14,15]
```

အထက်ပါ program အား run ကြည့်လျှင် output အနေဖြင့် lst ထဲက valueများကိုသာ ရရှိမည်။ line number 1 မှ 3 သည် myFun ဆိုသည့် function အတွက် code line များဖြစ်ပြီး line 4 တွင် list တစ်ခုအား တည်ဆောက်ထားပါသည်။ program စသော အချိန်တွင် line 4 ပြီးသောအခါ၌ line 5 တွင် myFun ဆိုသည့် function ကို lst ဆိုသည့် list အား parameter အနေဖြင့် ထည့်ခေါ်လိုက်ပါသည်။ ထို့ကြောင့် x နေရာတွင် lst ထဲမှ value များ ဖြစ်သည့် 10 , 11 , 12 ,13 ,14 ,15 စသည့် value များ ဝင်နေပါမည်။ line 2 သို့ ရောက်သောအခါတွင်မူ x ထဲသို့ object အသစ် assign လုပ်ခံလိုက်ရပါသည်။ ထို့ကြောင့် နဂိုမူရင်းရှိသော တန်ဖိုးများ x ထဲတွင် မရှိတော့ပဲ အသစ် assign လုပ်ခံလိုက်ရသော 20,30,40 စသည့် value များသာ ရှိနေပါမည်။ ထို့ကြောင့် line 3 တွင် print လုပ်သောအခါ [20,30,40] စသည့် value များကိုသာရရှိခြင်းဖြစ်သည်။ ထိုသို့ object အသစ် assign လုပ်ခံလိုက်ရသောအချိန်တွင် reference link သည် ပျက်စီးသွားသလို နဂိုမူရင်းရှိနေသော lst ဆိုသည့် list ကိုလည်း

modified လုပ်ခြင်းမရှိပါဘူး။ ထို့ကြောင့် line 6 တွင် output ထုတ်သောအခါ lst ဆိုသည့် list ထဲ၌ရှိသော မူရင်း value များသာ output အနေဖြင့် ထွက်လာခြင်းဖြစ်သည်။ C/C++ တို့တွင်မူ နဂိုမူရင်း ရှိသော value များ modified လုပ်ခြင်း ခံရပါသည်။ pass by reference အား ပိုမိုပြီး နားလည်စေရန် အောက်ပါ သင်ခန်းစာကို နားလည်အောင် ဆက်လက် ကြိုးစားကြည့်ပါ။

Sample Program (98)

```
def myFun(x):
  x = 20
x = 10
myFun(x)

print(x)
```

output အနေဖြင့် 10 ကို ရရှိပါမည်။အောက်တွင် ပြထားသော swap လုပ်သည့် program အားလည်း ပိုမိုနားလည်စေရန် ရေးသားကြည့်သင့်ပါသည်။

Sample Program (99)

```
def swapping(x, y):
   temp = x;
   x = y;
   y = temp;
x = 2
y = 3
swapping(x, y)
print(x)

print(y)
```

အထက်ပါ program ကို run ကြည့်လျှင် output အနေဖြင့် 2 and 3 ကိုသာ ပြန်ရပါသည်။ swapping ဆိုသည့် function ကို လှမ်းခေါ်သည့်အခါ ပထမဆုံး x တန်ဘိုး ဖြစ်သည့် 2 သည် temp ဆိုသည့် variable ထဲသို့ assign ထည့်လိုက်သည်။ ထို့နောက် y value ဖြစ်သည့် 3 အား x ထဲသို့ assign ထည့်လိုက်သည်။ temp=x , x=y စသည့် code နှစ်ကြောင်း run ပြီးသော အချိန်တွင် x တန်ဖိုးမှာ 3 ဖြစ်နေပါပြီ။ ထို့နောက် y = temp ဆိုသည့် code line ကို အလုပ်ဆက်လုပ်သောအခါတွင်မူ y တန်ဘိုးသည် temp ထဲမှ ရှိသော 2 ကို assign လုပ်ခံ ရပါသည်။ ထို့ကြောင့် code line 3 ခုလုံး run ပြီးသော အချိန်တွင် x တန်ဘိုးသည် 3 ဖြစ်ပြီး y တန်ဘိုးသည် 2 ဖြစ်နေပါသည်။ ထို code line 3 ခုရှိသော swapping function ထဲတွင် print လုပ်ကြည့်ပါက တွေ့နိုင်သည်။ သို့သော် swapping function အပြင်ဘက်ကို ရောက်သွားပြီးနောက် x and y တို့သည် အပြင်ဘက်က မူရင်း value 2 and 3 ကို တွေ့ရမည်ဖြစ်သည်။

Everything is an Object

Python မှာ အားလုံးနီးပါးဟာ object တွေပါ။ ဆိုလိုချင်တာက Data Types တွေဖြစ်ကြတဲ့ integers(int), floats(float), Booleans (bool), Strings(str), Lists(list),

Tuples(tuple), Sets(set), Dictionaries(dict), None(None Type) တို့အားလုံးဟာလည်း objects တွေပါပဲ။
opetrators (+ , - , * , / , ...) စတာတွေဟာလည်း objects တွေပါပဲ။ Object ဆိုတာ class ကို ကိုယ်စားပြု (instance of class) ထားခြင်းကို ဆိုလိုပါတယ်။

Sample Program (100)

```
a=10

print(type(a))
```

အထက်ပါ အတိုင်း Program ရေးပြီး run ကြည့်ပါက output အနေဖြင့် (<class 'int'>) ကိုရပါမည်။ အဘယ်ကြောင့်ဆိုသော် a သည် 10 ကို assign လုပ်ထားသောကြောင့် int ဖြစ်ပြီး int သည်လည်း class ဖြစ်သည်။ နောက်တစ်နည်းအနေဖြင့် int သည် object (instance of class) ဖြစ်ကြောင်းကို အောက်ပါ အတိုင်းဥပမာ ပြပါမည်။

Sample Program (101)

```
c=int()

print(c)
```

int() ဆိုသည့် object ကို c ထဲသို့ assign လုပ်ပါမည်။ထို့နောက် c အား print ထုတ်ကြည့်ပါက output အနေဖြင့် 0 ကိုသာရပါမည်။

```
c=int('101',base=2)

print(c)
```

အထက်ပါ Program အတိုင်း run ကြည့်ပါက output အနေဖြင့် 5 ကို ရရှိပါမည်။ ထို့ကြောင့် int သည် object ဖြစ်ပြီး int class ကို ကိုယ်စားပြုထားခြင်းဖြစ်သည်။
Functions, Classes and Types တွေဟာလည်း objects တွေပါပဲ functions, classes and types တွေမှာလည်း memory address ရှိပါတယ်။ class ကတော့ ဘယ်သူ့ကိုယ်မှ ကိုယ်စားပြုမနေပဲ သူကိုယ်တိုင်ပဲ ကြေငြာရတာပါ။ class အကြောင်းကို နောက်ပိုင်း သင်ခန်းစာများမှာ အသေးစိတ် ဆွေးနွေးသွားပါမည်။

```
def my_fun():

    ....
```

my_fun သည် သူ့နောက်က ( ) parentheses မပါလျှင် Python ၌ variable ဖြစ်သည်။ id(my_fun) ဟုရေးကြည့်လျှင် သူ့ memory address ကိုရပါမည်။ ထို့ကြောင့် my_fun သည် သူ့နှင့်သက်ဆိုင်သည့် object တစ်ခုကို reference လုပ်ထားသည် consequence နောက်ဆက်တွဲ အနေဖြင့် မည်သည့်

1. objects (including function) မဆို variable တစ်ခုထဲကို assign လုပ်နိုင်ပါသည်။ eg: a = my_fun()
2. မည်သည့် object မဆို function တစ်ခုအတွင်းသို့ parameter အနေဖြင့် passing လုပ်နိုင်ပါတယ်။

3. Function များ သည် return value အနေဖြင့် function တစ်ခုကို return ပြန်ပေးနိုင်သည်။

Sample Program (102)

```
def square(a):
  return a**2

print(type(square))
```

Sample Program (102)အတိုင်း run ကြည့်လျှင် <class 'function'> ကို output အနေဖြင့် မြင်ရပါမည်။ ထို့ကြောင့် square သည် memory address တစ်ခုသာဖြစ်ပြီး အခြားသော variable တစ်ခုထဲသို့ assign လုပ်နိုင်သည်။

Sample Program (103)

```
def square(a):
  return a**2
print(type(square))
fun=square
print(id(square))

print(id(fun))
```

Sample Program (103)အတိုင်း run ကြည့်ပါက square and fun တို့သည် တူညီသော memory address ကိုရပါမည်။ ထို့ကြောင့် program တွင် function ကို fun ဆိုသည့် variable အား အသုံးပြု၍လဲ အောက်ပါ အတိုင်းခေါ်ဆိုနိုင်သည်။

Sample Program (104)

```
def square(a):
  return a**2
print(type(square))
fun=square
print(fun(2))
```

Sample Program (104) run ကြည့်လျှင် error မတက်ပဲ output အနေဖြင့် 4 ကို ရရှိပါမည်။

Returning function from a function

Function ကနေ function များ return ပြန်ပေးခြင်းကို လေ့လာရန် ပထမဆုံးအနေဖြင့် function နှစ်ခုတည်ဆောက်ပါမည် ။

ပထမ function သည် square(a): ဖြစ်ပြီး ဒုတိယသည် cube(a): ဖြစ်သည်။

```
def square(a):
  return a**2
def cube(a):
  return a**3
```

ဒုတိယအနေဖြင့် တတိယမြောက် function တည်ဆောက်မည်ဖြစ်ပြီး ထို function ထဲကို ဖြတ်လာသော arguments အလိုက် သက်ဆိုင်ရာ function များကို return ပြန်ပေးပါမည်။

```
def fun(number):
  if number==1:
    return square
  else:
    return cube
```

တတိယ အနေဖြင့် ထို function များကို လှမ်းခေါ်ပါမည်။

```
f=fun(1) #getting square function

print(f(1))
```

fun(1) ဟု fun function ကိုလှမ်းခေါ်သောအခါ argument သည် 1 ဖြစ်သည့်အတွက် return အနေဖြင့် square function ရဲ့ address ကိုပြန်ပေးပါသည်။ ထို address ကို f ဆိုသည့် variable၊ f=fun(1) ဟုရေးကာ assign လုပ်လိုက်ပါသည်။ ထို့ကြောင့် f သည် ယခုအချိန်တွင် square function ရဲ့ address နှင့် အတူတူပင်ဖြစ်သည်။ ဆိုလိုသည်မှာ f = square() ဖြစ်နေပါပြီ။ ထို့နောက် print(f(1)) ဟုရေးလိုက်သောအခါ output အနေဖြင့် 1 ကို ရရှိခြင်းဖြစ်သည်။ အထက်ပါ ရှင်းလင်းချက်များအား ပိုပြီး detail ကျစေရန် အောက်ပါအတိုင်း memory address များဖြင့် debug လုပ်ပြထားပါသည်။

အဝါရောင် မျဉ်းလိုင်းကို ကြည့်ပါ။ အနီရောင်ဖြင့် ပြထားသော f ရဲ့ memory address သည် အပေါ်က square function's memory address နှင့် အတူတူ ဖြစ်နေသည်ကို မြင်နိုင်ပါသည်။

```
f=fun(2) #getting cube function

print(f(2))
```

ယခုတစ်ခါမှာတော့ fun ဆိုတဲ့ function ကို parameter 2 ဖြင့်လှမ်းခေါ်ပါတယ်။ ယခုအခါတွင်မူ 1 နှင့်မတူပဲ 2 ဖြစ်နေသည့်အတွက် cube ရဲ့ memory address ကို ပြန်ပေးပါတယ်။ f=fun(2) လို့ ရေးထားသောကြောင့် cube function ရဲ့ memory address သည် variable f ထဲသို့ ရောက်သွားပါသည်။ ယခုအခြေနေတွင် f=cube() ဖြစ်နေသည်ကို

imaging လုပ်နိုင်သည်။ ထို့နောက် print(f(2)) လို့ရေးလိုက်သည့်အတွက် cube function ထဲသို့ argument 2 ဖြင့် ဖြတ်သွားပြီး output အနေဖြင့် 8 ကို ရရှိခြင်းဖြစ်သည်။အထက်ပါ ရှင်းပြချက်များကို အောက်ပါ ပုံထဲက အတိုင်း memory address များဖြင့် အသေးစိတ် ရှင်းပြထားပါသည်။

```
Book > app.py > ...
1   def square(a):
2       return a**2
3   def cube(a):
4       return a**3
5
6   def fun(number):
7       if number==1:
8           return square
9       else:
10          return cube
11
12  f=fun(2) #getting square function
13  print(f(2))
14
15
```

```
<function cube at 0x0000022AAC949E50>
<function fun at 0x0000022AAC949EE0>
<function cube at 0x0000022AAC949E50>
```

f ရဲ့ memory address နှင့် cube function ရဲ့ memory address တို့ အတူတူ ဖြစ်နေသည်ကို မြင်နိုင်ပါသည်။ program အပြည့်အစုံကို အောက်တွင် ဖော်ပြထားပါတယ်။

Sample Program (105)

```
def square(a):
  return a**2
def cube(a):
  return a**3
def fun(number):
  if number==1:
    return square
  else:
    return cube

#Start calling function
f=fun(1) #getting square function
print(f(1))
f=fun(2) #getting cube

print(f(2))
```

Function Passing to a Function

အထက်တွင် ဖော်ပြပြီးသော square and cube function များကို function တစ်ခုအတွင်းမှာ parameter အနေဖြင့် passing လုပ်ပါမည်။ အောက်တွင် ပထမဆုံး အနေဖြင့် function သုံးခု တည်ဆောက်ထားပါသည်။

Sample Program (106)

```
def square(a):
   return a**2
def cube(a):
   return a**3
def fun_fun(fn , n):

   return fn(n)
```

fun_fun function ထဲတွင် fn သည် function ဖြစ်ပြီး n သည် fn function က ပြန်သုံးမည့် parameter ဖြစ်သည်။ တကယ်လို့ fn နေရာတွင် cube ဝင်လာလျှင် return fn(n) ဆိုသည့် code line ကို ရောက်သောအခါ cube() function ကိုလှမ်းခေါ်ပါမည်။ထိုနည်းတူပင် square() ဖြစ်ခဲ့ပါက square function ကိုလှမ်းခေါ်မည်ဖြစ်ပါသည်။

အထက်တွင် ဖော်ပြထားသော နည်းလမ်းအတိုင်း function တစ်ခုထဲတွင် function တစ်ခုကို parameter အနေဖြင့် ထည့်ခေါ်နိုင်ပါသည်။ a value သည် cube function ကိုလှမ်းခေါ် သောကြောင့် 27 ရမည်ဖြစ်ပြီး b value သည် square function ကို လှမ်းခေါ်သောကြောင့် 9 ရမည်ဖြစ်ပါသည်။

Avoidable Error

- my_fun သည် function ရဲ့ name သာဖြစ်သည် ထို name ထဲတွင် memory address ပါဝင်နေသည်။
- my_fun() ဟု ရေးမှ သာလျှင် function သည် စတင် အသက်ဝင်လာမည်ဖြစ်သည်။

Positional and Keyword Arguments

Function တစ်ခုကနေ အခြား function တစ်ခုကို argument များနှင့် ခေါ်ဆိုသောအခါ အခေါ်ခံရသော function တွင် မည်သည့် parameter သည် မည်သည့် argument ၌ ဖြတ်သွားမည် ဖြစ်ကြောင်း positioning လုပ်နိုင်ပါသည်။ ထိုသို့ positioning လုပ်ရာ၌ keyword arguments များကို အသုံးပြုနိုင်သည်။

Sample Program (107)

```
def square(a,b):#parameters
   return a*b
square(3,4) #arguments
```

Sample Program (107) တွင် 3 ဆိုသည့် argument သည် a ဆိုသည့် parameter နှင့် သက်ဆိုင်ပြီး 4 ဆိုသည့် argument သည် b ဆိုသည့် parameter နှင့် သက်ဆိုင်သည်။ အထက်ပါ နည်းလမ်းသည် အစဉ်လိုက် နေရာယူသော positional နည်းလမ်းဖြစ်သည်။

```
def square(a,b,c):#parameters
   return a*b

a=square(3,4)
```

ယခုအတိုင်းရေးမည်ဆိုလျှင်တော့ square function ခေါ်တဲ့ နေရာမှာ error တက်မှာ ဖြစ်ပါတယ်။ အဘယ်ကြောင့်ဆိုသော် 3 သည် a သို့ သွားနိုင်သော်လည်း 4 သည် b or c ဘယ်ကို သွားရမလဲ ဆိုတာ python က မသိပါဘူး။

Sample Program(108)

```
def square(a,b,c):#parameters

  return a*b*c
print(square(1,2,3))
```

Sample Program(108)တွင် 1 သည် a သို့ pass မည်ဖြစ်ပြီး 2 သည် b သို့ pass လုပ်မည်။ ထိုနည်းတူ 3 သည်လည်း c သို့ pass လုပ်ပါမည်။ အကယ်လို့ 1 ကို c သို့ pass လုပ်စေလိုပြီး 3 ကို a သို့ pass လုပ်စေလိုလျှင် မည်သို့ ရေးရမည်နည်း။ ထိုအချိန်တွင် keyword arguments ကို အောက်ပါ အတိုင်း အသုံးပြုနိုင်ပါသည်။

Sample Program (109)

```
def square(a,b,c):#parameters
  return a*b*c

print(square(c=1,b=2,a=3))
```

အထက်ပါ နည်းလမ်းကို အသုံးပြုပြီး function တစ်ခုအတွင်းသို့ arguments များသည် မိမိတို့စိတ်ကြိုက် parameters အတိုင်းpass လုပ်နိုင်ပါသည်။

Unpacking

Unpacking or Iterable unpacking ဆိုတာ = ညီမျှခြင်း တစ်ခုရဲ့ ညာဘက်မှာရှိတဲ့ value(object) တွေကို ဘယ်ဘက်မှာရှိတဲ့ variable များဆီသို့ assign လုပ်ခြင်းဖြစ်ပါတယ်။ object သုံးခု နှင့် variable သုံးခုဖြစ်ရင်တော့ iterable unpacking ကို သုံးနိုင်သော်လည်း object တွေများနေပြီး variable တွေနည်းနေခဲ့မယ်ဆိုရင်တော့ Extended Unpacking ကို သုံးနိုင်ပါတယ်။ အခု ဖော်ပြမှာကတော့ Iterable unpacking ဖြစ်ပါတယ်။ ပထမဆုံးအနေဖြင့် tuple တစ်ခုကို စတင်ကြေငြာပါမယ်။ a = (1,2,3) ယခုပုံစံအတိုင်း ကြေငြာလိုက်တာနဲ့ tuple ဖြစ်သွားပါပြီ။ tuple ဟုတ်မဟုတ်ဆိုတာကို print(type(a)) ကိုရေးကာ စစ်ဆေးနိုင်ပါသည်။ သတိပြုရန် တစ်ချက်မှာ tuple တစ်ခုကို declaring လုပ်ရာတွင် ( ) parentheses ကြောင့် tuple ဖြစ်ရခြင်း မဟုတ်ပဲ 1,2,3 တို့ကြားတွင်ရေးထားသော comma (,) များကြောင့်သာ tuple ဖြစ်ရခြင်း ဖြစ်သည်။ tuple သင်ခန်းစာကို နောက်ပိုင်းမှာ အသေးစိတ် ဆက်လက်ဆွေးနွေးသွားပါမည်။

Sample Program (110)

```
>>> a,b,c=1,2,3
>>> a
1
>>> b
2
>>> c
3
>>> 
```

```
a,b,c=1,2,3
print(a)
print(b)

print(c)
```

Sample Program (110)အား run ကြည့်လျှင် a,b,c တို့ တန်ဘိုးသည် 1,2,3 ဖြစ်နေသည်ကို မြင်နိုင်ပါသည်။ ယခု program သည် 1,2,3 ဆိုသည့် tuple ထဲမှ a,b,c ဆိုသည့် variable များထဲသို့ data များ unpacking လုပ်ပြီး ထည့်ခြင်းဖြစ်ပါသည်။

Unpacking list to tuple

List တစ်ခု ထဲမှာ ရှိတဲ့ data များကို tuple တစ်ခုထဲသို့လည်း အောက်ပါအတိုင်း unpack လုပ်နိုင်သည်။

```
>>> (a,b,c)=[1,2,3]
>>> a
1
>>> b
2
>>> c
3
>>> 
```

```
(a,b,c)=[1,2,3]
print(a)
print(b)

print(c)

a,b,c = 10,'a',3.14
print(a)   #10
print(b)   #a

print(c)   #3.14
```

အခြားသော data type များကိုလည်း unpack လုပ်နိုင်ပါတယ်။ unpacking ကို သုံးပြီး variable value များကို swapping လုပ်ရာမှာ အရမ်းကိုအသုံးဝင်ပါတယ်။ ဥပမာ အနေဖြင့် a,b=10,20 တွင် a value သည် ပုံမှန်အားဖြင့် unpack လုပ်လျှင် 10 ဖြစ်ပြီး b value သည် 20 ဖြစ်သည်။ a value ကို 20 ဖြစ်စေလိုပြီး b value ကို 10 ဖြစ်စေလိုသော အခါမျိုးတွင် အချို့သော programming များ၌ variable တစ်ခုကို အစားထိုးပြီး ထို variable ထဲကို value ကို ချိန်းထည့်ခြင်းဖြင့် swapping လုပ်ကြသည်။ Python တွင်မူ အောက်ပါ အတိုင်း ရေးသားရုံဖြင့် swapping လုပ်နိုင်သည်။

```
a,b=10,20
a,b=b,a
print(a)   #20
print(b)   #10
```

ပုံမှန်အားဖြင့် ကြည့်လျှင် b value ကို a ထဲသို့ ထည့်လိုက်သည်ဟု ထင်ရသော်လည်း python programming တွင် ညီမျှခြင်းရဲ့ right hand sight မှာရှိသော , ခံပြီးရေးထားသော value များသည် tuple ဖြစ်သည်။ ဒုတိယ code line တွင် ဘယ်ဘက်မှာရှိသော a သည် ညာဘက်မှာ ရှိသော b ရဲ့ memory address ကို reference လုပ်ထားခြင်းဖြစ်ပြီး ဘယ်ဘက်မှာ ရှိသော b သည် ညာဘက်မှာ ရှိသော a ရဲ့ memory address ကို reference လုပ်ထားခြင်းဖြစ်သည်။

Unpacking String, Set, Dictionary

```
a,b,c = 'XYZ'
print(a)   #X
print(b)   #Y

print(c)   #Z
```

အထက်တွင် list and tuple တို့ကို unpack လုပ်ပြထားခဲ့ပြီး ဖြစ်ပြီး string သည်လည်း ထိုနည်းအတိုင်းပင်ဖြစ်သည်။ သို့သော် python programming တွင် set and dictionary တို့သည် unorder ဖြစ်သည်။ unorder ဆိုသည်မှာ set and dictionary ထဲမှာရှိသော value များကို index ကိုသုံးပြီး access မလုပ်နိုင်ပါ။

```
a={1,2,3,4}
print(a[0])
# Traceback (most recent call last):
#   File ".\app.py", line 2, in <module>
#   print(a[0])

# TypeError: 'set' object is not subscriptable
```

python programming တွင် set ကို curly bracket သုံးပြီးကြော်ငြာသည်။ set အကြောင်းကို နောက်ပိုင်း သင်ခန်းစာများတွင် အသေးစိတ်ဆွေးနွေးသွားပါမယ်။ a ဆိုသည့် set တစ်ခုကို ကြေငြာထားပြီး ထို set ထဲမှ index 0 ကိုထုတ်ကြည့်သည့်အခါ subscriptable ဆိုသည့် Type Error ကိုပြပါသည်။

```
a={'a':1, 'b':2 , 'c':3}

print(a[1])
# Traceback (most recent call last):
#   File ".\app.py", line 2, in <module>
#    print(a[1])
# KeyError: 1
a="ABC"

print(a[0]) #A
```

ထိုနည်းတူ dictionary သည်လည်း unorder ဖြစ်ပြီး Index ကို support မလုပ်ပါ။ Python programming တွင် dictionary ကို key value များဖြင့် ကြေငြာသည်။ ယခု

ကြေငြာထားသော dictionary တွင် a ,b ,c တို့သည် key များ ဖြစ်သည်။ ထို key value များနောက်၌ : colon ကိုရေးပြီး value များကို ရေးသားသည်။ dictionary a တွင် a[1] ဟုရေးကြည့်လျှင် error ပြနေသည်ကို မြင်နိုင်သည်။ String အဖြစ်ကြေငြာထားသည့် a တွင်မူ a[0] ဟု index 0 ကို ထုတ်ကြည့်သည့်အခါ 'A' ထွက်လာသည်ကို မြင်နိုင်သည်။ Python တွင် list , tuple , string တို့သည် set and dictionary နှင့် မတူပဲ order ဖြစ်ကြသည်။

ထို့ကဲ့သို့ set နှင့် dictionary တို့သည် order မဖြစ်သည့်အတွက် unpack လုပ်ရာတွင် အခက်ခဲရှိသည်။ Iterables နည်းလမ်းဖြင့်ထုတ်မည်ဆိုလျှင်တော့ data တွေ အစဉ်လိုက်ထွက်လာသည်။ သို့သော် ထိုနည်းလမ်းသည် unpacking မဟုတ်ပါ။

Sample Program (111)

```
ite={1,2,3,4,5,}
for e in ite:
   print(e)
# 1
# 2
# 3
# 4

# 5
```

အထက်ပါ program အတိုင်း iterable နည်းလမ်းဖြင့် for loop သုံးပြီးထုတ်လျှင် value များ အစဉ်လိုက်ထွက်လာသည်။

Sample Program (112)

```
ite={1,2,3,4,5,}
print(ite)
#{1,2,3,4,5,}
ite={'a', 'b' , 'c' , 'd'}
print(ite)

#{'c', 'a' , 'b' , 'd'}
```

အထက်ပါ program ကိုကြည့်ပါ။ ite ထဲမှ number များကို print ထုတ်ကြည့်သော အခါတွင် အစဉ်လိုက် ထွက်လာသော်လည်း ite ထဲသို့ စကားလုံးများထည့်ပြီး ထုတ်ကြည့်သောအခါတွင်မူ အစဉ်လိုက် ထွက်မလာတော့သည်ကို တွေ့နိုင်ပါသည်။

Sample Program (113)

```
a,b,c,d = {'A' , 'B' , 'C' , 'D'}

print(a)   #C
print(b)   #D
print(c)   #A

print(d)   #B
```

Sample Program (114)

```
d = {'A', 'B' , 'C' , 'D' , 'E'}
for i in d:
   print(d)
# {'E', 'C', 'A', 'D', 'B'}
# {'E', 'C', 'A', 'D', 'B'}
# {'E', 'C', 'A', 'D', 'B'}
# {'E', 'C', 'A', 'D', 'B'}
# {'E', 'C', 'A', 'D', 'B'}
a,b,c,d,e = d
print(a)   #E
print(b)  #C
print(c)   #A
print(d)   #D

print(e)  #B
```

Sample Program (113) တွင် unpacking လုပ်သောအခါတွင် a သည် ပထမဆုံး C အကြီးကို ထုတ်ပေး သော်လည်း b အလှည့်တွင် D အကြီးကို ထုတ်ပေးပါသည်။ set ထဲတွင် value များ မည်သို့ နေရာ ယူနေသလဲ ဆိုတာကို သိနိုင်ရန် ပထမဆုံး variable တစ်ခုထဲသို့ assign ထည့်ပါမည်။ ထို့နောက်မှ ထို variable အား iterable လုပ်ပြီး ထုတ်ကြည့်ပါမည်။ Sample Program (114) တွင်ကြည့်ပါ။ 'E', 'C', 'A', 'D', 'B အဖြစ် နေရာ ယူနေသည်ကို တွေ့ရပါမည်။ ထို့နောက်မှ'E', 'C', 'A', 'D', 'B အား a,b,c,d,e ထဲသို့ unpack လုပ်ပါမည်။ ယခုနည်းအတိုင်း unpack လုပ်မည်ဆိုလျှင်တော့ မိမိလိုသလို value and variable များကို ချိန်ညှိနိုင်ပါသည်။ a နေရာမှာ A ထားနိုင်သလို b နေရာတွင် variable ကိုနေရာ ချိန်းခြင်းဖြင့် B ကိုထားနိုင်သည်။ သတိပြုရန်မှာ ပထမဆုံး iterable သို့မဟုတ် print ထုတ်ကြည့်ရန်လိုအပ်သည်။ သို့မှသာ မိမိလိုသလို ချိန်ညှိနိုင်မည်ဖြစ်သည်။ အထက်တွင် ဖော်ပြထားသော နည်းလမ်း များ အတိုင်း dictionary ကိုလည်း unpack လုပ်နိုင်ပါသည်။

Extended Unpacking (Python 3.5 ထိ အနိမ့်ဆုံး လိုအပ်ပါတယ်)

Iterable unpacking သင်ခန်းစာမှာ ညာဘက်မှာ ရှိတဲ့ objects တွေဟာ ဘယ်ဘက်မှာ ရှိတဲ့ variable တွေနဲ့ အရေအတွက် တူနေရင်တော့ အဆင်ပြေပါတယ်။ သို့သော် objects တွေများနေခဲ့မယ်ဆိုရင်တော့ Extending Unpacking ကိုသုံးနိုင်ပါတယ်။ Extended Unpacking အား symbol အနေဖြင့် * (asterisk) ကို သုံးပါတယ်။ Extending Unpacking ကို = ရဲ့ left hand sight ရော right hand sight မှာပါ အသုံးပြုနိုင်ပါတယ်။ Extended Unpacking မှာ star တစ်လုံးတည်းကိုသာ သုံးလျှင် Dictionary ကို unpack လုပ်သောအခါ Key ကိုသာ ရပါသည်။ Key's value ကို မရပါ။ Dictionary တွင် Key တင်မကပဲ Key's value ပါ လိုချင်သောအခါတွင်မူ double star (**) ကို အသုံးပြုပါသည်။ သို့သော် သတိပြုရမည့် အရေးကြီးဆုံး အချက်မှာ ** operator ကို LHS (left hand sight) တွင် အသုံးပြုလို့ မရပဲ RHS(right hand sight) တွင်သာ အသုံးပြုလို့ ရပါသည်။

Example Program and Explanation

ပထမဆုံး အနေဖြင့် l = [1,2,3,4,5,6] ဟု list တစ်ခု တည်ဆောက်လိုက်ပါမည်။

List extended unpacking

Sample Program (115)

```
l=[1,2,3,4,5,6]
a,*b=l
print(a) #1
print(b) #[2, 3, 4, 5, 6]
```

a,*b=l တွင် 1 သည် a ထဲသို့ assignment လုပ်မည်ဖြစ်ပြီး ကျန်သော object များအားလုံးသည် b ထဲသို့ list တစ်ခုအနေဖြင့် assignment လုပ်မည် ဖြစ်သည် ။

String extended unpacking

Sample Program (116)

```
s='greenhackers'
a,*b=s
print(a) # g

print(b) # ['r', 'e', 'e', 'n', 'h', 'a', 'c', 'k', 'e', 'r', 's']
```

string များကိုလည်း အထက်ပါ နည်းလမ်းအတိုင်း extended unpacking လုပ်နိုင်သည်။

```
s='WinHtut'
a,*b,c=s
print(a) # W
print(b) # ['i', 'n', 'H', 't', 'u']

print(c) # t
```

Sample Program (116) အရ extended unpacking ကို အလည်မှာထားသောအခါတွင် a သည် ပထမဆုံးစာလုံးကိုယူပြီး c သည် နောက်ဆုံးစာလုံးကို ယူပါသည်။ *b သည် အလယ်မှ ကျန်စာလုံးများအားလုံးကို ယူပါသည်။ နောက်ထပ် variable d အပိုတစ်လုံးထည့်ပြီး စမ်းမည်ဆိုလျှင် a သည် ရှေ့ဆုံးစာလုံး၊ c သည် နောက်ဆုံး ရှေ့ကတစ်လုံး၊ d သည် နောက်ဆုံး စာလုံးကို ယူပြီး *b သည် အလယ်မှ ကျန်စာလုံးများအားလုံးကို ယူပါမည်။

Tuple extending unpacking

Sample Program (117)

```
t =('a', 'b', 'c')
a,*b=t
print(a) # a

print(b) # ['b', 'c']
```

tuple ကိုလည်း extended unpacking လုပ်နိုင်သည်။ သို့သော် 'b','c' သည် list တစ်ခု အနေဖြင့် b ထဲသို့ assignment လုပ်ပါသည်။

(*) ကို RHS (right hand sight) ဘက်မှာ ရေးသားပြီးလည်း သုံးနိုင်သည် ။ အောက်ပါ program တွင် list နှစ်ခုကို * သုံးပြီး တစ်ခုတည်းအဖြစ် ပေါင်းပြထားပါသည်

```
l1 = [1,2,3]
l2 = [4,5,6]
l=[*l1 , *l2]

print(l) #[1,2,3,4,5,6]
```

list နှင့် string တို့လည်း ယခု နည်းအတိုင်းသုံးပြီး ပေါင်းနိုင်ပါသည်။

list extended unpacking using tuple

Sample Program (118)

a,*b,(c,d,*e)=[1,2,3,'python'] တွင် a သည် 1 ဖြစ်ပြီး *b သည် 2,3 ကို list တစ်ခု အနေဖြင့် Output ပေးမည်။ ထို့ပြင် c သည် p ဖြစ်ပြီး d သည် y ဖြစ်ပါသည်။ ကျန်သောစာလုံး အားလုံးသည် *e ဖြစ်ပါသည်။

```
a,*b,(c,d,*e)=[1,2,3,'python']
print(a) # 1
print(b) # [2, 3]
print(c) # p
print(d) # y

print(e) # ['t', 'h', 'o', 'n']
```

indexing ပုံစံဖြင့်လည်း unpack လုပ်နိုင်ပါသေးသည်။

Sample Program (119)

```
l =[1,2,3,'python']

print(l[0], l[1:-1] , l[-1][0] , l[-1][1], l[-1][2:])  #( 1 [2, 3] p y thon)
```

Sample Program (119) output ကိုကြည့်လျှင် thon သည် string ပုံစံဖြစ်နေပါသေးသည် ။ list ပုံစံဖြင့် ရယူလိုပါက list function ကိုသုံးနိုင်ပါသည်။

Sample Program (120)

```
 l =[1,2,3,'python']

print(l[0], l[1:-1] , l[-1][0] , l[-1][1], list(l[-1][2:])) #( 1 [2, 3] p y ['t', 'h', 'o', 'n'] )
```

အထက်ပါ ပုံစံ အတိုင်း list function ကို ထည့်ရေးကြည့်မည်ဆိုလျှင် output သည် list ဖြစ်နေပါမည်။ သို့သော် Output အားလုံးသည် ( ) ထဲတွင် ရှိနေပြီး tuple ဖြစ်နေပါသည်။ ထို့ကြောင့် tuple ထဲမှာရှိသော elements များကို အောက်ပါအတိုင်း variable များကိုသုံးပြီး tuple ကို ဖြုတ်ထုတ်နိုင်ပါသည်။

Sample Program (121)

```
l =[1,2,3,'python']

# print(l[0], l[1:-1] , l[-1][0] , l[-1][1], list(l[-1][2:]))
```

```
a,b,c,d,e = l[0], l[1:-1] , l[-1][0] , l[-1][1], list(l[-1][2:])
print(a ,b ,c ,d ,e)
#output

# 1 [2, 3] p y ['t', 'h', 'o', 'n']
```

ထို့အပြင် list and set တို့ကိုလည်း unpack လုပ်နိုင်သည်။ သို့သော် set သည် unorder ဖြစ်သောကြောင့် set ထဲတွင်ရှိသော object များသည် list ကဲ့သို့ အစဉ်လိုက်ပြမည်မဟုတ်ပါ။ အောက်ပါ program ကိုကြည့်နိုင်ပါတယ်။

Program 1

```
l=[1,2,3]
s='XYZ'
ls=[*l,*s]
print(ls)
#output
#[1, 2, 3, 'X', 'Y', 'Z']
```

Program 2

```
l=[1,2,3]
s={'a', 'b', 'c'}
ls=[*l,*s]
print(ls)
#output
#[1, 2, 3, 'a', 'c', 'b']
```

Set နှစ်ခုကိုပေါင်းမည်ဆိုလျှင်လည်း extending unpacking ကိုသုံးနိုင်ပါသေးသည်။ example အနေဖြင့် s1={1,2,3} နှင့် s2={3,4,5} နှစ်ခုကို ပေါင်းမည်ဆိုလျှင် c={ *s1,*s2} ဟု ရေးသားနိုင်ပါသည်။ သို့မဟုတ် union function ကိုလည်း သုံးပြီးရေးသားနိုင်ပါသေးသည်။ c=s1.union(s2) ကို ယခုနည်းအတိုင်းလည်း ရေးသားနိုင်ပါသည်။

Sample Program (122)

```
s1 = {1,2,3}

s2 = {3,4,5}
c ={*s1 , *s2}
print(c)
print(s1.union(s2))
# {1, 2, 3, 4, 5}
# {1, 2, 3, 4, 5}
```

Sample Program (122) ကိုကြည့်ပါ။ Set နှစ်ခုအားပေါင်းမည်ဆိုလျှင်လည်း extending unpacking ကိုသုံးနိုင်ပါသေးသည်။ example အနေဖြင့် s1={1,2,3} နှင့် s2={3,4,5} နှစ်ခုကို ပေါင်းမည်ဆိုလျှင် c={ *s1,*s2} ဟု ရေးသားနိုင်ပါသည်။ သို့မဟုတ် union function ကိုလည်း သုံးပြီးရေးသားနိုင်ပါသေးသည်။ c=s1.union(s2) ကိုယခုနည်းအတိုင်းလည်း ရေးသားနိုင်ပါ သည်။

Sample Program (123)

```
s1 = {1,2,3}
s2 = {3,4,5}
s3 = {5,6,7}
s4 = {7,8,9}
```

```
print(s1.union(s2,s3,s4))

# {1, 2, 3, 4, 5, 6, 7, 8, 9}
```

သတိပြုရန် အချက်မှာ set များကို set{ } ပုံစံဖြင့် unpacking လုပ်ကာ ပေါင်းလျှင်သော်လည်းကောင်း သို့မဟုတ် union ကိုသုံးကာ ပေါင်းလျှင်သော်လည်းကောင်း တူညီသော elements များကို ဖြတ်ထုတ်ပစ်သည်။ ထို့သို့ မဖြစ်စေလိုပါက list[ ] ပုံစံဖြင့် unpacking လုပ်ခြင်းကို သုံးနိုင်ပါသည်။

Sample Program (124)

```
s1 = {1,2,3}

s2 = {3,4,5}
s3 = {5,6,7}
s4 = {7,8,9}
print(s1.union(s2,s3,s4))
total =[*s1,*s2,*s3,*s4]
print(total)
# {1, 2, 3, 4, 5, 6, 7, 8, 9}

# [1, 2, 3, 3, 4, 5, 5, 6, 7, 8, 9, 7]
```

set နှင့် union သုံးထားသော Output တွင် 1,2,3,4 ... ဟု အစဉ်လိုက်သွားသော်လည်း list ပုံစံဖြင့် unpacking လုပ်ထားသော output တွင်မူ 1,2,3,3,4,5,5..... ဟု အစဉ်လိုက်သွားသည်ကို တွေ့နိုင်ပါသည်။

Arbitrary Arguments

ကျွန်တော်တို့ ကြေငြာထားတဲ့ function တစ်ခုထဲကို arguments ဘယ်လောက် ဖြတ်သွားမည်ကို မသိသော အခြေအနေမျိုးတွေမှာဆိုရင် parameters တွေကို ကြိုတင် ကြေငြာထားဖို့ ခက်ခဲလှပါတယ်။ Python က ထိုသို့သောအခြေနေမျိုးတွေကို အကောင်းဆုံး ကိုင်တွယ်ထားပါတယ်။ အဲဒါကတော့ *args (arbitrary arguments) ဖြစ်ပါတယ်။ Example အနေဖြင့် myFun ဆိုသည့် function တစ်ခု တည်ဆောက်ပါမည်။ ထို function ထဲသို့ arguments ဘယ်လောက် ဖြတ်လာမည်ကို မသိသောကြောင့် parameter အနေဖြင့် *args လို့သာ ကြေငြာထားပါမည်။ def myFun(*args) ထို့နောက် ဖြတ်လာသော parameters များအား print(args) ဟုရေးကာ print ထုတ်လိုက်ပါမည်။ ထို့နောက် function ကို 1,2,3,4,5 ဆိုသည့် arguments များ ထည့်ပေးပြီး ခေါ်လိုက်ပါမည်။

Sample Program (125)

```
def myFun(*args):
   print(args)

myFun(1,2,3,4,5)

# (1,2,3,4,5)
```

Sample Program (125) အား run ကြည့်လျှင် output အနေဖြင့် (1,2,3,4,5) ကို ပြန်ရပါမည်။ output ကို သေချာကြည့်လျှင် tuple အနေဖြင့် ပြန်ပေးခြင်းဖြစ်သည်။ list အနေဖြင့် မဟုတ်ပါ။ myFun အား မည်သည့် arguments မှ မထည့်ပဲ myFun() ဟုခေါ်ကြည့်ပါက tuple တစ်ခု ရမည်ဖြစ်သည်။

Positional arguments and Arbitrary arguments

Positional arguments တွေနဲ့ arbitrary arguments နှစ်ခုလုံးကို သုံးပြီးတော့လည်း function တစ်ခုထဲကို pass လုပ်နိုင်ပါတယ်။ သို့သော် positional arguments တွေကို အရင်ရေးရမည် ဖြစ်ပြီး arbitrary argument နောက်တွင် positional argument ကိုထပ်ရေးလျှင် error တက်မည်ဖြစ်ပါသည်

Sample Program (126)

```
def myFun(a,b,*args):

  print(a)      #1
  print(b)      #2
  print(args)   #(3,4)

myFun(1,2,3,4)
```

Sample Program (126) ကို run ကြည့်လျှင် output အနေဖြင့် 1 and 2 သည် ပုံမှန်အတိုင်း ရမည်ဖြစ်ပြီး 3,4 သည် tuple တစ်ခုအနေဖြင့် ရမည်ဖြစ်ပါသည်။

Sample Program (127)

```
def myFun(a,b,*args,c):
  print(a)
  print(b)
  print(args)

myFun(1,2,3,4)
```

အထက်ပါ အတိုင်း arbitrary argument နောက်မှာ parameter တစ်ခု ထပ်ထည့်မည်ဆိုလျှင် error တက်မည်ဖြစ်ပါသည်။ နောက်ထပ် program တစ်ပုဒ်အနေဖြင့် len and sum function များကို အသုံးပြုပြီး average ရှာတဲ့ program တစ်ပုဒ်ရေးပါမည်။ len() function သည် အရေအတွက် ဘယ်လောက်ရှိသလဲ parameter ဘယ်နှစ်ခုရှိသလဲ ဆိုတာကို သိနိုင်သော function ဖြစ်ပြီး sum သည် ထို parameter အားလုံးကို ပေါင်းပေးသော function ဖြစ်သည်။ ဥပမာ အနေဖြင့် len(args) တွင် args ထဲတွင် ပါလာသော parameter အရေအတွက်ကိုပြန်ရမည်ဖြစ်သည်။ sum(args) ဟုသုံးလျှင် args ထဲတွင် ပါလာသော parameter အားလုံးပေါင်းခြင်းကို ရမှာ ဖြစ်ပါတယ်။

Sample Program (128)

```
def avg(*args):

  length= len(args)
  total= sum(args)
```

```
    return total/length
result=avg(1,2,3,4)

print(result) #2.5
```

Sample Program (128) ကို run ကြည့်လျှင် output အနေဖြင့် 2.5 ကို ရမည်ဖြစ်သည်။ သတိပြုရန် အချက်မှာ avg() function အား မည်သည့် arguments မှ မထည့်ပဲခေါ်မည်ဆိုလျှင် result=avg() Zero Division Error: ကို ရမည်ဖြစ်သည်။ အဘယ်ကြောင့်ဆိုသော် length သည် zero ဖြစ်နေသောကြောင့် total ကို မစားနိုင်သောကြောင့် ဖြစ်သည်။ ထို့ကြောင့် အထက်ပါ program အား if else ကို သုံးပြီး အောက်ပါအတိုင်း fix လုပ်နိုင်ပါသည် ။ length သည် zero ဖြစ်နေလျှင် return value အနေဖြင့် zero ကို ပြန်ပေးမည်ဖြစ်ပြီး length သည် zero မဟုတ်မှသာ total ကို length ဖြင့်စားပြီး return value ပြန်ပေးပါမည်။

Sample Program (129)

```
def avg(*args):
    length= len(args)
    total= sum(args)
    if length == 0:
        return 0
    else:
        return length/total
result=avg()

print(result)
```

အထက်ပါ program ကို run ကြည့်လျှင် output အနေဖြင့် zero ကိုသာ ရပါမည်။ function ထဲတွင် မည်သည့် arguments မျှမထည့်လိုက်သော်လည်း အရင် Program ကဲ့သို့ error တက်တော့မည့် မဟုတ်ပါ။

List Pass to Function

Python programming မှာ function တစ်ခုထဲကို list တစ်ခုက တိုက်ရိုက် ဖြတ်သွားလို့ ရသော်လည်း အခြားသော parameters များဖြင့် ဖြတ်လျှင်မူ missing argument required ဆိုတဲ့ error တတ်ပါတယ်။ သို့သော် extending unpacking ကိုသုံးပြီးတော့ ဖြတ်နိုင်ပါတယ်။ အောက်တွင်ပြထားသော code သည် Error တက်ကြောင်းပြထားခြင်း ဖြစ်သည် ။

Sample Program (130)

```
def myFun(a,b,c):

    print(a)
    print(b)
    print(c)
myList=[10,20,30]

myFun(myList)
```

Extended unpacking ကိုသုံးပြီး အောက်ပါအတိုင်းရေးမည် ဆိုလျှင်တော့ error တက်တော့မည် မဟုတ်ပဲ 10,20,30 တို့ကို unpack လုပ်ပြီးသား output များကို ရပါမည်။

Sample Program (131)

```
def myFun(a,b,c):
   print(a)
   print(b)
   print(c)
myList=[10,20,30]

myFun(*myList)
```

Sample Program (131)မှာ list ထဲတွင် elements သုံးခုသာရှိတဲ့အတွက် myFun(a,b,c) ထဲကို ဖြတ်သော အခါ error မတက်ခြင်းဖြစ်သည်။တကယ်လို့ list ထဲမှာ elements သုံးခု ထက်ပိုသွားသော အခါတွင်မူ ဖြတ်စရာ parameter လိုနေသည့်အတွက် error တက်ပါမည်။ ထိုသို့သော ပြဿနာကို ဖြေရှင်းရန် *arbitrary arguments ကို နောက်ဆုံးတွင် ထားပေးခြင်းဖြင့် ဖြေရှင်းနိုင် ပါသည်။

Sample Program (132)

```
def myFun(a,b,*c):
   print(a)      # 10
   print(b)      # 20
   print(c)      # (30, 40, 50, 60)
myList=[10,20,30,40,50,60]

myFun(*myList)
```

အထက်ပါ program အား run ကြည့်လျှင် a သည် 10 နှင့် b သည် 20 ကို သွားမည်ဖြစ်ပြီး ကျန်သည့် အားလုံးသည် *c ထဲသို့ သွားမည်ဖြစ်သည်။

Keyword Argument a Deeper Look

Keyword argument အကြောင်းကို အထက်ပိုင်းသင်ခန်းစာတွေမှာ introduction ရေးသားခဲ့ပြီးပါပြီ။ python programming မှာ function တစ်ခုရဲ့ arbitrary arguments နောက်မှာပါလာတဲ့ parameter တွေကို ပုံမှန်တိုင်း ခေါ်ခဲ့မိရင် error တက်ကြောင်းကို အထက်သင်ခန်းစာမှာ ဖော်ပြပြီးပါပြီ။ သို့သော် function လှမ်းခေါ်တဲ့ အချိန်မှာ keyword argument ကို သုံးပြီး arbitrary argument နောက်က parameter ကိုလှမ်းခေါ်နိုင်ပါတယ်။ ထိုနည်းအတိုင်း မခေါ်ခဲ့ရင်တော့ required keyword-only argument ဆိုသည့် error တက်မှာပါ။

Sample Program (133)

```
def myFun(a,b,*c,d):

   print(a)      # 1
   print(b)      # 2
```

```
        print(c)      # (3, 4, 5)
        print(d)      # 10

    myFun(1,2,3,4,5,d=10)
```

Sample Program (133)တွင် 1 and 2 သည် a နှင့် b သို့ သွားမည်ဖြစ်ပြီး keyword argument ဖြင့် ခေါ်ထားသော d မှလွဲကာကျန် number များသည် arbitrary argument ဖြစ်သည့် *c သို့ သွားမည်ဖြစ်ပါသည်။

Arbitrary argument နှင့် keyword argument ကိုတွဲသုံးလျှင် Python programming တွင် error အနည်းဆုံး ဖြစ်အောင် ရေးသားနိုင်ပါသည်။ ဥပမာအားဖြင့် function ထဲတွင် parameter တစ်ခုတည်းကို သုံးပြီးတော့လည်း ခေါ်နိုင်သည်။ အောက်ပါ program တွင် ကြည့်ပါ။

Sample Program (134)

```
def myFun(*c,d):
  print(c,d)

myFun(d='a')
```

Sample Program (134)အား run ကြည့်လျှင် output အနေဖြင့် empty tuple တစ်ခုနှင့် a ဆိုသည့် value တစ်ခုကို ပြန်ရလာပါမည်။

End of Positional Arguments (*)

Python programming မှာ function တစ်ခုကိုဖြတ်တဲ့အခါ ပုံမှန်အားဖြင့် positional အလိုက်ဖြတ်ပါတယ်။ သို့သော် positional arguments ဖြတ်ခြင်းကို လက်မခံချင်သော အခြေ အနေမျိုးမှာဆိုရင် (*) end of positional argument ကိုသုံးပါတယ်။ def myFun(* ,d) ဆိုပြီး define လုပ်ထားသည့် function မျိုးတွင်မည်သည့် positional arguments ကိုမှလက်ခံမည် မဟုတ်ပဲ keyword argument ဖြစ်သည့် d တစ်ခုတည်းကိုသာ လက်ခံမည်ဖြစ်ပါသည်။ * သည် end of positional argument ဖြစ်ပြီး သူ့ ရှေ့မှာ အခြားသော positional arguments များ ရှိသေးပါက လက်ခံပြီး သူ့နောက်က မည်သည့် positional arguments ကိုမှ လက်မခံပါ။

Sample Program (135)

```
def myFun(a,b,*,d):
  print(a,b,d)        # 1 2 a

myFun(1,2,d='a')
```

အထက်ပါ ပုံစံ အတိုင်းသုံးနိုင်သည်။ 1,2 သည် a နှင့် b ထိ ရပါသည်။ သို့သော် 1,2,3 ဖြစ်လာလျှင်တော့ မရတော့ပါ။ myFun ဆိုတဲ့ function ထဲတွင် positional arguments နှစ်ခုတည်းသာ ကြေငြာထားကြောင်းနှင့် သူ့ထဲသို့ နှစ်ခု ထပ် ပိုပြီး အဖြတ် မခံကြောင်း error log ကို မြင်ရပါမည်။ error sample program ကို အောက်တွင် ဖော်ပြထားပါသည်။

Error Sample Program (136)

```
def myFun(a,b,*,d):
```

```
    print(a,b,d)
myFun(1,2,3,d='a')
#output
# Traceback (most recent call last):
#   File ".\app.py", line 3, in <module>
#     myFun(1,2,3,d='a')

# TypeError: myFun() takes 2 positional arguments but 3 positional arguments (and 1 keyword-only argument) were given
```

**kwargs

Python Function ကို ကြေငြာရာမှာ နောက်ထပ် parameter အနေဖြင့်သုံးလို့ရတဲ့ **kwargs (keyword arguments) တစ်ခုရှိပါသေးတယ်။ အရင်သင်ခန်းစာမှာ ရေးသားခဲ့တဲ့ *args သည် arbitrary argument သည် data များအား tuple အနေဖြင့် ပြန်ပေးပါသည်။ **kwargs(keyword argument) သည် dictionary အနေဖြင့် ပြန်ပေးပါသည်။ **kwargs ပြီးသွားတဲ့အခါမှာလည်း positional arguments များထပ်ပါလို့မရပါဘူး။ def fun(**kwargs,not allowed) **kwargs လို့ ကြေငြာထားပြီး မည်သည့် arguments မှ ဝင်မလာသော အချိန်တွင်မူ empty dictionary ကို return ပြန်ပေးပါသည်။ function call လုပ်းခေါ်တဲ့နေရာမှာလည်း **kwargs နောက်မှာ အခြားသော positional arguments တွေကို ခွင့်မပြုပါဘူး။

Returning dictionary

Example 1

```
def myFun(**kw):
    print(kw)
myFun()
#output
#{} ဘာမှ မပါသော empty dictionary ကို ပြန်ပေးပါသည်
```

Example 2

```
def myFun(**kw):
    print(kw)    #{'a': 10, 'b': 20, 'c': 30} dictionary ကို return ပြန်ပေးပါသည်။

myFun(a=10,b=20,c=30)
```

End of positional argument and keyword argument

End of positional argument ဖြစ်တဲ့ * နဲ့ keyword argument ဖြစ်တဲ့ ** တို့ကို နှစ်ခု ကပ်ပြီး သုံးလို့ မရပါဘူး။ သုံးမယ်ဆိုလျှင် ကြားထဲမှာ name argument တစ်ခုရေးပေးရပါမယ်။ ထိုသို့ မထည့်ပေးလျှင် named arguments must follow bare * ဆိုတာ error ရပါလိမ့်မယ်။

Sample Program (137)

```
def myFun(a,b,*,**kw):
    print(kw)

myFun(a=10,b=20,c=30)
```

Sample Program (137) အတိုင်းရေးလျှင် error တက်မည်ဖြစ်ပါမည်။ ထို့ကြောင့် end of positional argument နှင့် keyword argument များကို အောက်ပါ နည်းလမ်းအတိုင်းတွဲဖက်သုံးရပါမည်။

Sample Program (138)

```
def myFun(a,b,*,na,**kw):
    print(a)      # 10
    print(b)      # 11
    print(na)     # 101
    print(kw)     # {'c': 20}

myFun(10,11,c=20,na=101)
```

Sample Program (138) function call ခေါ်ရာ၌ na သည် name argument ဖြစ်သည့် positional argument များအတိုင်းအစဉ်လိုက်ထားစရာမလိုပဲ မိမိအဆင်ပြေသလိုထားနိုင်ပါသည်။

Simple Function Timer

Function call တွေ အားလုံးခေါ်ပြီးတဲ့ ကြာချိန်ကို တွက်တဲ့ program တစ်ပုဒ်ရေးသားပါမည်။ ယခု Program တွင် time ဆိုသည့် modules ကိုခေါ်သုံးပါမည်။ ပထမဆုံး function တစ်ခု တည်ဆောက်ပါမည်။ ထို function ထဲတွင် ပထမဆုံး parameter အနေဖြင့် function တစ်ခု ဖြတ်မည်ဖြစ်ပြီး ဒုတိယ နှင့် တတိယသည် arbitrary argument နှင့် keyword argument တို့ ဖြတ်ပါမည်။ ပြီးလျှင် loop ပတ်ရန် rep ဆိုသည့် keyword only argument တစ်ခု ဖြတ်ပါမည်။

Sample Program (139)

```
1. import time
2. def myFun(fn,*args,rep=1,**kwargs):
3.     start = time.perf_counter()
4.     for i in range(rep):
5.     fn(*args,**kwargs)
6.     end = time.perf_counter()
7.     avTime = (end-start)/rep
8.     fn(avTime)
9.     myFun(print,1,2,3,sep=' - ',end=' *** \n',rep=5)
#output
# 1 - 2 - 3 ***
# 1 - 2 - 3 ***
# 1 - 2 - 3 ***
```

```
# 1 - 2 - 3 ***
# 1 - 2 - 3 ***

# 0.00016967999999999983
```

Sample Program (139)တွင် line 1 သည် function များကို စခေါ်တဲ့ အချိန်နဲ့ ခေါ်ပြီးလို့ ဆုံးသွားတဲ့ အချိန်တွေကို သိနိုင်ဖို့ perf_counter() ဆိုတဲ့ function ကိုသုံးရပါတယ်။ထို function သည် time module ကို မိမိတို့ program တွင် ခေါ်သုံးမှသာ သုံးနိုင်ပါသည်။ line 3 တွင် starting time ကို မှတ်ရန် start ဆိုသည့် variable ကို တည်ဆောက်ထားပြီး ထို variable ထဲသို့ perf_counter() မှရသော အချိန်ကို မှတ်ထားပါသည်။ line 4 တွင် for loop ကိုသုံးထားပြီး ထို loop တွင် range() function ကိုသုံးထားပါသည်။ range() function သည် သူ့ထဲတွင် ပါလာသော Parameter အကြိမ်အရေအတွက်အတိုင်း အလုပ်လုပ်ပါသည်။ Line 5 တွင် fn သည် function ဖြစ်သည်။ ထို fn နေရာတွင် print function ဝင်ရောက်လာမှာ ဖြစ်ပါတယ်။ line 6 တွင် program က အချိန်ကို perf_counter() fun ကိုသုံးကာ နောက် တစ်ကြိမ်ထပ်ယူလိုက်ပြီး end ဆိုသည့် variable ထဲသို့ သိမ်းထားပါသည်။ ယခု program တွင် perf_counter() function ကို နှစ်ခါသုံးထားပါသည်။ ပထမ တစ်ခါသည်လည်း အချိန်တစ်ခုကိုယူပြီး ဒုတိယတစ်ခါ သည်လည်း အချိန်တစ်ခုကို ယူပါသည်။ ပထမတစ်ခုအချိန်သည် function call မခေါ်ခင် အချိန်ဖြစ်ပြီး ဒုတိယတစ်ခါသည် function call ခေါ်ပြီးချိန်ဖြစ်ပါသည်။ ပထမအချိန်ထဲက ဒုတိယကြာချိန်ကို နုတ်ရင် ကြာချိန်ရလာပါမည်။ ထိုကြာချိန်အား အကြိမ်အရေအတွက်ဖြင့် ပြန်စားလျှင် average ကြာချိန်ပြန်ရပါမည် ထိုအကြောင်းအရာကို Line 7 တွင်ရေးထားပါသည်။ line 8 သည် average time ကို ပြန်ထုတ်ပြခြင်းဖြစ်သည်။ line 9 သည် လိုအပ်သော argument များ ထည့်ပြီး function call ခေါ်ခြင်းဖြစ်သည်။

Program output ရဲ့ နောက်ဆုံးတွင် ဖော်ပြထားသော number များသည် average time ဖြစ်သည်။ ထိုအချိန်သည် မိမိတို့သုံးထားသော environment or version ပေါ်မှာ မူတည်ပြီး ကွဲပြားနိုင်ပါသည်။ ပထမတစ်ကြိမ် run သောအချိန်နှင့် ဒုတိယတစ်ကြိမ် runသော ကြာချိန်တို့မှာလည်း ကွဲပြားနိုင်ပါသည်။

Default Values

Default Values problem ကိုဖြေရှင်းရန် sample program တစ်ပုဒ် စတင်ရေးသားပါမည်။ ယခု program တွင် date and time ကိုလိုချင်တဲ့အတွက် Python programming ရဲ့ datetime module ကို ခေါ်သုံးပါမည်။ အခြားသော c/c++ programs များတွင် library or module များကိုခေါ်သုံးချင်ပါက #include ကိုသုံးပြီး node.js တွင်မူ require ကိုသုံးပါသည်။ Python programming တွင်မူ from datetime import datetime ဟုသုံးပါသည်။ from နောက်မှ စာသားသည် module name ဖြစ်သည်။ python တွင် datetime module ထဲမှ utcnow() ဆိုသည့် function ကိုခေါ်သုံးလျှင် ယခုလက်ရှိ နာရီ အချိန် နေ့ရက် ခုနှစ်များကို ရရှိပါသည်။

Sample Program (140)

```
from datetime import datetime
time= datetime.utcnow()      #object

print(time)      #string representation
```

အထက် Sample Program (140)ကို run ကြည့်ပါက 2020-05-07 03:41:28.255490 date and time ကိုရပါမည်။ program ကို နောက်ထပ်တစ်ကြိမ် ထပ် run ကြည့်ပါက မတူညီသည့်အချိန်ကို ထပ်ရပါမည်။ datetime.utcnow() ကို function parameter အနေဖြင့် အောက်ပါအတိုင်း သုံးပါမည်။

Sample Program (141)

```
from datetime import datetime
def time_fun(msg,*,dt=datetime.utcnow()):
   print('{0}:{1}'.format(dt,msg))

time_fun('Text message1',dt='2019-09-22 00:00:12.0000')
```

Sample Program (141)ကို run ကြည့်ပါက argument အနေဖြင့် ထည့်ပေးလိုက်သော time and date များကိုပြန်လည်ရရှိပါမည်။အထက်တွင် ဖော်ပြထားသော program တွင် datetime.utcnow() function သည် time_fun() မှ ထည့်ခေါ်လိုက်သော keyword only argument အတိုင်း ပြန်လည်ဖော်ပြပေးပါသည်။ အကယ်လို့သာ time_fun() ဆိုသည့် function call ခေါ်သည့်နေရာတွင် Text message ဆိုသည့် positional argument တစ်ခုတည်းကိုသာ ထည့်ပေးလိုက်ပါက Text message 1 ကိုလည်း ပြန်လည်ဖော်ပြပေးမည်ဖြစ်ပြီး ယခုလက်ရှိ အချိန်နှင့် နေ့ရက်တို့ကိုလည်း ပြန်လည်ဖော်ပြပေးပါမည်။ အောက်ပါ program တွင် လေ့လာနိုင်သည်။

Sample Program (142)

```
from datetime import datetime
def time_fun(msg,*,dt=datetime.utcnow()):
   print('{0}:{1}'.format(dt,msg))
time_fun('Text message1')
#output

# 2020-05-07 03:43:14.667376:Text message1
```

အထက်ပါ program တွင် time_fun('Text message 2') ဟုရေးပြီး program ကို ပြန် run ကြည့်ပါ။ output တွင် date and time တို့သည် လက်ရှိ current အချိန်ကို ဖော်မပြပဲ အရင်တစ်ခေါက် run ထားသော date and time ကိုသာ ပြန်လည်ဖော်ပြပါသည်။ ထိုအကြောင်းခြင်းအရာကို default values problem ဟုခေါ်ဆိုခြင်းဖြစ်သည်။ ထိုပြဿနာဖြစ်ပေါ်လာရခြင်းမှာ time_fun ဆိုသည့် function ကို စတင်ကြေငြာလိုက်ကတည်းက datetime.utcnow() နေရာတွင် value တစ်ခုကို create လုပ်ပြီးဖြစ်သည်။ ထိုvalue သည် lifetime ရှိနေသောကြောင့် ပြန်run သောအခါတွင်လည်း အရင် value ကိုသာ ပြန်ရနေခြင်းဖြစ်သည်။ ထို default values problem ကို ဖြေရှင်းရန် function ကြေငြာသောအချိန်၌ dt ဆိုသည့် keyword argument နေရာတွင် မည်သည့် value မှ assign မလုပ်ပဲ None ဆိုပြီးသာ ကြော်ငြာသင့်သည်။ ပြီးမှ ထို function body ထဲတွင် dt ထဲသို့

မိမိထည့်ချင်သော date and time များကို assign လုပ်ရပါမည်။ sample program ကိုအောက်တွင် ဖော်ပြထားသည်။

Sample Program (143)

```
from datetime import datetime

def time_fun(msg,*,dt=None):
  if not dt:
    dt=datetime.utcnow()
  print('{0}:{1}'.format(dt,msg))
time_fun('Text message solved 1')
#output

# 2020-05-07 03:44:38.539994:Text message solved 1
```

Sample Program (143) တွင်ရေးသားထားသော None သည် None datatype ဖြစ်ပြီး မည်သည့် value မှ assign မလုပ်လိုသောအခါတွင် အသုံးပြုပါသည်။ none သည် null ဖြစ်ပြီး မည်သည့် value မှ မဟုတ်သလို 0 zero လည်း မဟုတ်ပါ၊ Boolean အနေဖြင့် False လည်း မဟုတ်ပါ။ if not dt ဆိုလိုသည်မှာ dt သည်ဘာမှမဟုတ်ဘူးဆိုလျှင် သူ့အောက်က code များကို အလုပ်ဆက်လုပ်ပေးပါသည်။ တကယ်လို့သာ time_fun() ထဲတွင် dt=1 ဟုထည့်ပေးလိုက်ပါက return value အနေဖြင့် date and time ကိုပြန်မရတော့ပဲ 1 ကိုသာပြန်ရပါမည်။ အဘယ်ကြောင့်ဆိုသော dt သည် 1 ဖြစ်သွားသောကြောင့် ဖြစ်သည်။ သတိပြုရန်မှာ dt သည် မည်သည့် value မှ မဟုတ်မှသာလျှင် dt ထဲသို့ date and time ကို assign လုပ်မည်ဖြစ်သည်။

Parameter Defaults problem

Parameter Defaults problem ကို နားလည်ရန် ပထမဆုံး shop ဆိုသည့် function တစ်ခု တည်ဆောက်ပါမည်။ ထိုfunction ထဲတွင် parameter အနေဖြင့် ပစ္စည်း name၊ ပစ္စည်းအရေအတွက် quantity၊ ပစ္စည်းအမျိုးအစား unit၊ နှင့် ပစ္စည်းများအားလုံးကိုသိမ်းရန် grocery_list ဆိုသည့် list တစ်ခုကို ထည့်ထားပါမည်။ function body ပိုင်းတွင် parameter အနေဖြင့် ဝင်လာသော name, quantity, unit တို့ကို grocery_list ဆိုသည့် list ထဲသို့ ထည့်လိုက်ပါမည်။ ထို့နောက် return value အနေဖြင့် grocery_list ကို ပြန်ပေးပါမည်။ ဝင်လာတဲ့ parameter တွေကို parameter list ထဲကိုပဲ ပြန်ထည့်လိုက်တာပါ။ function ထဲတွင် list ကို parameter အနေဖြင့်သုံးထားသည့်အတွက် ထို function ထဲသို့ list များ ဖြတ်ရန် လိုအပ်ပါသည်။ ထို့ကြောင့် store1 and store2 ဆိုသည့် list နှစ်ခုကို တည်ဆောက်ပါမည်။ ပြီးလျှင် လိုအပ်သော parameter များထည့်ပြီး shop function ကိုခေါ်ပါမည်။ Sample program ကို အောက်တွင် ဖော်ပြထားသည်။

Sample Program (144)

```
1. def shop(name,quantity,unit,grocery_list):
2.     grocery_list.append("{0} {1} {2}".format(name,quantity,unit))
3.     return grocery_list
4. store1=[]
5. store2=[]
6. shop('banana',2,'units',store1)
7. shop('milk',1,'liter',store2)
8. print(store1)   # ['banana 2 units']
9. print(store2)   # ['milk 1 liter']
```

Sample Program (144) အား run ကြည့်လျှင် မိမိတို့ ထည့်ပေးလိုက်သော parameter များအတိုင်း အောက်ပါတို့ကို ရရှိပါမည်။

```
['banana 2 units']
['milk 1 liter']
```

Line number 6 shop function call တွင် normal arguments သုံးခုနှင့် list argument တစ်ခု ထည့်ပေးလိုက်သည်။ ထို့နောက် name , quantity and unit တို့သည် format အတိုင်း grocery_list ဆိုသည့် list ထဲသို့ အစဉ်တိုင်း append (ထပ်ထည့်) ပါသည်။ ပြီးနောက် grocery_lsit ကို return ပြန်ပေးပါသည်။ သတိပြုရန်မှာ grocery_list သည် store1 ဆိုသည့် argument မှ လာခြင်းဖြစ်သည့်အတွက် grocery_list ထဲသို့ထည့်လိုက်သော data များသည် store1 ဆိုသည့် list ထဲသို့ ရောက်ရှိသွားပါသည်။ print(store1) ဟုရေးလိုက်သောအခါ output အနေဖြင့် store1 ထဲတွင် append လုပ်ထားသော data များကို list အနေဖြင့် ပြန်လည်ဖော်ပြ ပေးခြင်းဖြစ်သည်။ ထိုနည်းတူပင် line number 7 တွင် milk,1,liter တို့သည်လည်း format အတိုင်း grocery_list ထဲသို့ append လုပ်ပါသည်။ ယခု တခါတွင် grocery_list သည် store2 ဖြစ်သည့်အတွက် store2 ကို print ထုတ်လိုက်သောအခါတွင် milk,1,liter တို့ကို list တစ်ခု အနေဖြင့် မြင်ရခြင်းဖြစ်သည်။ program ကို အောက်ပါအတိုင်း 2nd time implementation ထပ်လုပ်ကြည့်ပါ မည်။

Sample Program (145)

```
1. def shop(name,quantity,unit,grocery_list=[]):
2.     grocery_list.append("{0} {1} {2}".format(name,quantity,unit))
3.     return grocery_list
4. store1=shop('python',1,'book')
5. shop('java',1,'book',store1)
6. print(store1)   #['python 1 book', 'java 1 book']
```

line number 4 တွင် shop function ကို parameter သုံးခုထည့်ပြီး လှမ်းခေါ်ပါသည်။ ထို့နောက် function မှ return ပြန်လာသော value ကို store1 ထဲတွင် သိမ်းထားပါသည်။ ထို့ကြောင့် store1 ထဲတွင် python 1 book တို့ရှိနေပါမည်။ line number 5 တွင် shop function ကို နောက်တစ်ခါ ထပ်ခေါ်ပါသည်။ ယခုအခါတွင်မူ store1 ဆိုသည့် list တစ်ခုပါ

ထပ်ပေးလိုက် ပါသည်။ store1 ဆိုသည့် List ထဲတွင် line 4 က python 1 book တို့ ရှိနှင့်နေပြီး နောက်တစ်ခေါက် function call ခေါ်သောအခါတွင် ပါဝင်သွားသော parameter တို့သည် နဂိုမူရင်းရှိပြီးသား data နောက်တွင် နေရာယူမည် ဖြစ်ပါသည်။ ထို့ကြောင့် အထက်ပါ program ကို run ပြီးသောအချိန်တွင် store1 ထဲ၌ python 1 book java 1 book တို့ output အနေဖြင့် ထွက်လာသည်ကို မြင်ရပါသည်။ program ကို 3rd time implementation ထပ်လုပ်ကြည့်ပါမည်။

Sample Program (146)

```
1. def shop(name,quantity,unit,grocery_list=[]):
2.     grocery_list.append("{0} {1} {2}".format(name,quantity,unit))
3.     return grocery_list
4. store1=shop('python',1,'book')
5. shop('java',1,'book',store1)
6. store2=shop('c++',1,'book')
7. print(store1)
8. print(store2)
```

Line number 6 and 8 တို့ကို ထပ်ထည့်လိုက်ပါသည်။ အထက်ပါ အတိုင်းရေးပြီး program ကို run ကြည့်လျှင် အောက်ပါ အတိုင်းရပါမည်။

```
['python 1 book', 'java 1 book', 'c++ 1 book']

['python 1 book', 'java 1 book', 'c++ 1 book']
```

Line number 6 တွင် shop function ကို လှမ်းခေါ်သောအချိန်၌ c++ 1 book တို့ကို parameter အနေဖြင့် ထည့်ပေးလိုက်သော်လည်း line number 8 တွင် print ထုတ်သောအခါ၌ အရင် run ထားသော data များပါ output အနေဖြင့် ပြန်လည်ပါဝင်လာသည်ကို တွေ့ရသည်။ အဘယ်ကြောင့်ဆိုသော် shop ဆိုသည့် function ကို ခေါ် လိုက်သည်နှင့် တပြိုင်နက် default list တစ်ခုကို create လုပ်သောကြောင့်ဖြစ်သည်။ grocery_list နေရာတွင် list တစ်ခုကို ထည့်ခေါ်သည် ဖြစ်စေ မခေါ်သည်ဖြစ်စေ list တစ်ခုကို create လုပ်ပါသည်။ program ပြီးဆုံးသော အခါ၌ store1 နှင့် store2 တို့အား memory address တစ်ခုတည်း၌ data များကို သိုလှောင်ပါသည် ထို့ကြောင့် store1 and 2 တို့သည့် memory address တူနေသောကြောင့် data များလည်း တူနေကြသည်။ store1 and 2တို့ memory address တူမတူစစ်ရန် အောက်ပါ အတိုင်း code အနည်းငယ်ထပ်ထည့်လိုက်လျှင် သိနိုင်သည်။

```
if store1 is store2:

  print("same")
```

အထက်ပါ ကဲ့သို့ data များတူညီနေသော ပြဿနာကို ဖြေရှင်းရန် None ကို သုံးပြီး function body ပိုင်းရောက်မှ grocery_list ကိုlist တစ်ခု ဖန်တီးရမည်။ မိမိတို့ လိုချင်သည်မှာ function call ကို ခေါ်သောအခါ၌ list ကိုထည့်ပေးလိုက်လျှင် list တစ်ခုကို မဖန်တီးတော့ပဲ if statement ကို ကျော်ကာ parameter အနေဖြင့် ဝင်လာသော list ထဲတွင် data များကို ပေါင်းကာ Output အနေဖြင့် ပြရမည်။ function call ခေါ်ရာ၌ မည်သည့် list မှ မထည့်ပေးလိုက်လျှင် list ကို if statement အောက်တွင်ဖန်တီးပြီး ဝင်လာသော parameter

များကို list တစ်ခုဖြင့် return value ပြန်ပေးရမည်။ ထိုသို့ program ကောင်းမွန်စွာ အလုပ်လုပ်နိုင်ရန် parameter list နေရာ၌ None ကိုသုံးရမည်။ None ကိုသုံးခြင်းသည် ရှေ့ပိုင်း program မှာကဲ့သို့ အလိုအလျောက် create လုပ်ခြင်းကို ကာကွယ်နိုင်သည်။ Sample program ကို အောက်တွင် ဖော်ပြထားပါသည်။

Sample Program (147)

```
def shop(name,quantity,unit,grocery_list=None):
   if not grocery_list:
      grocery_list=[]
   grocery_list.append("{0} {1} {2}".format(name,quantity,unit))
   return grocery_list
store1=shop('python',1,'book')
shop('java',1,'book',store1)
store2=shop('c++',1,'book')
print(store1)
print(store2)
if store1 is store2:
   print("same")
else:
   print("not same")
#output
# ['python 1 book', 'java 1 book']
# ['c++ 1 book']

# not same
```

Memory address များလည်း တူတော့မည် မဟုတ်သောကြောင့် not same ဆိုသည့် output ကိုသာ ရပါမည်။ အဘယ်ကြောင့်ဆိုသော် value များ မတူတော့သောကြောင့်ဖြစ်သည်။

Docstrings

Docstrings ဆိုတာ Document Strings ကိုဆိုလိုခြင်းဖြစ်ပါတယ်။ Docstrings ကို ဘယ်နေရာမှာသုံးသလဲ ဆိုရင် python ရဲ့ built in function ဖြစ်တဲ့ print function အကြောင်းကို သိစေရန် help(print) လို့ရေးကြည့်ပါ။ အောက်ပါအတိုင်း စာသားများပေါ်လာပါမည်။ထိုစာသား များသည် docstrings ဖြစ်ပါသည်။

```
help(print)

# PS C:\Users\USER\Desktop\Book> python .\app.py
# Help on built-in function print in module builtins:
# print(...)
#    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)
#    Prints the values to a stream, or to sys.stdout by default.
```

```
#    Optional keyword arguments:
#    file:  a file-like object (stream); defaults to the current sys.stdout.
#    sep:   string inserted between values, default a space.
#    end:   string appended after the last value, default a newline.

#    flush: whether to forcibly flush the stream.
```

အထက်တွင်ပေါ်လာသော အကြောင်းအရာများသည် print အတွက် ရေးထားသော docstrings များဖြစ်ပြီး ထိုနည်းတူ int အကြောင်းကိုသိချင်ရင်လည်း help(int) လို့ ရိုက်လိုက်ရင် class int နဲ့ ပတ်သက်သော အကြောင်းအရာများကို မြင်ရပါမည်။ built ပါသော function များမှာ ထိုကဲ့သို့ docstrings များရှိသကဲ့သို့ မိမိတို့ ဖန်တီးလိုက်သော module , function , class or method တွေမှာလည်း docstrings တွေရှိသင့်ပါတယ်။ Function or Class တွေရဲ့ docstrings တွေကို ဖန်တီးတဲ့အချိန်မှာ header ပိုင်းပြီးသည်နှင့် တပြိုင်နက် body ပိုင်းရဲ့ ထိပ်ဆုံးမှာ ရေးသားရပါမည်။

Sample Program (148)

```
def myFun( x , y):

  """ return x*y"""
  return x*y

help(myFun)
```

အထက်ပါအတိုင်း program ကိုရေးပြီး run ကြည့်မည်ဆိုလျှင် အောက်ပါအတိုင်း output များကို ရရှိမှာ ဖြစ်ပါတယ်။

```
Help on function myFun in module __main__:
myFun(x, y)
  return x*y
```

Docstrings ကိုရေးတဲ့အခါမှာ triple quotes ကို သုံးသင့်ပါတယ်။ Triple quotes ကိုသုံးခြင်းအားဖြင့် Docstrings ထဲမှာ တခြားသောစာသားများကိုလည်း ထပ်ထည့်နိုင်ပါတယ်။ တစ်ခု သတိပြုရန်မှာ Docstrings သည် string ဖော်ပြချက်ဖြစ်ပါတယ်။ comment မဟုတ်ပါဘူး python programming တွင် comment သည် # sign နှင့် ရေးရပြီး program က comment ကိုလုံးဝ မသိပါဘူး။ သို့သော် docstring သည် triple quotes နဲ့ ရေးပြီး program က သိပါတယ်။ ထို့ကြောင့် အထက်ပါ program တွင် myFun ဆိုသည့် function ထဲ၌ docstrings များကို ရေးထားပြီး help(myFun) ဟုရေးလိုက်သောအခါ myFun ထဲမှာ ရေးထားသော docstrings များကို ပြန်လည်ဖော်ပြခြင်း ဖြစ်ပါတယ်။ အောက်ပါ အတိုင်း program ရေးပြီး run ကြည့်ပါ #နောက်တွင် ရေးထားသော comment သည် output တွင် ပြန်ပါလာမည် မဟုတ်ပါ။

Sample Program (149)

```
def myFun( x , y):
  #This is comment
  'This is first line'
```

```
    """ return x*y"""
    return x*y

help(myFun)
```

Sample Program (149) ကို run ပြီးသောအချိန်တွင် return x*y ဆိုသည့် docstring လည်း ပြန်လာမည်မဟုတ်ပါ။ အဘယ်ကြောင့်ဆိုသော် This is first line ဆိုသည့် စာသားကို အရင် ရေးထားသောကြောင့် ဖြစ်သည်။ ထို့ကြောင့် မိမိ ရေးချင်သော docstrings သည် first line ဖြစ်ရပါမည်။ docstring များကို __doc__ ကိုသုံးပြီးတော့ ကြည့်နိုင်ပါတယ်။ myFun.__doc__ လို့ရေးမယ်ဆိုရင် docstrings များကို တွေ့ရပါတယ်။

Sample Program (150)

```
def myFun( x , y):

    """documentation for myFun"""
    return x*y

print(myFun.__doc__)
```

Function Annotations

Functions တွေရဲ့ body ပိုင်းမှာ document တွေ ဖော်ပြဖို့အတွက် docstrings ကိုသုံးပြီး function ရဲ့ parameter တွေနဲ့ return တွေကို document အနေနဲ့ ဖော်ပြဖို့အတွက် ဆိုရင်တော့ annotations ကိုသုံးပါတယ်။ Python programming ရဲ့ features အသစ်တွေကို အမြဲ ဖော်ပြတဲ့ PEP (Python Enhancement Proposal) ရဲ့ 3017 မှာ Annotations ရေးသားပုံတွေကို ဖော်ပြထားပါတယ်။ Python 2 ရဲ့ external libraries တွေမှာဆိုရင်တော့ annotations ကို သုံးနေပြီးသားဖြစ်ပြီး Python 3 မှာဆိုရင်တော့ PEP 3107 မှာ စတင်ဖော်ပြခဲ့တာပါ။

```
def myFun(a: <expression> , b: <expression>) -> <expression>:
```

a နောက်တွင်ရေးထားသော expression နေရာတွင် မိမိ ရေးသားလိုသော document များကိုရေးသားနိုင်ပြီး b နောက်တွင်လည်း ထို parameter အတွက် document များကို ရေးသားနိုင်သည်။ annotation တွင် return value အတွက် -> ကိုသုံးပါသည်။ ထို -> တွင်ရေးထားသော expression နေရာတွင် ယခု function အတွက် return ပြလိုသော document များကိုရေးနိုင်သည်။ ဥပမာ အနေဖြင့် a ရဲ့ expression နေရာတွင် a သည် string ဖြစ်ကြောင်း ဖော်ပြနိုင်သလို b ရဲ့ expression နေရာတွင်လည်း b သည် integer ဖြစ်ကြောင်း ဖော်ပြနိုင်သည်။ ထိုနည်းတူ -> နောက်က expression နေရာတွင်လည်း string ကို return ပြန်မည်ဖြစ်ကြောင်း မိမိ စိတ်ကြိုက်ဖော်ပြနိုင်ပါတယ်။

Sample Program (151)

```
def myFun( x:'annotation of x ' , y:'annotation of y')->'annotation of function
return':
    return x*y

print(myFun.__annotations__)
```

Sample Program (151) ကို run ကြည့်လျှင်အောက်ပါအတိုင်း output များကိုရပါမည်။ annotations ကို ဖော်ပြလိုသောအခါများတွင်မူ __annotations__ ကိုသုံးပါတယ်။

```
#output
{'x': 'annotation of x ', 'y': 'annotation of y', 'return': 'annotation of function return'}
```

Output များကို ကြည့်လျှင် Dictionary ပုံစံဖြင့် ရလာသည်ကို မြင်ရပါမည်။ key and value များ တွဲလျက်ပါဝင်လာပါသည်။ အထက်တွင်မူ x သည် key ဖြစ်ပြီး annotation of x သည် value ဖြစ်သည်။အထက်တွင် ဖော်ပြထားသော program သည် function parameter နှင့် return အတွက်သာ annotation ဖော်ပြထားခြင်းဖြစ်ပြီး docstrings အတွက် မပါဝင်ပါ။ annotations နှင့် docstring ကို ပေါင်းပြီးအောက်ပါ အတိုင်းရေးသားနိုင်ပါသေးသည်။

Sample program (152)

```
def myFun( x:'annotation of x ' , y:'annotation of y')->'annotation of function return':
   """This is documentation for myFun"""
   return x*y
print(myFun.__doc__)
print(myFun.__annotations__)
```

```
#output

#This is documentation for myFun

# {'x': 'annotation of x ', 'y': 'annotation of y', 'return': 'annotation of function return'}
```

Sample Program နောက်တစ်ခုအနေဖြင့် number နှစ်ခုထဲက ကြီးတဲ့ number ကို ရှာပြီး အခြား number တစ်လုံးနဲ့ မြှောက်တဲ့ function တစ်ခုကို ရေးပါမည်။ ထို့နောက်မှ ထို function ကို docstrings and annotations များ ရေးသားပါမည်။

Sample Program (153)

```
x=3
y=5
def myFun(a):
   return a*max(x,y)
data=myFun(2)
print(data)
#output

#10
```

Sample Program (153) ကို docstrings and annotations များရေးပါမည်။ ပထမဆုံး အနေဖြင့် a အတွက် annotation စရေးပါမည်။

Sample Program (154)

```
x=3

y=5
def myFun(a:'some character')->'multiply '+str(max(x,y))+'times':
  """doc... Will return multiply by a of max """
  return a*max(x,y)
data=myFun(2)
print(data)
print(myFun.__doc__)

print(myFun.__annotations__)
```

Line number 3 ကို ကြည့်ပါ str(max (x,y)) ကဲ့သို့သော statement များကိုလည်း annotations ထဲ၌ ထည့်ရေးနိုင်ပါသည်။ ယခုနည်းကို သုံးခြင်းဖြင့် document များကို တိကျရှင်းလင်းစွာ ဖော်ပြနိုင်မည်ဖြစ်ပါသည်။

Output

```
10
doc... Will return multiply by a of max

{'a': 'some character', 'return': 'multiply 5times'}
```

Lambda Expressions/function

```
def myFun(x):

  return x*2
```

ပုံမှန် function တစ်ခုကို ကြေငြာမယ်ဆိုရင် def ဆိုတဲ့ keyword ကိုသုံးရသလို ထို function ရဲ့ function name ကိုလည်း ထည့်ပေးရပါတယ်။ သို့သော် lambda function တွင်မူ def လည်း မသုံးသလို function name ကိုလည်း ရေးပေးစရာမလိုပါဘူး။ ထိုနည်းတူ return ပြန်ရန် အတွက်လည်း return ဆိုသည့် keyword ကိုသုံးပေးစရာမလိုပါဘူး ။ ထိုကဲ့သို့ function name မရှိတဲ့အတွက် anonymous function လို့လည်းခေါ်ကြပါတယ်။

Declaration of lambda function

```
lambda y: y*2
```

lambda function ကို ကြေငြာရန် lambda ဆိုတဲ့ keyword ကို သုံးပါတယ်။ သူ့နောက်မှာ ကြေငြာထားတဲ့ y သည် parameter ဖြစ်ပြီး function header အဆုံးကို colon နဲ့ ပြပါတယ်။ y*2 ထို lambda function ရဲ့ return value ဖြစ်ပါတယ်။ return value သည်

function object အနေနဲ့ return value ပြန်ပေးမှာ ဖြစ်တဲ့အတွက် variable ထဲသို့လည်း assign လုပ်နိုင်ပါတယ်။

Sample Program (155)

```
x=lambda y: y*2

print(x(3))
```

Sample Program (155) ကို run ကြည့်လျှင် 6 ကို ပြန်ရပါမည်။ lambda သည် function object ကို ပြန်ပေးပါသည်။ ထို့ကြောင့် x သည် function object ဖြစ်သွားပြီး x ထဲတွင် parameter အနေဖြင့် 3 ကို ထည့်လို့ရခြင်းဖြစ်သည်။ parameter အနေဖြင့် ထည့်ပေးလိုက်သော 3 သည် y နေရာတွင် ဝင်ရောက်သွားမည်ဖြစ်ပြီး y*2 ဆိုသောကြောင့် output အနေဖြင့် 6 ကို ရရှိခြင်း ဖြစ်သည်။ ထို program ထဲတွင် print(x) ဟု ရေးပြီး run ကြည့်ပါက အောက်ပါအတိုင်း function lambda ကို ရရှိပါမည်။ မိမိအသုံးပြုသော python version အပေါ်မူတည်ပြီး output မှာလည်း အနည်းငယ်ကွဲပြားနိုင်ပါသည်။

```
x=lambda y: y*2
print(x)

# <function <lambda> at 0x0000020B64BA8AF0>
```

More Example For lambda with two parameters

```
x=lambda y , z=8: y+z

print(x(3 , 2))
```

အထက်ပါ program အား run ကြည့်လျှင် output အနေဖြင့် 5 ကို ရရှိပါမည်။ 3 သည် y နေရာသို့ သွားပြီး 2 သည် z နေရာသို့ သွားပါသည်။ ယခု program တွင် z နေရာတွင် 8 ကိုထည့်ထားသော်လည်း z နေရာသို့ 2 ဝင်လာသောအခါ ထို 8 သည် မရှိတော့ပါဘူး။ သို့သော် z နေရာသို့ မည်သည့် argument မျှရောက်မလာသော အချိန်တွင်မူ ထို 8 သည် ပျောက်မသွားပဲ y+z နေရာတွင် သုံးမည် ဖြစ်ပါသည်။ အောက်ပါ sample program ကို run ကြည့်သောအချိန်တွင် output အနေဖြင့် 11 ကို ရရှိပါမည်။

```
x=lambda y , z=8: y+z

print(x(3))
```

Arbitrary arguments များ keyword arguments များကို သုံးပြီးတော့လည်း lambda function ကိုရေးသားနိုင်သည်။ pack or unpacking ပုံစံလည်း ရေးနိုင်သည်။

Sample Program (156)

```
lam= lambda x,*args,y,**kwargs:(x,args,y,kwargs)
print(lam(10,'a','b',y=100,i=10,j=20,k=30))
#output

# (10, ('a', 'b'), 100, {'i': 10, 'j': 20, 'k': 30})
```

Sample Program (156) တွင် a,b တို့သည် tuple package ပုံစံဖြင့် လာပြီး i,j,k တို့သည် dictionary package ပုံစံဖြင့် လာပါသည်။ args ရှေ့တွင် * ထည့်ပြီး unpack လုပ်ကြည့်ပါက အောက်ပါအတိုင်း a , b တို့အား unpacking လုပ်ထားသည်ကို ရရှိပါမည်။

```
(10, ('a', 'b'), 100, {'i': 10, 'j': 20, 'k': 30})
```

Higher-Order Function

Higher order function ဆိုတာ အခြားသော function တွေကို argument အဖြစ် ယူသုံးသော function တွေကို Higher order function လို့ ခေါ်ပါတယ်။ Lambda function သည် Higher order function များရဲ့ argument အနေဖြင့်လည်း အသုံးပြုပါတယ်။

Sample Program (157)

```
def myFun(x,fn):#higher order function
   return fn(x) #lambda will come here
value=myFun(5, lambda y:y**2) #to multi 5 for two times

print(value)
```

Sample Program (157) အား run ကြည့်လျှင် output အနေဖြင့် 25 ကို ရရှိပါမည်။ line number 3 မှာ myFun ဆိုသည့် function ကို လှမ်းခေါ်သောအခါတွင် lambda function ကို parameter အနေဖြင့် ထည့်ပေးလိုက်ပါတယ်။ ထို parameter သည် fn နေရာတွင် argument အနေဖြင့် ဝင်ရောက်သွားပါသည်။ ထိုနည်းတူ x နေရာတွင်လည်း 5 ဝင်ရောက်လာသည်။ lambda ရဲ့ return expression သည် y**2 ဖြစ်ပြီး y value ကို y*y အဖြစ် ပြန်ပေးရန် ဖြစ်သည်။ fn(5) ဆိုသောကြောင့် 5*5 ဖြစ်ပြီး print ထုတ်ကြည့်သောအခါတွင် 25 ကို ရရှိခြင်း ဖြစ်ပါသည်။ y**2 သည် yy ဖြစ်သည်။ 5 ကို သုံးခါ မြှောက်လိုသောအခါတွင်မူ y**3 ဟုရေးပေးရမည်။

Sample Program (158)

```
def myFun(x,fn):#higher order function
   return fn(x) #lambda will come here
value=myFun(5, lambda y:y**3) #to multi 5 for three time

print(value)
```

arbitrary arguments များ keyword argument များနှင့် တွဲသုံးသော အခါတွင်မူ function ကို arbitrary and keyword များရဲ့ ရှေ့တွင် ရေးပေးရပါမည်။

Lambda Function with *args and **kwargs

Sample Program (159)

```
def vijja(fn , *args , **kwargs):
   return fn(*args,**kwargs)
value=vijja(lambda x,y: x+y , 3 , 5)

print(value)
```

Sample Program (159)ကို run ကြည့်လျှင် output အနေဖြင့် 8 ကို ရရှိပါမည်။ line number 1 တွင်မူ lambda ဝင်ရောက်လာမည့် fn အား args and kwargs ရှေ့ ၂ ခု ရေးထားပါသည်။ line number 3 တွင် lambda expression သည် x and y အား ပေါင်းခြင်းဖြစ်သည် နောက်ထပ် parameters များ ဖြစ်ကြသည့် 3 သည် args သို့သွားမည်ဖြစ်ပြီး 5 သည် kwargs နေရာသို့ သွားမည် ။ expression သည် args + kwargs ဖြစ်သောကြောင့် 3 and 5 အား ပေါင်းပြီး output အနေဖြင့် ပြန်ပေးမည်ဖြစ်သည်။ သတိပြုရန်မှာ return ၂ ခု args and kwargs တို့သည် unpack ပုံစံဖြင့် ပြန်လည် ဖော်ပြရပါမည်။ သို့မဟုတ်ပါက arguments error များတက်မည်ဖြစ်သည်။ ပိုပြီး save and complex ပုံစံဖြင့် ရေးလိုလျှင် end of positional argument ကိုသုံးပြီးတော့လည်း ရေးနိုင်ပါသည်။

Sample Program (160)

```
def vijja(fn , *args , **kwargs):
   return fn(*args,**kwargs)
value=vijja(lambda x,*,y: x + y , 10 , y=11 )

print(value)
```

Sample Program (160) အား run ကြည့်လျှင် 21 ရမည်ဖြစ်ပြီး သတိပြုရန်မှာ lambda x နောက်က end of positional argument ဖြစ်သည်။ ထို end ပါလာသည့်အတွက် x = 10 နေရာနောက်၂ခု ကပ်လျက်နေရာယူလာမည့် 11 သည် positional အလိုက် နေရာယူလို့ရတော့မည် မဟုတ်ပါ။ ထို့ကြောင့် y နေရာတွင် အစားဝင်လာမည့် 11 အား y=11 ဟု keyword only argument ပုံစံဖြင့် ရေးပေးရမည်ဖြစ်သည်။ သို့မဟုတ်လျှင် argument required error ကိုတွေ့ရမည် ဖြစ်သည်။

Lambda, args and sum() function

```
value
def vijja(fn , *args , **kwargs):
    return fn(*args,**kwargs)
value=vijja(lambda *args: sum(args),1,2,3,4,5,6 )
print(value)
```

```
args (1, 2, 3, 4, 5, 6)
kwargs
value
```

အထက်ပါ Sample Program အား run ကြည့်လျှင် 1,2,3,4,5,6 တို့အား sum function ကိုသုံးပြီး အစဉ်လိုက် ပေါင်းထားသောကြောင့် 21 ကို ရရှိပါမည်။ line number 3 မှ vijja function ကို လှမ်းခေါ်သောအခါတွင် ထည့်ပေးလိုက်သော 1,2,3,4,5,6 တို့သည် args တစ်ခုတည်းကို သာ ရောက်သွားပါမည် အဘယ်ကြောင့်ဆိုသော် lambda parameter နေရာတွင် *args ဟူပြီး တစ်ခုတည်းသာ ဖော်ပြထားသောကြောင့်ဖြစ်သည်။

Lambda with Sorted ()

Sorted() function ကို အသုံးပြုလျှင် ပေးထားတဲ့ အချက်အလက်တွေကို သတ်မှတ်ထား တဲ့အတိုင်း sorted လုပ်ပေးပါတယ်။ ဥပမာ a,b,c ကို အစဉ်လိုက်စီချင်ရင် sorted() function ထဲကို ထည့်ပေးလိုက်တာနဲ့ စီစဉ်ပေးပါတယ်။ သတ်မှတ်ထားတဲ့အတိုင်း ဆိုတာက ရှေ့ကနေ စီချင်လား သို့မဟုတ် နောက်ဆုံး စာလုံးကိုပဲ အစဉ်လိုက်စီချင်တာလား

မိမိလိုသလို sorted() function ကိုသုံးပြီး ပြုလုပ်နိုင်ပါတယ်။ sorted() အကြောင်းကိုအသေးစိတ်သိလိုပါက အောက်ပါအတိုင်း ရေးကြည့်နိုင်ပါတယ်။

Sample Program (161)

```
print(help(sorted))
# Help on built-in function sorted in module builtins:
# sorted(iterable, /, *, key=None, reverse=False)
#     Return a new list containing all items from the iterable in ascending order.
#     A custom key function can be supplied to customize the sort order, and the
#     reverse flag can be set to request the result in descending order.

# None
```

Sorted() သည် parameter အနေဖြင့် iterable , key , reverse တို့ကို တစ်ခု သို့မဟုတ် တစ်ခုထက်ပိုပြီး ယူပါတယ်။ iterable နေရာတွင် list , tuple ,set ,dictionary စသည့် data များကို ထည့်နိုင်ပါတယ်။ return value အနေဖြင့် သတ်မှတ်ပေးလိုက်တဲ့ နည်းလမ်းအတိုင်း ပြုလုပ်ပြီး list ကို ပြန်ပေးပါတယ်။

Sample Program (162)

```
data=[1,4,8,9,2,3]

print(sorted(data))
```

Sample Program (162) အား run ကြည့်လျှင် output အနေဖြင့် number များကို ငယ်စဉ်ကြီးလိုက် စဉ်ထားပြီး list တစ်ခု အနေဖြင့် ပြန်ပေးမှာ ဖြစ်ပါတယ်။

Sample Program (163)

```
data=['c','A','a','B','D']

print(sorted(data))
```

အထက်ပါ program အား run ကြည့်လျှင် output အနေဖြင့်
['A', 'B', 'D', 'a', 'c']

ယခု အတိုင်းရရှိမှာဖြစ်ပါတယ် ။ A အကြီးသည်ရှေ့ဆုံးမှ လာပြီး a , c အသေးတို့သည် အကြီး စာလုံးများပြီးမှ လာပါတယ် ။ အဘယ်ကြောင့်ဆိုသော် A အကြီးရဲ့ ascii value သည် 65 ဖြစ်ပြီး a အသေးရဲ့ ascii value သည် 97 ဖြစ်ပါတယ် ။ ထို့ကြောင့် Ascii value အရ ကြည့်လျှင် A အကြီးသည် a အသေးထက်ငယ်ပါသည်။ ထိုနည်းတူ A – Z အကြီးသည် 65 ကနေ 90 ထိ ဖြစ်ပြီး a – z အသေးသည် 97 မှ 122 ထိ ဖြစ်သည် ။ sorted() function သည် default အားဖြင့် ငယ်စဉ်ကြီးလိုက်စဉ်သောကြောင့် A , B , C တို့သည် ရှေ့က ထွက်လာခြင်းဖြစ်ပါသည်။ ASCII table ကို အောက်မှာ ဖော်ပြထားပါတယ် ။ ASCII ဆိုတာ American Standard Code for Information Interchange ကိုဆိုလိုခြင်း ဖြစ်ပါတယ် ။

ထိုကဲ့သို့ ASCII table မကြည့်လိုပါက python ရဲ့ built in function ဖြစ်တဲ့ ord() ကိုသုံးနိုင်ပါတယ် ။ ord('a') မိမိသိလိုသော character ကိုထည့်လိုက်ရင် return value

အနေဖြင့် ထို character ရဲ့ ascii value ကို ပြန်လည်ဖော်ပြပေးပါတယ်။ Number system အားဖြင့် ကြည့်မည်ဆိုလျှင် decimal number system ဖြစ်ပါတယ်။ ဥပမာ character A သည် decimal အားဖြင့် 65 ဖြစ်ပါတယ် ။ ထို 65 ကို 2 နဲ့အကြိမ်ကြိမ် စားပြီး ရလာသော 0 1 binary value များကို မိမိတို့ computer က နားလည်ခြင်းဖြစ်ပါတယ်။

Dec	HxOct	Char		Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (back space)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC 1 (device contral 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC 2 (decive control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC 3 (decive control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC 4 (decive control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

MR.WIN HTUT (GREEN HACKERS)

GREEN HACKERS UNITY IS ADVERS

အထက်ပါ program များသည် sorted() function ရဲ့ iterable parameter အတွက် ဖြစ်ပြီး key အတွက်ဆက်လက် ဖော်ပြပါမည်။ lambda သည် object တစ်ခုတည်းသို့ assign လုပ်နိုင်သည် ဟု ဖော်ပြခဲ့ပြီး ဖြစ်သည်။

Sample Program (164)

```
data=['c','A','a','B','D']
ls=sorted( data , key=lambda ld: ld.upper())
print(ls)
```

Sample Program (164) တွင် lambda, upper and sorted တို့ကို ပူးတွဲသုံးထားသည်။ upper သည် string handling အခန်းတွင် ဖော်ပြခဲ့ပြီး ဖြစ်သည်။ Line number 2 တွင် sorted ထဲ၌ lambda fun ကိုသုံးထား်းသည်။ ld သည် lambda ရဲ့ parameter ဖြစ်ပြီး ld.upper() သည် lambda ရဲ့ expression ဖြစ်သည်။ data သည် char များထည့်ထားသော list ဖြစ်ပြီး ထို data သည် lambda ရဲ့ parameter အဖြစ် အသုံးပြုမည်ဖြစ်ပါသည်။ ထို program ကို run ကြည့်သော အခါ characters များ အစဉ်လိုက် စီစဉ်ထားသည်ကို တွေ့ရပါမည်။

Ascending dictionary with Lambda and sorted()

Previous program တွင် list တစ်ခုအား lambda ကိုသုံးပြီး ascending လုပ်ခဲ့ကြသည် ။ ယခု program တွင် dictionary ထဲမှ key များကို အသုံးပြုပြီး key ရဲ့ value များကို ascending(စီစဉ်) ပါမည်။ ထို့ကြောင့် ပထမဆုံးအနေဖြင့် dictionary တစ်ခုကို ကြေငြာပြီး for statement ဖြင့် sample ပြထားပါသည် ။

Sample Program (165)

```
di={'a':300 ,'c':500 , 'd':100}

for i in di:

   print(i)
```

Sample Program (165)အား run ကြည့်လျှင် Output အနေဖြင့် a, c , d တို့ကို ရပါမည်။

Sample Program (166)

```
di={'a':300 ,'c':500 , 'd':100}

dictionary=sorted(di,key=lambda i: di[i])

print(dictionary)
```

Sample Program (166) ကို run ကြည့်လျှင် output အနေဖြင့် d, a, c တို့ကို ရပါမည် ။ line number 2 တွင် i သည် di ဆိုသည့် dictionary ထဲက value များကို ထုတ်ရန်ဖြစ်ပြီး ထိုကဲ့သို့ ထွက်လာသော value များကို sorted fun မှ ascending လုပ်မှာ ဖြစ်ပါတယ် ထို့ကြောင့် အငယ်ဆုံး ဖြစ်သည့် 100 ရဲ့ key ဖြစ်သည့် d သည် ရှေ့ဆုံးမှ ထွက်လာပြီး အကြီးဆုံးဖြစ်သည့် 500 ရဲ့ key ဖြစ်သည့် c သည် နောက်ဆုံးမှ ထွက်လာခြင်း ဖြစ်သည်။

Ascending last elements

ယခု သင်ခန်းစာတွင် list တစ်ခုထဲမှာ ရှိတဲ့ နောက်ဆုံး element တွေကို ascending လုပ်ပါမည်။

Sample Program (167)

```
a_l=['Guido','win','htut','green','hackers','team']
result=sorted(a_l, key=lambda ac: ac[-1])
print(result)
#output

['team', 'win', 'green', 'Guido', 'hackers', 'htut']
```

Sample Program (167) ကို run ကြည့်ပြီး output များရဲ့ နောက်ဆုံးစာလုံး[-1] ကိုကြည့်ပါ။ စာလုံးများ အစဉ်လိုက်စီထားသည်ကိုတွေ့ရပါမည်။ team သည်

နောက်ဆုံးစာလုံး m ဖြင့်ဆုံးသည့်အတွက် m သည် ascii အရ အငယ်ဆုံး ဖြစ်သောကြောင့် m ပါသော team သည် ရှေ့ဆုံးသို့ ရောက်လာခြင်းဖြစ်သည်။ ထို team ပြီးမှ m ထက်ကြီးသော n ဖြင့်ဆုံးသည့် win သည် team နောက်မှ ပါလာခြင်း ဖြစ်သည်။ နောက်ဆုံး element ကို စစ်လိုသော အခါတွင် [-1] ကို သုံးရမည့် အကြောင်းကို ရှေ့ပိုင်း chapter များတွင် ဖော်ပြခဲ့ပြီး ဖြစ်သည် ။

Random Module

Random module သည် python programming တွင် Random number တွေကို ထုတ်ပေးရန် သုံးပါသည်။ယခု lesson တွင် random number တွေကို generate လုပ်ပုံအကြောင်းကို အသေးစိတ် မဖော်ပြပဲ random ကိုအသုံးပြုပုံနှင့် random နှင့် sorted တို့ တွဲသုံးပုံတို့ကို ဖော်ပြသွားပါမည်။

```
import random as ran

print(ran.random())
```

random() function ကိုမည်သည့် parameter မျှ မထည့်ပဲ သုံးမည်ဆိုလျှင် 0 ကနေ 1 ထိ ကြားမှာ ရှိသော float number များကို return ပြန်ပေးပါမည်။ တစ်ကြိမ် run တိုင်းမှာ မတူညီသော random number တစ်မျိုးစီ ထွက်လာမှာ ဖြစ်ပါတယ်။ Precision အားဖြင့် 53bit ထိ ရှိမှာဖြစ်ပါတယ်။

Uniform from random

Random module ထဲမှ uniform function သည် parameter အားဖြင့်နှစ်ခု ယူပါသည်။ ပထမ parameter သည် အနည်းဆုံး number ဖြစ်ပြီး ဒုတိယ parameter သည် အမြင့်ဆုံး parameter ဖြစ်သည်။

Sample Program (168)

```
import random as ran
print(ran.uniform(1,10))

# Random float x, 1.0 <= x < 10.0
```

Sample Program (168) အတိုင်း run ကြည့်လျှင် 1 နှင့် 10 ကြားမှာ ရှိသည့် မတူညီသော ဒသမ ကိန်းများကို ထုတ်ပေးပါမည်။

Randint from random

Random module ထဲမှ randint သည်လည်း parameter နှစ်ခုယူပြီး return value အားဖြင့် 1 နှင့် 10 ကြားမှာ ရှိသော မတူညီသည့် random values များကို integer ဖြင့် ထုတ်ပေးပါသည်။ ပထမ parameter သည် အငယ်ဆုံးဖြစ်ရမည့် နံပါတ်ဖြစ်ပြီး ဒုတိယ သည်အမြင့်ဆုံး ဖြစ်ရမည့် number ဖြစ်သည်။

Sample Program (169)

```
import random as ran

print(ran.random(1,10))
```

```
# Integer from 1 to 10, endpoints included
```

Randrange from random

Randrange function သည် parameter အားဖြင့် သုံးခု ယူပါသည်။ ပထမ parameter သည် အနိမ့်ဆုံး ဖြစ်ပြီး ဒုတိယ parameter သည် အမြင့်ဆုံးဖြစ်သည်။ တတိယ parameter သည် number အလုံး အရေအတွက်ကို Limit လုပ်ရန်ဖြစ်ပါသည်။

Sample Program (170)

```
import random as ran

print(ran.randrange(0,100,2))

# Even integer from 0 to 100
```

Sample Program (170) တွင် 1 သည် အနိမ့်ဆုံး number ဖြစ်ပြီး100 သည် အမြင့်ဆုံး number ဖြစ်သည်။ သို့သော် even integer များ ဖြစ်ရန် အတွက်2 ကို ထည့်ပေးထားခြင်းဖြစ်သည်။ ရှေ့ဆုံး parameter ဖြစ်သည့် 0 နေရာတွင် 1 ပြောင်းခဲ့ပါက ထွက်လာမည့် output များသည့် မ ကိန်း များသာ ဖြစ်ပါလိမ့်မည်။ even number များထားပါက even number များသာ ထွက်လာပါမည်။ မိမိတို့လိုချင်သော precision အတိုင်း parameter သုံးခုစလုံးကို ပြောင်းလဲကစားနိုင်ပါသည်။

```
import random as ran
print(ran.randrange(1,1000,3))
```

Choice from random

Choice function() သည် သူ့ထဲတွင် parameter အဖြစ်ပါဝင်လာသော data များမှ random element တစ်လုံးကို ထုတ်ပေးပါသည်။ အသုံးပြုပုံသည် အောက်ပါအတိုင်း ဖြစ်ပါသည်။

Sample Program (171)

```
import random as ran

print(ran.choice('WinHtut@GreenHackersTeam'))
```

Sample Program (171)အတိုင်း run ကြည့်လျှင် Output အနေဖြင့် character တစ်လုံးကို ရပါမည်။

Random, sorted and lambda

List တစ်ခုထဲမှာရှိသော elements များကို random , sorted and lambda များကို သုံးပြီး

shuffle လုပ်ပါမည်။

Sample Program (172)

```
import random as ran

elements=[1,2,3,4,5,6,7,8,9,10]
shuffle=sorted(elements , key=lambda x: ran.random())

print(shuffle)
```

Sample Program (172) ကို run ကြည့်တိုင်းမှာ မတူညီသော shuffle လုပ်ထားသည့် list တစ်ခုကို ရမှာဖြစ်ပါတယ်။

Function Introspection

Python programming တွင် almost everything is object ဖြစ်သလို functions တွေဟာလည်း first class object တွေပါ။ ထို function တွေမှာ attributes တွေ ရှိပါတယ်။ ဥပမာ __doc__ တို့ __annotations__ တို့ဖြစ်ပါတယ်။ ထို့ပြင် အခြားသော attributes တွေ ရှိနေသလို မိမိကိုယ်ပိုင် attributes တွေကိုလည်း ထပ်ပြီးထည့်နိုင်ပါတယ်။ ထိုကဲ့သို့ ပြုပြင်ခြင်း analyzing လုပ်ခြင်းကို introspection လို့ခေါ်ပါတယ်။

Sample Program (173)

```
#function introspection

def myFun(x,y):
   return x+y

print(dir(myFun))
```

Sample Program (173)အတိုင်း ရေးပြီး run ကြည့်ပါက myFun ရဲ့ attributes များကို မြင်တွေ့ရပါမည်။

```
['__annotations__', '__call__', '__class__', '__closure__', '__code__', '__defaults__',
'__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__',
'__get__', '__getattribute__', '__globals__', '__gt__', '__hash__', '__init__',
'__init_subclass__', '__kwdefaults__', '__le__', '__lt__', '__module__', '__name__',
'__ne__', '__new__', '__qualname__', '__reduce__', '__reduce_ex__', '__repr__',
'__setattr__', '__sizeof__', '__str__', '__subclasshook__']
```

ထို Attributes များထဲတွင် __annotations__ နှင့် __doc__ တို့သည် ရှေ့ပိုင်း သင်ခန်းစာများ တွင် ဖော်ပြခဲ့ပြီး ဖြစ်သည်။ ဥပမာအားဖြင့် myFun ဆိုသည့် function တစ်ခု တည်ဆောက်မည်ဆိုပါစို့။ ထိုသို့တည်ဆောက်ပြီးနောက် myFun.subject='bio' ဟုရေးပြီး myFun ထဲသို့ subject ဆိုသည့် attributes တစ်ခုထပ်ထည့်နိုင်ပါတယ်။ print(myFun.subject) ဟုရေးကြည့်ပါက bio ဆိုတဲ့ value ထွက်လာမှာဖြစ်ပါတယ်။

Sample Program (174)

```
#function introspection

def myFun(x,y):
   return x+y
```

```
myFun.subject='bio'

print(myFun.subject)
```

Sample Program (174) သည် myFun ဆိုသည့် function ထဲသို့ attributes တစ်ခု ထပ်အပ်ပြီး ပြန်ထုတ်ခြင်း ဖြစ်သည်။ function or object တစ်ခုရဲ့ attributes တွေကို ကြည့်နိုင်ဖို့ dir ဆိုသည့် function ကိုသုံးနိုင်သည် ။ dir သည် built-in function တစ်ခုဖြစ်ပြီး argument အနေဖြင့် object တစ်ခုယူပါသည် ။ return value အနေဖြင့် ထည့်ပေးလိုက်သော object ရဲ့ attributes တွေကို list တစ်ခုအနေဖြင့် ပြန်ပေးပါတယ်။ အောက်တွင် ပုံမှိုထိရောက်နိုင်သော sample program နှင့် output ကို ဖော်ပြထားသည်။

Sample Program (175)

```
#function introspection Adding attributes

def myFun(x,y):
   return x+y
myFun.subject='bio'
myFun.mark=80
print(dir(myFun))
print(myFun.subject)

print(myFun.mark)
```

အထက်ပါ အတိုင်းရေးပြီး program ကို run ကြည့်ပါက output များကို အောက်ပါ အတိုင်း မြင်ရပါမည်။

```
['__annotations__', '__call__', '__class__', '__closure__', '__code__', '__defaults__',
'__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__',
'__get__', '__getattribute__', '__globals__', '__gt__', '__hash__', '__init__',
'__init_subclass__', '__kwdefaults__', '__le__', '__lt__', '__module__', '__name__',
'__ne__', '__new__', '__qualname__', '__reduce__', '__reduce_ex__', '__repr__',
'__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'mark', 'subject']
bio

80
```

Output များကို ကြည့်လျှင် မိမိတို့ ထပ်ထည့်လိုက်သော mark and subject တို့ကို မြင်ရပါမည်။ သူတို့ value များ ဖြစ်ကြသည့် 80 and bio ကိုလည်း မြင်ရပါမည်။

__name__,__default__,__kwdefaults__

ယခု သင်ခန်းစာတွင် name, default and kwdefaults တို့ကို ဖော်ပြသွားပါမယ်။ name သည် function ရဲ့ name ဖြစ်ပြီး default သည် function ရဲ့ positional default parameters များဖြစ်ပါတယ်။ kwdefaults သည် keyword-only defaults များဖြစ်ပါတယ်။ ပထမဆုံး အနေဖြင့် function တစ်ခု တည်ဆောက်ပါမည်။ ထို function ထဲတွင် parameter များ ထည့်ပါမယ်။ ပြီးနောက် ထို function ရဲ့ attributes များကို ပြန်ထုတ်ပါမည်။

Sample Program (176)

```
def myFun(x,y,z=10,*,kw1,kw2=2):

   return x+y
print(myFun.__name__)
print(myFun.__defaults__)
print(myFun.__kwdefaults__)
# myFun
# (10,)

# {'kw2': 2}
```

Name သည် ထို function name ကို ပြန်ပေးပါသည်။ defaults သည် default သတ်မှတ်ထားသော positional parameter ကို tuple အနေဖြင့် ပြန်ပေးပါသည်။ kwdefaults သည် keyword-only argument ကို dictionary ပုံစံဖြင့် key နှင့် value ပါ ပြန်ပေးသည်။ အထက်ပါ program အား docstring များ annotations များနှင့် arguments များ ထပ်ထည့်ပါမည်။

Sample Program (177)

```
def myFun(x:"first positional ",

      y:"second positional",
      z:"defaults arguments " =10,
      *args:"end of positional arguments ",
      kw1:"keyword only argument",
      kw2:"defaults keyword only argument"=2):
   """This function is just for edu purpose for function
   introspection"""
   a=10
   b=20
print(myFun.__annotations__)
print(myFun.__doc__)
#output
# {'x': 'first positional ', 'y': 'second positional', 'z': 'defaults arguments ', 'args': 'end
of positional arguments ', 'kw1': 'keyword only argument', 'kw2': 'defaults keyword
only argument'}
# This function is just for edu purpose for function

#    introspection
```

__code__ attribute

အထက်ပါ program တွင် print(myFun.__code__) ဟု ရိုက်ကြည့်ပါ။ အောက်ပါအတိုင်း အောက်ပါ code object ကို မြင်ရမည်။

```
print(myFun.__code__)
#output

# <code object myFun at 0x000001CAC17C1D40, file ".\app.py", line 1>
```

__code__ object မှာလည်း သူ့ကိုယ်ပိုင် properties များစွာရှိနေပါသေးသည် ။ code object attributes များကို သိနိုင်ရန် print(dir(myFun.__code__)) ဟုရေးကြည့်ပြီး သိနိုင်ပါသည်။

```
print(dir(myFun.__code__))
#output

# ['__class__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__le__', '__lt__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'co_argcount', 'co_cellvars', 'co_code', 'co_consts', 'co_filename', 'co_firstlineno', 'co_flags', 'co_freevars', 'co_kwonlyargcount', 'co_lnotab', 'co_name', 'co_names', 'co_nlocals', 'co_posonlyargcount', 'co_stacksize', 'co_varnames', 'replace']
```

Co_varnames

Co_varnames သည် code object ထဲမှဖြစ်ပြီး code object သည် myFun ရဲ့ attribute ဖြစ်သည်။ ထို့ကြောင့် co_varnames ကို print လုပ်ကြည့်လျှင် myFun function ထဲမှ variable names အားလုံးကို မြင်ရမည်ဖြစ်သည်။ ထို variables names များသည် function header ပိုင်းမှ parameters များတင်မကပဲ function body ပိုင်းမှ variables name များကိုပါ tuple ပုံစံ ဖြင့် ပြန်ပြပေးပါသည်။

Sample Program (178)

```
print(myFun.__code__.co_varnames)

#output

#('x', 'y', 'z', 'kw1', 'kw2', 'args', 'a', 'b')
```

Co_argcount

```
print(myFun.__code__.co_argcount)
#output

#3
```

အထက်ပါ program အတိုင်း run ကြည့်လျှင် output အနေဖြင့် 3 ကို ရပါသည်။ ဘာကြောင့်လဲဆိုလျှင် co_argcount သည် myFun parameters များထဲမှ positional arguments များ ဖြစ်သည့် x,y,z တို့ကိုသာ ရေတွက်သောကြောင့်ဖြစ်သည်။

Inspect module

Object တစ်ခုအား function သို့မဟုတ် method ဟူ၍ ခွဲခြားပြီး သိစေနိုင်ရန် python တွင် isfunction , ismethod , isroutine ဆိုသည့် function များရှိပါသည်။ isfunction သည် object တစ်ခုအား function ဖြစ်လား မဖြစ်သလား ဆိုတာ ကို စစ်ဆေးပေးပြီး parameter အနေဖြင့် object တစ်ခုကို ယူပါသည်။ ismethod သည် parameter အဖြစ် object တစ်ခု အား ယူပြီး ထို object သည် method ဖြစ်လျှင် true ပြန်ပေးပြီး မဟုတ်လျှင် false ပြန်ပေးပါသည်။ isroutine သည် function or method ကို စစ်ဆေးပါသည်။ parameter အနေဖြင့် object တစ်ခုကို ယူပြီး ထို object သည် function or method ဖြစ်လျှင် true ပြန်ပေးပါသည်။ ထိုသို့ ခွဲခြားပြီး သိစေနိုင်ရန် ဘယ်ဟာက method or ဘယ်ဟာက function ဖြစ်ကြောင်းကို ခွဲခြားပြီး သိရန်လိုအပ်ပါသည်။

Different between Function and Method

Function သည် သူ့ကိုယ်ပိုင် ရပ်တည်နိုင်ပြီး မည်သူကမှ ထိန်းချုပ်မထားပါဘူး ။ Python တွင် def myFun(): ဆိုပြီး လွတ်လပ်စွာ ကြေငြာသုံးနိုင်သလို မည်သူ့ရဲ့ property မှလည်း မဟုတ်ပါဘူး။ Method သည် Class တစ်ခုထဲ၌ ကြေငြာထားပြီး class က ထိန်းချုပ်ထားသည်။ ထို့ကြောင့် Method သည် သူနှင့် သက်ဆိုင်သည့် class တစ်ခုရဲ့ property ဖြစ်ပါသည်။

```
def myFun():#function
   pass
 def myClass:
   def cFun():#method

      pass
```

အထက်ပါ program တွင် myFun သည် သူ့ကိုယ်ပိုင် ရပ်တည်နေသော function တစ်ခု ဖြစ်ပြီး cFun() သည် method ဖြစ်သည်။ cFun() သည် myClass ဆိုသည့် class အောက်တွင် တည်ရှိပြီး myClass ရဲ့ property ဖြစ်သည်။ ထိုကြောင့် cFun() ခေါ်သုံးချင်လျှင် myClass ဆိုသည့် class အတွက် object တစ်ခု တည်ဆောက်ရပြီး ထို object မှ တစ်ဆင့် cFun() ကို ခေါ်သုံးရသည်။ သို့သော် myFun() ဆိုသည့် function သည် လွတ်လပ်စွာ ခေါ်သုံးနိုင်သည်။ object လည်း တည်ဆောက်ရန် မလိုပါ။

Sample Program (179)

```
import inspect

from inspect import isfunction,ismethod,isroutine
def myFun(a,b):
   pass
print('Checking is function',isfunction(myFun))
print('Checking is method ',ismethod(myFun))
```

```
print('Checking is fun/method',isroutine(myFun))
#output
# Checking is function True
# Checking is method  False

# Checking is fun/method True
```

Getsource

Inspect module ထဲမှ getsource သည်လည်း အသုံးများသည်။ parameter အနေဖြင့် function object တစ်ခုကိုယူပြီး ထို function ထဲမှာ ရေးထားသော source code အားလုံးကို ပြန်ပေးပါတယ်။

```
print(inspect.getsource(myFun))
```

Getmodule

Inspect module ထဲမှ getmodule သည် function object တစ်ခုကို parameter အဖြစ်ယူပြီး builtins or function type ကိုပြန်ပေးပါသည်။ inspect.getmodule(print) ဟုရေးကြည့်မည်ဆိုလျှင် <module 'builtins' (built-in)> ဆိုသည့် output ကို ပြန်ပေးပါမည်။ example အနေဖြင့် math module ထဲမှ sin function အားကြည့်မည်ဆိုလျှင် math module ကို သုံးမှာ ဖြစ်သည့်အတွက် import math ကို အရင်ရေးပေးရပါမည်။

print(inspect.getmodule(math.sin)) ဟုရေးကြည့်မည်ဆိုလျှင် <module 'math' (built-in)> ဆိုသည့် output ကို ပြန်ရပါမည်။

Getcomments

Getcomments သည် function တစ်ခုရဲ့ comments များကို သိရှိလိုသောအခါတွင် အသုံးပြုပါသည်။ getcomments သည် parameter အနေဖြင့် function တစ်ခုကို ယူပြီး ထို function အပေါ်တွင် ရေးသားထားသော comment များကို return အနေဖြင့် ပြန်ပေးပါသည်။

```
import inspect
from inspect import getcomments
#TODO: to explain getcomment
#Hello im win htut
def myFun():
   pass
print(getcomments(myFun))
```

Callable

Python programming တွင် object တွေကို ( ) parentheses operator ဖြင့် ခေါ်လိုက်လျှင် return value အနေဖြင့် True or False ပြန်ပေးပါသည်။ true ပြန်ပေးလျှင် ထို object သည် callable ဖြစ်ပြီး False ပြန်ပေးပါက callable မဟုတ်ပါ။ Python မှာ

တော်တော်များများက callable တွေဖြစ်ပါတယ်။ callable ဖြစ်မဖြစ်ကို သိရှိနိုင်ရန် built-in function ဖြစ်သည့် callable() ကိုသုံးပြီး စစ်ဆေးနိုင်ပါသည်။ အောက်ပါအတိုင်းရေးပြီး run ကြည့်လျှင် True များကိုသာ ရပါမည်။

```
print(callable(print))

print(callable('xyz'.upper))
```

သတိပြုရန်မှာ callable အားလုံးသည် return value ပြန်ပါသည်။ ဥပမာအားဖြင့် print() ကိုအောက်ပါ အတိုင်းစမ်းကြည့်နိုင်သည်။

```
test=print('hello')

print(test)
```

အထက်ပါ program အား run ကြည့်လျှင် output အနေဖြင့် hello နှင့် None ကို ရပါမည်။ နောက် example တစ်ခုအနေဖြင့် append ကို callable ဖြစ်မဖြစ် အောက်ပါအတိုင်း စမ်းကြည့်နိုင်သည်။

Sample Program (180)

```
aL=[1,2,3,4]

result=aL.append(4)
print(aL)
print(result)
#output
# [1, 2, 3, 4, 4]

# None
```

Different Types of Callable

Built-in functions များဖြစ်ကြတဲ့ print , len , callable တို့သည် callable များဖြစ်ကြပါတယ်။ Built-in methods များ ဖြစ်ကြတဲ့ str.upper , al.append တို့သည် callable method များဖြစ်ကြပါတယ်။ User – defined functions များဖြစ်ကြတဲ့ def and lambda စတဲ့ function တွေဟာလည်း callable function များဖြစ်ပါတယ်။ classes နှင့် class ရဲ့ကိုယ်စားဖြစ်တဲ့ obj တွေဟာလည်း callables ဖြစ်နိုင်ပါတယ်။ classes အားလုံးသည် callables ဖြစ်သည်။ class များအကြောင်း အသေးစိတ်ကို Object Oriented Programming နောက်ပိုင်းသင်ခန်းစာများ တွင် ဖော်ပြပေးသွားပါမည်။ class များသည် callable ဖြစ်မဖြစ်ကိုအောက်ပါ lesson တွင် လေ့လာကြည့်ပါ။

Sample Program (181)

```
class myClass:          #1

  def __init__(self,x=0): #2
    print('from class') #3
    self.counter=x     #4
```

```
print(callable(myClass))   #5
test = myClass(10)        #6
print(test)             #7
print(test.counter)       #7
print(callable(test))     #8
# output
# True
# from class
# <__main__.myClass object at 0x0000025DB68A7880>
# 10

# False
```

Sample Program (181) တွင် line 5 ၌ myClass ကို print ထုတ်ကြည့်သည့်အခါ True ကို ပြန်ပေးပါသည်။ line 6 တွင် myClass 10 ဆိုသည့် value တစ်ခုလွှမ်းထည့်ပြီး ခေါ်ပါသည် ။ line 7 တွင် myClass ကိုယ်စားပြုထားသည့် test ကို print ထုတ်ကြည့်သောအခါ from class ဆိုသည့် output ကိုရသလို 10 သည်လည်း x ထဲသို့ ဝင်ရောက်သွားပါသည်။ ထို့ကြောင့် myClass ကို ကိုယ်စားပြုထားသည့် test ထဲမှ test.counter ကို print ထုတ်ကြည့်သည့်အခါ 10 ကို ပြန်ရပါသည်။ သို့သော် test သည် callable မဖြစ်ပါဘူး။ ထို့ကြောင့် output တွင် False ကို မြင်ရပါမည်။ test သည် class တစ်ခုကို ကိုယ်စားပြုထားသော်လည်း callable မဖြစ်သည့် object တစ်ခုဖြစ်ပါသည်။

Sample Program (182)

```
class myClass:          #1

  def __init__(self,x=0): #2
    print('from class') #3
    self.counter=x     #4

  def __call__(self , x=1):          #5
    print('Updating counter value....') #6
    self.counter +=x              #7
b=myClass()                       #8
print(myClass.__call__(b,10))          #9
print(b.counter)                  #10
print(callable(b))                #11
# output
# from class
# Updating counter value....
# None
# 10

# True
```

Sample Program (182) သည် class တစ်ခုထဲမှာရှိတဲ့ function တစ်ခုကို callable ဖြစ်အောင် ဘယ်လိုခေါ်ရမလဲ ဆိုတာကို ပြထားပါတယ်။ line 5 တွင်ရေးထားသော call ထဲမှ self ဆိုသည့် နေရာတွင် class ကိုကိုယ်စားပြုထားသော object ကိုထည့်ပေးရပါမည်။ b ကို callable ဖြစ်ကြောင်း စစ်ကြည့်လျှင် True ကို ပြန်ရပါမည်။ အထက်ပါ ဖော်ပြချက်များသည် myClass ထဲမှ __call__ အတွက်ဖြစ်ပြီး __init__ အတွက်လည်း ထိုသို့ callable ဖြစ်အောင် ပြုလုပ်နိုင်သည်။ callable ဖြစ်သွားလျှင် မည်သည့် class မှ ရှေ့တွင် ထည့်ခေါ်စရာမလိုပဲ b ကို ( ) ဖြင့် တိုက်ရိုက်လှမ်းခေါ်နိုင်သည်။ Sample Program (182) ၌ b() ကို တိုက်ရိုက်လှမ်းခေါ် တိုင်း counter value သည် 1 ပေါင်းသွားမှာဖြစ်ပါတယ်။ b.counter ဟု print ထုတ်ကြည့်လျှင် counter တန်ဖိုးတိုးမတိုးကို သိနိုင်သည်။ myClass ထဲမှ __init__ ကို callable ဖြစ်စေရန်လည်း အောက်ပါအတိုင်းရေးနိုင်သည်။

Sample Program (183)

```
class myClass:          #1

  def __init__(self,x=0): #2
    print('from class') #3
    self.counter=x     #4
  def __call__(self , x=1):          #5
    print('Updating counter value....') #6
    self.counter +=x              #7
b=myClass()
z=myClass()
print(myClass.__call__(b,10))
print(myClass.__init__(z,20))
print(b.counter)
print(callable(b))
print(z.counter)
print(callable(z))

print(b())
```

Some Object are not callable

အချို့သော object တွေကတော့ callable မဟုတ်ပါဘူး။ class တစ်ခုကို ကိုယ်စားပြုထားတဲ့ object တွေကသာ callable ဖြစ်နိုင်ပါတယ်။ အောက်ပါ program တွင် ကြည့်ပါ။ a သည် callable မဟုတ်ကြောင်းကို ပြထားပါတယ်။

Sample Program (184)

```
from decimal import Decimal

print(callable(Decimal))#will be true
a=Decimal('10.5')
print(type(a))
print(callable(a))
```

```
# output
# True
# <class 'decimal.Decimal'>

# False
```

Iterable

Object တစ်ခုဟာ သူ့မှာရှိတဲ့ member data တွေကို တစ်ကြိမ်မှာ တစ်ခုသာလျှင် ထုတ်ပေးခြင်းကို iterable ဖြစ်တယ်လို့ ခေါ်ပါတယ်။ အခြေခံအားဖြင့် iterable သုံးမျိုးခွဲနိုင်ပါတယ်။ ပထမတစ်ခုကတော့ sequences type တွေဖြစ်တဲ့ list , string , tuple တို့ဖြစ်ပါတယ်။ ဒုတိယတစ်ခုကတော့ non-sequence types တွေဖြစ်တဲ့ dictionary , file objects တို့ဖြစ်ပါတယ်။ တတိယတစ်ခုကတော့ class ကို ကိုယ်စားပြုထားတဲ့ objects တွေပါ။ ဘယ်လို class တွေလည်းဆိုတော့ method အနေဖြင့် __iter__() သို့မဟုတ် __getitem__() စသည်တို့ ပါဝင်တဲ့ classes တွေ ဖြစ်ပါတယ်။

Iterables ဖြစ်တဲ့ objects တွေထဲကနေ data တွေ တစ်ခုခြင်းစီထုတ်တဲ့အချိန်မှာ for loop ကိုအသုံးပြုနိုင်သလို sequence types တွေအတွက်ဆိုရင် zip(), map() စတဲ့ function တွေကိုလည်း သုံးနိုင်ပါတယ်။

Sample Program (185)

```
list=['a','b','c','d','e']
iterator = iter(list)
print(next(iterator))
print(next(iterator))
print(next(iterator))
print(next(iterator))
```

Sample Program (185) ကို run ကြည့်မည်ဆိုလျှင် a,b,c,d တို့ထွက်လာသည်ကို မြင်ရမှာပါ။ iter function ကတော့ သူ့ကိုထည့်ပေးလိုက်တဲ့ parameter ကို object တစ်ခုဖန်တီးလိုက်ပါတယ် ပြီးလျှင် သူ့ကိုတစ်ကြိမ်မှာ data တစ်ခုစီထုတ်ပေးပါတယ်။ print ထုတ်တဲ့ အခါမှာ next function ကိုသုံးပြီး ထုတ်ထားပါတယ် ။ next function သည် parameter အနေဖြင့် ထည့်ပေးထားတဲ့ iterator ထဲက next item ကိုယူပေးပါတယ်။ ထို့ကြောင့် a,b,c,d ဟု Output များကို အစဉ်လိုက် ရခြင်း ဖြစ်ပါတယ်။ print code လေးကြောင်းမရေးခင် print(iterator) ဟုရေးပြီး run ကြည့်လျှင် object ဆိုတာကိုတွေ့ရပါမည်။

map()

Map function သည် higher order ထဲမှ function တစ်ခုဖြစ်ပြီး function တစ်ခုကို parameter အနေဖြင့် သုံးနိုင်ပါသည်။ map function မှာ ပုံမှန်အားဖြင့် parameter နှစ်ခု ယူပါသည်။ ပထမတစ်ခုသည် function ဖြစ်ပြီး ဒုတိယတစ်ခုသည် iterable ဖြစ်ပါတယ်။ iterable ကို တစ်ခုထက်ပိုပြီးလည်း parameter အနေဖြင့် သုံးနိုင်ပါသည်။

Sample Program (186)

```
def fact(n):
    return 1 if n<2 else n*fact(n-1)

results=map(fact,range(6))
print(results)
```

ပထမဆုံး အနေဖြင့် factorial ရှာသည့် function တစ်ခုကို စတင်ရေးသားပါမည်။ ထို့သို့ ရေးသားရာတွင် recursive function ကိုသုံးပြီးရေးသားထားပါသည်။ line 3 တွင် map function သည် return အနေဖြင့် map object ကို ပြန်ပေးပါတယ်။ list or tuple , set တို့ကို ပြန်ပေးတာမျိုး မဟုတ်ပဲ မိမိတို့ လိုချင်သည့်ပုံစံဖြင့် ပြန်ပြောင်းနိုင်သည်။ ထို့ကြောင့် အထက်ပါ program ကို run ကြည့်သောအခါတွင် အောက်ပါအတိုင်း map object ရကြောင်းကို မြင်ရပါမည်။

```
<map object at 0x0085A208>
```

Results ဆိုသည့် object ထဲမှ elements များကို အောက်ပါအတိုင်း for loop ကို သုံးပြီး ထုတ် ကြည့်နိုင်သည်။

```
for i in results:
    print(i)
```

သို့သော် results ဆိုသည့် Object ကို နောက်တစ်ကြိမ် ပြန်အသုံးပြုသောအခါတွင်မူ မည်သည့် output ကိုမှ ရမည်မဟုတ်ပါ။ အဘယ်ကြောင့်ဆိုသော် map function သည် list, tuple, set တို့ကို return ပြန်ခြင်းမဟုတ်ပဲ generator ကို return ပြန်ခြင်းဖြစ်သည်။ generator ကို နောက် lesson တွင် ဖော်ပြသွားပါမည်။ program ကို အောက်ပါအတိုင်း အစအဆုံးရေးကြည့်ပါ က output တစ်ခုတည်းထွက်သည်ကိုသာ မြင်ရပါမည်။ ပထမ looping ဖြစ်သည့် i အတွက် output များကို မြင်ရမည်။ ဒုတိယ looping အတွက်မူ ဘာကိုမှရမည်မဟုတ်ပါ။

Sample Program (187)

```
def fact(n):
    return 1 if n<2 else n*fact(n-1)
results=map(fact,range(6))
print(results)
for i in results:
    print(i)
for z in results:
    print(z)
```

Output:

1

1

2
6
24
120

ထိုကဲ့သို့သော ပြဿနာကို ဖြေရှင်းရန် map function ရှေ့တွင် အောက်ပါအတိုင်း list ဆိုသည့် function ကို ထပ်သုံးလိုက်ပါက map က ပြန်ပေးလာသော return value များကို list ပုံစံဖြင့် ပြန်ရမည် ဖြစ်သည်။

```
results=list(map(fact,range(6)))
```

Map with Lambda expression

Map function ထဲတွင် parameter အနေဖြင့် lambda expression ကိုလည်း သုံးနိုင်ပါတယ်။ ယခု program တွင် map function ထဲသို့ lambda အပြင် list နှစ်ခုလည်း ဖြတ်ပါမည်။

Sample Program (188)

```
list1=[1,2,3,4,5,6]
list2=[5,10,15,20]
results=list(map(lambda x,y:x+y , list1,list2))
print(results)
```

Output :
[6, 12, 18, 24]

Generator

Generator ဆိုတာလည်း function တစ်ခုဖြစ်ပါတယ်။ သို့သော် generator သည် return အနေဖြင့် yield ဆိုသည့် keyword ကိုသုံးပါသည်။ return သည် function တစ်ခုကို အဆုံးသတ်(terminate) လုပ်ခြင်းဖြစ်ပြီး yield ကတော့ function တစ်ခုအား ခဏရပ်တန့်ခြင်း
(pause) လုပ်ခြင်းသာ ဖြစ်ပါသည်။

Sample Program (189)

```
def myFun(x):
    return x

def myGen(y):
    yield y
```

Program တွင် myFun သည် function တစ်ခု ရေးသားသော ပုံစံ ဖြစ်ပြီး အောက်တွင် ရေးထားသော myGen သည် generator တစ်ခုအား ရေးသားထားသော ပုံစံ ဖြစ်သည်။

Sample Program (190)

```
def myGen():
    yield 10
results=myGen()
print(results)
```

Output:

<generator object myGen at 0x0345A2C8>

Sample Program (190) အား run ကြည့်လျှင် output အနေဖြင့် generator object ကို ပြန်လည် ရရှိပါမည်။ ထို generator object ထဲမှ data များကို အောက်ပါအတိုင်း __next__() ကိုသုံးပြီး ထုတ်နိုင်ပါသည်။

```
def myGen():
    yield 10
    yield 20
    yield 30
    yield 40
results=myGen()
print(results.__next__())
print(results.__next__())
print(results.__next__())
print(results.__next__())
```

Output:

10

20

30

40

အထက်ပါအတိုင်း print လေးကြောင်းကို မရေးလိုလျှင် for loop ကိုသုံးပြီးလည်း data များကို ထုတ်နိုင်ပါသည်။ နောက်ထပ် program တစ်ပုဒ်အနေဖြင့် yield များ နေရာတွင် while loop ကိုသုံးပြီး yield တစ်ခုတည်း ရေးသားပါမည်။

Sample Program (191)

```
def myGen():            #line(1)
    x=1                 #line(2)
    while x<=10:        #line(3)
        square=x*x      #line(4)
        yield square    #line(5)
        x+=1            #line(6)
results=myGen()         #line(7)
for i in results:       #line(8)
    print(i)            #line(9)
```

Sample Program (191) အား run ကြည့်လျှင် 1 မှ စတင်ပြီး 9 ထိ နှစ်ထပ်ကိန်းများကို ရရှိမှာ ဖြစ်ပါတယ်။ program အလုပ်လုပ်ပုံမှာ ပထမဆုံး အနေဖြင့် results = myGen() ဆိုသည့် line တွင် results ဆိုသည့် object တစ်ခု ကိုတည်ဆောက်လိုက်သည်။ ထို object သည် generator object ဖြစ်သည်။ ထို့နောက် for loop အကြောင်းတွင် ပထမဆုံး element ကို ထုတ်ရန် myGen() ဆိုသည့် generator ကိုသွားပါသည်။ step by step အလုပ်လုပ်သွားပြီး yield နေရာကို ရောက်သွားသောအခါတွင် square တန်ဘိုးကို result ဆီသို့ (line 8) ပြန်ပေးပါသည်။ ထို့ကြောင့် ပထမဆုံး တန်ဖိုးသည် 1 ကစတဲ့ အတွက် 1 ဖြစ်နေပြီး i တန်ဖိုးသည်လည်း 1 ဖြစ်နေပါမည်။ ပြီး လျှင် ကျန်ရှိနေသေးသော line 6 ကို အလုပ်ဆက်လုပ်ပါသည်။ ထို့နောက် line 3 ကို ဆက်သွားပါသည်။ ယခုတစ်ကြိမ်တွင် x တန်ဖိုးသည် 2 ဖြစ်နေသည့်အတွက် square တန်ဖိုးသည် 4 ဖြစ်နေပါမည်။ ထို square တန်ဖိုးအား result ဆီသို့ ပြန်ပေးပါသည် ။ ထိုနည်းအတိုင်းပင် generator ကိုသုံးပြီး value

များကို တစ်ခုချင်းစီ ထုတ်နိုင်ပါသည် ။ Database ထဲမှ data များကို ထုတ်ပြီး တိုက်စစ်လိုသောအခါတွင် ယခုနည်းလမ်းကိုသုံးနိုင်သည်။ Database ထဲမှ data အားလုံးကို ထုတ်စရာမလိုပဲ generator ကိုသုံးပြီး တစ်ခုချင်းစီထုတ်ကာ မိမိလိုသလို သုံးနိုင်သည်။ Generator အား အောက်ပါ နည်းလမ်းကို သုံးပြီးလည်း စမ်းသပ်နိုင်ပါသေးသည်။

Sample Program (192)

```
def myGen():
    value=10
    print('First calling')
    yield value
    value +=10
    print('Second calling')
    yield value
    value +=30
    print('Third calling')
    yield value
results=myGen()
print(next(results))
print(next(results))
print(next(results))
```

Generator နှင့်ပတ်သက်ပြီး အောက်ပါ တို့ကို သိထားရမည်။

- Generator ထဲတွင် yield များ တစ်ခုထက် ပိုပြီး ပါဝင်နိုင်သည်။
- For loop ကိုသုံးနိုင်သလို next() function ကိုလည်း သုံးနိုင်သည်။
- Yield တစ်ခုစီ တိုင်းသည် program process ကို ခနရပ်တန့် suspend လုပ်ပါသည်။
- ထိုသို့ ရပ်တန့်ထားသော အခါမျိုးတွင်လည်း value များ၊ statement များကို မှတ်ထားပါသည်။

Zip() Function

Zip function သည် parameter အနေဖြင့် iterables ကို ယူပြီး elements များအားလုံးကို tuple အနေဖြင့် ပြန်ပေးပါတယ်။ parameter အနေဖြင့် list , stirng ,dict တို့သာမက user-defined iterables တို့ကိုလည်း အသုံးပြုနိုင်ပါတယ်။

Syntax of zip()

zip(*iterables)

Sample Program (193)

```
itr1=[1,2,3,4,5]
itr2=[10,20,30]
results= zip(itr1,itr2)
print(results)
for i in results:
    print(i)
```

Output:
<zip object at 0x00C48AA8>
(1, 10)
(2, 20)
(3, 30)

Zip သည်လည်း map function အတိုင်းပင် zip object ကိုပြန်ပေးသည်။ for loop တစ်ခု ထပ်ရေးပြီး အသုံးပြုလျှင်မည်သည့် output ကိုမှရမှာမဟုတ်ပါဘူး။ zip function သည် ထည့်ပေးလိုက်သော itrables များကို element တစ်ခုချင်းစီပေါင်းပေးပါသည်။ ထို့ကြောင့် ပထမ ဆုံး output တွင် itr1 မှ 1 နှင့် itr2 မှ 10 တို့ကို tuple ပုံစံဖြင့် ရရှိခြင်းဖြစ်သည်။

Filter ()

Filter function သည် parameter အနေဖြင့် နှစ်ခုယူပါသည်။ ပထမတစ်ခုသည် function ဖြစ်ပြီး ဒုတိယ တစ်ခုသည် iterable ဖြစ်ပါသည်။ ပထမ parameter ဖြစ်သည့် function သည် iteralbe ထဲမှ element တစ်ခုချင်းစီအားစစ်ဆေးပြီး true or false တစ်ခုခုကို ပြန်ပေးပါသည်။ ဒုတိယတစ်ခုဖြစ်သည့် iterable သည်စစ်ဆေးရန်ဖြစ်ပြီး sets, lists, tuples သို့မဟုတ် အခြားသော iterable များ ဖြစ်နိုင်သည်။

Sample Program (194)

```
alphabet=['a','b','c','d','e','f','i','o','u']
def VowelFilter(alphabet):
    vowel=['a','e','i','o','u']
    if(alphabet in vowel):
        return True
    else:
        return False
VowelFilter = filter(VowelFilter,alphabet)
print('The filtered vowels are:')
for i in VowelFilter:
    print(i)
```

Sample Program (194) အား run ကြည့်လျှင် a , e ,i ,o ,u ဆိုသည့် output များကိုမြင်ရပါမည်။ Vowel များကို စစ်ဆေးရန် ပထမဆုံး အနေဖြင့် vowelfilter ဆိုသည့် function တစ်ခုတည်ဆောက်ပါသည်။ ထို function ထဲတွင် user မှထည့်ပေးလိုက်သော alphabet ဆိုသည့် iterable တစ်ခုဖြတ်ပါမည်။ ထို့နောက် if statement ထဲတွင် alphabet ဆိုသည့် list ထဲတွင် vowel ထဲမှစကားလုံးများပါသလား ဆိုတာကို တစ်လုံးချင်းစီတိုက်စစ်ရန်ရေးသားထားခြင်းဖြစ်သည်။

line number 8 တွင် filter function ထဲ၌ vowelfilter ဆိုသည့် function နှင့် alphabet ဆိုသည့် user ထည့်မည့် iterable တို့ကို ထည့်ပေးလိုက်ပါသည်။ Program သည် line number 8 မှစ run မည်ဖြစ်ပြီး line number 9 နှင့် line number 10 သို့ရောက်သောအခါ VowelFilter object ထဲမှ element များကို တစ်ခုခြင်းစီ စစ်ဆေးပြီးမှန်လျှင် True ပြန်ပေးပါသည် ဆိုလိုသည်မှာ ပထမဆုံးအနေဖြင့် alphabet ဆိုသည့် list ထဲမှ ပထမဆုံးစာလုံးသည် vowel ဆိုသည့် list ထဲမှ ပထမဆုံးစာလုံးနှင့်တူသလားစစ်ပြီး တူလျှင် True ပြန်ပေးခြင်း ဖြစ်သည်။ ထိုကဲ့သို့ True ဆိုလျှင် output အနေဖြင့် a ကိုပြန်ပေးပါသည်။ e ဆိုသည့် စာလုံး အလှည့်တွင်လည်း ထိုအတိုင်းပင် ဖြစ်သည်။ True ဆိုမှသာ VowelFilter ထဲသို့ စကားလုံးများ ရောက်ရှိမည် ဖြစ်ပြီး False ဆိုလျှင် VowelFilter ထဲသို့ စကားလုံးများရောက်မည် မဟုတ်ပါ ထို့ကြောင့်

false အလှည့်သို့ ရောက်သောအခါတွင်မူ စကားလုံးများကို ထုတ်ပေးမည်မဟုတ်ပါ။

Reduce () function

Python 2 မှာဆိုရင်တော့ reduce function သည် built in function ဖြစ်ပြီး 3 တွင်မူ functools ဆိုသည့် module ကို import လုပ်ပြီးမှ အသုံးပြုရပါမည်။ reduce function သည် ပုံမှန် အားဖြင့် parameter နှစ်ခုယူပါသည်။ ပထမတစ်ခုသည် function ဖြစ်ပြီး ဒုတိယတစ်ခုသည် iterable ဖြစ်ပါသည်။ lambda expression ကိုသုံးပြီး အောက်တွင် sample program တစ်ပုဒ်ရေးပြထားပါသည်။

Sample Program (195)

```
import functools
li = [1,2,3,4,5,6]
print("The sum of all elements from list: ",end="")
print((functools.reduce(lambda a,b: a+b , li)))
```

အထက်ပါ program ကို run ကြည့်လျှင် output အနေဖြင့် 21 ကို ရပါမည်။

Program ရှင်းလင်းချက်

- အလုပ်လုပ်သည့် ပုံစံမှာ ပထမဆုံးအနေဖြင့် list ထဲမှ ပထမနှင့်ဒုတိယ element နှစ်ခုကို ယူပါသည် ။ ထို့နောက် ထို element နှစ်ခုအား lambda function ထဲသို့ထည့်ပြီး lambda function ထဲမှ instruction အတိုင်း အလုပ်လုပ်ပါသည်။ အထက်တွင် ရေးထားသော lambda function ထဲတွင် element နှစ်ခုကို ထပ်ပေါင်းခြင်းဖြစ်သောကြောင့် ပေါင်းလဒ်ကို ရပါမည်။
- အထက်တွင်ရလာသော ပေါင်းလဒ် နှင့် list ထဲမှ ကျန်ရှိနေသော တတိယမြောက် element ကို ထပ်ယူပြီး lambda function ထဲသို့ထပ်ပို့ပါသည်။ပြီးလျှင် ထို element များမှ ရလာသော result နှင့် နောက်ထပ် list ထဲမှ element တစ်ခုကိုယူကာ lambda function ထဲသို့ ထပ်ပို့ပါသည်။ ဤနည်းဖြင့် list ထဲမှ element အားလုံးပေါင်းခြင်းကို ရခြင်း ဖြစ်ပါသည်။

List တစ်ခုထဲမှ element အားလုံး၏ အကြီးဆုံး element ကို ရလိုလျှင်လည်း reduce function ကို သုံးပြီး အောက်ပါအတိုင်းရေးနိုင်ပါသည်။

Sample Program (196)

```
import functools
li = [1,2,3,4,5,6]
print("The sum of all elements from list: ",end="")
print((functools.reduce(lambda a,b: a if a>b else b , li)))
```

Sample Program (196) ကို run ကြည့်ပါက output အနေဖြင့် 6 ကို ရရှိပါမည်။ အကယ်၍ အငယ်ဆုံး min value ကို ရလိုပါကလည်း အောက်ပါ အတိုင်း ရေးနိုင်ပါသည်။

Sample Program (197)

```
import functools
li = [1,2,3,4,5,6]
print("The sum of all elements from list: ",end="")
print((functools.reduce(lambda a,b: a if a<b else b , li)))
```

Reduce function ကို မသုံးပဲ မိမိတို့ကိုယ်တိုင် ရေးမည်ဆိုပါက အောက်ပါအတိုင်း ရေးရပါမည်။

Sample Program (198)

```
def mySum(x1, x2):              #line 1
    return x1 + x2              #line 2
def myReduce(mySum, seq):       #line 3
    first = seq[0]              #line 4
    for i in seq[1:]:           #line 5
        first = mySum(first, i) #line 6
    return first                #line 7
list = [1, 2, 3, 4]             #line 8
print(myReduce(mySum, list))    #line 9
```

Sample Program (198) အား run ကြည့်ပါ output အနေဖြင့် 10 ကို ရှိပါမည်။

Program ရှင်းလင်းချက်

- First အနေဖြင့် mySum ဆိုသည့် function တစ်ခု ရေးသားပါမည်။ ထို function သည် ပေးလာသော parameter နှစ်ခုကို ပေါင်းပြီးရလဒ်ကို ပြန်ပေးမှာ ဖြစ်ပါတယ်။
- Second အနေဖြင့် myReduce ဆိုသည့် function ကို parameter နှစ်ခုဖြင့် ရေးပါသည် ပထမ parameter သည် function တစ်ခု ဖြတ်ရန်ဖြစ်ပြီး ဒုတိယ parameter သည် sequence အလိုက် element များ ဝင်လာရန် ဖြစ်သည်။
- Third အနေဖြင့် list မှ index 0 အား first ဆိုသည့် variable ထဲသို့ ထည့်လိုက်ပါမည်။ ထို့နောက် list အား looping ပတ်ပြီး list ထဲမှ value အားလုံးကို ထုတ်ယူကာ parameter အနေဖြင့်သုံးမည် ဖြစ်ပါသည်။
- Fourth အနေဖြင့် line number 6 တွင် mySum ဆိုသည့် function အား အထက်တွင် ရလာသော elements များကို parameter အနေဖြင့် အသုံးပြုပြီးလှမ်းခေါ်ပါသည်။

mySum function မှ parameter နှစ်ခုအားပေါင်းပြီး return အနေဖြင့် ပြန်ပေးလာသော value ကို first ဆိုသည့် variable ထဲသို့ ပြန်ထည့်လိုက်ပါသည်။

- Fifth အနေဖြင့် ကျန်ရှိနေသေးသော element များကို တစ်ခြင်းစီထပ်ထုတ်ပြီး parameter အနေဖြင့် အသုံးပြုကာ mySum Function ကို လှမ်းခေါ်ပါသည်။ element များ ကုန်သွားသော အချိန်တွင် first ထဲမှ value ကို return အနေဖြင့် function call ခေါ်သည့် နေရာသို့ ပြန်ပေးလိုက်ပါသည်။ ထို့ကြောင့် အားလုံးပေါင်းထားသည့် result ကို နောက်ဆုံးတွင် ရ ခြင်းဖြစ်ပါသည်။

Warning Our Reduce Function

မိမိတို့ ကိုယ်တိုင် ရေးထားသော myReduce function သည် set or unsequence ဖြစ်သည့် data များကိုမူ handle မလုပ်နိုင်ပါ။ *TypeError: 'set' object is not subscriptable* ဖော်ပြထားသည့် error ကို တွေ့ရပါမည်။ သို့သော် functools module ထဲမှ reduce function သည် set or unsequence ဖြစ်သည့် data များကိုပါ handle လုပ်နိုင်ပြီး error တက်မည် မဟုတ်ပါ။

Partial () Function

Partial function ကို အသုံးပြုခြင်းအားဖြင့် function တစ်ခုရဲ့ arguments တွေကို လျှော့ချနိုင်ပါတယ်။ Partial function ကို ခေါ်သုံးနိုင်ရန် functools ဆိုသည့် module ကို import လုပ်ပေးရန် လိုအပ်ပါသည်။ partial function ကို ခေါ်မသုံးခင် မိမိတို့ကိုယ်တိုင် အောက်ပါအတိုင်း ရေးကြည့်ပါမည်။

Sample Program (199)

```
def myFun(x,y,z):                # line 1
    print(x,y,z)                 # line 2
def fn(yy,zz):                   # line 3
    return myFun(10,yy,zz)       # line 4
print(fn(100,200))               # line 5
```

Sample Program (199) အား run ကြည့်ပါက output အနေဖြင့် 10 ,100 ,200 တို့ကို ရှိပါမည်။ program အလုပ်လုပ်ပုံမှာ -

1. ပထမဆုံးအနေဖြင့် myFun ဆိုသည့် function ကို x,y,z ဆိုသည့် arguments 3 ခုဖြင့် ရေးထားပါသည်။
2. ဒုတိယအနေဖြင့် fn ဆိုသည့် function အား yy ,zz ဆိုသည့် arguments 2 ခုဖြင့် တည်ဆောက်ထားပါသည်။ fn ဆိုသည့် function အားခေါ်သောအခါတွင် return အနေဖြင့် myFun ဆိုသည့် function ကို ပြန်ပေးပါမည်။ ထိုသို့ return ပြန်ရာတွင် 10 ဆိုသည့် value ကိုတစ်ခါတည်း ထည့်ပေးလိုက်ပါသည်။ ထိုသို့ထည့်ပေးမှသာ အလုပ်လုပ် မည်ဖြစ်ပြီး မထည့်ပေးပါက error တက်မည် ဖြစ်သည်။
3. နောက်ဆုံးအနေဖြင့် fn ဆိုသည့် function အား 100 ,200 ဆိုသည့် value များထည့်ပြီး ခေါ်လိုက်ပါသည်။ ထို့ကြောင့် line 3 မှ yy and zz နေရာတွင် 100 နှင့် 200 တို့ ဝင်ရောက်လာပါမည်။
4. ထို့နောက် fn function သည် myFun အား return ပြန်ပေးသည့်အတွက် myFun ဆိုသည့် function အား 10 ,100 ,200 တို့ဖြင့် လှမ်းခေါ်ပါတော့သည်။ ထို့ကြောင့် myFun ထဲသို့ 10, 100 ,200 တို့ arguments value အနေဖြင့် ဝင်သွားပြီး ထို data များအား output

အနေဖြင့် ပြန်ပေးခြင်းဖြစ်ပါသည်။

Partial function သည် higher order function လည်းဖြစ်သကဲ့သို့ ပုံမှန်အားဖြင့် argument နှစ်ခုယူပါသည်။ ပထမတစ်ခုသည် function ဖြစ်ပြီး ဒုတိယတစ်ခုသည် pre-set လုပ်ရန် value ဖြစ်သည်။ pre-set လုပ်ရန်ဆိုသည်မှာ value တစ်ခုအား သတ်မှတ်ပေးထားခြင်း ဖြစ်သည်။ အထက်ပါ Program တွင် 10 ကို pre-set value အဖြစ်သုံးထားပါသည်။ အောက်တွင် partial function ကို သုံးပြီး program ရေးပြထားပါသည်။

Sample Program (200)

```
from functools import partial
def myFun(x,y,z):
    print(x,y,z)
pFn = partial(myFun , 10) #line 4
print(pFn(100,200))
```

Partial function သည် return value အနေဖြင့် new function တစ်ခုအား return ပြန်ပေးပါသည်။ ထိုသို့ ပြန်ပေးသည့် function တွင် မိမိတို့ partial function ကို ခေါ်တုန်းက ထည့်ပေးလိုက်သော keyword and arguments တွေကို default အနေဖြင့် function အသစ် မှာ ထည့်ပေးလိုက်ပါသည်။ အထက်ပါ program တွင် line number 4 ၌ ထည့်ပေးလိုက်သော 10 သည် pre-set value or default အနေဖြင့် pFun ဆိုသည့် function အသစ်မှာ ပါဝင်နေမှာ ဖြစ်ပါတယ်။ ထို့ကြောင့် line number 5 တွင် pFn ဆိုသည့် function ကို ခေါ်ရာ၌ 100 , 200 ဆိုသည့် argument နှစ်ခုသာ ထည့်ပေးလိုက်သော်လည်း output တွင် 10, 100, 200 တို့ကို ရနေခြင်းဖြစ်သည်။ အဘယ်ကြောင့်ဆိုသော် 10 သည် pre-set value အဖြစ် partial function မှ ပြုလုပ်ပေးလိုက်သောကြောင့် ဖြစ်သည်။

နောက်ထပ် program တစ်ပုဒ်အနေဖြင့် ရှုပ်ထွေးသည့် arguments များပါဝင်သည့် function တစ်ခုကို partial function သုံးပြီး arguments များ လျှော့ချကြည့်ပါမည်။

Sample Program (201)

```
from functools import partial                          #line 1
def myFun(x , y , *args , x1 ,y1 ,**kwargs):           #line 2
    print(x , y , args , x1 ,y1 , kwargs)              #line 3
pFn = partial(myFun , 10 ,x1='test')                   #line 4
pFn(20 ,30 ,40 ,y1='hello', aa=100 ,bb=200 , cc=300)   #line 5
```

Output:

10 20 (30, 40) test hello {'aa': 100, 'bb': 200, 'cc': 300}

10 သည် pre-set value အနေဖြင့် ထည့်ထားသော value ဖြစ်သည်။ 20 သည် y အတွက် ထည့်ပေးလိုက်သော value ဖြစ်သည်။ 30 and 40 တို့မှာ arbitrary argument အတွက် ထည့်ပေးလိုက်သော value များ ဖြစ်သည်။ test သည်လည်း pre-set အနေဖြင့် partial function ၌ ထည့်ထားပြီးသော value ဖြစ်ပြီး hello သည် y1 အတွက် ထည့်ပေးလိုက်သော value ဖြစ်သည်။ နောက်ဆုံးတွင် ကျန်ခဲ့သော aa, bb, cc တို့သည် **kwargs argument အတွက် ဖြစ်ပါသည်။ အထက်ပါ program တွင် partial function ကိုသုံးပြီး arguments နှစ်ခုကို pre-set လုပ်ကာ လျှော့ချလိုက်ပါသည်။

Partial function က လုပ်ပေးလိုက်သော pre-set value ကိုလည်း ပြန်လည်ချိန်း

နိုင်ပါသေးတယ်။ အောက်တွင် sample program ရေးပြထားပါသည်။

Sample Program (202)

```
from functools import partial          #line 1
def pow(base , exponent):              #line 2
    return base ** exponent            #line 3

sq = partial(pow , exponent =3)        #line 4
print(sq(4))                           #line 5

cu = partial(pow , exponent =2)       #line 6
print(cu(2))                           #line 7

print("test with base 5 expo 2 = ",cu(base=5)) #line 8
```

Output:

64

4

test with base 5 expo 2 = 25

Line number 2 တွင် power ကိုရှာသည့် fun တစ်ခုရေးထားပါသည်။ line number 4 တွင် partial function ဖြင့် exponent အား 3 အဖြစ် preset လုပ်ပါသည်။ ထို့နောက် sq(4) ဟု လှမ်းခေါ်ရာတွင် output အနေဖြင့် 64 ကို ရရှိပါသည်။ အဘယ်ကြောင့်ဆိုသော် 4 သည် base ဆိုသည့် နေရာသို့ရောက်သွားမည်ဖြစ်ပြီး မူရင်း pre-set value အဖြစ်ရှိနေသော 3 ဖြင့် 4 to the power 3 ဖြစ်သောကြောင့် ဖြစ်သည်။ line number 6 တွင် exponent = 2 ဟုရေးပြီးမူရင်း pre-set value အားလှမ်းပြင်ပါသည်။ ထို့ကြောင့် cu(2) ဟုလှမ်းခေါ် လိုက်သော အချိန်တွင် 2 to the power 2 ဖြစ်သွားသောကြောင့် output အနေဖြင့် 4 ကို ရရှိခြင်း ဖြစ်ပါသည်။ line number 8 တွင်မူ base အား 5 ဟု လှမ်းပြင်ပါသည် line number 6 တွင် exponent အား 2 ဟု ပြင်ခဲ့သည့်အတွက် 5 to the power 2 ဖြစ်သွားသောကြောင့် output ကို 25 ဟု ရရှိခြင်း ဖြစ်ပါသည်။

Operator Module

Operator ကို အသုံးပြုနိုင်ရန်အတွက် operator module ကို import လုပ်ပေးရန် လိုအပ်ပါသည်။ operator module ထဲတွင် ပါဝင်သော attribute များကိုသိလိုပါက dir(operator) ဟုရေးပြီး run ကြည့်ပါက သိနိုင်ပါသည်။ အောက်တွင် operator module ကိုသုံးပြီး sample program အနည်းငယ် ရေးပြထားပါသည်။

Sample Program (203)

```
import operator
print(operator.add(10,20))
print(operator.mul(2,3))
print(operator.truediv(3,2))
print(operator.floordiv(13,2))
```

Sample Program (203) တွင် operator module ထဲမှ add, mul , truediv , floordiv တို့ကို သုံးပြထားပါသည်။ add function သည် argument နှစ်ခုအား ပေါင်းပြီး retrun ပြန်ပေးပါသည်။ mul function သည်လည်း argument နှစ်ခုအားမြှောက်ပြီး return အဖြစ်

ပြန်ပေးပါသည်။ truediv function သည် ရှေ့က argument အား နောက်မှ argument ဖြင့် ပြတ်သည်အထိ စားပြီး return ပြန်ပေးပါသည်။ floordiv function သည် စားလဒ်ကိုသာ ပြန်ပေးပါသည်။ ပြတ်သည်ထိ မစားပါဘူး။

ရှေ့ပိုင်းသင်ခန်းစာများ၌ function များလျှော့ချနိုင်ရန် reduce function နှင့် lambda expression တို့ကိုတွဲသုံးခဲ့ပါသည်။ lambda expression ကိုမသုံးပဲ reduce နှင့် operator module ကိုလည်း တွဲသုံးနိုင်သည်။ အသုံးပြုပုံမှာ lambda expression နေရာတွင် operator module ထဲမှ မိမိလုပ်ဆောင်ချက်ပေါ် မူတည်ပြီး မိမိတို့ကြိုက်နှစ်သက်ရာ function ကိုသုံးနိုင်ပါသည်။ sample program ကို အောက်တွင် ကြည့်ပါ။

Sample Program (204)

```
import operator
from functools import reduce
list=[1,2,3,4,5,6]
output = reduce(lambda x,y:x*y ,list)
print("output from reduce and lambda",output)

output2= reduce(operator.mul , list)
print("output from reduce and operator",output2)
```

အထက်ပါ program ကို run ကြည့်သောအခါ output နှစ်ခုလုံးသည် 720 ကို ရရှိပါမည်။

Comparison operator from operator module

Operator module ထဲတွင် comparison operator များလည်း ကျန်ရှိနေသေးသည်။ ပုံမှန် အားဖြင့် Programming language များတွင် ရေးသားသည့် (<) operator သည် python တွင် lt ဆိုသည့် operator နှင့် ဆင်တူပါသည်။ ထို lt function သည် ပုံမှန်အား ဖြင့် argument နှစ်ခု ယူပြီး ပထမတစ်ခုသည် ဒုတိယတစ်ခုထက်ငယ်သလားဟု စစ်ဆေးပါသည်။ အကယ်၍ ငယ်ခဲ့လျှင် True ကိုပြန်ပေးပြီး မငယ်ခဲ့လျှင် false ကို ပြန်ပေးပါသည်။ ထို့ပြင် တူညီလားဆိုသည့် အခါ စစ်ချင်သည့် == အတွက်လည်း is_ function ရှိနေပါသေးသည်။ ထို function သည်လည်း ရှေ့ argument နှင့် နောက် argument တို့ကို စစ်ဆေးပြီး တူညီလျှင် True ပြန်ပေးပြီး မတူလျှင် false ပြန်ပေးပါသည်။ ထို့ပြင် (>) greater than function (gt) နှင့် truth function တို့ကိုလည်း ဖော်ပြထားပါသည်။ truth function သည် argument ထဲ၌ data ရှိသလား မရှိသလား ဆိုတာကို စစ်ဆေးပါသည် ရှိလျှင် True ကို ပြန်ပေးပြီး မရှိလျှင် False ကို return ပြန် ပေးပါသည်။

Sample Program (205)

```
import operator
list=[1,2,3,4,5,6]
output1=operator.lt(10 , 20)
output2= operator.gt(10 , 20)
output3= operator.is_('win','htut')
output4= operator.is_not('win','htut')
output5= operator.truth(list)
print("For lt function output1 :",output1)
print("for gt function output2 :",output2)
print("For is_ function output3: ",output3)
```

```
print("For is_not function output4 :",output4)
print("For truth function output5: ",output5)
Output::
For lt function output1 : True
for gt function output2 : False
For is_ function output3:  False
For is_not function output4 : True
For truth function output5:  True
```

Scope of Variable

Variable တစ်ခုရဲ့ scope ဆိုတာ ထို variable ကို access လုပ်နိုင်ခြင်း မလုပ်နိုင်ခြင်း ကို ဆိုလိုခြင်း ဖြစ်ပါတယ်။ global and local scope ဟူ၍ scope နှစ်မျိုး ရှိပါတယ် global scope ထဲမှာ ကြေငြာထားတဲ့ variable ကို program ရဲ့ မည်သည့်နေရာကမဆို access လုပ်နိုင်သလို local scope ထဲမှာ ကြေငြာထားတဲ့ variable ကိုတော့ သက်ဆိုင်ရာ local scope ကနေပဲ access လုပ်နိုင်ပါတယ်။ Python ရဲ့ global variable ကတော့ function တစ်ခုထဲမှာ ခေါ်သုံးရုံနဲ့ ထို global variable ရဲ့ value ကိုမပြောင်းလဲနိုင်သလို မူရင်း value ကို ထိခိုက်မှုမရှိစေပဲ အသုံးပြုနိုင်ပါတယ်။ အကယ်၍ မူရင်း global value ကို ပြောင်းလဲလိုသောအခါတွင်လည်း global ဆိုသည့် keyword ကိုသုံးပြီး
ပြောင်းလဲနိုင်ပါတယ်။

Sample Program (206)

```
g=10                        #line 1
def myFun(n):               #line 2
    g =20                   #line 3
    v = g ** n              #line 4
    return v                #line 5
print(myFun(2))             #line 6
print('global var g:',g)    #line 7
```

Output:

400

global var g: 10

Sample Program (206) တွင် line 1 ၌ g ဆိုသည့် global variable တစ်ခုကို value 10 ဖြင့် ကြေငြာထားပါသည်။ line 3 တွင် myFun ဆိုသည့် function ထဲ၌ g ကို20 ဆိုသည့် value အသစ်အား assign လုပ်လိုက်ပါသည်။ line 4 တွင်မူ 20*20 ဖြစ်သည့် အတွက် v တန်ဖိုးသည် 400 ဖြစ်ပါသည်။ line number 7 တွင် g တန်ဖိုးအား output ထုတ်ကြည့်သောအခါ 10 ကိုသာ ပြန်လည် ရရှိပါသည်။ line 4 တွင် g value သည် 20 ဖြစ်ခဲ့သော်လည်း line 7 တွင်မူ g value သည် 10 သာ ဖြစ်နေသည်ကို မြင်ရပါမည်။ အဘယ်ကြောင့်ဆိုသော် myFun ဆိုသည့် function တစ်ခု အတွင်း၌ g ဆိုသည့် global variable ကိုယူသုံးခြင်းသာ ဖြစ်သည်။ global variable g ထဲသို့ 20 ဆိုသည့် value အသစ်ကို assign လုပ်သော်လည်း ထို g value သည် local scope တစ်ခုအနေဖြင့် myFun ဆိုသည့် function တစ်ခု အတွင်း၌သာ အသုံးပြုလို့ရမည်ဖြစ်သည် myFun function အပြင်ဘက်သို့ ရောက်သွားသောအခါတွင်မူ မူရင်း value ဖြစ်သည့် 10 ကိုသာ အသုံးပြုလို့ရမည် ဖြစ်ပါသည်။ အကယ်၍ g အား myFun

အတွင်းခ္ဍ လည်း global အနေဖြင့် အသုံးပြုလိုပါက အောက်ပါ program ထဲတွင် ဖော်ပြထားသည့် အတိုင်း global keyword ကို သုံးနိုင်ပါသည်။

Sample Program (207)

```
g=10                        #line 1
def myFun(n):               #line 2
    global g                #line 3
    g =20                   #line 4
    v = g ** n              #line 5
    return v                #line 6
print(myFun(2))             #line 7
print('global var g:',g)    #line 8
```

Sample Program (207) ကို run ကြည့်ပါက output အနေဖြင့် 400 and 20 ကို ရရှိပါမည်။ အကယ်၍ global ကို အရင်မကြေငြာပဲ အသုံးပြုပါက UnboundLocalError တက်မည် ဖြစ်ပါသည်။ အောက်တွင် sample program ကို run ကြည့်ပါ error တက်သည်ကို မြင်ရပါမည်။

Sample Program (208)

```
a =10
def myFun():
    print('global a :',a) #line 3
    a= 'hello world' #line 4
    print(a)
myFun()
```

ထိုသို့ error တက်ရခြင်းမှာ line number 3 တွင် a ကို assignment မလုပ်ခင် အသုံးပြုထားသောကြောင့် ဖြစ်သည်။ line 4 တွင်ရေးထားသည့် အတိုင်း assignment မလုပ်ခင် အသုံးပြုလိုပါက function ထဲတွင် global keyword ကိုသုံးပြီး အရင်ဆုံး ကြေငြာထားသင့်ပါသည်။ ထိုကဲ့သို့ မကြေငြာလိုပါက assignment လုပ်မည့် instruction ကို အရင်ဆုံးရေးပြီး ထို instruction အောက်တွင်မှ global variable ကို အသုံးပြုရပါမည်။ အောက်တွင် sample program ရေးပြထားပါသည်။ ထို program ထဲတွင် assignment အရင်လုပ်ပြီးမှ အသုံးပြုသည်ကို သတိပြုပါ။

Sample Program (209)

```
a =10
def myFun():
    a= 'hello world'
    print('global a :',a)
myFun()
```

Nonlocal Variables

Nonlocal variables တွေကို inner function or nested function တွေထဲမှာ အသုံးပြုပါတယ်။ အသုံးပြုပုံသည် global နှင့် ဆင်တူသော်လည်း global သည် မည်သည့် function ထဲမှာမှ ကြေငြာထားခြင်း မဟုတ်ပဲ nonlocal ကတော့ function တစ်ခုအတွင်းမှာ ကြေငြာပါတယ်။ nonlocal အား ရှင်းလင်းစွာသိစေနိုင်ရန် program တစ်ပုဒ်ရေးကြည့်ပါမည်။ ထို program တွင် function နှစ်ခု ပါဝင်ပါမည်။ ပထမတစ်ခုသည် outer function ဖြစ်ပြီး ဒုတိယ function သည် ထို outer function ထဲတွင်ပင် အသုံးပြုမည့် inner function ဖြစ်ပါသည်။

Sample Program (210)

```
def outerFun():              #line 1
    d = 'green'              #line 2
    def innerFun():          #line 1
        d = 'python'         #line 4
        print('inner: ',d) #line 1
    innerFun()               #line 6
    print('outer:',d)        #line 7
outerFun()                   #line 8
```

Output :

```
inner:  python
outer : green
```

Sample Program (210) တွင် outerFun ထဲ၌ d = 'green' ဆိုသည့် variable တစ်လုံးကို ကြေငြာထားသည်။ ထို variable ကို line 4 တွင် python ဆိုသည့် data အသစ် ထပ်ထည့်ပါသည် ။ line number 6 တွင် innerFun ကို လှမ်းခေါ်သောအခါ၌ line 5 ကို အလုပ်လုပ်သည့်အတွက် ပထမ ဆုံး output အနေဖြင့် inner: python ဆိုသည့် output ကိုရခြင်းဖြစ်ပါသည်။ line number 8 တွင် outerFun ကို ခေါ်သောအခါ line 7 မှ output သည် green ကို သာရရှိပါသည် line 4 တွင် ပြောင်းလဲလိုက်သော python သည် innerFun ထဲ၌သာ အကျိုးသက်ရောက်ပြီး outerFun တွင်မူ အကျိုးသက်ရောက်မှုမရှိပါ။ variable d သည် global variable မဟုတ်သော်လည်းပဲ ပြောင်းလဲ နိုင်ခြင်းမရှိပါ အဘယ်ကြောင့်ဆိုသော် scope မတူသောကြောင့် ဖြစ်သည်။

Nested function များထဲ၌ ကြေငြာထားသော variable ကို ပြောင်းလဲလိုပါက nonlocal ဆိုသည့် keyword ကိုသုံးပြီး ပြောင်းနိုင်ပါသည်။

Sample Program (211)

```
def outerFun():                 #line 1
    d = 'green'                 #line 2
    def innerFun():             #line 3
        nonlocal d              #line 4
        d = 'python'            #line 5
        print('inner: ',d)      #line 6
    innerFun()                  #line 7
    print('outer:',d)           #line 8
outerFun()                      #line 9
```

Sample Program (211) တွင် line number 4 ၌ d variable အား nonlocal keyword ဖြင့် သုံးထားပါသည်။ ထို့ကြောင့် output များကို စစ်ဆေးကြည့်သောအခါ နှစ်ခုလုံးသည် python ဆိုသည့် စာသားများသာ ထွက်လာသည်ကို မြင်ရပါမည်။ nonlocal variable အား ပိုမိုနားလည် စေရန် အောက်တွင် sample program တစ်ပုဒ် ထပ်မံရေးပြထားပါသည်။

Sample Program (212)

```
def outerFun():
    d = 'green'
    def innerFun():
        def innerFun1():
            print('the most inner1:',d)
        innerFun1()
        nonlocal d
        d = 'python'
        print('inner: ',d)
    innerFun()
    print('outer:',d)
outerFun()
```

Closure

Parameter pass လုပ်ခြင်း မရှိပဲ function တစ်ခုအတွင်းသို့ data ထည့်ခြင်းကို closure လို့ ခေါ်ပါတယ်။ closure သည် function object တစ်ခုဖြစ်ပြီး ထည့်ပေးလိုက်တဲ့ data ကိုလည်း သိမ်းထားနိုင်ပါတယ်။ ပိုမိုရှင်းလင်းစေရန် အောက်ပါ program တွင် ကြည့်ပါ။

Sample Program (213)

```
def myFun1():
    data = 'i am belong to myFun1'
    def myFun2():  #nested function
        print(data)
    return myFun2
```

အထက်တွင် ဖော်ပြထားသော sample program ၌ nested function ဖြစ်သည့် myFun2() ကို ခေါ်ထားတာမျိုး မရှိပါဘူး။ သို့သော် myFun1() ဆိုသည့် function ကို ခေါ်သောအခါတွင်မူ myFun2 ကို return အနေဖြင့် ပြန်ပေးပါသည်။ ယခု နည်းလမ်းကို အသုံးပြုပြီး myFun2 ဆိုသည့် function တစ်ခုလုံးကို return ပြန်ပေးနိုင်သလို myFun2 ဆိုသည့် function ကို object တစ်ခုထဲ သို့လည်း assign လုပ်နိုင်ပါသည်။ ထို့ပြင် myFun2 ကိုသုံးကာ object များကို မိမိလိုသလို တည်ဆောက်ပြီး အသုံးပြုနိုင်ပါသေးသည်။ အောက်တွင် sample program တစ်ခု ဖော်ပြထားပြီး object အသစ်ထဲသို့ assign လုပ်သည့် အပိုင်းကို သေချာကြည့်ရန် လိုအပ်ပါသည်။

Sample Program (214)

```
def myFun1():                          #line 1
    data = 'i am belong to myFun1'     #line 2
```

```
    def myFun2():  #nested function   #line 3
        print(data)                   #line 4
    return myFun2  #closure           #line 5
obj1= myFun1()                        #line 6
obj1()                                #line 7
```

Sample Program (214) ကို run ကြည့်ပါက i am belong to myFun1 ဆိုသည့် output ကို ရပါမည်။ line 5 မှ myFun2 သည် closure function ဖြစ်ပါသည်။ သူသည် myFun2 ဆိုသည့် function တစ်ခုလုံးကို return ပြန်နိုင်ပါသည်။ ထို့ကြောင့် line 6 တွင် myFun1() ဆိုသည့် function ကိုလှမ်းခေါ်သောအခါ ပြန်ပေးလာသော myFun2 closure ကို obj1 ဆိုသည့် object ထဲသို့ assign လုပ်ပါသည်။ ထို့နောက် obj1() ကို function အနေဖြင့် အသုံးပြုပြီး ခေါ်ကြည့်သည့် အခါ myFun2() function ထဲမှ print(data) ဆိုသည့် statement မှ အလုပ်ထလုပ်ခြင်း ဖြစ်ပါသည်။ ထိုသို့ အလုပ်လုပ်ရာတွင် အသုံးပြုသော data သည် myFun1 မှ data ဖြစ်ပါသည်။ Closure သဘောတရားသည် ဖော်ပြထားသည်လောက်နှင့် မပြည့်စုံသေးပါ ပိုမိုနားလည် နိုင်စေရန် အောက်ပါ source code များကို သေချာ ကြည့်ရန် လိုအပ်ပါသည်။

Sample Program (215)

```
def myFun1():                         #line 1
    data = 'i am belong to myFun1'    #line 2
    def myFun2():  #nested function   #line 3
        print(data)                   #line 4
    return myFun2  #closure           #line 5
obj1= myFun1()                        #line 6
del myFun1                            #line 7
myFun1()                              #line 8
```

Output :

```
Traceback (most recent call last):
  File "test.py", line 8, in <module>
    myFun1()
NameError: name 'myFun1' is not defined
```

Sample Program (215) ကို run ကြည့်ပါက myFun1 is not defined ဆိုတာကို တွေ့ရပါမည်။ အဘယ်ကြောင့်ဆိုသော် line 7 တွင် myFun1 ကို delete လုပ်လိုက်သောကြောင့် ဖြစ်ပါသည်။ အောက်တွင် ဖော်ပြထားသော program ကို ဆက်ကြည့်ရာတွင် line 8 ၌ obj1 ဆိုသည့် object ကို function အနေဖြင့် ပြန်ခေါ်ကြည့်ရာ output ထွက်လာသည်ကို မြင်ရပါမည်။ဆိုလိုသည်မှာ closure သည်မူရင်း function ပျက်သွားသော်လည်း သူ့ထဲတွင်ရှိနေသော data ကို မှတ်ထားပါသည်။ myFun2 ဆိုသည့် closure ထဲမှ data သည် myFun1 ဆိုသည့် outer function မှ data ဖြစ်သည်။ ထို outer function ကိုဖျက်လိုက်သော်လည်း closure သည် data ကို မှတ်ထားခြင်းကြောင့် output ကို ပြန်ရလာခြင်း ဖြစ်ပါသည်။

Sample Program (216)

```
def myFun1():                          #line 1
    data = 'i am belong to myFun1'     #line 2
    def myFun2():  #nested function    #line 3
        print(data)                    #line 4
    return myFun2  #closure            #line 5
obj1= myFun1()                         #line 6
del myFun1                             #line 7
obj1()                                 #line 8
```

Closure ဖြစ်ရန် အောက်ပါ အချက်တို့ လိုအပ်ပါသည်။

- Nested Function တစ်ခု ရှိရန် လိုအပ်ပါသည်။
- Nested Function ထဲမှ variable သည် သူ့အပြင်ဘက် function တွင် ကြေငြာထားသော variable ကို refer လုပ်ထားရမည်။
- Enclosing or Outer Function သည် Nested Function ကို return ပြန်ပေးရပါမည်။

Closure သည် global variables များအသုံးပြုခြင်းကို လျော့နည်းစေပြီး data များကို hidden လုပ်ထားပြီး callback function အနေဖြင့်လည်း အသုံးပြုနိုင်သည်။ ထို့ပြင် closure သည် မိမိတို့ program ထဲ၌ function များ ကို လျော့နည်းစေရန်အတွက်လည်း သုံးနိုင်ပါသည်။

Closure ကို မိမိတို့လိုချင်သော expression များ ရေးပြီးလည်း အသုံးပြုနိုင်ပါသေးသည်။

Sample Program (217)

```
def myFun(n):                                  #line 1
  def adding(data):                            #line 2
    return data+n                              #line 3
  return adding                                #line 4
add10=myFun(10)                                #line 5
print('this will be 10 + 10 =',add10(10))      #line 6
add20 = myFun(20)                              #line 7
print('This will be 20 + 20 =',add20(20))      #line 8
print('This will be 20 + 15 =',add20(add10(5))) #line 9
```

```
Output:
this will be 10 + 10 = 20
This will be 20 + 20 = 40
This will be 20 + 15 = 35
```

Sample Program (217)တွင် enclosing function အား parameter ဖြင့် အသုံးပြုထားပြီး nested function ကိုလည်း parameter ဖြင့် အသုံးပြုထားပါသည်။ enclosing function ကို လှမ်းခေါ်သောအခါတွင် parameter တစ်ခုထည့်ပေးရမည် ဖြစ်ပြီး ထို parameter ကို nested function ထဲတွင် ပြန်သုံးထားပါသည်။ line 5 တွင် myFun ကို လှမ်းခေါ်သောအခါ adding function သည် add10 ဆိုသည့် object ထဲသို့ assign

လုပ်လာမည်ဖြစ်ပြီး add10(10) ဟု ပြန်ခေါ်လိုက်သောအခါတွင် adding function ရဲ့ expression အရ data + n ကို ပေါင်းပေးပြီး return အနေဖြင့် ပြန်ပေးပါသည်။ ဤနည်းကို အသုံးပြုပြီး မိမိတို့အသုံးပြုလိုသော arithmetic operation များ ကို closure ကိုသုံးပြီးရေးထားနိုင်ပါသည်။

__closure__ Attribute

Function object တွေမှာဆိုရင် __closure__ attribute ရှိပါတယ်။ ဥပမာ myFun.__closure__ လို့ရေးမယ်ဆိုရင် myFun သည် closure function ဖြစ်နေမည်ဆိုလျှင် cell object ကို tuple အနေဖြင့် return ပြန်ပေးပါတယ်။

Sample Program (218)

```
def myFun(n):                                   #line 1
  def adding(data):                             #line 2
    return data+n                               #line 3
  return adding                                 #line 4
add=myFun(10)                                   #line 5
print(add.__closure__)                          #line 6
```

Output:

```
(<cell at 0x01ACF8F0: int object at 0x61666540>,)
```

Sample Program (218) ကို run ကြည့်သောအခါ output အနေဖြင့် cell object ကို ပြန်ပေးပါသည်။ အဘယ်ကြောင့်ဆိုသော် add သည် closure function ဖြစ်နေသောကြောင့် ဖြစ်သည်။ ထို့ပြင် မိမိတို့အသုံးပြုနေသည့် closure function ထဲတွင် မည်သည့် data or value ရှိနေသည်ကိုလည်း အောက်တွင်ဖော်ပြထားတဲ့နည်းလမ်းအတိုင်းရေးပြီး သိရှိနိုင်ပါသေးသည်။

Sample Program (219)

```
def myFun(n):                                   #line 1
  def adding(data):                             #line 2
    return data+n                               #line 3
  return adding                                 #line 4
add=myFun(10)                                   #line 5
print(add.__closure__)                          #line 6
print(add.__closure__[0].cell_contents)         #line 7
```

Sample Program (219) ကို run ကြည့်မည်ဆိုလျှင် output အနေဖြင့် 10 ကို ရရှိမှာ ဖြစ်ပါတယ်။ အဘယ်ကြောင့်ဆိုသော် line 5 တွင် enclosing function ထဲသို့ 10 ကို ထည့်ပေးလိုက်သောကြောင့် cell_contents ကိုသုံးပြီး closure ထဲမှ ထုတ်ကြည့်သည့်အခါ 10 ကို မြင်ရခြင်း ဖြစ်ပါသည်။

Decorator

Decorator ဆိုတာ မူရင်းရှိပြီးသား function or object တစ်ခုထဲကို လုပ်ဆောင်ချက် အသစ်တွေ ထပ်ထည့်လိုတဲ့အခါမှာ အသုံးပြုပါတယ်။ ထိုသို့ လုပ်ဆောင်ချက်တွေ

ထပ်ထည့်တဲ့ အခါမှာ မူရင်းရှိပြီးသား function or object တွေထဲမှာရှိတဲ့ code structure ကို modify or ပြုပြင် ပြောင်းလဲခြင်းမရှိပါဘူး။ Decorator တွေဟာ map function တို့လို function တစ်ခုကို argument အနေဖြင့် ယူပါတယ်။ return အနေဖြင့်တော့ closure တစ်ခုကို ပြန်ပေးပါတယ်။

Sample Program (220)

```
def myFun(x,y):
  print(x/y)
myFun(5,10)
```

Sample Program (220) ကို run ကြည့်မည်ဆိုလျှင် output အနေဖြင့် 0.5 ကို ရရှိပါမည်။ line 3 function call တွင် ငယ်သည့် ဂဏန်းသည် ရှေ့တွင်ရှိနေပြီး ကြီးသည့်ဂဏန်းသည် နောက်တွင် ရှိနေပါသည်။ အကယ်၍ မိမိတို့အနေဖြင့် user ထည့်ပေးလိုက်သော value တွေကို ကြီးသည့် ဂဏန်းရှေ့မှာထားလိုပြီး ငယ်သည့် value ကိုနောက်တွင် ထားလိုသောအခါမျိုးတွင် မူရင်း function ဖြစ်သည့် myFun အတွင်း၌ code structure ကို ပြုပြင်ပြောင်းလဲရပါမည်။ သို့သော် မူရင်း function ကို ပြုပြင်ပြောင်းလဲဖို့ ခွင့်မပြုထားသော အခြေနေမျိုးမှာဆိုလျှင် မည်သို့ပြုလုပ်မည်နည်း။ ထိုသို့သော အခြေနေမျိုးမှာဆိုလျှင် decorator ကို အသုံးပြုနိုင်သည်။

အောက်ပါ sample program ကို ကြည့်ပါ။

Sample Program (221)

```
def myFun(x,y):  # မူရင်း မပြောင်းလဲ လိုသော function
  print(x/y)

def working(func):    #line 3
  def inner(x,y):     #line 4
    if x< y:          #line 5
      x,y = y, x      #line 6
    return func(x,y) #line 7
  return inner        #line 8

result=working(myFun) #line 9
result(5,10)          #line 10
```

Output:: 2.0

အထက်ပါ program တွင် line 1 and 2 သည် မူရင်းမပြောင်းလဲလိုသော function ဖြစ်ပြီး line 3 တွင် working ဆိုသည့် decorator တစ်ခု တည်ဆောက်ထားပါသည်။

- ပထမဆုံး line 9 ကို စ run သော အခါတွင် working ကို လှမ်းခေါ်ပါသည်
- Working သည် inner ကို return ပြန်ပါသည်။
- ထို့ကြောင့် result ထဲတွင် inner ရောက်ရှိလာပါသည်။
- Line 10 တွင် result ထဲ၌ parameter အနေဖြင့် 5 ,10 ကို ထည့်ပေးလိုက်ပါသည်။ ထို့ကြောင့် line 4 မှ inner function ကို ထည့်ပေးလိုက်သော parameter ဖြင့် သွားခေါ်ပါသည်။
- x နေရာတွင် 5 ဝင်ရောက် လာမည်ဖြစ်ပြီး y နေရာတွင် 10 ဝင်ရောက်လာပါသည်။

- Line 5 တွင် x တန်ဖိုးသည် y တန်ဖိုးထက် ငယ်နေသည့် အတွက် if condition အောက်မှ line 6 statement ကို အလုပ်ဆက်လုပ်ပါသည်။
- Line 6 တွင် x တန်ဖိုးနှင့် y တန်ဖိုးကို ချိန်းလိုက်ပါသည်။
- Line 7 တွင် func(x , y) ဆိုပြီး return ပြန်ပေးပါသည်။ func သည် myFun အစား parameter ဖြစ်သည့်အတွက် line 1 မှာ myFun ကို x ,y parameter ဖြင့် သွားခေါ်ပါသည်။
- Line 2 တွင် x အား y ဖြင့် စားပြီးရလာသော တန်ဖိုးကို print ထုတ်ပေးလိုက်ပါသည်။ Line 3 မှ line 8 ထိ သည် decorator တစ်ခုဖန်တီးခြင်းဖြစ်ပြီး မိမိတို့မထိခိုက်လိုသော မူရင်း function ဖြစ်သည့် myFun ကိုလည်း သွားရောက် modify လုပ်ခြင်းမရှိပါဘူး။

Counter Application with Decorator

Decorator ကိုအသုံးပြုပြီး counter application တစ်ခုရေးသားပါမည်။ ပထမဆုံး အနေဖြင့် closure တစ်ခုရေးသားပါမည်။ ထို closure ၏ inner function ထဲတွင် nonlocal variable ကိုသုံးပြီး count ဆိုသည့် variable တစ်ခုတည်ဆောက်ပါမည်။ ထို nonlocal variable ကိုသုံးပြီး မည်သည့် function ကအကြိမ်ရေတွက် ဘယ်လောက်ခေါ်သလဲဆိုတာကို မှတ်ထားမှာ ဖြစ်ပါတယ်။ ထို့ပြင်ယခု program ၌ parameter ပြဿနာများကို ဖြေရှင်းနိုင်ရန် arbitrary argument နှင့် keyword only argument တို့ကို သုံးပါမည်။

Sample Program (222)

```
def counter(func):                    #line 1
  count=0                             #line 2
  def inner(*args ,**kwargs):         #line 3
    nonlocal count                    #line 4
    count +=1                         #line 5
print('Function{0}wascalled {1} times'
.format(func.__name__,count))
#line 6
    return func(*args , **kwargs)   #line 7
  return inner                      #line 8

def add(a , b =0):                  #line 9
  """
  adding two values using decorator
  """
  print(a+b )                      #line 10
result=counter(add)                #line 11
result(10 ,20)                     #line 12
```

Output:

Function add was called 1 times
30

Sample Program (222) တွင် line 6 ၌ add function ကို ဘယ်နှစ်ခေါက်ခေါ်သလဲဆိုတာ သိနိုင်ရန် count ကို print ထုတ်ထားသည်။ ထို့ပြင် function name ကိုပါ သိနိုင်ရန် __name__ attribute ကိုသုံးထားပါသည်။ program ကို run ပြီးချိန်၌ output တွင် add function အား 1 times ခေါ်သည် ဆိုတာကို တွေ့ရပါမည်။ ပိုမိုရှင်းလင်းစေရန် အောက်ပါ code line တစ်ကြောင်းအား ထပ်ထည့် ကြည့်ပါ ။

```
print('add for second calling ',result(20 ,30))
```

Program ကို ပြန် run ပြီးချိန်၌ output တွင် add function အား နှစ်ကြိမ်ခေါ်သည် ဆိုတာကို မြင်ရပါမည်။

@Property

@Property ကိုတော့ decorator များနှင့်တွဲဖက်ပြီး အသုံးပြုပါတယ်။ ထိုသို့ အသုံးပြုခြင်းအားဖြင့် ဖတ်ရလွယ်ကူစေသလို ရေးရတာလည်း ရိုးရှင်းပါတယ်။ အသုံးပြုပုံကို အောက်မှာ ဖော်ပြထားပါတယ်။

Sample Program (223)

```
def dec_1(fn):
    def inner():
        print('running decorator 1')
        return fn()
    return inner

def dec_2(fn):
    def inner():
        print('running decorator 2')
        return fn()
    return inner
@dec_1                          #line 12
@dec_2                          #line 13
def myFun():
    print('Running myFun')
myFun()
#myNewFun = dec_1(dec_2(myFun))  Line 17
```

Output:
running decorator 1
running decorator 2
Running myFun

Sample Program (223) ၌ line 12 and 13 တွင် @property ကိုသုံးထားပါသည်။ @

နောက်တွင် မိမိတို့ အသုံးပြုလိုသော decorator name ကို ထည့်ပေးရသည်။ @property ကို မိမိတို့အသုံးပြုလိုသော function ရဲ့ အပေါ်မှာ ရေးပေးရပါသည်။ ထိုသို့ ရေးပေးလိုက်လျှင် ဥပမာ myFun() ဆိုသည့် function အပေါ်မှာ ရေးသည်ဆိုပါစို့။ dec_1(myFun) ယခုပုံစံနှင့် ဆင်တူသွားပါသည်။ အထက်ပါ program အတိုင်း function တစ်ခုအပေါ်မှာ decorator နှစ်ခု ရေးထားသည် ဆိုပါစို့ ။ dec_1(dec_2(myFun)) ယခု ပုံစံနှင့် ဆင်တူပါသည်။ သို့သော် သတိပြုရန် အချက်မှာ ပထမဆုံးအနေဖြင့် ပထမဆုံး decorator ကိုသာ အလုပ်လုပ်ဆောင်ပါသည်။ အထက်ပါ program တွင် dec_1 ကို ဦးဆုံးအလုပ်လုပ်ပါမည်။ ပြီးမှ dec_2 နောက်ဆုံးမှ myFun ကို အလုပ်လုပ်ပါသည်။

Wraps

Decorator function တွေနဲ့ wraps ကို တွဲသုံးပါတယ်။ ဘာကြောင့် သုံးပေးရလည်းဆိုတော့ function တစ်ခုဟာအခြား decorator function တစ်ခုစီသို့ parameter အနေဖြင့် pass လုပ်တဲ့အချိန်မှာ ထို function ရဲ့ docString ဟာပျောက်သွားပါတယ် (သို့မဟုတ်) None ဟုပေါ်နေတာကို မြင်ရပါတယ်။ ထိုကဲ့သို့သော အခြေနေမျိုးမှာဆိုရင် wraps ကို သုံးပေးရပါတယ်။ wraps function သည် functools module ထဲမှ ဖြစ်သည့် အတွက် အသုံးပြုမည် ဆိုလျှင် functools ကိုအရင်ဆုံး import လုပ်ပေးရပါတယ်။ အောက်တွင် wraps ကို မသုံးပဲ sample program တစ်ပုဒ်ရေးပြထားပါတယ်။

Sample Program (224)

```
def log(fn):                                    #line 1
    def with_log(*args , **kwargs):             #line 2
        print(fn.__name__+"was called")         #line 3
        return fn(*args , **kwargs)             #line 4
    return with_log                             #line 5
@log                                            #line 6
def f(x):                                       #line 7
    """Some operation"""                       #line 8
    return x+x+x                                #line 9
f(5)                                            #line 10
print(f.__name__)                               #line 11
print(f.__doc__)                                #line 12
```

Output:

```
fwas called
with_log
None
```

Sample Program (224) ကို run ကြည့်မည် ဆိုလျှင် line number 11 ၌ function ရဲ့ name ကို ခေါ်ရာတွင် function name ကို ပြန်ပေးသော်လည်း function ရဲ့ doc ကို ခေါ်ကြည့်သောအခါ None ကိုသာ ရပါသည်။ အဘယ်ကြောင့်ဆိုသော် ထို function ရဲ့ doc string ပျောက်သွားသော ကြောင့် ဖြစ်သည်။ ထိုပြဿနာကို အောက်ပါ program တွင် ဖြေရှင်းပေးထားပါသည်။

Sample Program (225)

```
from functools import wraps                    #line 1
def log(fn):                                   #line 2
    @wraps(fn)                                 #line 3
    def with_log(*args , **kwargs):            #line 4
        print(fn.__name__+"was called")        #line 5
        return fn(*args , **kwargs)            #line 6
    return with_log                            #line 7
@log                                           #line 8
def f(x):                                      #line 9
    """does some math"""                       #line 10
    return x+x+x                               #line 11
f(5)                                           #line 12
print(f.__name__)                              #line 13
print(f.__doc__)                               #line 14
```

Sample Program (225) ကို run ကြည့်သောအခါ function name ရော function ရဲ့ docstring နှစ်ခုလုံး ပေါ်လာသည်ကို မြင်ရပါမည်။ line number 3 တွင် paramerter အနေဖြင့် pass လုပ်လာသော function အား wraps ထဲသို့ ထည့်ထားသောကြောင့် ဖြစ်သည်။ ထိုကြောင့် wraps သည် function တစ်ခုရဲ့ name, docstring and arguments list တွေကို coping လုပ်ရန်အသုံး ပြုနိုင်ပါသည်။

Logger Application

Decorator function ကို အသုံးပြုပြီး logger application တစ်ခုစတင် ရေးသားကြည့်ပါမည်။ အလုပ်လုပ်ပုံမှာ function ကို ဘယ်နေ့ ဘယ်အချိန်က ခေါ်တယ်ဆိုတာကို အတိအကျ မှတ်ပေးထားတဲ့ ပုံစံမျိုးပါ။ ယခင် သင်ခန်းစာများတွင် module များ ကို အပြင်ဘက် ဆုံးတွင် ရေးသော်လည်းပဲ ယခုအခါတွင်မူ မိမိတို့အသုံးပြုလိုသော decorator function ထဲတွင် ရေးထားပါမည်။ ထိုသို့ရေးရခြင်းမှာ မိမိတို့အနေဖြင့် source code တွေကိုတစ်ခြား module ထဲတွင် သွားပေါင်းထည့်ပြီး အသင့်ခေါ် သုံးနိုင်ရန် ဖြစ်သည်။ ယခု program တွင် datetime module နှင့် functools နှစ်ခုလုံးကိုသုံးမှာ ဖြစ်သည့်အတွက် မိမိတို့တည်ဆောက်လိုက်သော decorator function ထဲတွင် ထို module နှစ်ခုလုံးကို ထည့်ရေးပေးပါမည်။

Program ရှင်းလင်းချက်

Logger Application ဟုရေးထားသည့် ပုံ(page-238)၌ line number 20 တွင် myFun1(10,20) ဆိုသည့် value များထည့်ပြီး ခေါ် လိုက်ပါသည်။ myFun1 အပေါ် line 12 တွင် logger decorator ကိုခေါ်ထားသည့်အတွက် logger decorator function ဆီသို့ ရောက်သွားပါသည်။ logger function သည် inner function ကို line number 11 တွင် return ပြန်ပေးသည့်အတွက် line number 6 သို့ program ရောက်သွားပါသည်။

Inner function ထဲသို့ရောက်သောအခါ date and time zone ကိုယူပြီး dt ထဲသို့ ထည့်ထားပါသည်။ ထို့နောက် line number 8 တွင် fn သည် myFun1 ဖြစ်သည့်အတွက် myFun1 function ကို သွားခေါ်ပါသည်။ထိုသို့သွားခေါ်ရာတွင် နဂိုမူရင်း 10 , 20 value နှစ်ခုပါ ပါသွားပါသည်။ ထို့ကြောင့် myFun1 ထဲတွင် 10 + 20 အား ပေါင်းပေးပြီး return

ပြန်ပေးပါသည်။ ထိုသို့ return ပြန်ပေးလာသော value ကို result ထဲသို့ ထည့်ထားပါသည်။ line number 9 တွင် datetime , function name နှင့် result မှ ရလာသော value တို့အား print ထုတ်ပေးလိုက်ပါသည် ပြီးလျှင် result ကို return ပြန်ပေးပြီး function အဆုံး သတ်သွားပါသည်။

Sample Program (226)

```
1  def logger(fn):
2      from functools import wraps
3      from datetime import datetime , timezone
4
5      @wraps(fn)
6      def inner(*args , **kwargs):
7          dt = datetime.now(timezone.utc)
8          result = fn(*args ,**kwargs)
9          print('{0} : called {1} : recent Data: {2}'.format(dt , fn.__name__ ,result))
10         return result
11     return inner
12 @logger
13 def myFun1(x, y):
14     """From function 1"""
15     return x+y
16 @logger
17 def myFun2():
18     """From function 2"""
19     return 10
20 myFun1(10,20)
21 myFun2()
22 print(myFun1.__doc__)
```

Logger Application

Sample Program (226) အား run ကြည့်လျှင် output အနေဖြင့် အောက်ပါ တို့ကို ရရှိပါမည်။

2020-04-07 16:18:03.983676+00:00 : called myFun1 : recent Data: 30

2020-04-07 16:18:03.983676+00:00 : called myFun2 : recent Data: 10

From function 1

Timer Application Using Decorator

ဆက်လက်ပြီး Decorator ကိုပဲ အသုံးပြုကာ timer application တစ်ခု ရေးသားသွားပါမည်။ ထို timer app သည် function တစ်ခု အလုပ်ဘယ်လောက်ကြာကြာလုပ်သွားသလဲ ဆိုတာကို သိနိုင်ရန် ဖြစ်သည်။ ထို့ကြောင့် time module ထဲမှ perf_counter ဆိုသည့် function ကို ခေါ်သုံးရပါမည်။ ပထမဆုံးအနေဖြင့် timer decorator function ထဲတွင် logger application ထဲကအတိုင်း functools နှင့် time module တို့ကို import လုပ်ပေးထားရပါမည်။ ပြီးလျှင် inner function တစ်ခု ရေးပြီး

Function call ခေါ်သည့် အချိန်နှင့် ခေါ်ပြီးသည့် အချိန်တို့ကို မှတ်ကာ ကြာချိန်ကို ရယူပါမည်။ ဒုတိယအဆင့်အနေဖြင့် factorial ကိုရှာရန် fact ဆိုသည့် function တစ်ခု တည်ဆောက်ပါမည်။ ထို function ထဲတွင် reduce function နှင့် operator module ထဲမှ mul ဆိုသည့် မြှောက်ပေးသည့် function ကို အသုံးပြုပါမည်။

Sample Program (227)

```
def time(fn):
    from functools import wraps
    from time import perf_counter
    @wraps(fn)
    def inner(*args ,**kwargs):
        start = perf_counter()
        result = fn(*args , **kwargs)
        end = perf_counter()
        timer= end - start
                                    print('{0}    ranfor    {1:.6f}
s'.format(fn.__name__,timer))
        return result
    return inner
@time
def fact(n):
    from operator import mul
    from functools import reduce
    return reduce(mul , range(1,n+1))
print(fact(10))
```

Output::
fact ran for 0.000009 s
3628800

Sample Program (227) သည် Timer application တစ်ခုအား မိမိ ဘာသာ ရေးသား်လိုက်ခြင်း ဖြစ်ပြီး factorial ကို ရှာတဲ့အချိန်မှာ ကြာသည့်အချိန်ကို တွက်ချက်ရန် ဖြစ်သည်။ ထို့ပြင် အချိန် နေ့ရက်များကိုပါ မှတ်သားထားလိုပါက logger decorator ကိုလည်း သုံးနိုင်သည်။ အရင် သင်ခန်းစာတွင် ရေးသားခဲ့သော logger decorator ကို ယခု program ထဲတွင် ကူးထည့်ပြီး @logger ဟု @time အပေါ် သို့မဟုတ် အောက်တွင် မိမိ အဆင်ပြေသလို ရေးနိုင်သည်။

Cache Fibonacci using Decorator

Fibonacci numbers တွေဆိုတာ sequence အလိုက်သာသွားတဲ့ number တွေပါ။ အထက်ပါ ပုံထဲတွင် ပထမဆုံး number ဖြစ်သည့် 0 နှင့် ဒုတိယ number ဖြစ်သည့် 1 တို့ကို ပေါင်းပြီး တတိယ number ရလာပါတယ်။ ထိုနည်းတူပဲ ဒုတိယ number နှင့် တတိယ number တို့ကို ပေါင်းပြီး လေးခုမြောက် number ရလာပါတယ်။ ထိုကဲ့သို့သော sequence အလိုက် Fibonacci number များ ရလာအောင် ရေးတဲ့နည်းများစွာ ရှိတဲ့အထဲကမှ recursive function ကိုသုံးပြီး ရေးတဲ့နည်းဟာ ထင်ရှားပါတယ်။

Fibonacci Using Recursive

Fibonacci number တွေရှာဖို့ function တစ်ခုစရေးပါမည်။ ထိုသို့မရေးခင် condition အနည်းငယ်ကိုတော့သိရန် လိုအပ်ပါသေးသည်။ 3 အောက်ငယ်သည့် number တွေဆိုရင် function ထဲကိုဝင်လာပါက return အနေဖြင့် 1 ကိုသာ ပြန်ပေးပါမည်။ အဘယ်ကြောင့် ထိုသို့ပြန်ပေးရသည်ကို အောက်ပါ program ထဲတွင်ရှင်းပြပါမည်။

Sample Program (228)

```
def fibo(n):
    return 1 if n <3 else fibo(n-1)+fibo(n-2)
print(fibo(6))
```

Sample Program (228) ကို run ကြည့်လျှင် output အနေဖြင့် 8 ကို ရရှိပါသည်။ 8 ဘယ်လို ရလာသလဲဆိုလျှင် အောက်ပါ ပုံကို ကြည့်ခြင်း ဖြင့် သိနိုင်ပါသည်။

User မှ fibo ဆိုသည့် function ထဲသို့ 6 ထည့်ပေးလိုက်သည်။ ထို့နောက် fibo(n-1)+fibo(n-2) n သည် 6 ဖြစ်သည့်အတွက် ပထမဆုံးအနေဖြင့် fibo(5) + fibo(4) ယခုပုံစံအတိုင်း ဆက်မြင်ရမည်။ ထို့ကြောင့် fibo(6) သည် fibo(5) နှင့် fibo(4) နှစ်ခုရပါသည်။ ထိုနည်းတူ fibo(5) ကို ထပ်ခွဲလျှင်(recursive) fibo(4) နှင့် fibo (3) နှစ်ခုကို ရပါသည်။ (ပုံထဲတွင် 5) နေရာကို ကြည့်ပါ။ ထို့ပြင် fibo(4) ကို ထပ်ခွဲလျှင်လည်း fibo(3) နှင့် fibo(2) တို့ ရရှိပါသည်။ fibo(2) ကို ထပ်ခွဲလျှင် fibo(1) and fibo(0) ကို ရပါသည်။ ထို့ကြောင့် 3 ထက် ငယ်နေသော value များဖြစ်လျှင် return အနေဖြင့် 1 ကို ပြန်ပေးခြင်း ဖြစ်ပါသည်။(အထက်ပါ program ၏ return နေရာတွင် ဖော်ပြထားသည်) နောက်ဆုံးတွင် 1 စခါ ရလာပါသည် ။ ထို 1 စခါ လုံးကို ပေါင်းပြီး output အနေဖြင့် 8 ကို ရရှိခြင်းဖြစ်ပါသည်။

ကျွန်ုပ်တို့ အဓိကသိလိုသည့်အချက်မှာ ထိုကဲ့သို့ နောက်ဆုံးအဖြေရလာဖို့ program မှာ function call တွေကို ဘယ်နှစ်ခေါက်ခေါ်ပြီး step တွေ ဘယ်လောက်လုပ်သွားသလဲ ဆိုတာ သိလိုတာပါ။ အောက်ပါအတိုင်း code ကိုအနည်းငယ် modify လုပ်ကြည့်ပါမည်။

Sample Program (229)

```
def fibo(n):
    print('Calculating fibo({0})'.format(n))
    return 1 if n <3 else fibo(n-1)+fibo(n-2)
print(fibo(6))
```

Output::

Calculating fibo(6)

Calculating fibo(5)

Calculating fibo(4)

Calculating fibo(3)

Calculating fibo(2)

```
Calculating fibo(1)
Calculating fibo(2)
Calculating fibo(3)
Calculating fibo(2)
Calculating fibo(1)
Calculating fibo(4)
Calculating fibo(3)
Calculating fibo(2)
Calculating fibo(1)
Calculating fibo(2)
```

Function call ကို 15 ခါတောင် ခေါ်ပြီး အလုပ်လုပ်သွားသည်ကို တွေ့ရပါမည်။ အထက်ပါ output များကိုကြည့်လျှင် 4 ,3 ,2, 1 တို့သည် ထပ်ခါ ထပ်ခါ အလုပ်လုပ်နေသည်ကို မြင်ရပါမည်။ ထို့ကြောင့် memory usage လည်းပိုများသလို အချိန်လည်းပိုကြာပါသည်။ ထိုပြဿနာကို ဖြေရှင်းရန် Decorator ကို သုံးနိုင်ပါသည်။

Sample Program (230) - Fibonacci Using Recursive and Decorator

```
def memoize(fibo):
    cache={1:1 , 2: 1}
    def inner(n):
        if n not in cache:
            cache[n]=fibo(n)
        return cache[n]
    return inner
@memoize
def fibo(n):
    print('Calculating fibo({0})'.format(n))
    return 1 if n <3 else fibo(n-1)+fibo(n-2)
print(fibo(6))
```

```
Output::
Calculating fibo(6)
Calculating fibo(5)
Calculating fibo(4)
Calculating fibo(3)
8
```

Output များကိုကြည့်ပါက program တွင် အဆင့်လေးဆင့်သာ အလုပ်လုပ်သွားသည်ကို တွေ့ရပါမည်။ အဘယ်ကြောင့်ဆိုသော် လုပ်ထားပြီးသားအလုပ်များကို line number 2 မှ cache ထဲတွင် မှတ်ထားသောကြောင့်ဖြစ်သည်။ အထက်ပါ program ကို နားလည်ရန်ဦးစွာ function ထဲကနေ function ကိုပြန်ခေါ်သော recursive function ကိုနားလည်ရန် လိုအပ်ပါသည်။ လုံးဝ အပြည့်အစုံနားလည်နိုင်ရန် debugging လုပ်ပါဟု အကြံပေးလိုပါသည်။

ဆက်လက်ပြီး print(fibo(7)) ဟူသည့် code line ထပ်ထည့်ပြီး run ကြည့်ပါက 7

အတွက် အလုပ်တစ်ခါသာ လုပ်သည်ကို မြင်ရပါမည်။ အဘယ်ကြောင့်ဆိုသော် သူ့အပေါ်မှ line 13 တွင် fibo(6) ထိ cache လုပ်ထားပြီး မှတ်သားထားသောကြောင့် ဖြစ်သည်။

```
1
2   def memoize(fibo):
3       cache={1:1 , 2: 1}
4       def inner(n):
5           if n not in cache:
6               cache[n]=fibo(n)
7           return cache[n]
8       return inner
9   @memoize
10  def fibo(n):
11      print('Calculating fibo({0})'.format(n))
12      return 1 if n <3 else fibo(n-1)+fibo(n-2)
13  print(fibo(6))
14  print(fibo(7))
```

```
Calculating fibo(5)
Calculating fibo(4)
Calculating fibo(3)
8
Calculating fibo(7)
13
PS C:\Users\User\Desktop\Desktop\python>
```

အထက်တွင် ဖော်ပြထားသော ပုံ၌ output များကိုကြည့်လျှင် fibo(7) အတွက်တစ်ကြောင်းသာ အလုပ်လုပ်ပါသည်။ fibo(8) အတွက်ဆိုလျှင် 8 အတွက်တစ်ခုသာ အလုပ်လုပ်ပါတော့မည် ။ ထို့ကြောင့် decorator ကို သုံးခြင်းအားဖြင့် မိမိတို့ program ကို ပိုမို effective ဖြစ်ပြီး မြန်ဆန်အောင် ရေးသားနိုင်ပါသည်။

Factorial with Decorator

Decorator ကိုအသုံးပြုပြီး memoize လုပ်တာကို ပိုမိုနားလည်စေရန် factorial ဖြင့်လည်း အောက်ပါ အတိုင်းရေးကြည့်နိုင်သည်။ factorial ရှာသည့် နေရာတွင်လည်း recursive function ကို သုံးထားပါသည်။

Sample Program (231)

```
def memoize(fibo):
    cache=dict()
    def inner(n):
        if n not in cache:
            cache[n]=fibo(n)
```

```
            return cache[n]
        return inner
@memoize
def fact(n):
    print('Calculating {0}!'.format(n))
    return 1 if n<2 else n*fact(n-1)
print(fact(6))
```

Output:

Calculating 6!

Calculating 5!

Calculating 4!

Calculating 3!

Calculating 2!

Calculating 1!

720

Sample Program (231)တွင် cache နေရာ၌ အသေမှတ်မထားတော့ပဲ dict() ဆိုသည့် dictionary အလွတ်တစ်ခုကို သုံးထားသည် ။

@lru_cache

LRU(Least Recently Used) cache သည် memoizing အတွက် အသုံးပြုနိုင်သည့် Python တွင် ရေးပေးထားသော built in decorator ဖြစ်သည်။ ထို lru_cache အားအသုံးပြုရန် functools module ကို import လုပ်ပေးရန် လိုအပ်ပါသည်။ lru_cache decorator တွင် cache လုပ်နိုင်သည့် max size သည် 128 ဖြစ်ပါသည်။ lru_cache decorator ကို သုံးရာတွင် max size ပေးပြီး အသုံးပြု နိုင်သလို မပေးဘဲနှင့်လည်း အသုံးပြုနိုင်ပါသည်။ မပေးဘဲ သုံးမည်ဆိုလျှင် maxsize=None ကိုသုံးနိုင်သည်။ None သည် အကန့်အသတ်မထားပါဟု ဆိုလိုခြင်းဖြစ်သည်။

Sample Program (232)

```
from functools import lru_cache
@lru_cache(maxsize=9)
def fibo(n):
    print('Calculating fib{0}'.format(n))
    return 1 if n <3 else fibo(n-1)+fibo(n-2)
print(fibo(9))
```

Output:

Calculating fib9

Calculating fib8

Calculating fib7

Calculating fib6

Calculating fib5

```
Calculating fib4
Calculating fib3
Calculating fib2
Calculating fib1
34
```

Sample Program (232) တွင် maxszie ကို ကျွန်တော်တို့အနေဖြင့် 9 ဟုသတ်မှတ်ထားခဲ့သည်။ ထို့ကြောင့် cache ကို 9 ခုပဲမှတ်နိုင်ပါသည်။ ဥပမာ- maxsize ကို 9 ဟုပေးပြီး fibo ကို fibo(18) ဟုပေးလိုက်ပါက 1 မှ 9 ထိ cache အားလုံးသည် ပျက်သွားမည်ဖြစ်ပြီး 10 မှ 18 ထိ cache များကိုသာ မှတ်သားထားပါမည်။

Sample Program (233)

```
from functools import lru_cache
@lru_cache(maxsize=9)
def fibo(n):
    print('Calculating fib{0}'.format(n))
    return 1 if n <3 else fibo(n-1)+fibo(n-2)
print(fibo(9))
print('After printing 9 ...........')
print(fibo(18))
print('After printing 18 ......')
print(fibo(3))
```

Sample Program (233) ကို run ကြည့်ပြီး output များကို စီစစ်ကြည့်လျှင် lru_cache decorator ကိုပိုမိုနားလည်သွားမှာ ဖြစ်ပါတယ်။

```
Calculating fib9
Calculating fib8
Calculating fib7
Calculating fib6
Calculating fib5
Calculating fib4
Calculating fib3
Calculating fib2
Calculating fib1
34
After printing 9 ...........
Calculating fib18
Calculating fib17
Calculating fib16
Calculating fib15
Calculating fib14
Calculating fib13
Calculating fib12
```

```
Calculating fib11
Calculating fib10
2584
After printing 18 ......
Calculating fib3
Calculating fib2
Calculating fib1
2
```

Modules

ကျွန်တော်တို့ Python interpreter ကိုသုံးပြီး program တစ်ပုဒ်ထဲမှာ function တွေ variable တွေ statements and conditions တွေရေးကြတယ်။ပြီးတော့ program run လိုက်တယ်။ Program run ပြီး ပြန်ပိတ်လိုက်တာနဲ့ ကျွန်တော်တို့ရဲ့ codes တွေအားလုံးမရှိတော့ပါဘူး။ အဲ့တော့ ကျွန်တော်တို့ အနေနဲ့ အချိန်အကြာကြီး run ချင်တယ်။ ide or text editor တစ်ခုခုကို သုံးပြီး program အများကြီးရေးထားမယ်။ လိုမှ run ခိုင်းလိုက်မယ်ဆိုရင် အဲဒါကိုတော့ script file လို့ ခေါ်ပါတယ်။ အဲ့တော့ ကျွန်တော်တို့ ရေးလိုက်တဲ့ file ထဲက source code တွေကို အခြားသော program တွေကနေလည်း copy ကူးလိုက်တာမျိုးမဟုတ်ပဲ အလွယ်တကူ ခေါ်သုံးချင်တယ် ။ ခနခန အသုံးပြုနေသော function တွေ decorator တွေကို တစ်ခါပဲရေးထားပြီး ခနခန ခေါ်သုံးချင်တယ်ဆိုရင်တော့ code တွေအားလုံးကို file တစ်ခုထဲမှာ ပေါင်းရေးထားနိုင်ပါတယ်။ ထိုကဲ့သို့သော file ကို module လို့ခေါ်ဆိုခြင်း ဖြစ်ပါတယ်။

ကျွန်တော်တို့အနေနဲ့ ကိုယ်ပိုင် module တစ်ခုကို စတင်ရေးသားပါမည်။ ပထမဆုံးအနေ ဖြင့် myModule ဆိုသည့် project file တစ်ခုတည်ဆောက်မည်။ ထို project file ထဲတွင် python file နှစ်ခုတည်ဆောက်ပါမည်။ ပထမတစ်ခုသည် module file အတွက်ဖြစ်ပြီး ဒုတိယတစ်ခုသည် script file အတွက်ဖြစ်သည်။ ပထမ file ကို mymodule.py ဟုနာမည်ပေးပြီး ထို file ထဲတွင် အောက်ပါ code များကို ရေးသားပါမည်။

Sample Program (234)

```python
def fibo(n):   #write fibonacci series up to n
    """Normal Fibo """ #function annotation
    a,b = 0,1
    while a<n:
        print(a,end=' ')
        a,b=b,a+b
def listFibo(n):
    """List Fibo""" #function annotation
    result=[]
    a,b= 0,1
    while a<n:
        result.append(a)
        a,b= b,a+b
```

```
        return result
```

ဒုတိယ file ကို test.py ဟု နာမည်ပေးပြီး ထို file ထဲတွင် အောက်ပါ source code များကို ရေးသားပါမည်။

```
from mymodule import fibo,listFibo
fibo(10)
print(listFibo(100))
```

အထက်ပါ file နှစ်ခုလုံးအား save ပြီး test.py ဟု run ကြည့်ပါက fibonacci series number များကို ရရှိပါလိမ့်မည်။ ပထမ mymodule.py ဆိုသည့် module file ထဲတွင် function နှစ်ခုကို ရေးထားပါသည်။ ပထမ function သည် fibonacci number များကို ပုံမှန်အတိုင်းပြရန် ဖြစ်ပြီး ဒုတိယ functionသည် list အနေဖြင့်ပြန်ရန်ဖြစ်ပါသည်။ function နှစ်ခုလုံးသည် ထည့်ပေးလိုက်သော parameter မရောက်ခင်အထိ အလုပ်လုပ်မည်ဖြစ်သည်။ test.py ထဲတွင်မူ ပထမဆုံးအနေဖြင့် mymodule ဆိုသည့် module ကို import လုပ်ပါသည်။ ထို module ထဲမှ fibo နှင် listfibo တို့ကို import လုပ်မှာ ဖြစ်သည့်အတွက် from mymodule import fibo,listfibo ဟု ရေးထားခြင်း ဖြစ်ပါသည်။ ထိုသို့ မရေးလိုလျှင် အောက်ပါအတိုင်းလည်း ရေးနိုင်ပါသည်။ မည်သို့ ပင်ရေးစေကာမူ လုပ်ဆောင်ချက်များမှာ အတူတူပင် ဖြစ်သည့်အတွက် output များလည်း အတူတူပင် ဖြစ်ပါသည်။

Sample Program (235)

```
import mymodule
mymodule.fibo(10)
print(mymodule.listFibo(100))
```

Globals Function

Global Function က ယခုလက်ရှိ run နေတဲ့ program file ရဲ့ အချက်အလက်အားလုံးကို ပြပေးပါတယ်။ သူ့ကို အသုံးပြုပုံကတော့ print(globals()) လို့ရေးလိုက်တာနဲ့ရပါတယ်။ သို့သော် key and value များကို ခွဲထုတ်လိုလျှင်တော့ အောက်ပါအတိုင်းလည်း ရေးသားနိုင်ပါတယ်။

Sample Program (236)

```
def print_dict(header , d):
    print('*********{0}*********'.format(header))
    for key,value in d.items():
        print(key , value)
    print('*****end of information******')
print_dict('about globals',globals())
Output::
*********about globals*********
__name__ __main__
```

```
__doc__ None
__package__ None
__loader__ <_frozen_importlib_external.SourceFileLoader object at 0x0140EBF0>
__spec__ None
__annotations__ {}
__builtins__ <module 'builtins' (built-in)>
__file__ mymodule.py
__cached__ None
fibo <function fibo at 0x014894B0>
listFibo <function listFibo at 0x014894F8>
print_dict <function print_dict at 0x01489540>
*****end of information******
```

Output များကို စစ်ဆေးကြည့်ပါက name နေရာတွင် __main__ ကိုတွေ့ရမည် အဘယ်ကြောင့်ဆိုသော် name အားလုံးကို __main__ ဖြင့်သာ ဖော်ပြပါသည်။ file name ကိုတော့ သက်ဆိုင်ရာ file ရဲ့ name ဖြင့် ပြန်ပေးပါသည်။ ထို့ပြင် globals သည် သက်ဆိုင်ရာ file အတွင်းမှာ ရှိသော function များကိုလည်း ပြန်လည်ဖော်ပြပေးပါသည်။ စာရေးသူသည် mymodule.py ဆိုသည့် file ထဲတွင် ရေးထားသည့်အတွက် ယခင်သင်ခန်းစာတွင် ရေးသားထား သော fibo function နှင့် listFibo function တို့ကိုပါ ပြန်လည်ဖော်ပြခြင်း ဖြစ်သည်။

Sample Program (237)

```
__name__='__main__'
print('it will be output under this line...if name == main')
if __name__ == "__main__":
   print('name is equal to main')
print(__name__)
Output ::
it will be output under this line...if name == main
name is equal to main
__main__
```

Sample Program (237) သည် name ကို main အဖြစ်သာ ပြောင်းလိုက်ကြောင်းကို ဖော်ပြထားခြင်း ဖြစ်သည်။

Delete Module

Python တွင် script file တစ်ခုထဲမှာ module တွေကိုများပြီး နှစ်ခေါက် import လုပ်လို့ ရပါတယ်။ သို့သော် မိမိတို့အနေဖြင့် ပိုနေသော module များကိုလည်း del and globals keyword များကိုသုံးပြီး ဖျက်ပစ်နိုင်ပါတယ်။ test.py program ထဲ၌ အောက်ပါ program များကို ရေးရပါမည်။

Sample Program (238)

```
import mymodule
mymodule.fibo(10)
import mymodule
```

အထက်ပါ အတိုင်းရေးပြီး run ကြည့်ပါက program အားလုံး အလုပ်လုပ်သည်ကို တွေ့ရပါမည်။ mymodule ဆိုသည့် module ကိုဖျက်ထုတ်ပြစ်ရန် အောက်ပါ အတိုင်းရေးနိုင်ပါသည်။

Sample Program (239)

```
import mymodule
del globals()['mymodule']
mymodule.fibo(10)
Output:
 File "test.py", line 3, in <module>
        mymodule.fibo(10)
NameError: name 'mymodule' is not defined
```

Line number 1 တွင် module ကို import လုပ်ထားသော်လည်း line number 2 တွင် ဖျက်ပစ်လိုက်သောကြောင့် line number 3 တွင် module ကိုပြန်ခေါ်ရာတွင် မသိတော့သည့်အတွက် not defined ဆိုသည့် NameError တက်ခြင်း ဖြစ်ပါသည်။

Object and Class

Object and Class ကိုနားလည်နိုင်ရန် ပထမဆုံးအနေဖြင့် class တစ်ခု စတင်ဖန်တီးပါမည်။ class တစ်ခုဖန်တီးဖို့အတွက် python မှာ code တွေအများကြီးရေးစရာ မလိုပါဘူး။ class တစ်ခုကို စတင်ဖန်တီးဖို့အတွက် class ဆိုတဲ့ keyword ကို အသုံးပြုရပါမည်။ ထို့နောက်မှ မိမိ အသုံးပြုလိုသည့် name ကို ထည့်ပေးရပါတယ်။ထို name နောက်တွင်တော့ : (colon) နဲ့ ပိတ်ပေးရပါတယ်။

```
class MyClass:
    pass
```

Python ရဲ့ class name ဟာလည်း variable naming rules ကိုလိုက်နာရပါတယ်။ letter သို့မဟုတ် underscore(_) တွေနဲ့ စတင်လို့ရပါတယ်။ သို့သော် (PEP 8) python enhancement proposal မှာတော့ class name တွေရဲ့ ပထမဆုံးအစကို CapWords notation စကားလုံးအကြီးနဲ့ စဖို့ အကြံပေးထားပါတယ်။ အခြားသော programming language တွေမှာ brackets တွေနဲ့ ရေးသော်လည်း python မှာတော့ indentation နဲ့ပဲရေးပါတယ်။ ဆိုလိုချင်တာ ဒုတိယလိုင်းမှာ ရေးထားတဲ့ pass သည် ဘေးနဲ့ space လေးခုခြားပါတယ်။ pass ဆိုတာကတော့ လုပ်ဆောင်စရာ မရှိသေးဘူးလို့ ဆိုလိုချင်တာပါ။

Sample Program (240)

```
class MyClass:
    pass
objA= MyClass()
objB= MyClass()
```

```
print(objA)
print(objB)
#Output::
<__main__.MyClass object at 0x02C20AB0>
<__main__.MyClass object at 0x02C20B90>
```

Object တစ်ခုမှာ သူနဲ့သက်ဆိုင်ရာ class တစ်ခုရှိပါတယ်။ Line 3 မှာ MyClass() ဆိုတဲ့ class ကိုလှမ်းခေါ်ပါတယ်။ ခေါ်တဲ့အချိန်မှာ parentheses တွေပါနေသော်လည်း python သည် class ကိုခေါ်တယ်ဆိုတာ သိပါတယ်။ ထို့ပြင် class ကိုခေါ်တဲ့အချိန်မှာ သူ့ကိုယ်စား object တစ်ခု ဆောက်ပေးပါတယ်။ ထို့ကြောင့် line 3 left sight တွင် objA ဆိုသည့် object တစ်ခုကို ဖန်တီးပေးပါတယ်။ line 4 မှာလည်း ထိုနည်းအတိုင်းပင် MyClass ဆိုသည့် class ကို နောက်တွင် parenthesis များသုံးပြီး လှမ်းခေါ်ပါတယ်။class ကို ခေါ်လိုက်သည့်နှင့် တစ်ပြိုင်နက် သူ့တာဝန်သည် object ဖန်တီးပေးရန် ဖြစ်ပါသည်။ ထို့ကြောင့် objB ဆိုသည့် object တစ်ခုကို ဖန်တီးလိုက်ပါတယ်။ အထက်ပါ နည်းလမ်းများအတိုင်း object နှစ်ခုကို ဖန်တီးပြီးသည့်အခါ ထို object များကို print ထုတ်ကြည့်ရာတွင် မတူညီသည့် memory address များ၌ နေရာ ယူထားသည်ကို တွေ့မြင်ရပါမည်။

Adding attributes

အထက်ပါ lesson မှာ class တစ်ခုဖန်တီးပြီးသော်လည်း ထို class သည် အသုံးမဝင်သေး သလို အသက်လည်း မဝင်သေးပါဘူး။ ထို့ကြောင့် အသက်ဝင်သွားအောင် attributes လေးတွေ စထည့်ကြည့်ပါမည်။ class ထဲကို attributes တွေ ထည့်သည့်အခါမှာ မည်သည့် code မှ class ထဲမှာ ထပ်မပေါင်းပဲ ထည့်နိုင်ပါတယ်။ ထိုသို့ attributes များထည့်ရာတွင် dot notation ကို အသုံးပြုပြီး ထည့်နိုင်ပါတယ်။

Sample Program (241)

```
class MyClass:        #class defination
    pass
objA= MyClass()       #creating object
objB= MyClass()

objA.x=5              #adding attributes #line 5
objA.y=10
objB.x=20
objB.y=30
print('ObjA x={0} and y={1}'.format(objA.x,objA.y))
print('ObjB x={0} and y={1}'.format(objB.x,objB.y))
```

<object>.<attributes> = <value>

အထက်မှာ ရေးထားတဲ့ line 5 မှာတော့ attributes တွေ စထည့်တာပါ။ ထည့်တဲ့အချိန်မှာ ပထမဆုံးမိမိထည့်လိုတဲ့ class ရဲ့ object name ပြီးလျှင် . (dot notation) ပြီးမှ assignment operator ကိုသုံးပြီး value ကို ထည့်ပါတယ်။

Adding Behaviors

Class တစ်ခုထဲမှာ action or behaviors တွေထည့်တယ်ဆိုတာ method တွေထပ်

ထည့်တာပါ။ ယခု class ထဲမှာတော့ အရိုးရှင်းဆုံးဖြစ်အောင် myMethod ဆိုသည့် method တစ်ခုကို ဖန်တီးပါမည်။ ထို method ထဲတွင် x and y ဆိုသည့် attributes များပါ ထည့်ထားပါမည်။

Sample Program (242)

```
class MyClass:          #class defination
    def myMethod(self): #creating a method
        self.x=0
        self.y=0
myObj = MyClass()
myObj.myMethod()
print(myObj.x,myObj.y)
```

Sample Program (242) ကို run ကြည့်ရာတွင် 0 0 တို့ကိုသာရရှိပါမည်။ python မှာ method တစ်ခုကို ဖန်တီးခြင်းဟာ function တစ်ခုကို ဖန်တီးတာနဲ့ ပုံစံချင်းဆင်တူပြီး def ဆိုတဲ့ keyword ကို အသုံးပြုပါတယ်။ ထို့နောက် parenthesis ထဲမှာ parameter တွေ ထည့်ပါတယ် နောက်ဆုံးမှာတော့ colon နဲ့ အဆုံးသတ်ပါတယ်။ syntactically အရ method နဲ့ normal function ကြားက ခြားနားချက်သည် python ရဲ့ method အားလုံးမှာ argument တစ်ခုပါဝင်ပါတယ်။ python ရဲ့ conventionally အရ argument နေရာမှာ self ကိုအသုံးပြုပါတယ်။ python programmer တော်တော်များများဟာ ထို self နေရာမှာ အခြားသော name တွေကို အသုံးမပြုကြ ပါဘူး။ self argument သည် သူနှင့်သက်ဆိုင်သည့် class ကိုသာ ကိုယ်စားပြုထားခြင်း ဖြစ်သည်။ self ဆိုသည့် keyword ကိုသုံးခြင်းအားဖြင့် class တစ်ခုရဲ့ attributes and method တို့ကို access လုပ်နိုင်ပါတယ်။ ဆိုလိုချင်တာက self ဆိုတဲ့နေရာမှာ သက်ဆိုင်ရာ object တစ်ခု pass လုပ်တာပါ။ အောက်ပါ ပုံတွင်ကြည့်ပါ။

Self

Sample Program (243)

```
1  class MyClass:        #class defination
2      def __init__(self,xAxis,yAxis):
3          self.xAxis=xAxis
4          self.yAxis=yAxis
5  o = MyClass(10,20)
6  print(o.xAxis,o.yAxis)
```

Sample Program (243) တွင် line 2 ၌ __init__ ဆိုသည့် special method ကိုသုံးထားသည်။ ထို method သည် သူနဲ့သက်ဆိုင်ရာ class ကိုခေါ်လိုက်သည်နှင့် တစ်ပြိုင်နက် အလိုလျောက် အလုပ်ထလုပ်ပေးပြီး object ထဲသို့ရောက်လာပါသည်။ init method တွင် parameter သုံးခု ရေးထားပါသည်။ သို့သော်ထို method ကိုလှမ်းခေါ်သောအခါတွင် line 5 ၌ parameter နှစ်ခုသာ ထည့်ပေးလိုက်သည်။ သို့သော်ထို program သည် ကောင်းမွန်စွာ အလုပ်လုပ်ပါသည် အကြောင်းမှာ ပထမဆုံး parameter

ဖြစ်သည့် self နေရာသို့ line 5 မှ o ဆိုသည့် object မှ pass လုပ်ပါသည်။ ထို့ကြောင့် line 3 တွင် ရေးထားသည့် self သည် object ကို ညွှန်းထားခြင်း ဖြစ်ပြီး self.xAxis သည် object ထဲမှ xAxis သို့ xAxis ဆိုသည့် value ထည့်မည်ဟု ဆိုလိုခြင်း ဖြစ်ပါသည်။ yAxis သည်လည်း ထိုနည်းတူပင် ဖြစ်သည်။ ထို့ကြောင့် line 6 တွင် o.xAxis နှင့် o.yAxis တို့ print ထုတ်သည့်အခါ 10, 20 တို့ ပြန်လည်ရရှိလာခြင်း ဖြစ်သည်။ class ထဲရှိ method များအတွက်လည်း self ကို သုံးနိုင်သည်။ sample program ကို အောက်တွင်ရေးပြထားပါသည်။

Sample Program (244)

```
1   class MyClass:      #class defination
2       def __init__(self,xAxis,yAxis):
3           self.xAxis=xAxis
4           self.yAxis=yAxis
5       def mInfo(self):
6           print('from info method x={0}:y={1}'.format(self.xAxis,self.yAxis))
7   o1 = MyClass(10,20)
8   o2 = MyClass(30,40)
9
10  print(o1.xAxis , o1.yAxis)
11  o1.mInfo()
12  print(o2.xAxis , o2.yAxis)
13  o2.mInfo()
14
```

Output::

```
10 20
from info method x=10 :y=20
30 40
from info method x=30 :y=40
```

Sample Program (244) တွင် line 5 ၌ mInfo ဆိုသည့် method တစ်ခုကို ဖန်တီးထားပါသည်။ ထို method တွင်လည်း parameter အနေဖြင့် self ကိုသာ သုံးထားပါသည်။ previous program တွင်ရှင်းပြထားသည့်အတိုင်းပဲ ယခု lesson မှာလည်း line 11 မှ o1.minfo() ဟုလှမ်းခေါ်လိုက်သောအခါ ထို o1 ဆိုသည့် object သည် self နေရာသို့ pass လုပ်ပါသည် ။ ထို့ကြောင့် line 6 တွင် self.xAxis ဟု print လုပ်သည့်အခါ 10 ကို ထွက်လာခြင်း ဖြစ်ပါသည်။ ထို့ကြောင့် self သည် special method ဖြစ်သည့် init တွင်လည်း သုံးနိုင်သလို normal method များတွင်လည်း data binding လုပ်ထားသောကြောင့် သုံးနိုင်သည်။ အကယ်၍ self ကို သိပ်နားမလည်သေးဘူးဆိုရင် အောက်ပါ program လေးကို စမ်းရေးကြည့်ပြီး self ရဲ့ memory address နှင့် object ရဲ့ memory address တို့ကို print ထုတ်ကြည့်နိုင်ပါတယ်။

Sample Program (245)

```
class MyClass:      #class defination
    def __init__(self,xAxis,yAxis):
        print('this is from init:',id(self))
        self.xAxis=xAxis
        self.yAxis=yAxis
o1 = MyClass(10,20)
print('this is from outside',id(o1))
```

Sample Program (245) တွင် init method ထဲ၌ print ထုတ်ထားသော memory address နှင့် အောက်ဆုံး စာကြောင်းတွင် print ထုတ်ထားသော o1 ရဲ့ memory address တို့ တူနေသည်ကို တွေ့မြင်ရပါမည်။

Sample Program (246) - Comparing Objects

```
class MyClass:      #class defination
    pass
o1 = MyClass() #constructor
o2 = MyClass()
print('id of o1:',id(o1))
print('id of o2:',id(o2))
```

Sample Program (246) တွင် class တစ်ခုတည်းကို လှမ်းခေါ်တာတူသော်လည်း memory address များမတူကြောင်းကို ပြထားပါသည်။ output ကို စစ်ဆေးကြည့်သောအခါတွင် memory address များမတူကြောင်းကို တွေ့မြင်နိုင်ပါသည်။ ထို့ကြောင့် class တူသော်လည်း object များသည် memory space နေရာများမတူကြောင်းကို သတိပြုပါ။ အခြား programming များတွင်မူ class ရဲ့ ကိုယ်စား object တစ်ခုတည်ဆောက်လိုက်သည်နှင့် တစ်ပြိုင်နက် class ထဲမှ function တစ်ခုသည် အလုပ်ထလုပ်ပါသည်။ ထို function ကို constructor ဟုခေါ်ဆိုပါသည်။ python တွင်မူ special method ဖြစ်သည့် init မှ အလုပ်ထလုပ်ပေးပါသည်။ သို့သော် constructor သည် init မဟုတ်ပဲ constructor သည် class name နှင့် တစ်ပုံစံတည်းတူပြီး parenthesis ပါသည့် MyClass() ဖြစ်သည်။ (အထက်ပါ program တွင်မူ line 3 မှ ဖြစ်သည်)

Update Attribute

Class တစ်ခုထဲမှာရှိတဲ့ attribute တွေကို update လုပ်ချင်ရင် မိမိကြိုက်နှစ်သက်ရာ နည်းလမ်းကို သုံးနိုင်ပါတယ်။ စာရေးသူအနေဖြင့် နည်းလမ်းနှစ်ခုနဲ့ ဖော်ပြပေးပါမည်။ ပထမဆုံး အနေဖြင့် class တစ်ခု တည်ဆောက်မည်။ ထို class ထဲတွင် init method တစ်ခုနှင့် normal method တစ်ခုတည်ဆောက်ပါမည် ။ init method သည် attribute များကို data ထည့်ရန်ဖြစ်ပြီး normal method ကတော့ attrubute များကို update လုပ်ရန် ဖြစ်သည်။

Sample Program (247)

```
class MyClass:      #class defination
    def __init__(self):
        self.name = "WinHtut"
        self.age = 25
        self.hobby="hacking"
```

```
    def update(self):
        self.name = "Vijja"
        self.age ="26"
        self.hobby="research"
h1 = MyClass()
h2= MyClass()
h1.name="htut"
h1.age="27"
h1.hobby="scientist"
print(h1.name ,':',h1.age,':',h1.hobby)
#Output
#htut : 27 : scientist
```

Sample Program (247) သည် class ထဲမှာရှိတဲ့ attribute တွေကို class ကနေပြီး object မှတစ်ဆင့် update လုပ်ခြင်း ဖြစ်ပါတယ်။

Sample Program (248)

```
class MyClass:      #class defination
    def __init__(self):
        self.name = "WinHtut"
        self.age = 25
        self.hobby="hacking"
    def update(self):
        self.name = "Vijja"
        self.age ="26"
        self.hobby="research"
h1 = MyClass()
h2= MyClass()
print('Before Update:')
print(h1.name ,':',h1.age,':',h1.hobby)
print(h1.name ,':',h1.age,':',h1.hobby)
h1.update()
h2.update()
print("After Update:")
print(h1.name ,':',h1.age,':',h1.hobby)
print(h1.name ,':',h1.age,':',h1.hobby)
#Output ::
#Before Update:
#WinHtut : 25 : hacking
#WinHtut : 25 : hacking
#After Update:
#Vijja : 26 : research
#Vijja : 26 : research
```

Sample Program (248) ဖော်ပြထားသည့် program မှာ class ထဲတွင် update ဆိုသည့် method ကို သုံးပြီး attribute များကို update လုပ်ခြင်းဖြစ်ပါသည်။

Class Attribute

Python တွင် ယေဘုယျ အားဖြင့် class attribute 5 ခုရှိပါတယ်။ attribute အားလုံးသည် သူနှင့်သက်ဆိုင်သည့် class ထဲမှ information တွေကို ထုတ်ပြပါတယ်။

- __dict__ class ထဲမှာရှိတဲ့ အချက်လက်တွေအားလုံးကို dictionary ပုံစံမျိုးနဲ့ ပြန်ပြပေးပါတယ်။
- __doc__ class ရဲ့ documentation တွေကို ရလိုတဲ့အခါမှာ သုံးပါတယ်။
- __name__ class ရဲ့ name ကို ရလိုတဲ့အခါမှာ သုံးပါတယ်။
- __module__ class ရဲ့ သူနဲ့သက်ဆိုင်တဲ့ module ကို သိလိုတဲ့အခါမှာ သုံးပါတယ်။
- __bases__သူကတော့ class တစ်ခုရဲ့ base class တွေအားလုံးကို သိလိုတဲ့အခါမှာ သုံးပါတယ်။ return tuple ပုံစံမျိုးနဲ့ ပြန်ပေးပါတယ်။

__dict__ သည်လည်း special method တစ်ခု ဖြစ်ပါတယ်။ class တွေထဲမှာ ရှိတဲ့ အချက်အလက်တွေကို ထုတ်ကြည့်နိုင်ပါတယ်။ အသုံးပြုပုံမှာ <class name >.__dict__ ယခုပုံစံ အတိုင်းရေးခြင်းဖြင့် class ထဲမှာရှိတဲ့ အချက်အလက်အားလုံးကို ပြန်ရမှာ ဖြစ်ပါတယ်။

Sample Program (249)

```
class Myclass:
    variable = "programming"

    def active():
        print(f'Hello from {Myclass.variable}')

print(Myclass.__dict__)
Myclass.active()
```

Output:

{'__module__': '__main__', 'variable': 'programming', 'active': <function Myclass.active at 0x038A94B0>, '__dict__': <attribute '__dict__' of 'Myclass' objects>, '__weakref__': <attribute '__weakref__' of 'Myclass' objects>, '__doc__': None}

Hello from programming

Output များကို စစ်ကြည့်ပါက class ထဲတွင်ရှိသည့် အချက်အလက်များအားလုံးကို မြင်တွေ့ရမှာပါ။

Class Function

ယေဘုယျ အားဖြင့် python ရဲ့ class များအတွက် built-in function 4 ခုရှိပါတယ်။

- getattr(obj , name , default) - class တစ်ခုရဲ့ attribute တွေကို ရလိုတဲ့အခါမှာ သုံးပါတယ်။ parameter အနေဖြင့် သုံးခုယူပါတယ်။
- setattr(obj , name , value) - object တစ်ခုရဲ့ attribute တွေကိုသီးခြား value တွေ

ထည့်လိုတဲ့အခါမှာ အသုံးပြုပါတယ်။ parameter အနေဖြင့် သုံးခုယူပြီး ပထမတစ်ခုသည် object name, ဒုတိယတစ်ခုသည် attribute name ဖြစ်ပြီး တတိယတစ်ခုသည် မိမိသတ်မှတ်ပေးလိုသော value ဖြစ်ပါတယ်။

- delattr(obj , name) - ကိုယ်ဖျက်ချင်တဲ့ attribute တွေကိုဖျက်လိုတဲ့အခါမှာ အသုံးပြုပါတယ်။သူကတော့ parameter နှစ်ခုပဲ ယူပါတယ်။ ပထမတစ်ခုသည် object name ဖြစ်ပြီး ဒုတိယတစ်ခုသည် မိမိဖျက်လိုသော attribute ရဲ့ name ဖြစ်ပါတယ်။
- hasattr(obj , name) - object ထဲမှာ မိမိလိုချင်တဲ့ attribute တွေရှိနေတယ်ဆိုရင် true ကို return ပြန်ပေးပါတယ်။ parameter နှစ်ခုယူပြီး ပထမတစ်ခုသည် object name ဖြစ်ပြီး ဒုတိယတစ်ခုသည် attribute name ဖြစ်ပါတယ်။

Sample Program (250)

oop.py > ...

```
1   class Student:
2       def __init__(self,name,id,age):
3           self.name = name;
4           self.id = id;
5           self.age = age
6
7   sobj = Student("John",101,25)
8   print(getattr(sobj,'name'))
9   setattr(sobj,"age",23)
10  print(getattr(sobj,'age'))
11  print(hasattr(sobj,'id'))
12  delattr(sobj,'age')
13  print(sobj.age)
```

Built in Function
of a Python Class

WinHtut(greenhackers)

Sample Program (250) တွင် line number 1 - 6 ထိ class တစ်ခုကို ဖန်တီးထားပါတယ်။
ထို class ထဲတွင် init special method ကို parameter လေးခုဖြင့် ရေးထားပါတယ်။ line number 7 တွင် student ဆိုသည့် class ကိုသုံးပြီး sobj ဆိုသည့် object တစ်ခုကို တည်ဆောက်ထားပါတယ်။ line 8 တွင် getattr ဆိုသည့် class function ကို စတင်သုံးထားပါတယ်။ ထို function ၏ parameter နှစ်ခုတွင် ပထမတစ်ခုသည် sobj နှင့် ဒုတိယနေရာတွင် attribute name ဖြစ်သည့် name ကိုထည့်ပေးလိုက်ပါသည်။ ထို့ကြောင့် line 8 ကို run ပြီးသောအချိန်တွင် attribute name ရဲ့ data ဖြစ်သည့် John ကိုရရှိမှာ ဖြစ်ပါတယ်။ line 9 မှာတော့ setattr function ကိုသုံးပြီး parameter သုံခုထည့်ပေးလိုက်ပါသည်။ ပထမတစ်ခုသည် မိမိပြောင်းလဲလိုသော attribute ရဲ့ object name ဖြစ်ပြီး ဒုတိယပြောင်းလဲလိုသော attribute ရဲ့ name ဖြစ်ပါသည် တတိယတစ်ခုသည် attribute

ထဲတွင်ထည့်လိုသော value or data ဖြစ်ပါသည်။ ထို့ကြောင့် line 9 ကို run ပြီးသောအခါတွင် sobj ဆိုသည့် object ထဲ၌ age ရဲ့ value သည် 23 သို့ပြောင်းလဲ သွားမှာ ဖြစ်ပါတယ်။

Line 10 မှာတော့ မိမိတို့ line 9 တွင် ပြောင်းလဲပေးလိုက်သော attribute ကို ပြန်လိုချင်သည့်အတွက် getattr ဆိုသည့် function ကိုသုံးထားပြီး ထို function သည် parameter အနေဖြင့် ပထမနေရာတွင် object name ကိုယူပြီး ဒုတိယနေရာတွင် attribute name ကို ယူပါသည်။ ထို့ကြောင့် line 10 ကို run ပြီးသောအခါတွင် output အနေဖြင့် 23 ကိုရရှိမှာ ဖြစ်ပါတယ်။

Line 11 တွင် hasattr function ကိုသုံးထားပါသည်။ ထို function သည် ပထမ parameter အနေဖြင့် ထည့်ပေးလိုက်တဲ့ object ထဲမှာ ဒုတိယ parameter အနေဖြင့် ထည့်ပေးလိုက်တဲ့ attribute name ရှိသလားဆိုတာ ကြည့်ပါတယ်။တကယ်လို့ ရှိတယ်ဆိုရင်တော့ True ကို ပြန်ပေးပါတယ်။ ထို့ကြောင့် line number 11 ကို run ပြီးသည့်အချိန်မှာ output အနေဖြင့် True ကို ပြန်ရမှာ ဖြစ်ပါတယ်။ Line 12 မှာတော့ delattr function ကိုသုံးထားပါတယ် delattr function တွင် ပထမ parameter အနေဖြင့် object name ကိုထည့်ပေးလိုက်ပြီး ဒုတိယနေရာတွင် attribute name ကိုထည့်ပေးလိုက်ပါတယ်။ ထို့ကြောင့် line 12 ကို run ပြီးသည့်အချိန်မှာ age ဆိုသည့် attribute ပျက်သွားမှာ ဖြစ်ပါတယ်။ line 13 တွင် age ကိုပြန်ပြီး print ထုတ်ထားပါတယ်။ line 13 တွင် age ကိုဖျက်လိုက်တာ ဖြစ်သည့်အတွက် line 14 ကို run သောအခါ error တက်မှာ ဖြစ်ပါတယ်။ အဘယ်ကြောင့်ဆိုသော် age attribute ပျက်သွားသောကြောင့် ဖြစ်ပါတယ်။

Inheritance

Object-Oriented နယ်ပယ်မှာဆိုရင် inheritance သည် အရေးကြီးသည့် အစိတ်အပိုင်း တစ်ခု ဖြစ်ပါတယ်။ Inheritance သည် မူရင်း class (ပထမဆုံး) မှ attributes and behavoir တွေကို ဒုတိယ class မှတစ်ဆင့် access လုပ် နိုင်ခြင်း ဖြစ်ပါတယ်။ ဥပမာ ကျွန်တော်တို့ အနေဖြင့် parent ဆိုသည့် ပထမ class တစ်ခုတည်ဆောက်မည် ဆိုပါစို့။ ထို parent class ထဲတွင် working() function တစ်ခုနှင့် age, id စတဲ့ variable တွေကို ရေးထားပါမည်။ ပြီးနောက် child ဆိုသည့် class တစ်ခု ထပ်တည်ဆောက်ပါမည်။ ထို class ထဲတွင် hello ဆိုသည့် စာကြောင်းတစ်ကြောင်းသာ print function ကို သုံးပြီး ရေးသားထားပါမည်။ သို့သော် ဒုတိယ class ဖြစ်သည့် child class ကိုရေးရာတွင် class Child(Parent): ဟု ရေးမည်ဆိုလျှင် ပထမ class ဖြစ်တဲ့ parent ထဲမှာ ရှိတဲ့ attributes and behavoir တွေကို ဒုတိယ class မှ ခေါ်ယူ သုံးစွဲခွင့်ရသွားတာ ဖြစ်ပါတယ်။ ထိုသို့သော လုပ်ဆောင်ချက်ကို (ပထမ class ကနေ အမွေဆက်ခံ ရယူခြင်းကို) inheritance ဟု ခေါ် ဆို ခြင်း ဖြစ်ပါသည်။

Inheritance(OOP)

ပထမဆုံး Class ဖြစ်တဲ့ Parent ကိုတော့ Base Class ဟု ခေါ်ဆိုပြီး ဒုတိယ Class ဖြစ်တဲ့ Child class ကိုတော့ Derived Class ဟုလည်း ခေါ်ဆိုပါတယ်။

Sample Program (251)

```
Class derived-class (base - class ):
        Pass
```

Sample Program

```
class Animal:
    def eat(self):
        print("Animal is Eating")

class Dog(Animal):  #animal ဆိုသည့် class မှ derived လုပ်ပါသည်။
    def bark(self):
        print("dog is barking")
obj = Dog()
obj.bark()
obj.eat()
```

Sample Program (251)ကိုကြည့်မည် ဆိုလျှင် Dog Class ကိုသာ အသုံးပြုပြီး obj ဆိုသည့် object တစ်ခု တည်ဆောက်ထားပါသည်။ သို့သော် ထို obj သည် Animal ထဲမှ eat ဆိုသည့် method ကိုရော Dog ထဲမှ bark ဆိုသည့် method ကိုရော နှစ်ခုလုံးကို access လုပ်နိုင်ပါသည်။ အဘယ်ကြောင့်ဆိုသော် Dog class သည် Animal class ကို inheritance လုပ်ထားသောကြောင့် ဖြစ်သည်။ အောက်ပါ program တွင် Base(parent) class ထဲမှ ရှိသော Variable များကိုပါ access လုပ်ထားသည်ကို ပြထားပါသည်။ ပိုမို နားလည်စေရန် အောက်ပါ program ကို စမ်းရေးကြည့် နိုင်သည်။

Sample Program (252)

```
class Animal:
    color ='yello'
    age =2
```

```
    def eat(self):
        print("Animal is Eating")

class Dog(Animal):  #animal ဆိုသည့် class မှ derived လုပ်ပါသည်။
    def bark(self):
        print("dog is barking")
obj = Dog()
obj.bark()
obj.eat()
print(obj.color)
print(obj.age)
```

Multiple Inheritance

Multiple Inheritance

Derived Class တစ်ခုသည် base class များစွာ ရှိနိုင်ပါတယ်။ ဆိုလိုချင်တာ Derived Class တစ်ခုသည် base class များစွာဆီက inheritance လုပ်နိုင်ပါတယ်။ Derived Class name နောက်က bracket ထဲမှာ Base Class name များကိုရေးပေးခြင်းဖြင့် multiple inheritance လုပ်နိုင်ပါတယ်။ ထို့သို့ Multiple Inheritance လုပ်ခြင်းဖြင့် base class များဆီက Attribute and Behaviors တွေ အားလုံးကို derived class မှ access လုပ်နိုင်ပါတယ်။

Syntax of Multiple Inheritance

```
class Base1:
    Pass
class Base2:
    Pass
class Derived(Base1,Base2):
    Pass
```

Sample Program (253)

```
class Base1:
    def Multiplication(self,a,b):
```

```
        return a*b
class Base2:
    def Adding(self,a,b):
        return a+b
class Derived(Base1,Base2):
    def sub(self,a,b):
        return a-b

obj = Derived()
print('From Base1',obj.Multiplication(5,5))
print('From Base2',obj.Adding(5,5))
print('From Derived',obj.sub(10,5))
```

Sample Program (253) တွင် Derived Class တစ်ခုသည် Base Class နှစ်ခုဆီမှ Behavior တွေကို inheritance လုပ်ယူခြင်း ဖြစ်ပါတယ်။

Multi-level Inheritance

Multi - level Inheritance

Multi - level inheritance မှာတော့ class1 ကို class မှ inheritance လုပ်ပါတယ် ။ ထိုနည်းတူ class2 ကိုလည်း class3 မှ inheritance လုပ်ပါတယ်။ ထို့ကြောင့် class3 သည် class1 and class2 မှ attribute and behavior များကို inheritance လုပ်ယူနိုင်ပါတယ်။

Syntax :

```
class Class1:
    pass
class Class2(Class1):
    pass
class Derived(Class2):
    pass
```

Sample Program (254)

```
class Class1:
    def Multiplication(self,a,b):
        return a*b
class Class2(Class1):
    def Adding(self,a,b):
        return a+b
class Derived(Class2):
    def sub(self,a,b):
        return a-b
obj = Derived()
print('From Base1',obj.Multiplication(5,5))
print('From Base2',obj.Adding(5,5))
print('From Derived',obj.sub(10,5))
```

Sample Program (254) မှာဆိုလျှင် Class1 ကို Class2 မှ inheritance လုပ်ပါတယ်။ class2 ကိုမှ Derived Class ကနေ inheritance လုပ်ပါတယ်။ ထို့ကြောင့် နောက်ဆုံး Class ဖြစ်တဲ့ Derived Class သည် Class1 and Class2 နှစ်ခုလုံးမှ attribute and behavior တွေကို inheritance လုပ်နိုင်ပါတယ်။ Derived Class ကိုသုံးပြီး obj ဆိုသည့် object တစ်ခု တည်ဆောက်လိုက်ပါသည်။ ထို object သည် class များထဲမှ methodအားလုံးကို access လုပ်နိုင်ပါတယ်။

Issubclass () method

Issubclass method သည် parameter အနေဖြင့် ထည့်ပေးလိုက်သည့် class များရဲ့ ဆက်နွယ်မှုကို စစ်ဆေးတဲ့အခါမှာ အသုံးပြုပါတယ်။ဥပမာ parent class နှင့် child class နှစ်ခုရှိသည် ဆိုပါစို့။ issubclass(child , parent) ယခု ဖော်ပြထားတဲ့အတိုင်းရေးမည်ဆိုလျှင် child class သည် parent class မှ inheritance လုပ်ထားသည့် sub class or derived class ဖြစ်ပါက return အနေဖြင့် True ကို ပြန်ပေးပါတယ်။ထို့သို့ မဟုတ်ပါက False ကိုပြန်ပေးပါတယ်။

Sample Program (255)

```
class Class1:
    def Multiplication(self,a,b):
        return a*b
class Class2(Class1):
    def Adding(self,a,b):
        return a+b
obj = Class2()
print(issubclass(Class2,Class1))
```

Sample Program (255) ကို run မည်ဆိုလျှင် Class2 သည် Class1 ရဲ့ subclass ဖြစ်တာကြောင့် True ကို ပြန်ရမှာ ဖြစ်ပါတယ်။

isinstance(obj , class) method

Isinstance ကတော့ object and class ကြားမှာရှိတဲ့ ဆက်နွယ်မှုကို စစ်ဆေးလိုတဲ့အခါမှာ သုံးပါတယ်။ ပထမ parameter ဖြစ်တဲ့ object သည် ဒုတိယ parameter ဖြစ်တဲ့ class ကို ကိုယ်စားပြုထားတယ်ဆိုရင် True ကို return ပြန်ပေးပါတယ်။ အကယ်၍ class1 and class2 နှစ်ခုရှိသည် ဆိုပါစို့။ class2 သည် class မှ inheritance လုပ်ထားသည်ဆိုလျှင် class2 ရဲ့ object သည် class 1 and class 2 နှစ်ခုလုံးရဲ့ instance object ဖြစ်ပါတယ်။

Sample Program (256)

```
class Class1:
    def Multiplication(self,a,b):
        return a*b
class Class2(Class1):
    def Adding(self,a,b):
        return a+b
obj = Class2()
print(isinstance(obj , Class2))
print(isinstance(obj , Class1))
```

Encapsulation

Encapsulation သည်လည်း Object-oriented programming ရဲ့ အခြေခံ concept တစ်ခု ဖြစ်ပါတယ်။ Encapsulation ကို အသုံးပြုရခြင်း ရည်ရွယ်ချက်သည် attribute and behavior တွေကို မဆိုင်သည့်နေရာ ခွင့်မပြုထားသည့် နေရာတွေမှ modification လုပ်ခြင်းတွေကို ကာကွယ်ရန် ဖြစ်သည်။

အထက်ပါ ပုံထဲက အတိုင်းပါပဲ ဆေးလုံးလေးထဲမှာ method and variable တွေကို အပြင်ကနေပြင်လို့မရအောင် ကာထားပါတယ်။ ထိုသဘောတရားသည် encapsulation ရဲ့ သဘော တရား ဖြစ်ပါတယ်။ encapsulation မှာ data တွေကို private and protected ဆိုပြီး နှစ်မျိုးရှိပါတယ် ။ private ဆိုရင်တော့ double underscore နှစ်ခုကိုရှေ့က ခံရေးပြီး protected ဆိုရင်တော့ single underscore တစ်ခုတည်းကို name ရဲ့ ရှေ့မှာ ခံထားပါတယ်။

_data => protected

__data => private

C++ programming language မှာဆိုရင်တော့ public , private , protected ဆိုပြီး keyword သုံးမျိုးကို သုံးပါတယ်။ ဥပမာ public အနေနဲ့ အသုံးပြုမည်ဆိုလျှင်ထို public

keyword အောက်မှာ indentation ကိုသုံးပြီး data တွေ method တွေကိုသုံးပါတယ်။ public keyword အောက်မှာ ရေးထားတဲ့ data and method တွေကိုတော့ class အတွင်းအပြင်ကရော access လုပ်နိုင်ပါတယ်။ private အောက်မှာ ရှိတဲ့ data and method တွေကိုတော့ class အတွင်းကနေပဲ access လုပ်နိုင်ပါတယ် ။ protected အောက်မှာ ရှိတဲ့ data and method တွေကိုတော့ ထို class ကို derived လုပ်ထားတဲ့ derived class ကနေတစ်ဆင့် ပဲ access လုပ်နိုင်ပါတယ်။

သို့သော် python ရဲ့ protected ကိုတော့ class ရဲ့ အပြင်ဘက်နေ ခေါ်သုံးလို့ရသလို modify လည်း လုပ်နိုင်ပါတယ်။ အောက်မှာတော့ Python ရဲ့ protected and private ကို program များဖြင့် ပြပေးထားပါတယ်။ Python ရဲ့ Convention အရ protected ကို keyword နဲ့မရေးပဲ single underscore နဲ့ရေးတယ်ဆိုတာကို အထက်မှာ ဖော်ပြခဲ့ပြီးပါပြီ ။ protected ကို အသုံးပြုလျှင် အရေးကြီးဆုံး သိရမည့်အချက်သည် " don't touch this , unless you're a subclass" ကိုယ်က subclass ထဲက မဟုတ်ဘူးဆိုရင် protected နဲ့ရေးထားတဲ့ behavior and attribute တွေကို မထိပါနဲ့တဲ့။ ဆိုလိုချင်တာကတော့ subclass ထဲကပဲသုံးပါ။ အပြင်ကနေ တိုက်ရိုက် ခေါ်မသုံးပါနဲ့လို့ ဆိုလိုတာပါ။

Sample Program (257)

Protected

```
class Parent: #base class
    def __init__(self,age):
        # Protected member
        self._age = age
class Sub(Parent):
    def getData(self):
        print(self._age)
o = Sub(10)
o.getData()
```

Sample Program (257) တွင် Parent class ထဲမှ _age ဆိုတဲ့ attribute ကို အပြင်ကနေလည်း print(o._age) ယခုကဲ့သို့ရေးပြီး access လုပ်နိုင်ပါတယ်။ သို့သော် sub class ကနေတစ်ဆင့် getData ဆိုသည့် ထိုကဲ့သို့သော method များကို တည်ဆောက်ပြီးသား access လုပ်ပါဟု recommend ပေးထားပါတယ်။ private ကတော့ class အပြင်ဘက်ကနေ access လုပ်ဖို့ လုံးဝကို ခွင့်မပြုထားတာပါ။

Private

Private ကိုအသုံးပြုမယ်ဆိုရင်တော့ keyword ကိုသုံးစရာမလိုပါဘူး။ __ (double underscore) နဲ့ အထက်မှာ ဖော်ပြထားပြီး နည်းလမ်းအတိုင်း သုံးနိုင်ပါတယ်။

Sample Program (258)

```
class Parent: #base class
    def __init__(self,age):
```

```
        self.__age = age

p = Parent(30)
print(p.__age)
```

Sample Program (258) အတိုင်းရေးခဲ့မည်ဆိုလျှင် နောက်ဆုံး စာကြောင်းမှာ error တက်မှာပါ __age ဆိုတဲ့ attribute ကိုမသိဘူးလို့ပဲ ပြောမှာပါ။ subclass ကနေတော့ access လုပ်နိုင်ပါတယ်။ အောက်မှာ sample program ရေးပြထားပြီး sub class ထဲမှာ method နှစ်ခုရေးပြထားပါတယ်။ ပထမ data ဆိုတဲ့ method ကတော့ acccess လုပ်ပြထားတာ ဖြစ်ပြီး ဒုတိယ get_data ဆိုတဲ့ method ကတော့ private data ကို modification လုပ်ပြထားပါတယ်။

Sample Program (259)

```
class Parent: #base class
    def __init__(self,age):
        self.__age = age
class Child(Parent):
    def data(self,input):
        self.__age=input
        return self.__age
    def get_data(self):
        self.__age=30
        print(self.__age)
obj = Child(10)
print(obj.data(50))
obj.get_data()
```

Polymorphism

Polymorphism ဆိုတာ poly and morph ဆိုပြီး နှစ်ခုခွဲပြီး နားလည်လို့ရပါတယ်။ poly ကတော့ အများကို ဆိုလိုချင်တာပါ။ morph ကတော့ ပုံစံ(form) ပါ။ programming အနေနဲ့ ပြောမယ်ဆိုရင် method name တွေ အားလုံးဟာ အတူတူပါဘဲ။ သို့သော် လုပ်ဆောင်ချက်တွေ ကွာတာကို ဆိုလိုချင်တာပါ။ ဥပမာ - လူတွေအားလုံးကို လူ(human) လို့ပဲ ခေါ်ပါတယ်။ သို့သော် လူတစ်ယောက်နဲ့ တစ်ယောက် မတူသလို လုပ်ဆောင်ချက်တွေလည်း မတူပါဘူး ။ mgmg and aung aung လူနှစ်ယောက်ရှိသည် ဆိုပါစို့။ သူတို့နှစ်ယောက် human တွေပါပဲ။ human ဆိုတဲ့ method name တွေပေါ့။ mgmg class ထဲက method name ကလည်း human ဖြစ်သလို aung aung class ထဲက method name ကလည်း human ပါပဲ။ ထို human ဆိုသည့် method နှစ်ခုသည် name တူသော်ငြားလည်း အထဲက လုပ်ဆောင်ချက်များသည် မတူပါ။ ထို သဘောတရား သည် polymorphsim ဖြစ်ပါတယ်။ သို့သော် polymorphism မှာ Class's Method , Operator Overloading , Method Overloading , Method Overriding စတဲ့ သဘောတရားများ ရှိနေပါသေးတယ် ။ len ဆိုတဲ့ funtion ကို ကျွန်တော်တို့ သုံးခဲ့ကြဖူးပါတယ် len('winhtut') ဒီလို သုံးမယ်ဆိုရင် ဒါဟာ string ရဲ့ length ကိုရဖို့ သုံးတာပါ။ len([10,20,30]) ဒီလို အသုံးပြုမယ် ဆိုရင်တော့ list ရဲ့ length ကိုရဖို့ သုံးတာပါ။ နှစ်ခုလုံးက len ကို သုံးပါတယ်။ နှစ်ခုလုံးရဲ့ function name က len ပါပဲ။ သို့သော် parameter

တွေကတော့ မတူပါဘူး။ ဒါဟာလည်း polymorphism ရဲ့ သဘောတရားပါပဲ။

Polymorphism with class method

ဒီအခန်းကတော့ အထက်မှာ ရှင်းပြပြီးတဲ့ အတိုင်းပါပဲ။ class name တွေ မတူသော်လည်း method name တွေကတော့ တူနေပါတယ်။ method name တွေ တူနေသော်လည်း လုပ်ဆောင် ချက်တွေကတော့ ကွဲပြားနေပါတယ်။ လူသုံးယောက်နဲ့ ဥပမာပေးပြီး အောက်မှာ program ရေးပြထားပါတယ်။

Sample Program (260)

```
class agag:
    def human(self):
        print('AgAg is a programmer...')
        print('income per month is $2000')
class mgmg:
    def human(self):
        print('MgMg is a SecurityAnalytis...')
        print('income per month is $2500')
class koko:
    def human(self):
        print('KoKo is a DataScientist...')
        print('income per month is $5000')
class People:
    def fun(self,obj):
        obj.human()
agag=agag()
mgmg=mgmg()
koko=koko()
people=People()
mylist=[agag ,mgmg ,koko]
for i in mylist:
    people.fun(i)

output
AgAg is a programmer...
income per month is $2000
MgMg is a SecurityAnalytis...
income per month is $2500
KoKo is a DataScientist...
income per month is $5000
```

Operator Overloading

Overloading ဆိုတာ လုပ်ဆောင်ချက်တစ်ခုပါ။ ဘယ်လိုလုပ်ဆောင်ချက်လဲ ဆိုတော့

Function or method တစ်ခုကို pass လုပ်တဲ့ parameters တွေပေါ်မှာ မူတည်ပြီး ပုံမှန်တိုင်း မရတဲ့ လုပ်ဆောင်ချက်တွေကို မိမိလိုချင်တဲ့ ပုံစံရအောင် ဖန်တီးခြင်း သို့မဟုတ် ပုံမှန် operator တွေနဲ့ လုပ်ဆောင်လို့မရတဲ့ operands တွေကို မိမိလိုချင်တဲ့ လုပ်ဆောင်ချက်တွေ ရအောင် ဖန်တီးခြင်း ဖြစ်ပါတယ်။ အောက်မှာတော့ operator overloading အကြောင်းကို ဖော်ပြထားပါ တယ်။

ပေါင်းခြင်း သို့မဟုတ် နုတ်ခြင်း သို့မဟုတ် arithmetic လုပ်လို့မရတဲ့ operand တစ်ခု သို့မဟုတ် တစ်ခုထက်ပိုတဲ့ operands တွေကို ပေါင်းခြင်း ၊ နုတ်ခြင်း ၊ arithmetic လုပ်လို့ ရအောင် လုပ်ဆောင်ခြင်းကို operator overloading လို့ ခေါ်ပါတယ်။ python programming တွင် 10 + 10 ဆိုသည့် integer နှစ်ခုကို ပေါင်းလျှင် 20 ဆိုသည့် output ကို ပြန်ရသည်။ string နှစ်ခုကို ပေါင်းလျှင် string တွေကိုပြန်ရမှာ ဖြစ်သလို list နှစ်ခုကို ပေါင်းလျှင်လည်း နှစ်ခုလုံးကို ပြန်ရမှာ ဖြစ်ပါတယ်။ ထိုကဲ့သို့ အလုပ်လုပ်နိုင်ရန်အတွက် + နောက်မှာ အလုပ်လုပ်ပေးတဲ့ class တွေထဲက special method တွေရှိပါတယ်။ ဥပမာ 10 နဲ့ 20 ဘယ်လို အလုပ်လုပ်သလဲ ဆိုရင် int.__add__(10,20) ထိုကဲ့သို့ အလုပ်လုပ်ပါတယ်။ + operator တင်မကပဲ အခြားသော operator များအတွက်လည်း သူနဲ့ သက်ဆိုင်ရာ special method တွေရှိပါတယ်။ သို့သော် object တွေအတွက်တော့ မရှိပါဘူး။ object နှစ်ခုကိုဆိုရင် ဘယ်လို arithmetic လုပ်မလဲပေါ့။ ဒီနေရာမှာ ဆိုရင် operator overloading ဆိုတဲ့ နည်းစနစ်ကို သုံးရတာပါပဲ။

Sample Program (261)

```
class Age:
    def __init__(self,age):
        self.age =age
age1 = Age(10)
age2=Age(70)
liveAge= age1.age+age2.age
print(liveAge)
#output
#80
```

Sample Program (261) မှာတော့ age1(obj) ထဲက age နဲ့ age2(obj) ထဲက age တို့ ပေါင်းတာပါ object ထဲက data များကို arithmetic လုပ်ခြင်းဖြစ်ပြီး object တွေကို arithmetic လုပ်တာ မဟုတ်ပါဘူး။ အောက်ပါ program အတိုင်း object တွေကို arithmetic လုပ်ကြည့်မည်ဆိုလျှင် unsupported operand ဆိုတဲ့ error တက်မှာ ဖြစ်ပါတယ်။

Sample Program (262)

```
class Age:
    def __init__(self,age):
        self.age =age
age1 = Age(10)
age2 = Age(70)
liveAge= age1+age2
print(liveAge)
```

Output: unsupported operand error

Operator overloading ကို ရေးသားရန် ပထမဆုံးအနေဖြင့် Age ဆိုတဲ့ class ထဲ၌ special method တစ်ခု ရေးသားပါမည်။ ကျွန်တော်တို့ လုပ်ဆောင်လိုသည်မှာ ပေါင်းခြင်း(+) လုပ်ချင်တဲ့အတွက် __add__ ဆိုတဲ့ method ကို ရေးပါမည်။ parameter အနေဖြင့် object နှစ်ခု ဖြတ်လာမှာ ဖြစ်သည့်အတွက် object နှစ်ခုအတွက် self , self1 ဆိုသည့် argument နှစ်ခုကို ရေးထားပါမည်။ self သည် ပထမ object အတွက် ဖြစ်ပြီး self1 သည် ဒုတိယ object အတွက် ဖြစ်သည်။ ထို့နောက် return အနေဖြင့် ဖြတ်လာသော object နှစ်ခုထဲမှ data တွေကိုပေါင်းပြီး ပြန်ပေးလိုက်မှာ ဖြစ်ပါတယ်။

Sample Program (263)

```
class Age:
    def __init__(self,age):
        self.age =age
    def __add__(self,self1):
        return self.age + self1.age
age1 = Age(10)
age2=Age(70)
liveAge= age1+age2
print(liveAge)
#output
#80
```

Method Overloading

ကျွန်တော်တို့အနေနဲ့ method or function တစ်ခုကို ကိုယ်ပေးချင်တဲ့ parameter တွေပေးပြီး သတ်မှတ်ထားလို့ရသလို ပြန်ခေါ်တဲ့အချိန်မှာလည်း ကိုယ်သတ်မှတ်ထားခဲ့တဲ့ အတိုင်းပဲ parameter တွေ အတိအကျပေးပြီး ပြန်ခေါ်ရပါတယ်။ Method overloading ဆိုတာကတော့ method or function တစ်ခု ကြေငြာထားမယ်။ သို့သော် ကိုယ်ထည့်ပေးချင်တဲ့ parameter တွေထည့်ပြီး method or function တစ်ခုတည်းကိုပဲ မတူညီတဲ့ နည်းလမ်းတွေနဲ့ ခေါ်ခြင်း ဖြစ်ပါတယ်။

Sample Program (264) - Method Overloading with None

```
class Price:
    def payment(self, x = None, y = None):
        if x != None and y != None:
            return x * y
        elif x != None:
            return x * x
        else:
            return 0
obj = Price()
print(obj.payment(10,20))
```

Output::200

Sample Program (264) မှာတော့ method overloading လုပ်နိုင်ရန် payment method ထဲမှာ None များကို အသုံးပြုထားပါတယ်။ ပထမ condition သည် နှစ်ခုလုံး None မဟုတ်လျှင် x and y အား မြှောက်ပြီး return ပြန်ပေးပါသည်။ ဒုတိယ condition တွင်မူ parameter တစ်ခုတည်း ထည့်ပေးလိုက်လျှင် ပထမဆုံး parameter တွင်သာ pass လုပ်ပြီး ထို parameter ကိုပဲ ပြန်မြှောက်ကာ ပြန်ပေးလိုက်မှာ ဖြစ်ပါတယ်။

သို့သော် ထို program အားနည်းချက်သည် parameter များသတ်မှတ်ထားသည်ထက် ပိုများလာလျှင် error တက်သွားမှာ ဖြစ်ပါတယ်။ ထိုသို့သော ပြဿနာကို အောက်ပါအတိုင်း handle လုပ်နိုင်ပါတယ်။ ရှေ့ပိုင်း သင်ခန်းစာများတွင် ဖော်ပြထားပြီးသားဖြစ်သည့် arbitrary argument ကို သုံးနိုင်ပါတယ်။

Sample Program (265)

```
class Price:
    def payment(self,*args):
        a = 0
        count=0
        for i in args:
            count +=1
            a=a+i
                print("There are passed {} parameters:: ".format(count))
        return a

obj = Price()
print(obj.payment(10,20,30,40,50))
```

Sample Program (265)မှာတော့ parameter တွေ ဘယ်လောက်များများ အဆင်ပြေပါတယ်။ arbitrary argument က ကိုင်တွယ်နိုင်ပါတယ်။ passed လုပ်လာတဲ့ parameter တွေကိုလည်း count လုပ်ထားပါတယ်။ သို့သော် ထို program ရဲ့ အားနည်းချက်မှာ string တွေ pass လုပ်လာမည်ဆိုလျှင် error တက်မှာ ဖြစ်ပါတယ်။ဥပမာ string တွေ pass လုပ်လာပြီဆိုလျှင် မြှောက်သည့် အလုပ်ကိုမလုပ်တော့ပဲ ပေါင်းသည့် အလုပ်ကိုသာ လုပ်ပေးသင့်သည်။ ထိုကဲ့သို့ သော ပြဿနာကို ဖြေရှင်းရန် အောက်ပါအတိုင်း arbitrary argument နှင့် conditions များကို တွဲသုံးနိုင်သည်။

Sample Program (266)

```
class Price:
    def payment(self,dataType,*args):
        if dataType =="int":
            a = 0
            count=0
            for i in args:
                count +=1
                a=a+i
            print("There are passed {} parameters:: ".format(count))
```

```
            return a
        else:
            a = ''
            count=0
            for i in args:
                count +=1
                a=a+i
            print("There are passed {} parameters:: ".format(count))
            return a

obj = Price()
print(obj.payment("str",'i ','am',' win ','htut ','greenhackers'))
print(obj.payment('int',10,20,30,40,50,60))
```

Output::

There are passed 5 parameters::

i am win htut greenhackers

There are passed 6 parameters::

210

Sample Program (266) မှာတော့ method တစ်ခုတည်းကိုပဲ ပုံစံမျိုးစုံနဲ့ Method overloading လုပ်ထားပါတယ်။ str အတွက်လည်း ရေးပေးထားပြီး integer များအတွက်လည်း ရေးပေးထားပါတယ်။

Method Overriding

Method Overriding ဆိုတာ Object-Oriented Programming တွေရဲ့ feature တစ်ခု ဖြစ်ပြီး Super- Classes or Parent classes ထဲမှာ ရှိပြီးသားဖြစ်တဲ့ method ကို subclass or child class ထဲမှာ ထပ်ဖန်တီးခြင်း ဖြစ်ပါတယ်။ သို့မဟုတ် Parent class ထဲမှာ ရှိပြီးသား method ရဲ့ name, parameters အတိုင်း subclass ထဲမှာ ထပ်တူဖန်တီး အသုံးပြုခြင်း ဖြစ်ပါတယ်။

Sample Program (267)

```
class Price():
    def payment(self):
        print("From Parent Class")

class Child(Price):
    def payment(self):
        print("From child class")
obj = Child()
obj.payment()
```

Sample Program (267) ကို run ကြည့်မည်ဆိုလျှင် output အနေဖြင့် child class ထဲမှ

From child class ဆိုသည့် စာသားကိုပဲ ရမှာပါ။ Parent class ထဲမှ method name နှင့် child class မှ method name တူနေလျှင် child class ထဲမှ method ကို ခေါ်ပါတယ်။

Sample Program (268) - Method Overriding with Multilevel Inheritance

```
class Parent:
    def payment(self):
        print('This is from Parent Class')
class Child(Parent):
    def payment(self):
        print('This is from Child Class')
class GrandChild(Child):
    def payment(self):
        print('This is from GrandChild Class')
obj = GrandChild()
obj.payment()
```

Sample Program (268) ကို run ကြည့်မည်ဆိုလျှင်လည်း output အနေဖြင့် GrandChild class ထဲမှ payment method ကိုပဲ အလုပ်လုပ်မှာ ဖြစ်ပါတယ်။

Abstraction

ကားမောင်းဖို့အတွက် ကားသော့ကို လှည့်မယ်။ ကားစက်နိုးလာပြီဆိုတာနဲ့ သင့်တော်တဲ့ gear ထိုးပြီး ကျွန်တော်တို့ ကားကိုမောင်းနိုင်ပါပြီ။ ကားသော့လှည့်လိုက်တဲ့အချိန်မှာ battery ကနေတဆင့် starter motor ကို power ဘယ်လိုရောက်သလဲ။ starter motor ဘယ်လိုစလည် သလဲ လောင်စာဆီတွေ ဘယ်လိုပေါက်ကွဲသလဲဆိုတာ ကျွန်တော်တို့မသိလည်း ကားမောင်းနိုင်ပါတယ်။ real world မှာဆိုရင်တော့ ဒါကို information abstraction လို့ခေါ်ပြီး computer science ရဲ့ Programming နယ်ပယ်မှာ ဆိုရင်တော့ abstraction လို့ ခေါ်ပါတယ်။

ဥပမာ နောက်တစ်ခုအနေဖြင့် TV remote ကိုသုံးဖို့ အတွက် အပေါ်က ခလုတ်လေးတွေကို နှိပ်ရုံပါပဲ။ ဒါဆိုရင် TV channel တွေပြောင်းသွားပါမယ်။ remote ထုတ်တဲ့ company က ဒီ ခလုတ်ကို နှိပ်လိုက်ရင် ဘာဖြစ်သွားမယ် ဘာ signal ထွက်သွားမယ် ဟိုဘက်ကနေ signal ကို ဘယ်လို ပြန်ဖမ်းတယ် ဆိုတာလည်း ဖော်ပြမထားပါဘူး။ user ကို အသုံးပြုပုံကိုပဲ အလေးအနက်ထားပြီး ဖော်ပြပါတယ်။ ကျန်တာတော့ ဖော်ပြမထားပါဘူး။ ဒါသည်လည်း abstraction ဖြစ်ပါတယ်။

Python မှာဆိုရင်တော့ abstraction ကို application တစ်ခုရဲ့ လုပ်ဆောင်ချက်တွေကို hiding လုပ်ဖို့သုံးပါတယ်။ ဥပမာ method တစ်ခုပေါ့။ method တစ်ခုထဲမှာ ဘာတွေ ဘယ်လို ရေးထားသလဲ ဆိုတာကို hidden လုပ်ထားပါတယ်။ user က ခေါ်သုံးချင်တယ်ဆိုရင် method ကို ခေါ်သုံးလိုက်ရုံပါပဲ။ အသုံးပြုပုံကိုပဲ အလေးအနက်ထားပြီး ဖော်ပြပါတယ်။ ဘယ်လို လုပ်ဆောင် ထားတယ်ဆိုတာကိုတော့ ဖော်ပြထားခြင်း မရှိပါဘူး။ ဒါသည် abstraction ရဲ့ သဘောသဘာဝ ဖြစ်ပါတယ်။

Abstraction For What?

Abstraction ကို သုံးခြင်းအားဖြင့် programmer တစ်ယောက်က application တစ်ခုရဲ့

data and process တွေကို hidden လုပ်ထားနိုင်သလို ထိုသို့လုပ်ထားခြင်းဖြင့် application ရဲ့ ရှုပ်ထွေးမှုတွေကို လျှော့ချနိုင်ပြီး efficiency ပိုကောင်းအောင်လုပ်နိုင်ပါတယ်။

Python မှာဆိုရင်တော့ Abstract classes and Abstract methods တွေကိုသုံးပြီး Abstraction ကိုအသုံးပြုနိုင်ပါတယ်။

Abstract Method and Class

Class တစ်ခုထဲမှာ abstract method တွေ တစ်ခု သို့မဟုတ် တစ်ခု ထက်ပိုပြီး ပါဝင်နေတာကို abstract class လို့ခေါ်ပါတယ်။ Abstract method တွေထဲမှာ မည်သည့် codes(implementation) တွေမှ မပါဝင်ပါဘူး။ လုပ်ဆောင်ချက်(implementation) များအား abstract class တွေကို inheritance လုပ်ထားတဲ့ sub classes တွေထဲမှာသာ ရေးသားပါတယ်။ Abstract class တွေကို ဖန်တီးဖို့အတွက် abc(abstract class) module ထဲမှ ABC ကို import လုပ်ခြင်းဖြင့် ပြုလုပ်နိုင်ပါတယ်။ ထိုသို့ ပြုလုပ်ပြီးသည့် abstract class ကိုမှ နောက် class တစ်ခုတွင် inheritance လုပ်ပြီး ထို sub class ထဲတွင် implementation များရေးနိုင်ပါတယ်။ Abstract class တစ်ခု ဖန်တီးပုံကို အောက်တွင် ဖော်ပြထားပါတယ်။

```
from abc import ABC
class ClassName(ABC):
    pass
```

ထို abc module ထဲမှပင် abstract method decorator ကိုလည်း သုံးနိုင်ပါသည်။ ပထမဆုံးအနေဖြင့် Abstract class တစ်ခု တည်ဆောက်ပါမည်။ ထို class ထဲတွင် abstract method decorator ကို သုံးထားသည့် payment ဆိုသည့် method နှင့် abs decorator ကို မသုံးထားသော print_slip ဆိုသည့် method နှစ်ခု တည်ဆောက်ထားပါမည်။ သတိထားရန် အချက်မှာ abstract method ရှိနေတဲ့ abstract class ကို ခေါ်လို့ မရပါဘူး။ ခေါ်ကြည့်မည် ဆိုလျှင် Can't instantiate abstract class Price with abstract methods payment ဆိုသည့် error ကို တွေ့ရမှာ ဖြစ်ပါတယ်။

```
from abc import ABC, abstractmethod
class Price(ABC):
    def print_slip(self, amount):
        print('Payment Amout $', amount)
    @abstractmethod
    def payment(self, amount):
        return 'nothing'
p =Price()
```

ထို့ပြင် class နှစ်ခု ထပ်မံဖန်တီးပြီး ထို class များ Price class ကို inheritance လုပ်ထားပါမည်။ program အပြည့်အစုံကို အောက်တွင် ဖော်ပြထားသည်။

Sample Program (269)

```
from abc import ABC, abstractmethod
class Price(ABC):
    def print_slip(self, amount):
```

```
            print('Payment Amout $', amount)
        @abstractmethod
        def payment(self, amount):
            pass
    class CreditCardPayment(Price):
        def payment(self, amount):
            print('Payment With Credit Card $', amount)
    class MobileBanking(Price):
        def payment(self, amount):
            print('Payment With Mobile Banking $', amount)

    obj = CreditCardPayment()
    obj.payment(100)
    obj.print_slip(100)
    obj = MobileBanking()
    obj.payment(200)
    obj.print_slip(200)
```

Super()

Abstract class ထဲမှ abstract method ကို ပုံမှန်အားဖြင့် ပြန်ခေါ်လို့ မရပါဘူး။ ဘာကြောင့် ပြန်ခေါ်လို့ မရလဲဆိုလျှင် subclass များမှ နာမည်တူနေသည့်အတွက် override လုပ်သွားသောကြောင့် ဖြစ်သည်။ inheritance လုပ်ထားတာကနေ abstract method ကို ခေါ်လိုလျှင် super() ကို အသုံးပြုနိုင်ပါတယ်။ ထို့ကြောင့်မို့ abstract method ထဲမှ မိမိတို့ အနေဖြင့် implementation လုပ်လိုလျှင် လုပ်လို့ရပါသေးသည်။ အောက်ပါတွင် super() ကို အသုံးပြုပြီး abstract method ကို ပြန်လည်ခေါ်ပြထားပါသည်။

Sample Program (270)

```
    from abc import ABC, abstractmethod
    class Price(ABC):
        @abstractmethod
        def payment(self):
            print('from abstract method')

    class CreditCardPayment(Price):
        def payment(self):
            super().payment()
            print('Payment With Credit Card $')
    cdc = CreditCardPayment()
    cdc.payment()
```

File Handling

ယခုသင်ခန်းစာကို လေ့လာပြီးရင် file တွေကို ဘယ်လိုကိုင်တွယ်ရမလဲ ဆိုတာ သိသွားမှာပါ။ အထက်ပိုင်းသင်ခန်းစာများတွင် program control များ data types များ နှင့်

data structures တို့ကို လေ့လာခဲ့သော်လည်း ထို program များမှ ရရှိလာသော data များကို နောက်တစ်ကြိမ် ပြန်လည်အသုံးပြုရန် store မလုပ်ခဲ့ကြပါဘူး။ ယခု file handling သင်ခန်းစာကို လေ့လာပြီးရင် ထို data များကို files များတွင် store (write) လုပ်ခြင်း read ပြန်ထုတ်ခြင်းများကို သိရှိသွားမှာ ဖြစ်ပါတယ်။

ကျွန်တော်တို့ သိထားတဲ့အတိုင်းပဲ data တွေကို disk ထဲမှာ binary format နဲ့ store လုပ်ကြပါတယ်။ ယခုသင်ခန်းစာတွင် ထိုအကြောင်းများကိုလည်း ဆွေးနွေးသွားမှာပါ။

Python မှာ File တွေကို ကိုင်တွယ်တဲ့ လုပ်ငန်းစဉ်တွေက ရိုးရှင်းပါတယ်။ ပထမဆုံး အနေဖြင့် မိမိတို့ကိုင်တွယ်လိုသော file ကိုဖွင့်ပါတယ်။ ထိုသို့ဖွင့်ရန်လည်း open() ဆိုသည့် function ကိုသုံးပါတယ်။ open() function သည် arguments နှစ်ခု ယူပါတယ်။ ပထမတစ်ခုသည် file name ဖြစ်ပြီး ဒုတိယတစ်ခုသည် ထို file အား မည်သည့်ပုံစံနှင့် ကိုင်တွယ်မည် ဆိုသည့် mode ဖြစ်ပါတယ်။

Open() function ထဲမှာ arguments ကို သုံးခုထည့်ပြီး သုံးနိုင်ပါတယ် ။ တတိယမြောက် တစ်ခုကိုတော့ ယခုသင်ခန်းစာရဲ့ နောက်ပိုင်းမှာ ဆွေးနွေးသွားပါမယ်။ open function ကိုသုံးခြင်းဖြင့် returns အနေနဲ့ file ကို object အနေနဲ့ ပြန်ပေးပါတယ်။ ထိုကဲ့သို့ ရလာသော object ကိုမှ read လုပ်ခြင်း write လုပ်ခြင်း သို့မဟုတ် append လုပ်ခြင်းတို့ကို လုပ်ဆောင်ပါတယ်။ မိမိလုပ်လိုသော လုပ်ငန်းစဉ်များ လုပ်ဆောင်ပြီးရင်တော့ close() function ကိုသုံးပြီး file ကို ပြန် ပိတ်ပေးရပါတယ်။

File Handling Mechanism

အထက်တွင် ဖော်ပြထားသော ပုံသည် open() function ကို သုံးပြီး data.txt ဆိုသည့် file ကိုဖွင့်ခြင်းဖြစ်ပါသည်။ mode တွင် r ဟုရေးထားပြီး ထို r သည် read ကိုဆိုလိုခြင်း

ဖြစ်သည်။ file ကိုဖွင့်လို့ အောင်မြင်တယ်ဆိုရင် file object ကို return ပြန်ပေးပါသည်။ အထက်ပါပုံတွင် အောင်မြင်ခဲ့လျှင် f သည် file object ဖြစ်သည်။ မအောင်မြင်ခဲ့ဘူးဆိုလျှင်တော့ IOERROR ဆိုသည့် error ကို မြင်ရမှာပါ။ IOERROR ဆိုတာ input output error exception ကို ဆိုလိုခြင်း ဖြစ်ပါတယ်။ Exception အကြောင်းကို နောက်သင်ခန်းစာမှာ ဆွေးနွေးသွားမှာပါ။ File တစ်ခုကို ဖွင့်ရာမှာ အခြေခံအားဖြင့် mode သုံးမျိုးဖြင့် ဖွင့်ပါတယ်။ ပထမတစ်ခုကတော့ အထက်မှာ ဖော်ပြထားတဲ့ r (read) mode ပါ။ ဒုတိယတစ်ခုကတော့ w (write) mode ဖြစ်ပါတယ်။ file တစ်ခုကိုဖွင့်ပြီး အထဲတွင် data များရေးထည့်လိုတဲ့အခါမှာ သုံးပါတယ်။ တတိယ တစ်ခုကတော့ a (append) mode ပါ။ သူကတော့ file တစ်ခုကို ဖွင့်ပါတယ်။ ထို့နောက် ကိုယ်ရေးချင်တဲ့ data တွေကိုရေးပါတယ်။ တကယ်လို့ data တွေရှိနေပြီးသားဆိုရင်လည်း ထပ်ရေးပါတယ်။ အကယ်၍ append mode နဲ့ဖွင့်မယ်ဆိုရင် ကိုယ်ဖွင့်လိုက်တဲ့ file name နဲ့ file ရှိမနေဘူးဆိုရင် file အသစ် ဖွင့်ပေးပါတယ်။

Sample Program (271)

```
fileptr = open("whgh.txt",'r')

if fileptr:

    print('file is opened successfully')

fileptr.close()
```

Sample Program (271)ကတော့ read mode နဲ့ ဖွင့်ထားတာပါ။ program ကို မရေးခင် အရင်ဆုံး မိမိတို့ယခု လက်ရှိရေးနေတဲ့ python file ရှိတဲ့ directory နေရာမှာပဲ whgh.txt ဆိုတဲ့ text file တစ်ခုကို အရင်ဆုံး create လုပ်ပေးရပါမယ်။ ပြီးလျှင် ထို text file ထဲ၌ မိမိတို့ ရေးချင်သည့် စာသားများကို ရေးခဲ့နိုင်ပါတယ်။ ထို့သို့ ပြုလုပ်ပြီးမှ အထက်ပါ program ကို ရေးရမှာပါ။ အကယ်၍ ထိုကဲ့သို့ file ကို create မလုပ်ခဲ့ဘူးဆိုလျှင် FileNotFoundError ဆိုတဲ့ error တက်မှာ ဖြစ်ပါတယ်။

File တစ်ခုကို ဖွင့်ပြီးလျှင် အဆုံး၌ အမြဲတမ်း ပြန်ပိတ်ပေးရပါတယ်။ ထိုသို့ ပိတ်ရာတွင် close() ဆိုသည့် method ကိုသုံးနိုင်ပါတယ်။ fileobject.close() ယခု syntax အတိုင်း ရေးရ ပါတယ်။ ထို့ကြောင့် နောက်ဆုံးတွင် fileptr.close() ဆိုသည့် code line ကိုရေးသားထားခြင်း ဖြစ်ပါတယ်။

Reading From File

File တစ်ခုထဲမှာ ရှိတဲ့ contents တွေကို ဖတ်ဖို့ဆိုရင် read function ကို အသုံးပြုပါတယ်။ ထို function ကို အသုံးပြုရာမှာ argument အနေဖြင့် integer ကို အသုံးပြုနိုင်ပါတယ်။ File ထဲမှာ ရှိတဲ့ data အားလုံးကို ဖတ်ချင်တယ်ဆိုရင် read(-1) ဟုရေးနိုင်ပါတယ်။ အကယ်၍ integer arguments မပေးလျှင်လည်း

အဆုံးထိဖတ်ပေးပါတယ်။

Sample Program (272)

```
fileptr = open("whgh.txt",'r')
if fileptr:
    print('file is opened successfully')
    data = fileptr.read(10)
    print(data)
else:
    print('There have no files')
fileptr.close()
```

Sample Program (272)မှာတော့ file ထဲမှာ ရှိတဲ့ contents တွေထဲက စာလုံး(၁၀)လုံးပဲ ဖတ်လိုတဲ့အတွက် read function ထဲတွင် arguments 10 ကို ထည့်ပေးထားခြင်း ဖြစ်ပါတယ်။ ထို့ကြောင့် output အနေဖြင့် space အပါအဝင် စကားလုံးဆယ်လုံးသာ ထွက်လာမှာ ဖြစ်ပါတယ်။ အကယ်၍ အားလုံးဖတ်လိုတယ်ဆိုရင်တော့ 10 နေရာမှာ မည်သည့် arguments မှ မထည့်ပဲ run နိုင်ပါသည်။

Readlines and Readline

Readline method ကတော့ file ထဲမှာရှိတဲ့ content တွေထဲက ပထမဆုံးသော စာကြောင်းကိုပဲ ဖတ်ပြီး string အနေဖြင့် return ပြန်ပေးပါတယ်။ readlines method ကတော့ content ထဲမှာ ရှိသမျှအားလုံးကို ဖတ်ပြီး list တစ်ခုအနေနဲ့ return ပြန်ပေးပါတယ်။ list အနေနဲ့ ပြန်ပေးတဲ့အခါမှာ စာကြောင်းတစ်ကြောင်းကို index တစ်ခု အနေနဲ့ ပြန်ပေးပါတယ်။

Sample Program (273)

```
fileptr = open("whgh.txt",'r')
fileptr2= open("whgh.txt",'r')
if fileptr:
    print('file is opened successfully')
    data = fileptr.readlines()
    print('Reading with readlines method...')
    print(data)
    content = fileptr2.readline()
    print('Reading with readline method...')
    print(content)
else:
    print('There have no files')
fileptr.close()
fileptr2.close()
```

Output::

file is opened successfully

Reading with readlines method...

['Hello I am Win Htut\n', 'From Green Hackers Team\n', 'Hello \n', 'Hello']

Reading with readline method...
Hello I am Win Htut

Writing the File

File တစ်ခုအတွင်းသို့ data or contents များ write လုပ်မည်ဆိုပါက mode အနေဖြင့် a or w ကို သုံးနိုင်ပါတယ်။ သို့သော် ထို mode နှစ်ခုသည် ကွဲပြားပါသည်။ a mode သည် file ထဲမှာ ရှိပြီးသား မူရင်း content များဖျက်မပစ်ပဲ အသစ်ထပ်ပြီး append လုပ်ပေးပါသည်။ သို့သော် w mode သည် file ထဲမှာ ရှိသော မူရင်း contents များကို ဖျက်ပစ်ပြီးနောက် contents များကို ရေးသားပါသည်။ အထူးသတိပြုရန်အချက်မှာ Python တွင် file များကို ကိုင်တွယ်မည်ဆိုပါက operation တစ်ခုပြီးမှ တစ်ခုကို လုပ်ဆောင်သင့်ပါသည်။ operation အားလုံးကို တစ်ပြိုင်တည်း လုပ်ဆောင်ပါက incomplete ဖြစ်နိုင်ပါသည်။ အောက်တွင် sample program တစ်ပုဒ်ရေး ပြထားပြီး ထို program တွင် operation တစ်ခုပြီးမှ တစ်ခု ဆောင်ရွက်ထားသည်ကို မြင်ရပါမည်။

Sample Program (274)

```
fileptr = open("whgh.txt",'w')
contents= "Hello this is our contents \n newLine."
if fileptr:
    print("writing data to the file")
    fileptr.write(contents)
    fileptr.close()
    fileptr2=open("whgh.txt",'r')
    newContent=fileptr2.readlines()
    print(newContent)
    fileptr2.close()

else:
    print('There have no files')
```

Output:
writing data to the file
['Hello this is our contents \n', ' newLine.']

နောက်ထပ် program တစ်ပုဒ်အနေနဲ့ line number 1 တွင် w နေရာ၌ a ကိုသုံးပြီးစမ်းရေး ကြည့်နိုင်သည်။ append mode ကို သုံးတာ ဖြစ်တဲ့အတွက် မူရင်း ရှိသည့် contents များကို ဖျက်ပစ်မှာ မဟုတ်ပဲ နောက်ထပ် contents များပါ output ထွက်လာသည်ကို မြင်ရမှာပါ။

Creating a File

File အသစ်တစ်ခုကို create လုပ်ဖို့ဆိုရင် x mode ကိုသုံးနိုင်ပါသေးတယ်။ အသုံးပြုပုံကတော့ open("file.txt","x") ဖြစ်ပါတယ် ။ x mode သည် ကိုယ်သတ်မှတ်ပေးလိုက်တဲ့ file

name နဲ့ file တစ်ခုကို create လုပ်ပေးပါတယ်။ သတိပြုရန်အချက်မှာ same name နဲ့ file တစ်ခု ရှိနေရင်တော့ error တက်မှာ ဖြစ်ပါတယ်။
Sample Program (275)

```
fileptr = open("exit.txt",'x')
print(fileptr)
if fileptr:
    print("File was created")
```

Output:
<_io.TextIOWrapper name='exit.txt' mode='x' encoding='cp1252'>
File was created

File Accessing Mode

File တွေကို access လုပ်ရာမှာ mode 12 မျိုးရှိပါတယ်။ a , r ,w ကတော့ base 3 ခုပါ။

- r : r သည် file တစ်ခုကို read လုပ်ရန်အတွက် အသုံးပြုပါတယ်။
- rb : rb သည် file တစ်ခုကို binary format ဖြင့် ဖတ်ရန် အသုံးပြုပါတယ်။
- r+ : r+ သည် file တစ်ခုကို read and write လုပ်ရန် အသုံးပြုပါသည်။
- rb+ : rb+ သည် file တစ်ခုကို binary format ဖြင့် read and write လုပ်ရန် သုံးပါသည်။
- w : w သည် file တစ်ခုကို write လုပ်ရန် အသုံးပြုပါသည်။
- wb : wb သည် file တစ်ခုကို binary format ဖြင့် write လုပ်ရန် သုံးပါသည်။ w သည် file ထဲတွင် မူရင်းရှိနေသော contents များကို ဖျက်ပစ်ပြီး အသစ်ပြန် write လုပ်ပါသည်။
- w+ : w+ သည် file တစ်ခုကို write လုပ်ရန် သုံးပါသည်။ r+ နှင့်ခြားနားချက်သည် r+ သည် မူရင်း ရှိသော content များကို ဖျက်မပစ်ပါဘူး။ w+ ကတော့ ဖျက်ပစ်သလို file မရှိဘူးဆိုရင်လည်း file တစ်ခုကို create လုပ်ပေးပါတယ်။
- wb+ : wb+ သည် file တစ်ခုကို binary format ဖြင့် read and write လုပ်ရန် သုံးပါသည်။
- a : file တစ်ခုကို write လုပ်ရာတွင် append mode ဖြင့် write လုပ်ရန် သုံးပါသည်။ file pointer သည် pervious က ရေးခဲ့သော content ရဲ့ နောက်ဆုံးမှာ ရှိပါတယ်။ ထို့ကြောင့် မူရင်း content များကို ဖျက်မပစ်ပဲ ထပ်တိုးပြီးရေးပေးတာပါ။ same name ဖြင့် file မရှိဘူးဆိုရင်လည်း file တစ်ခုကို create လုပ်ပေးပါတယ်။
- ab : ab ကတော့ a နှင့် တစ်ပုံစံတည်းဖြစ်ပြီး binary format ဖြင့် သုံးလိုတဲ့အခါမှာ အသုံးပြုပါတယ်။
- a+ : a+ file တစ်ခုကို append and read လုပ်လိုသောအခါတွင် သုံးပါသည်။ file မရှိဘူး ဆိုရင်လည်း create လုပ်ပေးပါတယ်။
- ab+ : ab+ သည် file တစ်ခုကို binary format ဖြင့် append and read လုပ်ရန် သုံးပါသည်။

With Statement with File

With statement ကို file handling နဲ့ ပတ်သက်ပြီး များစွာအသုံးပြုပါတယ်။ ထိုကဲ့သို့ အသုံးပြုရခြင်းသည် resourceစား သက်သာစေလို့ပါ။ file တွေကို သုံးတဲ့အခါမှာ ဖွင့်လည်း

ဖွင့်ပေးရသလို ပိတ်လည်း ပိတ်ပေးရပါတယ်။ open ကို ထိပ်ပိုင်းမှာ ရေးပါတယ်။ close ကိုတော့ program အားလုံးပြီးမှ ပိတ်လေ့ရှိပါတယ်။ ထိုကဲ့သို့သော အခြေအနေတွင် program မှ အဆုံးထိ မရောက်သေးပဲ အလယ်တွင် break or exception ဖြစ်ခဲ့တယ်ဆိုရင် file ကို ပြန်မပိတ်မိပါဘူး။ သို့သော် with ကို သုံးမည်ဆိုလျှင် break or return or exception များ ပေါ်ပေါက်လာပါက File ကို အလိုလျောက် ပြန်ပိတ်ပေးပါတယ်။

Sample Program (276)

```
with open("whgh.txt",'r') as file:
    if file:
        contents= file.read()
        print(contents)
    else:
        print('cannot open')
```

File Pointer

C/C++ မှာပဲ pointer ကိုသုံးစွဲခွင့်ရပါတယ်။ pointer variable ဆိုတာလည်း ပုံမှန် variable ပါပဲ။ သို့သော်ပုံမှန် variable များသည် value ကို store လုပ်သော်လည်း pointer variable ကတော့ memory address ကို store လုပ်ပါတယ်။ variable ကို pointer ထောက်ထားတယ်လို့လည်း သုံးနှုန်းပါတယ်။ ဆိုလိုတာက variable ရဲ့ value အစစ်အမှန် store လုပ်ထားတဲ့ memory address ကို ထောက်ထား(ရယူ) ထားတာကို ဆိုလိုတာပါ။ File Pointer သည်လည်း ထိုနည်းအတိုင်းပါပဲ။ file တစ်ခုရှိတဲ့နေရာရဲ့ memory address ကို store လုပ်ထားတာပါ။ python မှာတော့ tell ဆိုတဲ့ method ကိုသုံးပြီး file pointer ဘယ်နားကိုရောက်နေသလဲဆိုတာ ဖမ်းနိုင်ပါတယ်။ ထို့ကြောင့် mode တွေကိုလည်း tell method ကိုသုံးပြီး mode တစ်ခုချင်းစီမှာ pointer က ဘယ်နားကိုထောက်သလဲဆိုတာ စစ်ဆေးနိုင်ပါတယ်။ ဆိုလိုချင်တာ file ထဲမှာရှိတဲ့ အစစာလုံးကို ထောက်သလား သို့မဟုတ် file ထဲမှ အဆုံးစာလုံးကို ထောက်ထားသလား အလယ် စာလုံးကို ထောက်ထားသလားဆိုတာ သိနိုင်ပါတယ်။ အောက်မှာ pointer location နဲ့ ပတ်သက် sample program ရေးပြထားပါတယ်။

Sample Program (277)

```
with open("whgh.txt",'r') as file:
    if file:
        print("The file pointer at Byte:",file.tell())
        contents = file.read()
                    print("After Reading file pointer at
Byte:",file.tell())
    else:
        print('cannot open')
```

Sample Program (277) ကို run ပြီးချိန်တွင် ပထမ tell method ကိုသုံးထားသည့် output ၌ read mode နဲ့ဖွင့်ထားတာ ဖြစ်သည့်အတွက် ပထမဆုံး 0 နေရာကို ထောက်ထားမှာ

ဖြစ်ပါတယ်။ ဒုတိယ tell method ကို သုံးပြီးသည့်အခါတွင် နောက်ဆုံးစာလုံးထိဖတ်ထားတာ ဖြစ်သည့် အတွက် နောက်ဆုံးစာလုံးနေရာကို ထောက်ထားမှာ ဖြစ်ပါတယ်။

seek() Function

seek() function ကို file pointer အား မိမိလိုသလို ရွှေ့ချင်တဲ့နေရာကို ရွှေ့လိုသောအခါတွင် အသုံးပြုပါတယ်။ seek() function သည် parameter နှစ်ခုယူပါတယ်။ ပထမတစ်ခုသည် offset ဖြစ်ပြီး ဒုတိယသည် whence ဖြစ်ပါတယ်။ whence arguments နေရာမှာ value သုံးခုကို သုံးလို့ရပါတယ်။

- 0 သည် file ရဲ့ အစနေရာကို ညွှန်းချင်တဲ့အခါမှာ သုံးပါတယ်။ ဥပမာ file.seek(2,0) ဟု ရေးခဲ့သည် ဆိုပါစို့။ file pointer သည် 2 နေရာသို့ ရောက်ရှိသွားမည် ဖြစ်ပါသည်။ အဘယ်ကြောင့်ဆိုသော် ဒုတိယ arguments ဖြစ်သည့် whence သည် 0 ဖြစ်တဲ့အတွက် မူလနေရာမှ 2 နေရာသို့ရောက်သွားမည် ဖြစ်ပါသည်။ file.seek(2,0) ဟု run ပြီးသော အချိန်တွင် file pointer သည် 2 နေရာသို့ ရောက်ရှိသွားမည် ဖြစ်သည်။
- 1 သည် file pointer ရဲ့ ယခုလက်ရှိနေရာကို ညွှန်းချင်တဲ့အခါမှာ သုံးပါတယ်။ ဥပမာ file.seek(2,1) ဟု ရေးခဲ့သည်ဆိုပါစို့။ file pointer သည် ယခုလက်ရှိ ရှိနေတဲ့နေရာကနေပြီး နောက်ထပ်နှစ်ခု ထပ်တိုးသွားမှာဖြစ်ပါတယ်။ ယခင်က file pointer သည် 5 နေရာတွင် ရှိနေသည် ဆိုပါစို့။ file.seek(2,1) ဟု run ပြီးလျှင် file pointer သည် 7 နေရာသို့ ရောက်ရှိ သွားမည် ဖြစ်ပါသည်။
- 2 သည် file ရဲ့ နောက်ဆုံးနေရာကို ညွှန်းချင်တဲ့အခါမှာ သုံးပါတယ်။ ဥပမာ file.seek(10,2) ဟု run ကြည့်မည်ဆိုလျှင် output အနေဖြင့် 10 ကိုပဲ ပြန်လည်ရရှိမှာ ဖြစ်ပါတယ်။ Sample program ကို အောက်မှာ ဖော်ပြထားပါတယ်။

အောက်ပါ program ကို run ရန် အတွက် Python 2 အားအသုံးပြုရန် လိုအပ်ပါသည်။ Python 3 တွင် position arguments များထည့်ခြင်းကို ယခု အချိန်တွင် ခွင့်မပြုထားသေးပါ။ python 2 ယခု https://repl.it/languages/python တွင် သွားရောက် run နိုင်ပါသည်။

Sample Program (278)

```
with open("whgh.txt",'r') as file:
   if file:
       print("The file pointer at Byte:",file.tell())
       contents = file.read()
       print("After Reading file pointer at Byte:",file.tell())
       file.seek(5)
       print("After moving file pointer at 5:",file.tell())
       file.seek(2,1)
       print("After Reading file pointer at 7:",file.tell())
       file.seek(10,2)
       print("using end file pointer:",file.tell())

   else:
       print('cannot open')
```

Output:

('The file pointer at Byte:', 0)

('After Reading file pointer at Byte:', 0)
('After moving file pointer at 5:', 5)
('After Reading file pointer at Byte:', 7)
('After Reading file pointer end:', 10)

More Example
Sample Program (279)

```
with open("whgh.txt",'a') as file:
 if file:
   print("the file pointer current pos:",file.tell())
   file.seek(2,1)
   print("the file pointer after 2 moved pos:",file.tell())
   file.seek(5,0)#start
   print("fp from 0 after 5 moved pos:",file.tell())
   file.seek(5,1)# 1 current
   print("current file pointer",file.tell())
   file.seek(20,2) # 52
   print("going to the end",file.tell())
   file.seek(10,2)# 42
   print("the final ending",file.tell())
   file.seek(15,2) # 32+15=47
   print("end 15 ",file.tell())
```

Sample Program (278) ကို စမ်းကြည့်မည်ဆိုလျှင် whgh.txt ဆိုသည့် text file တစ်ခု ဆောက်ပြီး ထို text file ထဲတွင် i am win htut from green hackers ဆိုသည့် စာသားလေးကိုရေးထည့်ထားပါ။ ထိုစာသားသည် စုစုပေါင်း space အပါအဝင် 32 လုံး ရှိပါတယ်။ ဥပမာ file.seek(10,2) ဟုရေးလျှင် output အနေဖြင့် 42 ကို ရရှိမှာပါ။ အဘယ်ကြောင့်ဆိုသော် နောက်ဆုံး မူရင်းရှိသည့် 32+10=42 ဖြစ်သွားသောကြောင့် ဖြစ်သည်။ ထို့ပြင် file.seek(20,2) ဟု ထပ်ရေးခဲ့မည်ဆိုလျှင် 52 ကို ပဲ ရရှိမှာပါ။ အဘယ်ကြောင့်ဆိုသော် end ဖြစ်သည့် 2 သည် file ရဲ့ နောက်ဆုံးကနေသာ ရေတွက်ပြီး အလုပ်လုပ်ပါသည်။ သူ့အပေါ်တွင် run ထားသော position ကို မကြည့်ပါ။ ထိုအချက်သည် 1 နှင့် လုံးဝကွဲပြားသည့်အချက် ဖြစ်ပါသည်။ ထို့ကြောင့် end ကို သုံးရာတွင် အထူးသတိပြုရန် လိုအပ်ပါသည်။

Python 3 မှာ seek ကိုသုံးပြီး file pointer position တွေကို ကစားချင်တယ်ဆိုရင် os module ကိုသုံးပြီးတော့ ပြုလုပ်နိုင်ပါသေးတယ်။ os module ထဲကမှ SEEK_SET and SEEK_END တို့ကိုသုံးပြီး မိမိလိုသလို ပြောင်းလို့ရပါတယ်။ ဥပမာ ယခုလက်ရှိရောက်နေတဲ့ နေရာကနေ 2 နေရာကို သွားချင်တယ်ဆိုရင် file.seek(2 , os.SEEK_SET) ကိုသုံးနိုင်ပါတယ်။ SEEK_SET သည် 0 နေရာကို ရည်ညွှန်းချင်တာပါ။ နောက်ဆုံးကို သွားချင်တယ် ဆိုရင်တော့ file.seek(0 , os.SEEK_END) ကို သုံးနိုင်ပါတယ်။ ယခုလက်ရှိ ရောက်နေတဲ့ နေရာကနေ နောက်ကို 5 နေရာစာ ဆုတ်ချင်တယ် ဆိုရင်လည်း file.seek(file.tell()-5 , os.SEEK_SET) ယခုကဲ့သို့ ရေးမည်ဆိုလျှင် ယခုရောက်နေသည့် position မှ file pointer သည် 5 နေရာစာ နောက်ဆုတ်သွားမှာ ဖြစ်ပါတယ်။ အောက်ပါမှာ sample program နှင့်အတူ output များကိုပါ

ဖော်ပြထားပါတယ်။ file ကိုဖွင့်ရာ တွင် a+ ဖြင့် ဖွင့်ထားသည်ကို သတိပြုပါ။

Sample Program (280)

```
import os
with open("whgh.txt",'a+') as file:
    if file:
        print("The file pointer at Byte:",file.tell())
        contents = file.read()
        print("After Reading file pointer at Byte:",file.tell())
        file.seek(5)
        print("After changing file pointer at Byte:",file.tell())
        file.seek(2,os.SEEK_SET)
        print("After changing file pointer at 2 :",file.tell())
        file.seek(0,os.SEEK_END)
        print("After changing file pointer to End:",file.tell())
        file.seek(file.tell()-5,os.SEEK_SET)
        print("Backward 5 position:",file.tell())
    else:
        print('cannot open')
```

Output:

The file pointer at Byte: 18
After Reading file pointer at Byte: 18
After changing file pointer at Byte: 5
After changing file pointer at 2 : 2
After changing file pointer to End: 18
Backward 5 position: 13

Exception

Exception ဆိုတာ ဖြစ်စဉ်တစ်ခုပါ။ ဘယ်လိုဖြစ်စဉ်လဲဆိုရင် program execution လုပ်နေ တဲ့အချိန်မှာ syntax error or logical error တစ်ခုခုကို detected လုပ်မိသွားခြင်းပါ။ ရှေ့ပိုင်း lessons များတွင်လည်း exception ကို သုံးခဲ့ပါတယ်။ ဥပမာ ZeroDivisionError ကိန်းတစ်ခုခုကို zero ဖြင့် စားမိတဲ့အခါမျိုးမှာ ပြတဲ့ Error ပါ။ နောက်တစ်ခုက NameError သူက variable တစ်ခုကို defined မလုပ်ပဲ သုံးမိတဲ့ အခါမျိုးမှာ ဖြစ်တတ်ပါတယ်။ နောက်တစ်ခုတော့ TypeError ပါ။ type မတူတာတွေကို arithmetic လုပ်ရာမှာ ဖြစ်တတ်ပါတယ်။ အထက်ပါ သုံးခုကတော့ python ကို စလေ့လာတဲ့ သူများမှာ ဖြစ်တတ်တဲ့ error သုံးခုပါ။ Exception မြောက်များစွာ ကျန်ရှိ နေပါသေးတယ်။

Sample Program (281)

```
a=10
b=0
c=a/b
print(c)
```

Sample Program (281) ကိုရေးကြည့်မည်ဆိုလျှင် zero နဲ့ စားတာ ဖြစ်တဲ့ အတွက် ZeroDivisionError ဆိုတာကို တွေ့ရမှာပါ။

Output::

```
c=a/b
ZeroDivisionError: division by zero
```

10+data*5 ယခု ကဲ့သို့ ရေးကြည့်မည်ဆိုလျှင် NameError: name 'data' is not defined ဆိုတာကို တွေ့မှာပါ။ 10+'data' ယခုကဲ့သို့ run မည်ဆိုလျှင် TypeError: unsupported operand type(s) for +: 'int' and 'str' ဆိုသည့် TypeError ကိုတွေ့ရမှာပါ။

Handling Exception

Exception တွေကို မိမိလိုသလို Handle လုပ်နိုင်ဖို့ try and except, try - except - else, try -except - finally တို့ကိုလည်း သုံးနိုင်ပါတယ်။

Sample Program (282)

```
try :
 { မိမိ run လိုသော code }
Except:
 { exception ဖြစ်လာခဲ့ရင် ယခု code များကို အလုပ်လုပ်ပါ }
```

```
try:
    a=10/0
except:
    print('some error occur')
```

Sample Program (282) မှာဆိုရင် try ထဲက အလုပ်များကိုလုပ်ပြီးလျှင် ZeroDivisionError ဆိုတာ တက်လာမှာပါ။ သို့သော် ထို error ကိုမပြပဲ exception ထဲက code များကို အလုပ် ဆက်လုပ် ပေးပါသည်။ Keyboard Interrupt ကိုလည်း အောက်ပါအတိုင်း program ရေးပြီး စမ်းကြည့်နိုင်ပါသေးသည်။ Keyboard Interrupt Error ကို မပေးပဲ except ထဲက စာသားကိုပဲ အလုပ် လုပ် ပေးသွားမှာပါ။

Sample Program (283)

```
try:
    data=int(input("please enter valid key:"))
except:
    print("You have entered invalid number..")
```

ထို့ပြင် else ဖြင့်လည်း တွဲသုံးနိုင်ပါသေးတယ်။ exception မဖြစ်ခဲ့ဘူးဆိုလျှင် else ထဲက program များကို အလုပ်လုပ်ပေးသွားမှာပါ။

Sample Program (284)

try :
{ မိမိ run လိုသော code }
except:
{ exception ဖြစ်လာခဲ့ရင် ယခု code များကို အလုပ်လုပ်ပါ }
else:
{ exception မဖြစ်ခဲ့ဘူးဆိုလျှင် ယခု code များကို အလုပ်လုပ်ပေးပါမည်။ }

Sample Program (285)

```
try:
    a=10+10
except:
    print("Some exception Occured")
else:
    print("No exception occur")
```

Sample Program (285) ကို run ကြည့်မည်ဆိုလျှင် no exception occur ဆိုသည့် output ကိုသာလျှင် ရရှိမှာ ဖြစ်ပါတယ်။ အဘယ်ကြောင့်ဆိုသော် error တစ်ခုခုဖြစ်မှသာလျှင် exception ထဲမှ code များအလုပ်လုပ်မှာ ဖြစ်ပြီး exception မဖြစ်လျှင် else ထဲက code များ အလုပ်လုပ် မှာပါ။

Try , except , finally

try :
{ မိမိ run လိုသော code }
except:
{ exception ဖြစ်လာခဲ့ရင် ယခု code များကို အလုပ်လုပ်ပါ }
else:
{ exception မဖြစ်ခဲ့ဘူးဆိုလျှင် ယခု code များကို အလုပ်လုပ်ပေးပါမည်။ }
Finally:
{ ဘယ်လို အခြေနေပဲ ဖြစ်ဖြစ် ယခု code တွေ အားလုံး အလုပ်လုပ်သွားမှာပါ။ }

Sample Program (286)

```
try:
    a=10/0
except:
    print("Some exception Occured")
finally:
    print("This is from finally")
```

အထက်တွင် ရေးသားထားခဲ့သော program များသည် except နေရာ၌ exception name များကို မရေးပဲ ရေးသားထားခြင်းဖြစ်သည်။ Python တွင် exception များစွာ ရှိပါသည်။ ၎င်းတို့ထဲမှာ အနည်းငယ်ကို အောက်တွင် ဖော်ပြသွားပါမည် ။

Sample Program (287)

```
try:
    a=10+'a'
except Exception as err:
    print(err)
finally:
    print("This is from finally")
```

Sample Program (287) ကို run ကြည့်မည်ဆိုလျှင် Type error ကိုရရှိမှာ ဖြစ်ပါတယ်။ ထို error ကို exception ကသိပြီး err ထဲသို့ ထည့်လိုက်ပါသည်။ ထို့ကြောင့် err ကို print ထုတ်ကြည့်ပါက unsupported operand type(s) for +: 'int' and 'str' ယခု error မျိုးကို မြင်တွေ့ရမှာပါ။

Sample Program (288)

```
try:
    f = open('myfile.txt')
except OSError as err:
    print("OS error: {0}".format(err))
finally:
    print("Unexpected error:")
```

Sample Program (288) မှာတော့ OSError ကို ဥပမာ သုံးပြထားပါတယ်။ ထို OSError သည် OS နှင့် ပတ်သက်သည့် function call များကို ခေါ်ဆိုရာမှ ဖြစ်ပေါ်လာသည့် Exception များနှင့် ပတ်သက်ပြီး အသုံးပြုပါသည်။ အထက်ပါ program ကို run ကြည့်မည်ဆိုလျှင် အောက်ပါ အတိုင်း error ကို မြင်တွေ့ရမှာပါ။

OS error: [Errno 2] No such file or directory: 'myfile.txt

Error ကို error number နှင့် ဖော်ပြသည်ကို သတိထားကြည့်နိုင်ပါသည်။ ဆိုလိုသည်မှာ OSError ရဲ့ error number 2 ကိုဆိုလိုခြင်း ဖြစ်သည်။ OSError တွင် error number 1 သည် operation not permitted ဖြစ်ပြီး error number 2 သည် No such file or directory ဖြစ်သည်။ error number 3 သည် No such process ဖြစ်သည်။ ထိုကဲ့သို့ error ပေါင်း 50 ဝန်းကျင် ခန့်ရှိပါသည်။

Raise

Exception တွေ အလုပ်လုပ်နိုင်ဖို့ raise ဆိုတဲ့ keyword ကို အသုံးပြုနိုင်ပါတယ်။ raise နောက်မှာ exception class name ကိုတော့ ဖော်ပြပေးဖို့ လိုအပ်ပါတယ်။ ဥပမာ try statement ထဲမှာ raise ကိုရေးပြီး ထို raise နောက်တွင် exception class name ကိုရေးခဲ့သည် ဆိုပါစို့။ program သည် raise ဆိုသည့် keyword နေရာကို ရောက်လာသည်နှင့် တစ်ပြိုင်နက် သူ့နောက်က exception name ကို program ထဲတွင် ရှာပါတယ်။ တွေ့ခဲ့ရင် ထို exception ထဲက code များကို အလုပ်လုပ် ပေးပါတယ်။

Sample Program (289)

```
try:
    print("activating exception")
    raise ZeroDivisionError
except ZeroDivisionError:
    print("From ZeroDivisionError")
```

Sample Program (288) တွင် line 3 ၌ ZeroDivisionError ကို ရေးခဲ့ကြသည်။ ထို့ကြောင့် program သည် line 3 သို့ရောက်သောအခါတွင် ZeroDivisionError ကိုသွားရှာပါသည်။ ပြီးနောက် ZeroDivisonError အောက်က code များကို အလုပ်လုပ်ပါသည်။ မိမိခေါ်ချင်သည့် exception ကို ခေါ်နိုင်ရန် အောက်ပါ sample program လေးကို စမ်းကြည့်နိုင်ပါသည်။

Sample Program (290)

```
try:
    data=int(input("please enter a number:"))
    if data==10:
        raise ZeroDivisionError
    else:
        raise TypeError
except ZeroDivisionError:
    print("From ZeroDivisionError")
except TypeError:
    print("This is TypeError")
```

Creating Own Exception

Python မှာ exception တွေကို ကိုယ်တိုင်ဖန်တီးဖို့လည်း ခွင့်ပြုထားပါတယ်။ ထိုသို့ဖန်တီးဖို့ဆိုရင် Exception ဆိုတဲ့ class ကိုတော့ ခေါ်သုံးပေးရပါမည်။ ဆိုလိုသည်မှာ ပထမဆုံး အနေဖြင့် myExcept ဆိုသည့် class တစ်ခုဖန်တီးမည် ဆိုပါစို့။ class myExcept(Exception) ဟု ရေးရမည်။ ထို့နောက် မိမိတို့ဖန်တီးလိုသည့် exception များကို myExcept ဆိုသည့် parent class ကို inheritance လုပ်ပြီး ဖန်တီးနိုင်ပါသည်။

Sample Program (291)

```
class myExcept(Exception):
    pass
class NameError(myExcept):
    pass
```

```
class ValueError(myExcept):
    pass
try:
    a=10
    b=20
    if a is b:
        raise ValueError
    else:
        raise NameError
except NameError:
    print("Name Error")
except ValueError:
    print("Value Error")
```

JSON (JavaScript Object Notation)

JSON (JavaScript Object Notation) သည် data တွေကို (interchange) လွှဲပြောင်းရယူ ရာတွင် အသုံးပြုသည့် lightweight data format ဖြစ်သည်။ လူတွေအတွက် read and write လုပ်ရန် လွယ်ကူသလို machine အတွက်လည်း လွယ်လွယ်ကူကူ လိုအပ်သလို ပြုပြင်ပြောင်းလဲ နိုင်ပါတယ် ။ syntax က JavaScript programming ဆင်တူပြီး client and server ကြားမှာ data တွေကို အလွှဲအပြောင်းလုပ်ဖို့ အဓိကအသုံးပြုပါတယ်။ JSON Data Format ကို Python programming တွင်မကပဲ အခြားသော programming များစွာဖြင့်လည်း အသုံးပြုနိုင်ပါသေး တယ်။

အပေါ်မှာ ပြထားတဲ့ပုံကတော့ Client ဖြစ်တဲ့ mobile application တွေ single web page application တွေ အခြားသော application တွေနှင့် server ကြားထဲမှာ data တွေကို အပို့အယူ လုပ်တဲ့ပုံစံ ဖြစ်ပါတယ်။ server and client ကြားထဲမှာ data interchange လုပ်တဲ့ အခြားသော format တွေလည်း ရှိနေပါသေးတယ်။

- HTML (Hyper Text Markup Language)
- Plain Text
- XML (Extensible Markup Language)
- JSON (JavaScript Object Notation)

အထက်ပါ data format လေးမျိုးသည် server and client ကြားမှာ အသုံးပြုသော format များဖြစ်ပြီး JSON သည် အသုံးများသည်။ အဘယ်ကြောင့်ဆိုသော် HTML format သည် format + data နှစ်ခုဖြစ်နေသည်။ ဆိုလိုသည်မှာ HTML တွင် HTML သည် data ဖြစ်သော်လည်း ဆိုသည့် structure ပါနေသည်။ အကယ်၍ data များအား လွှဲပြောင်းလိုသော အခါတွင်မူ HTML သည် parse လုပ်ရန်ခက်ခဲသည်။ Plain text သည် data များသာ ဖြစ်သော်လည်း ရှင်းလင်းသည့် data structure မရှိပါ။ XML သည် Machine မှ readable ဖြစ်သော်လည်း သူနဲ့သက်ဆိုင်သည့် XML parser လိုအပ်ပါသည်။ ထို့ကြောင့် human and machine မှ လွယ်ကူစွာဖတ်နိုင်သော JavaScript သို့လည်း လွယ်ကူစွာပြောင်းနိုင်သော JSON သည် အသုံးများသည်။

JSON တွေရဲ့ data stores လုပ်ပုံက အရမ်းကို ရိုးရှင်းပါတယ်။ key and value အစုံနဲ့ အလုပ်လုပ်တာပါ။ဥပမာ အနေနဲ့ဆိုရင် ရှေ့ပိုင်းသင်ခန်းစာတွေမှာ ဖော်ပြခဲ့ဖူးတဲ့ python dictionary နဲ့ ဆင်တူပါတယ်။

```
{
  "name": "WinHtut",
  "age": 25
}
```

အထက်တွင် ဖော်ပြထားသော ပုံသည် JSON data ပုံစံ ဖြစ်သည်။ name နှင့် age သည် keys များဖြစ်ပြီး WinHtut နှင့် 25 တို့သည် values များဖြစ်သည်။ keys and values များကိုရေးရာတွင် double quotes (“ “) နှစ်ခုနှင့် ရေးရမည် single quote (‘ ‘)

တစ်ခုတည်းဖြင့် ရေးလျှင် JSON format rule နှင့် မကိုက်ညီပါ။ JSON format ဖြစ်သလား မဖြစ်သလား ကို jsonlint.com တွင် စစ်ဆေးနိုင်သည်။

JSON Values To Python

Python ကနေ JSON ကိုပြောင်းလဲခြင်း သို့မဟုတ် JSON format ကနေ python ကို ပြောင်းမယ်ဆိုလျှင် JSON values များနှင့် Python Types များကို သိရန် လိုအပ်နေပါသေးသည်။ JSON တွင် object , array , string , true , false , null စသည့် value type များ ရှိပါသည်။ JSON object မှ python သို့ ပြောင်းလျှင် python dict type ကို ရရှိပါသည်။ JSON array မှ ပြောင်းလျှင် python list ကို ရရှိပါသည်။ JSON string မှ ပြောင်းလျှင် python string ကို ရရှိပါသည်။ ထိုနည်းတူ JSON true ကို ပြောင်းလျှင်လည်း python True ကို ရရှိပါသည်။ false ဆိုလျှင် Fasle , null ဆိုလျှင် python ၌ None တို့ကို ရရှိပါသည်။ အထက်တွင် ဖော်ပြချက် များသည် JSON နှင့် Python တွင် exchange လုပ်ရာ၌ အသုံးပြုသော format များ ဖြစ်သည်။ အောက်တွင်လည်းပုံနှင့် တကွ ဖော်ပြထားသည်။

Sample Program (292)

```
import json
myjson='''{
"name":"WinHtut",
"age":25,
"hobby":["coding","training","building"]

}'''
data = json.loads(myjson)
print(data)
print(type(data))
Output::
{'name': 'WinHtut', 'age': 25, 'hobby': ['coding', 'training', 'building']}
<class 'dict'>
```

JSON ပတ်သက်သည့် အလုပ်များကို လုပ်မည်ဆိုလျှင် json ဆိုသည့် module ကို ခေါ်သုံး ပေးရပါမည်။ထို module သည် built-in ဖြစ်သည့်အတွက် install လုပ်ပေးရန်မလိုအပ်ပါဘူး။

- ပထမဆုံးအနေဖြင့် myjson ဆိုသည့် string တစ်ခု တည်ဆောက်လိုက်သည်။ ထို string ထဲတွင် json format များဖြင့် မိမိရေးသားလိုသည့် အချက်အလက်များကို ရေးသားထားပါသည်။ ယခု program တွင်တော့ string ပုံစံဖြင့် သေချာပေါက်ရေးပေးရပါမည်။ ထိုသို့ မရေးလျှင် TypeError တက်ပါမည်။ (JSON object must be string)
- ထို့နောက် json module ထဲမှ loads ဆိုသည့် method ကို အသုံးပြုလိုက်ပါသည်။ loads method သည် return အနေဖြင့် python object ကို ပြန်ပေးပါသည်။
- Loads method ဖြင့်ရလာသော output များကို data ဆိုသည့် object ထဲတွင် သိမ်းလိုက်ပါသည်။ ထို data ဆိုသည့် obejct အားစစ်ဆေးကြည့်လျှင် dict ကို ရရှိပါမည်။ ထို့ကြောင့် print ထုတ်ကြည့်လျှင်လည်း data များကို python dictionary ပုံစံဖြင့် ပြန်လည်ရရှိပါမည်။

Serialization JSON

Python Type တွေကနေ JSON အဖြစ်သို့ ပြောင်းလဲခြင်းကို Serialization လုပ်တယ်လို့ ခေါ်ပါတယ်။ Python Object တွေကို JSON data အဖြစ်သို့ ပြောင်းနိုင်ဖို့ JSON module ထဲမှ dump() and dumps() methods တို့ကို သုံးနိုင်ပါတယ်။

Sample Program (293)

```
import json
myInfo={
"name":"WinHtut",
"age":25,
"hobby":["coding","training","building"],
"pro":["coding","making project","iot"]
```

```
}
with open("data.json","w") as jsFile:
  json.dump(myInfo,jsFile)
```

Sample Program (293)တွင် python dictionary object တစ်ခု တည်ဆောက်ထားသည်။ ထို့နောက် w mode ကိုသုံးပြီး data.json ဆိုသည့် json file တစ်ခုဖွင့်လိုက်ပါသည်။ ထို့နောက် dump ဆိုသည့် method ကိုသုံးပြီး myInfo ဆိုသည့် dict object ထဲမှ data များကို jsFile ဆိုသည့် json file ထဲသို့ dump လုပ်လိုက်ပါသည်။

ထို့ကြောင့် program အား run ပြီးချိန်တွင် ယခု python file ရှိသည့်နေရာတွင် data.json ဆိုသည့် json file တစ်ခု ဖြစ်လာပါမည်။ ထို file ထဲတွင် data များအားလုံး json format ဖြင့်ရောက်ရှိနေသည်ကို မြင်ရပါမည်။

Dump and Dumps

Dump method ကို python object မှ json format သို့ encode လုပ်ရန် အသုံးပြုပါသည်။ Dump method သည် parameters နှစ်ခုယူပါသည်။ ပထမတစ်ခုသည် Python object ဖြစ်ပြီး ဒုတိယတစ်ခုသည် json file ဖြစ်သည်။ အထက်ပါ program တွင် ဖော်ပြထားပြီးဖြစ်သည်။ dumps method သည် parameter တစ်ခုတည်းဖြင့်သာ သုံးနိုင်သော်လည်း indent အတွက် parameter ထည့်လိုပါက ထပ်ထည့်နိုင်သည်။

Sample Program (294)

```
import json
myInfo={
"name":"WinHtut",
"age":25,
"hobby":["coding","training","building"],
"pro":["coding","making project","iot"]
}
json_object = json.dumps(myInfo)
print(json_object)
```

Output:

```
{"name": "WinHtut", "age": 25, "hobby": ["coding", "training", "building"], "pro":
["coding", "making project", "iot"]}
```

Sample Program (294)ကို run ကြည့်ပြီး output များကို ကြည့်ပါက အစီစဉ်မကျသည်ကို တွေ့မြင်နိုင်ပါသည်။ ထို့ကြောင့် json.dumps(myInfo , indent =4) ဟု ထည့်ရေးပေးမည် ဆိုလျှင် output များသည် indent များနှင့် အစီအစဉ်ကျနေပြီး ဖတ်ရလွယ်ကူသည်ကို တွေ့မြင်နိုင်ပါ သည်။

```
{
"name": "WinHtut",
"age": 25,
"hobby": [
"coding",
"training",
```

```
        "building"
        ],
        "pro": [
        "coding",
        "making project",
"iot"
        ]
}
```

Deserializing JSON

Deserializing ဆိုတာကတော့ JSON data format တွေကို python objects များသို့ ပြောင်းခြင်းဖြစ်ပြီး serializing မှာတော့ dump and dumps တို့ကို သုံးခဲ့ကြသလို deserializing မှာတော့ load and loads methods တို့ကို သုံးပါတယ်။

Load and Loads

json.load () သည် file object တစ်ခုကိုယူပြီး return အနေဖြင့် key and value pair ဖြစ်တဲ့ json object ကို ပြန်ပေးပါတယ်။ အောက်မှာ sample program အနေဖြင့် python object ကို json file တစ်ခုထဲသို့ ရေးထည့်လိုက်ပါတယ်။ ထို့သို့ရေးထည့်ပြီးနောက် load method ကို သုံးပြီး json data ကို python object သို့ ပြန်ပြောင်းကာ deserializing ပြန်လုပ်လိုက်ပါတယ်။ ထို့ကြောင့် python object မှ json , json မှ python dictionary object သို့ ပြန်ပြောင်းထားပုံကို မြင်ရမှာပါ။

Syntax:

```
    json.load(file object )
```

Sample Program (295)

```
import json
jsonData={
    "name": "WinHtut",
    "age": 25,
    "hobby": [
        "coding",
        "training",
        "building"
    ],
    "pro": [
        "coding",
        "making project",
        "iot"
    ]
}
```

```
with open("data.json","w")as dataFile:
    json.dump(jsonData,dataFile)
with open("data.json","r")as rDataFile:
    data = json.load(rDataFile)
print(data)
print(type(data))
```

နောက်ဆုံးတွင် ရေးထားသော print(type(data)) output ကို ကြည့်ပါက dictionary object ဆိုတာကို မြင်ရပါမည်။

json.loads() method သည်လည်း json data တွေကို python object သို့ ပြောင်းလဲရန် သုံးပါတယ်။ json.load() method နှင့် အနည်းငယ်တူသည်လို့ ထင်ရသော်လည်းပဲ json.load() သည် parameter အနေဖြင့် file object ကို ယူပါသည်။ json.loads() သည် parameter အနေဖြင့် json object ကို သာယူပါသည်။

Sample Program (296)

```
import json
data= {
    "name": "WinHtut",
    "age": 25,
    "hobby": ["coding", "training", "building"],
    "pro": ["coding", "making project", "iot"]
}
jData = json.dumps(data)
pData = json.loads(jData)
print(pData)
print(type(jData))
print(type(pData))
```

Output:

{'name': 'WinHtut', 'age': 25, 'hobby': ['coding', 'training', 'building'], 'pro': ['coding', 'making project', 'iot']}

<class 'str'>

<class 'dict'>

သတိပြုရန် အချက်မှာ python object မှ တစ်ဆင့် json.dumps() ကို သုံးလိုက်လျှင် json string ကိုရရှိပါသည်။ ထို json string data ကို json.loads() method ထဲသို့ပြန်သုံးလိုက်လျှင် python dictionary အဖြစ်ပြန်လည် ရရှိပါသည်။ အထက်ပါ program ouput တွင် ဖော်ပြ ထားသည်။

Project Using Weather API

ယခု သင်ခန်းစာမှာတော့ https://openweathermap.org/ ကနေပြီး Weather API တစ်ခုကို ယူပြီး weather အချက်အလက်များကို ရယူပါမည်။ ထိုကဲ့သို့ ရလာတော့ JSON data များကို python object သို့ ပြန်ပြောင်းပြီး မိမိတို့ လိုအပ်သလို handling လုပ်သွားပါမည်။

ပထမဆုံးအနေဖြင့် အထက်ပါ website သို့သွားပြီး API သွားယူပါမည်။ ထို website တွင် မိမိတို့ ကိုယ်ပိုင်အကောင့် တစ်ခုဖွင့်ရပါမည်။ သို့မှသာ API key ကို ရရှိပါမည်။ ထို key ကိုအသုံးပြုမှသာ weather အချက်အလက်များကို ရရှိပါမည်။ အကောင့်ဖွင့်ပြီးလျှင် pricing ဆို သည့် အထဲကိုဝင်ပါ ။

ထို့နောက် အောက်ပါ ပုံထဲက အတိုင်း Free Get API Key ကို တစ်ချက် ထပ်နှိပ်ပါ။

အောက်ပါ

ပုံထဲက အတိုင်း API keys ကို ထပ်နှိပ်ပါ ။

အောက်ပါ ပုံထဲတွင် ဖော်ပြထားသည့် API keys ကို copy ကူးလိုက်ပါ

Copy ကူးပြီးရလာသော API keys ကို မိမိတို့ program ထဲတွင် အောက်ပါအတိုင်း သွားရောက် သိမ်းထားပေးပါ။

ထို့နောက် မိမိတို့ ယခုလက်ရှိ အသုံးပြုနေသော browser url တွင် အောက်ပါ ပုံထဲက အတိုင်းလုပ်ဆောင်ပါ။

http://api.openweathermap.org/data/2.5/weather?q= **မိမိတို့ ထည့်လိုသည့် မြို့အမည်** &appid=**မိမိတို့ သိမ်းထားသော keys**

အထက်ပါ ပုံများအတိုင်း browser url ၌ ရေးသားပြီးလျှင် enter နှိပ်လိုက်ပါက အောက်ပါအတိုင်း data များရရှိလာလျှင် API key ယူခြင်း အောင်မြင်ပါပြီ။ ထိုကဲ့သို့ အဆင်မပြေခဲ့ပါက အဆင့်တိုင်း စစ်ဆေးရန် လိုအပ်ပါသည်။

{"coord":{"lon":96.08,"lat":21.97},"weather":[{"id":500,"main":"Rain","description":"light rain","icon":"10n"}],"base":"stations","main":
{"temp":303.63,"feels_like":304.35,"temp_min":303.63,"temp_max":303.63,"pressure":1003,"humidity":55,"sea_level":1003,"grnd_level":994},"win
{"country":"MM","sunrise":1589842501,"sunset":1589890174},"timezone":23400,"id":1311874,"name":"Mandalay","cod":200}

ထို့နောက် browser မှာ URL တစ်ခုလုံးအား copy ကူးခဲ့ပြီး မိမိတို့ program ထဲတွင် သွားရောက်သုံးပါမည်။

```
keys='8a6d6b7f7fe81a7b09ebbfa266296ce2'
http://api.openweathermap.org/data/2.5/weather?q=Mandalay&appid=8a6d6b7f7fe81a7b09ebbfa266296ce2
```

မိမိတို့ သိလိုသော မြို့နယ် မိမိတို့ရဲ့ API keys

ပြီးလျှင် အောက်ပါအတိုင်း program ကို ပြောင်းရေးပါမည်။ မြို့နာမည်ကို မိမိကြိုက်တဲ့ ထည့်နိုင်ရန်အတွက် မြို့အတွက်ကို နောက်မှာထားရေးပါမည်။ တစ်လုံးမှ သေးသေးလေးမှ မှားလို့မရသည့်အတွက် အောက်ပါပုံကို သေချာ ကြည့်ရန် လိုအပ်ပါသည်။

```
key = '8a6d6b7f7fe81a7b09ebbfa266296ce2'
url ='http://api.openweathermap.org/data/2.5/weather?appid=8a6d6b7f7fe81a7b09ebbfa266296ce2&q='
```

အထက်ပါ ပုံထဲကအတိုင်း key ကိုရှေ့ပို့ပြီး &q= ကို နောက်ဆုံးမှာ ထားပါမည်။ အဘယ်ကြောင့် ဆိုသော် မိမိထည့်လိုသည့် မြို့ကို နောက်မှ ပေါင်းထည့်နိုင်ရန် ဖြစ်သည်။

ထို့နောက် terminal or command တွင် pip install requests ကို ရေးပြီး install လုပ်ပေးရန် လိုအပ်ပါသည်။

```
PS C:\Users\User> pip install requests
```

ခဏစောင့်ပြီးလျှင် requests လုပ်ပြီးသောအခါ မိမိတို့ program ထဲတွင် requests ကို ခေါ်သုံးလို့ရပါပြီ။ ထို requests သည် API data များကို ရယူရန် ဖြစ်သည်။

```
import json
import requests
key = 'မိမိတို့ API keys'
url   ='http://api.openweathermap.org/data/2.5/weather?appid=  မိမိတို့ API keys &q='
```

Program file ထဲတွင် ရေးရမည့် ပုံစံမှာ အထက်ပါအတိုင်းဖြစ်ပြီး program အပြည့်အစုံကို အောက်တွင် ဖော်ပြထားသည်။

Sample Program (297)

```
import json
import requests
key = '8a6d6b7f7fe81a7b09ebbfa266296ce2'
url
='http://api.openweathermap.org/data/2.5/weather?appid=8a6d6b7
f7fe81a7b09ebbfa266296ce2&q='
cityName=input('Please enter your city name :')
newUrl=url+cityName
```

Sample Program (297) သည် user ထည့်ပေးလိုက်သော မြို့အမည်ကို url နှင့် ပေါင်းပြီး ပြည့်စုံသည့် api request url တစ်ခုရရှိရန်အတွက် ဖြစ်သည်။

Sample Program (298)

```
import json
import requests
key = '8a6d6b7f7fe81a7b09ebbfa266296ce2'
url
='http://api.openweathermap.org/data/2.5/weather?appid=8a6d6b7
f7fe81a7b09ebbfa266296ce2&q='
cityName=input('Please enter your city name')
newUrl=url+cityName

jsonData = requests.get(newUrl).json()
print(type(jsonData))
print(jsonData)
```

Sample Program (298)ကို run ပြီးချိန်၌ အောက်ပါအတိုင်း JSON format ဖြင့် Weather data များကို ရရှိပါမည်။

```
PS C:\Users\User\Downloads> python .\100.py
Please enter your city name: Yangon
<class 'dict'>
{'coord': {'lon': 96.16, 'lat': 16.81}, 'weather': [{'id': 804, 'main': 'Clouds', 'description': 'overcast clouds
: 'stations', 'main': {'temp': 302.34, 'feels_like': 305.26, 'temp_min': 302.34, 'temp_max': 302.34, 'pressure':
_level': 1005, 'grnd_level': 1001}, 'wind': {'speed': 4.2, 'deg': 163}, 'clouds': {'all': 99}, 'dt': 1589897296,
sunrise': 1589842996, 'sunset': 1589889640}, 'timezone': 23400, 'id': 1298824, 'name': 'Yangon', 'cod': 200}
PS C:\Users\User\Downloads>
```

ထိုကဲ့သို့ ရရှိလာသော data များကို လွယ်ကူစွာဖတ်နိုင်ရန် အောက်ပါအတိုင်း program ကို ပြောင်းရေးလိုက်ပါမည်။

```
jsonData = requests.get(newUrl).json()
print(type(jsonData))
data = json.dumps(jsonData , indent=4)
print(type(data))
print(data)
```

```
PS C:\Users\User\Downloads> python .\100.py
Please enter your city name: Yangon
<class 'dict'>
<class 'str'>
{
    "coord": {
        "lon": 96.16,
        "lat": 16.81
    },
    "weather": [
        {
            "id": 804,
            "main": "Clouds",
            "description": "overcast clouds",
            "icon": "04n"
        }
    ],
    "base": "stations",
    "main": {
        "temp": 302.34,
        "feels_like": 305.26,
        "temp_min": 302.34,
        "temp_max": 302.34,
        "pressure": 1005,
        "humidity": 74,
        "sea_level": 1005,
```

Output များကို ကြည့်ပါ။ ရိုးရှင်းစွာ ဖတ်ရလွယ်သည်ကို တွေ့ရပါမည်။ dumps method ကို သုံးလိုက်လျှင် မူရင်း python dictionary object မှ string object သို့ပြောင်းသွားသည်ကို သတိပြုပါ။ အထက်ပါပုံတွင် လိမ္မော်ရောင်ဖြင့် ဝိုင်းပြထားပါသည်။ အထက်ပါ program တွင် jsonData ထဲမှ ပါဝင်သော keys များကိုသာ သိလိုလျှင်လည်း အောက်ပါအတိုင်း ရေးနိုင်သည်။

```
jsonData = requests.get(newUrl).json()
print(type(jsonData))
for i in jsonData:
    print(i)
```

Output:

```
Please enter your city name: yangon
<class 'dict'>
coord
weather
base
main
wind
clouds
dt
sys
timezone
id
name
cod
```

ထို့ပြင် coord ဆိုသည့် coordinate ထဲမှ lat and lon value များကို သိလိုလျှင်လည်း အောက်ပါတိုင်း ရေးကြည့်နိုင်သည်။

Sample Program (299)

```
jsonData = requests.get(newUrl).json()
print(type(jsonData))
for i in jsonData:
    print(i)
print(jsonData['coord'])
```

အထက်ပါ အတိုင်းရေးကြည့်ပါ။ ouput တွင် မိမိတို့ထည့်လိုက်သော မြို့ရဲ့ lat and lon value များကို ရရှိပါမည်။ ထို့ပြင် lat and lon value များကို မိမိတို့စိတ်ကြိုက် ပြောင်းလိုပါကလည်း အောက်ပါအတိုင်း ပြောင်းနိုင်ပါသေးသည်။ အောက်တွင် program အပြည့်အစုံကို ဖော်ပြထားပါသည်။

Sample Program (300)

```
import json
import requests
key = '8a6d6b7f7fe81a7b09ebbfa266296ce2'
url
='http://api.openweathermap.org/data/2.5/weather?appid=8a6d6b7f7fe81
a7b09ebbfa266296ce2&q='
cityName=input('Please enter your city name: ')
newUrl=url+cityName
jsonData = requests.get(newUrl).json()
print(type(jsonData))
for i in jsonData:
    print(i)
print(jsonData['coord'])
```

```
jsonData['coord']['lon']=100
print(jsonData['coord']['lon'])
```

Sample Program (300) မှာ အောက်ဆုံးလိုင်းရဲ့ အပေါ်တစ်ကြောင်းတွင် lon value အား 100 ဟု ပြောင်းလိုက်ပါသည်။ ထို့နောက် value များကို အောက်ဆုံးလိုင်းတွင် ပြန်ထုတ်ကြည့်ပါက lon value ပြောင်းလဲသွားသည်ကို မြင်ရပါမည်။

Database With MySQL

ပထမဆုံး အနေဖြင့် MySQL ကို Download ရယူနိုင်ရန် https://dev.mysql.com/downloads/installer/ ယခု link ကိုသွားပါ။

Download option နှစ်ခု တွေ့ပါမည်။ အပေါ်က တစ်ခုသည် web base installer ဖြစ်တဲ့အတွက် အောက်က ဒုတိယ တစ်ခုကိုသာ ရယူပါမည်။ Download ကို တစ်ချက်နှိပ်လိုက်သည်နှင့် နောက်ထပ် page တစ်ခုကို ရောက်သွားပါမည်။

No thanks, just start my download ဆိုတာကို နှိပ်ပြီး ဆက်သွားပေးပြီးလျှင် download ကျလာပါလိမ့်မယ်။ file size များသည့်အတွက် internet connection ပေါ်မူတည်ပြီး အချိန်အနည်းငယ် ကြာတတ်ပါသည်။ Download ပြီးပါက install

စလုပ်ပေးပါ။ အောက်ပါ အတိုင်း ပေါ်လာမည်ဖြစ်ပြီး ခဏစောင့်ပေးရပါသည်။

အခြားသော option များကို မရွေးပဲ Developer Default ကိုသာ ရွေးချယ်ပြီး next ဆိုတာကို ထပ်နှိပ်ပေးပါ။ နောက်ထပ်ပေါ်လာသော window form တွင်လည်း next ဆိုတာကိုသာ ထပ်နှိပ်ပေးပါ။

ထပ်ပေါ်လာသော window form အသေးလေးတွင် yes ဆိုတာကိုပဲ ထပ်နှိပ်ပေးပါ။

နောက်ထပ်ပေါ်လာသော နေရာတွင် Execute ကိုသာ ဆက်နှိပ်ပေးပါ။

အောက်ပါ ပုံအတိုင်း အားလုံးကို install လုပ်သွားပါလိမ့်မည်။ ထို့နောက် next ကို ထပ်နှိပ်ပေးပါ။

ယခုကဲ့သို့ ပေါ်လာလျှင် next ကိုပဲ ထပ်နှိပ်ပေးပါ။ နောက်ထပ်ပေါ်လာသော အကွက်တွင်လည်း next ကိုပဲ နှိပ်ပေးပါ။

Next ကိုပဲ ထပ်နှိပ်ပေးပါ။

အောက်ပါအတိုင်း ပေါ်လာသော နေရာတွင် password ကို နှစ်ခါထည့်ပေးဖို့

လိုအပ်ပါသည်။ မိမိတို့ ပေးချင်သည့်အတိုင်း secure ဖြစ်မည့်အတိုင်း ပေးနိုင်ပါသည်။ စာရေးသူအနေဖြင့် password နှစ်လုံးကို toor ဟုသာ ပေးခဲ့ပါမည်။

အထက်ပါ ပုံထဲကအတိုင်း password နှစ်ခုလုံးပေးပြီးလျှင် Add User ကို နှိပ်ပါ ။

စာရေးသူ အနေဖြင့် user name အား winhtut ဟု ပေးခဲ့ပြီး passoword နေရာတွင် toor ဟုသာ နှစ်ခုလုံးပေးခဲ့ပါသည်။ပြီးလျှင် ok ကို နှိပ်ပေးပါ။ မိမိတို့အနေဖြင့် ပေးခဲ့သည့် username and password များကို သေချာမှတ်သားထားရန် လိုအပ်ပါသည်။

အထက်ပါ ပုံထဲက အတိုင်း next ဆိုတာကို ထပ်နှိပ်ပေးပါ။ နောက်ထပ်ပေါ်လာသော window form တွင်လည်း Next ကိုပဲ ထပ်နှိပ်ပေးပါ။

အထက်ပါအတိုင်း ပေါ်လာလျှင် Execute ကို နှိပ်ပါ။ ပြီးသွားလျှင် Finish ကို နှိပ်ပါ။

ထို့နောက် next ကိုပဲ ထပ်နှိပ်ပေးပါ။

အထက်ပါအတိုင်း ပေါ်လာလျှင် Finish ကိုပဲ ဆက်နှိပ်ပေးပါ။ နောက်ထပ်ပေါ်လာသော အကွက်တွင်လည်း Next ကိုပဲ ထပ်နှိပ်ပေးပါ။

Password နေရာတွင် မိမိတို့ပေးထားခဲ့သော password ကိုသာ ပြန်ပေးလိုက်ပါ။ စာရေးသူအနေဖြင့် toor ကိုသာ အသုံးပြုခဲ့သည့်အတွက် toor ကိုသာထည့်လိုက်ပါသည်။ ထို့နောက် Check ကို နှိပ်ပေးပါ ။ check ကို နှိပ်ပြီးသည်နှင့် တစ်ပြိုင်နက် အပေါ်က running ဆိုသော နေရာတွင် အစိမ်းရောင် ပြောင်းသွားပါမည် ။ Connection succeeded ဆိုသည့် စာသားကိုလည်း မြင်ရပါမည်။ ထို့နောက် Next ကို နှိပ်ပေးရပါမည်။

နောက်ထပ် ပေါ်လာသော window form တွင် Execute ကိုသာ နှိပ်ပေးပါ။ ထို့နောက် Finish ပြီးလျှင် next ကို ထပ်နှိပ်ပေးပါ။ နောက်ဆုံးပေါ်လာသော နေရာတွင် အမှန်ခြစ်များကို ဖြုတ်ပေးပြီး Finish ကို နှိပ်ပေးလိုက်ပါ။

ထို့နောက် အောင်မြင်ကြောင်း စစ်ဆေးရန် အောက်ပါအတိုင်း ဆက်လုပ်လုပ်ဆောင်ပါ။

ထို့နောက် MySQL Workbench ကို ရှာပြီး ဖွင့်လိုက်ပါ။

အပေါ်ပုံအတိုင်း ပေါ်လာလျှင် Local instance MySQL80 ကို တစ်ချက် နှိပ်ပေးလိုက်ပါ။ အောက်ပါ အတိုင်း window form အသေးလေးပေါ်လာပါမည်။ ထို့သို့ ပေါ်လာသောနေရာတွင် password ကို မိမိတို့သတ်မှတ်ပေးခဲ့တဲ့အတိုင်း ပေးပြီး OK ကို နှိပ်ပေးလိုက်ပါ။

OK နှိပ်ပြီးလျှင် အောက်ပါအတိုင်း အကွက်တစ်ခု ပေါ်လာရပါမည်။

နောက်ဆုံးအနေဖြင့် အောင်မြင်ကြောင်း စစ်ဆေးရန် window + R ကို နှိပ်ပြီး cmd ဟုရေးကာ enter ခေါ်ပေးပါ။ command promt တွင် python -m pip install mysql-connector ကိုရေးပေးပြီး install လုပ်ပေးပါ။

```
C:\Users\User>python -m pip install mysql-connector
Collecting mysql-connector
  Downloading https://files.pythonhosted.org/packages/28/04/e40098f3730
f4fc4787a/mysql-connector-2.2.9.tar.gz (11.9MB)
     |                                | 11.9MB 731kB/s
Installing collected packages: mysql-connector
    Running setup.py install for mysql-connector ... done
Successfully installed mysql-connector-2.2.9
WARNING: You are using pip version 19.3.1; however, version 20.1.1 is a
You should consider upgrading via the 'python -m pip install --upgrade

C:\Users\User>python
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 19:29:22) [MSC v.191
Type "help", "copyright", "credits" or "license" for more information.
>>> import mysql
>>>
```

အထက်ပါ ပုံထဲအတိုင်း install ပြီးသွားလျှင် python ဆိုတာကို တစ်ချက်နှိပ်ပြီး python interpreter ကိုသွားပါ။ ထို့နောက် ပုံထဲကအတိုင်း import mysql ဟုရေးပြီး enter ခေါက်ကြည့် ပါ။ error တက်မလာခဲ့လျှင် mysql ကို install လုပ်ခြင်း အောင်မြင်ပါပြီ။ ကျွန်ုပ်တို့ python program တွင် ခေါ်သုံးနိုင်ပါပြီ။

Connecting to MySQL Database

MySQL database နှင့် python တို့ ချိတ်ဆက်ဖို့ အတွက် mysql.connector ထဲမှ connect() ဆိုသည့် method ကို ခေါ်သုံးရပါမည်။ ထို့ကြောင့် mysql connector python ကိုလည်း install လုပ်ပေးရန် လိုအပ်ပါသေးသည်။

```
\Users\User>pip3 install mysql-connector-python
quirement already satisfied: mysql-connector-python in c
lib\site-packages (8.0.20)
```

ထိုကဲ့သို့ အဆင်ေဆင့် ပြုလုပ်ပြီးသည့်နောက် မိမိတို့ program ထဲတွင် အောက်ပါ အတိုင်း ရေးပြီး ချိန်ဆက်နိုင်ပါပြီ။

Sample Program (301)

```
import mysql.connector
db_connection = mysql.connector.connect(
  host="localhost",
  user="root",
  passwd="toor"
)
print(db_connection)
```

Host နေရာတွင် localhost ကို ရေးပေးရမည်။ ကျွန်ုပ်တို့အနေဖြင့် mySQLကို localhost မှာ run ထားသောကြောင့်ဖြစ်သည်။

မိမိတို့ computer ညာဘက်အောက်က notification နေရာတွင် တွေ့နိုင်သည်။ User သည် မိမိတို့ MySQL install လုပ်တုန်းက ပေးခဲ့သည့်အတိုင်းပင် ဖြစ်ပြီး password နေရာတွင်လည်း ထိုနည်းအတိုင်းပင် ဖြစ်သည်။ အထက်တွင် ရေးထားသော program အား run ကြည့်ပါက output အနေဖြင့် object တစ်ခုရလာသည်ကို တွေ့ရပါမည်။

<mysql.connector.connection.MySQLConnection object at 0x03A8AD10>

အထက်ပါအတိုင်း output ထွက်လာပါက MySQL database သို့ python နှင့် connect လုပ်ခြင်း အောင်မြင်ပါပြီ။

Creating Database

Database တစ်ခုကို စတင် create လုပ်နိုင်ဖို့အတွက် cursor ဆိုသည့် object တစ်ခုကို အရင်ဆုံး ပြုလုပ်ပေးရန် လိုအပ်ပါသည်။ ထို cursor object သည် database ကို ကိုင်တွယ်ရာ တွင် အသုံးပြုရန် ဖြစ်ပြီး SQL နှင့် ပတ်သက်သည့် operation များကို လုပ်ဆောင်နိုင်သည်။ ဆိုလိုသည်မှာ database တစ်ခု ပြုလုပ်တာမျိုးပါ။

Sample Program (302)

```
import mysql.connector
db_connection = mysql.connector.connect(
  host="localhost",
  user="root",
  passwd="toor"
)
#creating a cursor object
db_cursor = db_connection.cursor()
db_cursor.execute("CREATE DATABASE myFirstDB") #line 9
db_cursor.execute("SHOW DATABASES") #line 10

for db in db_cursor: #line 11
  print(db)
#output:
('information_schema',)
('myfirstdb',)
('mysql',)
('performance_schema',)
('sakila',)
('sys',)
('world',)
```

Database တစ်ခု create လုပ်ခြင်း create လုပ်လိုက်သော database များကို ပြန်ထုတ်ခြင်းများကို လုပ်နိုင်ရန် execute ဆိုသည့် method ကို အသုံးပြုရပါမည်။ အထက်ပါ program တွင် line 9 ၌ execute ဆိုသည့် method ကိုသုံးပြီး database တစ်ခု create လုပ်ပါသည် ။ create လုပ်နိုင်ရန် CREATE DATABASE ဆိုသည့် SQL command ကို သုံးရမည် ဖြစ်ပြီး CREATE DATABASE နောက်မှ myFirstDB သည် Database name ဖြစ်သည်။ နောက်ထပ် code line တွင် ရေးထားသော SHOW DATABASES သည် user က ဆောက်ထားသမျှ သို့မဟုတ် SQL server ပေါ်မှာ ရှိသမျှ Database အားလုံးကို ပြန်ရန် ဖြစ်သည်။ ထို့နောက် db_cursor ထဲမှ database အားလုံးကို loop ပတ်ပြီး ထုတ်ပြလိုက်ပါသည်။

Creating a Table

အထက်ပါ program တွင် myFirstDB ဆိုသည့် Database တစ်ခုတည်ဆောက်ပြီး ဖြစ်ပြီး ယခုသင်ခန်းစာတွင် ထို database ထဲသို့ table တစ်ခုထည့်မှာပါ။ ထိုသို့ထည့်ရာတွင် မိမိတို့ တိုက်ရိုက်ထည့်လိုသည့် database name အား အောက်ပါအတိုင်း ဖော်ပြနိုင်သည်။

Sample Program (303)

```
import mysql.connector
dbConnection = mysql.connector.connect(
host="localhost",
user="root",
```

```
    passwd="toor",
    database="myFirstDB"
    )
```

ထို့နောက် table တစ်ခု တည်ဆောက်နိုင်ရန် အောက်ပါအတိုင်း ဆက်ရေးရမည်။

```
dbCursor=dbConnection.cursor()
dbCursor.execute("CREATE TABLE Students(id int primary key AUTO_INCREMENT,name VARCHAR(30),age SMALLINT,attend TINYINT)")
```

CREATE TABLE သည် table တစ်ခု တည်ဆောက်ရန် SQL command ဖြစ်ပြီး Students သည် table name ဖြစ်သည်။ ထို table ထဲတွင် column အနေဖြင့် id , name , age , attend ဟု column 4 ခု ထည့်ထားပါသည်။ first column ဖြစ်သည့် id အား int primary key AUTO_INCREMENT ဟု ပေးထားခြင်းမှာ column အသစ်တိုးလာတိုင်း id number များ အလိုလျောက်တိုးလာရန် ဖြစ်ပြီး primary key ဆိုတာကတော့ column တစ်ခုခြင်းစီတိုင်းကို key များဖြင့် သတ်မှတ်ထားရန် ဖြစ်သည်။ name နောက်မှ VARCHAR(30) သည် variable character အလုံး ၃၀ ခန့်ထည့်နိုင်ရန် ဖြစ်သည်။ age နောက်မှ SMALLINT သည် 2 bytes ခန့် integer များ stored လုပ်နိုင်ရန် ဖြစ်သည် mysql တွင် TINYINT 1 BYTE, SMALLINT 2 BYTES , MEDIUMINT 3 BYTES , INT 4 BYTES , BIGINT 8 BYTES တို့ အသီးသီး stored လုပ်နိုင်ကြပါသည်။ attend နောက်မှ TINYINT သည် 1 BYTE ခန့် stored လုပ်နိုင်ရန် ဖြစ်ပါသည်။ အောက်ပါအတိုင်းcode line ၃ ကြောင်း ထပ်ထည့်ပြီး run ကြည့်ပါက မိမိတို့ဆောက်လိုက်သော table အား တွေ့ရပါမည်။

```
dbCursor.execute("DESCRIBE Students")
for tb in dbCursor:
    print(tb)
```

Output:

```
('id', 'int', 'NO', 'PRI', None, '')
('name', 'varchar(30)', 'YES', '', None, '')
('age', 'int', 'YES', '', None, '')
('attend', 'int', 'YES', '', None, '')
```

သတိပြုရန် အချက်မှာ program ကို နောက်တစ်ကြိမ်ပြန် run သောအခါတွင် CREATE TABLE ဆိုသည့် code line အားဖြုတ်ထားရန် လိုအပ်ပါသည်။ အကယ်၍ မဖြုတ်ထားဘူးဆိုလျှင် နောက်တစ်ကြိမ် Students ဆိုသည့် table ကို ထပ်ဆောက်မှာ ဖြစ်တဲ့အတွက် အရင်ဆောက်ထား သော students table နှင့် name တူနေသောကြောင့် error တက်မှာ ဖြစ်ပါတယ်။

```
 File "mysqldb.py", line 11, in <module>
   dbCursor.execute("CREATE TABLE Students(id int(20)primary key,name VARCHAR(30),age int(3),a
3))")
py", line 569, in execute
   result = self._handle_result(self._send_cmd(ServerCmd.QUERY, query))
 File "C:\Users\User\AppData\Local\Programs\Python\Python37-32\lib\site-packages\mysql\connect
   raise errors.get_exception(packet)
ysql.connector.errors.ProgrammingError: 1050 (42S01): Table 'students' already exists
```

Adding More column

အထက်ပါ သင်ခန်းစာမှာဆိုလျှင် ကျွန်ုပ်တို့အနေဖြင့် columns လေးခုကိုသာ တည်ဆောက်ခဲ့ပါသည်။ အကယ်၍ application ရဲ့ လိုအပ်ချက်အရ နောက်ထပ် column တွေ ထပ်တိုးလိုတဲ့အခါမှာတော့ ALTER TABLE ဆိုသည့် sql command ကို အသုံးပြုနိုင်ပါတယ်။ အောက်မှာတော့ myFirstDB ထဲမှ Students ဆိုသည့် table ထဲကိုပဲ column တစ်ခုထပ်တိုးပြီး ပြပေးထားပါတယ်။

Sample Program (304) - ALTER TABLE

```
import mysql.connector
dbConnection = mysql.connector.connect(
host="localhost",
user="root",
passwd="toor",
database="myFirstDB"
)
dbCursor=dbConnection.cursor()
dbCursor.execute("alter table Students ADD hobby VARCHAR(20)")
dbCursor.execute("DESCRIBE Students")
for tb in dbCursor:
    print(tb)
```

Output::

```
('id', 'int', 'NO', 'PRI', None, '')
('name', 'varchar(30)', 'YES', '', None, '')
('age', 'int', 'YES', '', None, '')
('attend', 'int', 'YES', '', None, '')
('hobby', 'varchar(20)', 'YES', '', None, '')
```

Sample Program (304) ကို run ကြည့်ပါက output တွင် hobby ဆိုသည့် column တိုးလာ သည်ကို မြင်ရပါမည်။ alter table command ကိုအသုံးပြုပုံမှာ alter table နောက်တွင် table name ရှိရပါမည်။ ထို့နောက်တွင် ADD ဆိုသည့် command ကိုသုံးရမည်ဖြစ်ပြီး add နောက်တွင် column name နှင့် column ရဲ့ type ကို ထည့်ပေးရပါမည်။

Adding Data to the Table

Table တစ်ခုထဲသို့ data ထည့်မည်ဆိုလျှင် ထည့်မည့် data များရဲ့ type များကိုလည်း သတ်မှတ်ပေးဖို့ လိုအပ်ပါသည်။ ဥပမာ %s string အနေဖြင့် ထည့်မလား သို့မဟုတ် %d

decimal အနေဖြင့် ထည့်မလား စသဖြင့် ဆိုလိုသည်။ စာရေးသူအနေဖြင့် datatype and data ဆိုသည့် object နှစ်ခုတည်ဆောက်ပြီးမှ အသုံးပြုပါမည်။ ပိုမိုရှင်းလင်းစေပြီး ဖတ်ရလွယ်ကူစေရန် ဖြစ်သည်။ datatype object ထဲတွင် မိမိထည့်မည့် data type များကို သတ်မှတ်ပေးထားမည် ဖြစ်ပြီး data ထဲတွင် မိမိတို့ထည့်မည့် data များကို သတ်မှတ်ပေးထားပါမည်။

```
dataType = "INSERT INTO Students(id , name , age ,attend ,
hobby)values(%s , %s ,%s ,%s ,%s)"
data = (101,"WinHtut",20,3,"IoT")
```

ထို့နောက်တွင် execute ဆိုသည့် method ကိုသုံးပြီး ထို method ထဲတွင် datatype နှင့် data object တို့ကို အောက်ပါအတိုင်း ထည့်ပေးလိုက်ပါမည်။

```
dbCursor.execute(dataType,data)
```

အထက်ပါ code line တွင် execute သည် table ထဲသို့ object နှစ်ခုမှ သတ်မှတ်ထားသည့် အတိုင်း data များကို ထည့်ပေးပါသည်။ ထို့နောက် table ထဲသို့ data များ ရောက်မရောက် သိနိုင်ရန် rowcount ကို သုံးပြီး သိနိုင်ပါသည်။ အကယ်၍ data များ ရောက်သွားလျှင် output အနေဖြင့် row တစ်ခုတည်းရှိသေးတာ ဖြစ်သည့်အတွက် 1 ဆိုသည့် output ကို ပြန်ပေးမှာ ဖြစ်ပါတယ်။ sample code မှာ အောက်ပါ အတိုင်းဖြစ်သည်။

```
print(dbCursor.rowcount)
```

နောက်ဆုံးအနေဖြင့် Table ထဲမှ Data များကို ဆွဲထုတ်ရန် SELECT * FROM <table name > ဆိုသည့် sql command ကို သုံးနိုင်ပါသည်။

Sample Program (305) - SELECT * FROM

```
import mysql.connector
dbConnection = mysql.connector.connect(
host="localhost",
user="root",
passwd="toor",
database="myFirstDB"
)
dataType = "INSERT INTO Students(id , name , age ,attend ,
hobby)values(%s , %s ,%s ,%s ,%s)"
data = (101,"WinHtut",20,3,"IoT")
dbCursor=dbConnection.cursor()
dbCursor.execute(dataType,data)
dbConnection.commit()
print("Data are inserted")
print(dbCursor.rowcount)
dbCursor.execute("SELECT * FROM Students")
for d in dbCursor:
    print(d)
```

Output::
Data are inserted
1
(101, 'WinHtut', 20, 3, 'IoT')

Insert Many Data to the Table

ယခုတစ်ခါမှာတော့ Data တွေအများကြီးကို တစ်ပြိုင်တည်း table ထဲသို့ ထည့်ပါမည်။ ထိုသို့ထည့်ရန်အတွက် မိမိထည့်လို့သော data များကို list တစ်ခု ဆောက်ပြီး list ထဲတွင် tuple အနေဖြင့် အစုလိုက် အောက်ပါအတိုင်း ပြုလုပ်ထားပါမည်။

```
data = [(106,"Vij",20,3,"IoT"),(107,"codelock",20,3,"IoT")]
```

သတိထားရန်အချက်မှာ data အများကြီးကို တစ်ပြိုင်တည်း ထည့်လိုသောအခါတွင် execute အစား executemany ကိုသုံးပေးရပါမည်။ ထိုသို့မသုံးပေးလျှင် အောက်ပါပုံအတိုင်း error တက်နိုင်ပါသည်။

```
 to_mysql
    "MySQL type".format(type_name))
TypeError: Python 'tuple' cannot be converted to a MySQL type
```

ထို့နောက်တွင် မိမိထည့်လိုက်သော data များကို mySQL server ပေါ်တွင် သွားရောက်သိမ်းဆည်းရန် commit() ဆိုသည့် method ကို သုံးပေးရပါမည်။

```
dbConnection.commit()
```

Sample Program (306) - executemany

```
import mysql.connector
dbConnection = mysql.connector.connect(
host="localhost",
user="root",
passwd="toor",
database="myFirstDB"
)
dataType = "INSERT INTO Students(id , name , age ,attend ,
hobby)values(%s , %s ,%s ,%s ,%s)"
data = [(106,"Vij",20,3,"IoT"),(107,"codelock",20,3,"IoT")]
dbCursor=dbConnection.cursor()
dbCursor.executemany(dataType,data)
dbConnection.commit()
print("Data are inserted")
print(dbCursor.rowcount)
dbCursor.execute("SELECT * FROM Students")
for d in dbCursor:
    print(d)
```

Fetching

Fetching ဆိုတာကတော့ database ထဲမှ သက်ဆိုင်ရာ table ထဲကနေ မိမိလိုချင်သော data များကို ရယူတာ ဖြစ်ပါတယ်။ ထိုကဲ့သို့ data များ fetch လုပ်နိုင်ရန် SELECT FROM ဆိုသည့် sql command ကို အသုံးပြုရပါတယ်။

```
db_cursor.execute("SELECT id , name , age FROM Students")
```

အထက်ပါ code line တွင်ရေးသားထားသည်မှာ Students table ထဲမှ data များကို မိမိတို့ လိုအပ်သည့် id , name , age တို့ကိုသာ ထုတ်ယူရန် ဖြစ်ပါသည်။ ထိုကဲ့သို့ ရေးပြီးလျှင် table ထဲမှ ဘယ် column များကို ထုတ်ယူရမည် ဆိုတာကို db_cursor ဆိုသည့် object မှ သိသွားပါမည်။ ထို့နောက် data များအားလုံးကို အမှန်တကယ် ထုတ်ယူနိုင်ရန် fetchall() ဆိုသည့် method ကို အသုံးပြုပေးရပါမည်။

```
allData =db_cursor.fetchall()
```

အထက်ပါ code line run ပြီးချိန်တွင် allData ဆိုသည့် object ထဲ၌ data များအားလုံး ရှိနေမည် ဖြစ်ပါသည်။ program အပြည့်အစုံကို အောက်တွင် ဖော်ပြထားပါသည်။

Sample Program (307)

```
import mysql.connector
db_connection = mysql.connector.connect(
host ="localhost",
user="root",
passwd="toor",
database="myFirstDB"
)
db_cursor = db_connection.cursor()
db_cursor.execute("SELECT id , name , age FROM Students")
allData =db_cursor.fetchall()
for db in allData:
    print(db)
```

Fetching only one

ယခု သင်ခန်းစာမှာတော့ row တစ်ခုတည်းမှ data များကိုသာ ထုတ်ယူသွားမှာ ဖြစ်ပါတယ်။ အသုံးပြုရမည့် method မှာ fetchone() ဖြစ်ပါသည်။ sample program ကို အောက်တွင် ဖော်ပြထား ပါသည်။

Sample Program (308)

```
db_cursor = db_connection.cursor()
db_cursor.execute("SELECT id , name , age FROM Students")
data =db_cursor.fetchone()
print(data)
```

Format

Table ထဲမှ ရရှိလာသော data များကို မိမိတို့နှစ်သက်သည့်အတိုင်း format ပြုလုပ်ပြီး print ထုတ်နိုင်ပါသေးသည်။ ထိုသို့အသုံးပြုရန် string ပုံစံဖြင့်ထုတ်ချင်ပါက %s ကိုသုံးပေးရမည် ဖြစ်ပြီး decimal integer အနေဖြင့် ထုတ်ချင်ပါက %d ကို သုံးပေးရမည် ဖြစ်ပါသည်။ sample program ကို အောက်တွင် ဖော်ပြထားပါသည်။

Sample Program (309)

```
db_cursor = db_connection.cursor()
db_cursor.execute("SELECT id , name , age FROM Students")
data =db_cursor.fetchall()
for db in data:
    print("%d %s %d"%(db[0],db[1],db[2]))
```

Using where

ယခုတစ်ခါမှာတော့ မိမိလိုချင်သည့် data များသာ သီးသန့်ခွဲထုတ်မှာ ဖြစ်ပါတယ်။ ထို့သို့ ခွဲထုတ်နိုင်ရန် where command ကို အသုံးပြုနိုင်ပါသည်။ ပထမဆုံးအနေဖြင့် မိမိတို့ database ထဲတွင် သိမ်းဆည်းထားသော data များကို သိနိုင်ရန် MySQL workbench ထဲသို့ ဝင်ပါ။ MySQL workbench သည် မိမိတို့ SQL ကို install လုပ်ကတည်းက ပါဝင်နေပြီး ဖြစ်ပါသည်။ workbench ထဲမှ Schema ထဲသို့ဝင်ပါ။ ထို့နောက် မိမိတို့ Database ထဲကိုဝင်ပါ။ စာရေးသူအနေဖြင့် myfirstdb ဖြစ်ပါသည်။ myfirstdb ထဲမှ Tables ပြီးလျှင် students ထို့နောက် students ဘေးမှ ဇယားကွက်လေးကို တစ်ချက် နှိပ်လိုက်ပါက မိမိတို့ table ပေါ်လာပါမည်။

စာရေးသူရဲ့ database table ထဲတွင် vij နာမည်နှင့်က သုံးခု codelock နာမည် နှင့်က သုံးခု ဖြစ်ပါသည်။ ထိုထဲမှ စာရေးသူသည် vij name နှင့် data များကိုသာ ရယူလိုပါသည်။ ထိုသို့သော အခြေနေမျိုးတွင် where command ကို သုံးပါသည်။

```
db_cursor.execute("SELECT id , name , age FROM Students WHERE name
like 'v%'")
```

အသုံးပြုပုံမှာ execute method ထဲတွင်ပင် SELECT <မိမိလိုချင်သော data> FROM <table name > WHERE name like 'v%' ဖြစ်ပါသည်။ စာရေးသူသည် v နဲ့စသည့် နာမည်အားလုံးကို ထုတ်လိုသည့်အတွက် v% ဟု ရေးထားပေးခြင်း ဖြစ်ပါသည်။ program အပြည့်အစုံနှင့် output ကို အောက်တွင် ဖော်ပြထားသည်။

Sample Program (310)

```
db_cursor = db_connection.cursor()
db_cursor.execute("SELECT id , name , age FROM Students WHERE name
like 'v%'")
data =db_cursor.fetchall()
for db in data:
    print("%d %s %d"%(db[0],db[1],db[2]))
```

Output:

102 Vij 20

104 Vij 20

106 Vij 20

Where ကို id နှင့်ပါ အသုံးပြုနိုင်ပါသေးသည်။ အသုံးပြုပုံမှာ WHERE နောက်တွင် id ကိုရေးပြီး id နောက်တွင် in (မိမိထုတ်လို သော id များ) ဟု ရေးသားရပါမည်။ sample program ကို အောက်တွင် ဖော်ပြထားပါသည်။

Sample Program (311)

```
db_cursor = db_connection.cursor()
db_cursor.execute("SELECT id , name , age FROM Students WHERE id
in(102,103,104)")
data =db_cursor.fetchall()
for db in data:
    print("%d %s %d"%(db[0],db[1],db[2]))
```

Output:

102 Vij 20

103 codelock 20

104 Vij 20

Order

Name များကို Alphabet အလိုက် ordering လုပ်နိုင်ပါသေးသည်။ ထိုသို့ လုပ်နိုင်ရန် ORDER BY ဆိုသည့် sql command ကို အသုံးပြုနိုင်ပါသည်။ အသုံးပြုပုံမှာ မိမိ data ထုတ်လိုသော table name နောက်တွင် ORDER BY name ဟုရေးပြီး ordering လုပ်နိုင်ပါသည်။

Sample Program (312)

```
db_cursor = db_connection.cursor()
db_cursor.execute("SELECT id , name , age FROM Students ORDER BY name")
data =db_cursor.fetchall()
for db in data:
    print("%d %s %d"%(db[0],db[1],db[2]))
```

Output:

103 codelock 20

105 codelock 20

107 codelock 20

102 Vij 20

104 Vij 20

106 Vij 20

Update

ယခု သင်ခန်းစာမှာတော့ database table ထဲမှ data များကို update လုပ်မှာ ဖြစ်ပါတယ်။ ထိုသို့ update ပြုလုပ်နိုင်ရန် UPDATE SET ဆိုသည့် command ကို အသုံးပြုရပါမည်။

UPDATE <table name > SET name="mgmg" WHERE id=102

Update ပြုလုပ်ပြီးလျှင် commit() method ကို ပြန်သုံးပေးရပါမည်။ ထို့နောက် SELECT FROM command ကိုသုံးပြီး မည်သည့် data ကို ယူမည့်အကြောင်းရေးရမည်။ ပြီးလျှင် fetchall() method ကိုသုံးပြီး data အားလုံး ပြန်ကြည့်နိုင်ပါပြီ။

Sample Program (313) - Update and Fetching

```
db_cursor = db_connection.cursor()
db_cursor.execute("UPDATE Students SET name='WinHtut' WHERE id=102")
db_connection.commit()
db_cursor.execute("SELECT id , name , age FROM Students")
data =db_cursor.fetchall()

for db in data:
    print("%d %s %d"%(db[0],db[1],db[2]))
```

Output::

102 WinHtut 20

103 codelock 20
104 Vij 20
105 codelock 20
106 Vij 20
107 codelock 20

Delete

Database table ထဲမှ မိမိဖျက်လိုသည့် data များကိုဖျက်နိုင်ရန် DELETE FROM ဆိုသည့် sql command ကိုသုံးနိုင်ပါသည်။ id အတိအကျ နှင့် ဖျက်လိုပါက WHERE နှင့် အသုံးပြုရပါမည်။

Sample Program (314)

DELETE FROM <table name> WHERE id = 103

```
db_cursor = db_connection.cursor()
db_cursor.execute("DELETE FROM Students WHERE id=102")
db_connection.commit()
db_cursor.execute("SELECT id , name , age FROM Students")
data =db_cursor.fetchall()
for db in data:
    print("%d %s %d"%(db[0],db[1],db[2]))
```

Output:
103 codelock 20
104 Vij 20
105 codelock 20
106 Vij 20
107 codelock 20

Program ကို run ပြီးချိန်၌ id 102 မှ data အချက်လက်အားလုံး မရှိတော့သည်ကို တွေ့ရပါမည်။

Join Table

Join လုပ်တယ်ဆိုတာက database တစ်ခု အောက်မှာရှိတဲ့ table တွေကို မိမိတို့လိုသလို join လုပ်တာပါ ။ ဥပမာ myFirstDB ဆိုတဲ့ ကျွန်တော်တို့ရဲ့ database ထဲမှာ students ဆိုတဲ့ table တစ်ခု ရှိပါတယ်။ ထို table ထဲသို့ အခြား table တစ်ခုမှ လာပေါင်းတာမျိုးကို ဆိုလိုတာပါ။ နောက်ထပ် ပေါင်းနိုင်ဖို့အတွက် table တစ်ခုတည်ဆောက်ပါမည်။ table name ကို cost ဟု နာမည်ပေးလိုက်မည် ဖြစ်ပြီး ထို table ထဲတွင် id ,monthly နှင့် totalCost ဆိုသည် column နှစ်ခုကို တည်ဆောက်ထားပါမည်။

```
dbCursor=db_connection.cursor()
dbCursor.execute("CREATE TABLE cost(id int primary key AUTO_INCREMENT , monthly int , totalCost int)")
```

အထက်ပါအတိုင်း program ရေးပြီး run ပါက myFirstDB ဆိုသည့် database ထဲတွင်

နောက်ထပ် table တစ်ခု ထပ်ပေါ်လာပါမည်။ ထို့နောက် cost table ထဲသို့ data များ ထည့်ပါမည်။

```
dataType = "INSERT INTO cost(id ,totalCost , monthly)values(%s , %s
,%s )"
data = [(103,150,1500),(104,100,1200),(105,200,2000)]
dbCursor.executemany(dataType,data)
db_connection.commit()
```

ဒုတိယ ဆောက်သော table တွင်လည်း id ထည့်ပေးရန် လိုအပ်ပါသည်။ အဘယ်ကြောင့် ဆိုသော် ပထမ table မှ id နှင့် ဒုတိယ table မှ id ကို တို့ကို ချိန်ဆက်ခြင်းဖြင့် table နှစ်ခုအား ပေါင်းမှာ ဖြစ်ပါတယ်။
Cost.id = Students.id ထိုကဲ့သို့ id key များကို သုံးပြီး ပေါင်းပေးမှာ ဖြစ်ပါတယ်။

အထက်ပါ အဆင့်များ ပြီးပါက database ထဲတွင် table တစ်ခု ဆောက်ပြီးသွားမည် ဖြစ်ပြီး ထို table ထဲတွင် data များလည်းထည့်ပြီး ဖြစ်ပါသည်။ ထို့နောက် ပထမရှိထားသော table နှင့် ယခု အသစ်ဆောက်လိုက်သော table အား join ရန် အတွက် SELECT … FROM .. JOIN … ON ဆိုသည့် sql command များကို အသုံးပြုပေးရန် လိုအပ်ပါသည်။

```
SELECT ပထမ table မှ id နှင့် join လိုသော column များ ,ဒုတိယ table မှ id နှင့် join
လိုသော column များ FROM ဒုတိယ table join ပထမtable on ဒုတိယ table
id=ပထမ table id
```

Program ရေးသားပုံမှာ အောက်ပါအတိုင်း ဖြစ်ပါသည်။

```
dbCursor.execute("SELECT Students.id , Students.name ,
Students.age,cost.id,cost.totalCost,cost.monthly FROM cost join
Students on cost.id=Students.id")
```

Table ဆောက်ရန်နှင့် Data ထည့်ရန် အတွက် program

```
dbCursor=db_connection.cursor()
dataType = "INSERT INTO cost(id ,totalCost , monthly)values(%s , %s
,%s )"
data = [(103,150,1500),(104,100,1200),(105,200,2000)]
dbCursor.executemany(dataType,data)
db_connection.commit()
```

နောက်ဆုံး table နှစ်ခုအား join ရန် program

```
dbCursor=db_connection.cursor()
dbCursor.execute("SELECT Students.id , Students.name ,
Students.age,cost.id,cost.totalCost,cost.monthly FROM cost join
Students on cost.id=Students.id")
data =dbCursor.fetchall()
print("id    name   age  CostId monthly totalCost")
for db in data:
        print("%d     %s        %d          %d          %d          %d
```

```
"%(db[0],db[1],db[2],db[3],db[4],db[5]))
```

သတိပြုရန်အချက်မှာ program နှစ်ခုလုံးပေါင်းရေးလျှင် table ကို နှစ်ခါဆောက်သည့် အတွက် error တက်မှာ ဖြစ်ပါတယ်။

MySQL in CLOUD

ကျွန်ုပ်တို့ ရဲ့ mysql server ကို ယခု သင်ခန်းစာတွင် cloud တစ်ခုပေါ်၌ သွားရောက်တင်ထားပါမည်။ ထိုကဲ့သို့ တင်ထားခြင်းဖြင့် ကျွန်ုပ်တို့ database များကို online ပေါ်မှ မည်သည့်နေရာကမဆို မည်သည့် website ကနေမဆို access လုပ်နိုင်မှာ ဖြစ်ပါတယ်။ ထို့ကဲ့သို့ access လုပ်နိုင်ဖို့ကိုလည်း မိမိတို့ကိုယ်တိုင် limitation များ ပြုလုပ်ပေးနိုင်ပါတယ်။

ပထမဆုံးအနေဖြင့် github မှာ အကောင့်တစ်ခုဖွင့်ပေးပါ။ https://github.com/ အကောင့်ဖွင့်ရတာ အရမ်းလွယ်ကူတဲ့အတွက် ဒီသင်ခန်းစာမှာတော့ မဖော်ပြတော့ပါဘူး။ github အကောင့်ဖွင့်ပြီး verified လုပ်ပြီးပြီ ဆိုလျှင် https://www.clever-cloud.com/en/ ယခု ဖော်ပြပါ link သို့သွားပြီး SIGNUP သွားလုပ်ပေးပါ။

ထို့သို့ signup လုပ်ရာတွင် ယခင်က ဖောက်ထားသော github အကောင့်နှင့် ချိတ်ပေးရုံသာ ဖြစ်ပါသည့် အတွက် အသေးစိတ် မဖော်ပြလိုတော့ပါ။

အကောင့် ဖွင့်ပြီးလျှင် login ဝင်ပြီးနောက် အောက်ပါ အတိုင်း page တစ်ခု ပေါ်လာပါမည်။

ပေါ်လာသော page တွင် Personal space ဆိုတာကို တစ်ချက်နှိပ်ပေးလိုက်ပါ ထို့နောက် ဘေးတွင် Create ဆိုသည့် button တစ်ခု ထပ်ပေါ်လာပါမည်။ ထို create button ကို နှိပ်ပြီးလျှင် အောက်ပါအတိုင်း button များ ထပ်ပေါ်လာပါက an add-on ဆိုသည့် button ကို ထပ်နှိပ်ပေးပါ။

ခန နောင့်ပြီးလျှင် ညာဘက် ဘေးခ်ွ အောက်ပါ အတိုင်း database များ ပေါ်လာပါမည်။

ထိုကဲ့သို့ ပေါ်လာသော database များတွင် MySQL ကို SELECT လုပ်ပေးပါ။ ထို့နောက် အောက်ပါ အတိုင်း Pricing များပေါ်လာပါမည်။ ပေါ်လာသော နေရာများတွင် ပထမဆုံး ဖြစ်သည့် DEV ဆိုတာကို select လုပ်ပေးပါ။ DEV က free ဖြစ်ပြီး 10 MB အသုံးပြု နိုင်ပါတယ်။

What kind of MySQL do you need?

	PLAN NAME	BACKUPS	MAX CONNECTION LIMIT	MAX DB SIZE	MEMORY	TYPE
◉	DEV	Daily - 7 Retained	5	10 MB	Shared	Shared
○	XXS Small Space	Daily - 7 retained	15	512 MB	512 MB	Dedicated
○	XXS Medium Space	Daily - 7 retained	15	1 GB	512 MB	Dedicated
○	XXS Big Space	Daily - 7 retained	15	2 GB	512 MB	Dedicated
○	XS Tiny Space	Daily - 7 retained	75	2 GB	1 GB	Dedicated

အထက်ပါ အတိုင်းရွေးပြီးလျှင် အောက်ဆုံးသို့ သွားပြီး NEXT ကို နှိပ်ပေးပါ။

Dedicated	8	454.40 €
Dedicated	8	534.40 €
Dedicated	10	1022.00 €
Dedicated	10	1134.00 €
Dedicated	10	1358.00 €
Dedicated	10	1582.00 €

NEXT

နောက်ထပ် ပေါ်လာသော page တွင် မိမိနှစ်သက်သော name တစ်ခု ကိုပေးပြီး ZONE ကို ကြိုက်ရာ ရွေးနိုင်ပါသည်။ စာရေးသူ အနေဖြင့် Paris ကိုရွေးထားပါသည်။

ထို့နောက် next ကို ထပ်နှိပ်ပေးပါ။ အောက်ပါအတိုင်း connect လုပ်ရန် အချက်လက်များ ပေါ်လာပါမည်။ Host , Database Name , User , Password တို့သည် မိမိတို့ python program လှမ်းချိတ်ရန် လိုအပ်သော အချက်လက်များ ဖြစ်ပါသည်။

မိမိတို့ Python program တွင် အထက်ပါ အချက်လက်များကို ဖြည့်ပြီး ချိတ်ဆက် အသုံးပြု နိုင်ပါပြီ။

Connect To Cloud Database With Python

မိမိ တို့ python program ထဲမှ မူရင်း localhost ရေးထားသော နေရာတွင် အထက်ပါ ပုံထဲမှ host ဆိုတာကို copy ကူးပြီးဖြည့်ပါ ။ user and passwd နှင့် database များကိုလည်းထိုနည်း အတိုင်းပင်ဖြည့်ပါ။

Sample Program (315)

```
import mysql.connector
db_connection = mysql.connector.connect(
host ="bfmzg1fwpuklxzjxbwgl-mysql.services.clever-cloud.com",
user="umpnsxhi4ihv8lai",
passwd="UGbtIOaIGwQwa2py1",
database="bfmzg1fwpuklxzjxbwgl"
)
dbCursor=db_connection.cursor()
dbCursor.execute("SHOW databases")
for db in dbCursor:
    print(db)
Output
('bfmzg1fwpuklxzjxbwgl',)
('information_schema',)
```

အထက်ဖော်ပြပါ အချက်လက်များသည် စာရေးသူရဲ့ အချက်လက်များသာ ဖြစ်ပါသည်။ program run ကြည့်သောအခါ output အနေဖြင့် database name နှင့် information_schema ဆိုတာကို ပြန်လည်ရရှိခဲ့ပါက process များအားလုံးအောင်မြင်ပါပြီ။ ဆက်လက်ပြီး table များကို creat လုပ်ကာ data များ ထည့်နိုင်ပါပြီ။

Creating a Database Table in Cloud

Clever Cloud တွင် table တစ်ခု ဖန်တီးရန်မှာလည်း မိမိတို့ computer တွင် MySQL server အနေဖြင့် အသုံးပြုခဲ့သည့် အတိုင်းပင် ဖြစ်သည်။ အောက်တွင် sample program ဖြင့် ပြန်လည်ဖော်ပြထားသည်။

Sample Program (316)

```
import mysql.connector
db_connection = mysql.connector.connect(
host ="brop5fujxgipc5lj50-mysql.services.clever-cloud.com",
user="u4jfzuhd1wn2byx6",
passwd="FsdAZUgeaHMPgRGudL",
database="brop5fujxvdpc5lj50"
)
dbCursor=db_connection.cursor()
#dbCursor.execute("SHOW databases")
dbCursor.execute("create table Students(id int,name VARCHAR(30),age
SMALLINT,attend TINYINT)")
dbCursor.execute("DESCRIBE Students")
```

```
for tb in dbCursor:
    print(tb)
```

Sample Program (316)ကို run ပြီးလျှင် မိမိတို့ computer တွင်လည်း table တစ်ခု ဖန်တီးပြီးကြောင်း output များကို ကြည့်နိုင်သလို Clever Cloud တွင်လည်း သွားရောက်ကြည့်နိုင်သည်။

ပထမဆုံး အနေဖြင့် Addon dashboard ထဲသို့ ဝင်ပါ။ ပြီးလျှင် PHPMyAdmin ထဲသို့ ဝင်ပါ။ ပြီးလျှင် မိမိတို့ database ကို တစ်ချက်ထောက်ပြီး မိမိတို့ဆောက်ခဲ့သော table ကို တစ်ချက် နှိပ် ပေးလိုက်ပါ။ ထို့နောက်တွင်မူ မိမိတို့ ဆောက်လုပ်လိုက်သော table ပေါ်လာမည်ဖြစ်ပြီး data များကိုလည်း ထည့်နိုင်ပါပြီ။

အထက်တွင် ဖော်ပြထားသော သင်ခန်းစာများသည် MySQL ကို free ရသော cloud တစ်ခုတွင် connect ချိတ်ဆက်ပုံကို ဖော်ပြထားခြင်းသာ ဖြစ်သည်။ စာဖတ်သူများ အနေဖြင့် အမှန်တကယ် အသုံးပြုရန် လိုအပ်ပါက Google Cloud Platform , Amazon AWS , Azure စသဖြင့် မိမိတို့အဆင်ပြေရာကို သုံးနိုင်ပါသည်။ ထို cloud service များသည် credit card သို့မဟုတ် paypal များဖြင့် ချိတ်ဆက်ရသည့် အတွက် စာဖတ်သူများ အနေဖြင့် အခက်အခဲ ရှိနိုင်သောကြောင့် free ရသော clever cloud တွင် ချိတ်ဆက် ပြထားခြင်းသာ ဖြစ်ပါသည်။

MongoDB with Python

MongoDB သည် ယခုနောက်ပိုင်းတွင် ခေတ်စားလာသော Database တစ်ခုဖြစ်သလို MERN or MEAN ဆိုပြီး mongoDB , express , react , node ဆိုပြီး လွန်ခဲ့တဲ့ နှစ်အနည်းငယ်ကတည်းက တစ်ပြိုင်တည်း ခေတ်စားလာခဲ့တာပါ။ MongoDB သည် NoSQL ဟုလည်း သိကြပါတယ်။ ဆိုလိုသည်မှာ MongoDB ကို အသုံးပြုဖို့အတွက် MySQL ကဲ့သို့ SQL command များမလိုအပ်ပါဘူး။ MySQL မှာကဲ့သို့ သူ့မှာ table တွေ မရှိပါဘူး။ Table တွေအစား collection ဆိုတာတွေကို သုံးပါတယ်။ MySQL ကိုတော့ multi-row application တွေမှာ အသုံးများပါတယ်။ ဥပမာ accounting system application တွေမှာပါ။ MongoDB ကိုတော့ Content Management application တွေ Internet of things application တွေ mobile applications တွေမှာ အသုံးပြုပါတယ်။

Data Structure Between MySQL and MongoDB

MySQL မှာတော့ ယခင်သင်ခန်းစာမှာ ဖော်ပြထားတဲ့အတိုင်း data structured ကို schema ပုံစံနဲ့ တည်ဆောက်ထားပါတယ်။ MongoDB မှာတော့ Schema မလိုအပ်ပါဘူး ။ risk ပိုင်းအရ ဆိုရင်တော့ MySQL သည် SQL injection attacks လုပ်ခံရတာ တွေ့ရှိပါတယ် MongoDB မှာတော့ သူ့ရဲ့ design ကြောင့်မို့ attacks လုပ်ခံရတာ နည်းပါတယ်။ Structured ကျပြီးသား data တွေ အတွက်ဆိုရင်တော့ MySQL က ရွေးချယ်စရာကောင်းတဲ့ Database ဖြစ်သလို structured မကျပဲ အမြဲတိုးပွားနေတဲ့ database တွေအတွက်ဆိုရင်တော့ MongoDB က ရွေးချယ်စရာ ဖြစ်မှာပါ။

MongoDB အား Download ဆွဲရန် အောက်ပါ link သို့သွားပါ

https://www.mongodb.com/download-center/community

OS နေရာတွင် မိမိအသုံးပြုသည့် Operating System အားရွေးချယ်ပါ။ မိမိ အသုံးပြုလိုသည့် version နှင့် မိမိ download ဆွဲလိုသည့် package ကိုရွေးချယ်ပြီး download ဆွဲယူနိုင်ပါသည်။ စာရေးသူအနေဖြင့် version 4.2.6 နှင့် MSI, window64 တို့ကိုသာ ရွေးချယ်ထားပါသည်။ download ဆွဲပြီးလျှင် စတင် install လုပ်ပါ။

I accept ကို အမှန်ခြစ်ပေးပြီး next ကို နှိပ်ပါ။ ထို့နောက် complete ကိုရွေးချယ်ပါ ပြီးလျှင် next နောက်ထပ် ပေါ်လာသော နေရာတွင်လည်း next ကိုသာ ရွေးချယ်ပါ ။

ထို့နောက် Next and Next , Install တို့ကို နှိပ်ပေးပြီး လွယ်ကူစွာ install လုပ်နိုင်ပါသည်။ install လုပ်ပြီးသည်နှင့် တစ်ပြိုင်နက် C:\Program Files\MongoDB\Server\4.2\bin ယခု location အတိုင်း အတိအကျသွားပြီး location ကို copy ကူးခဲ့ပါ။

ထို့နောက် computer ထဲသို့ ဝင်ပြီး right click ထောက်ပါ။ ထို့နောက် properties ထဲသို့ သွားပါ

ပြီးလျှင် Advanced System Settings ထဲသို့ သွားပါ ။

Environment Variables ထဲသို့ ဆက်သွားပါ။

New ကို တစ်ချက် နှိပ်ပါ ထို့နောက် Variable name နေရာတွင် PATH ဟုရေးပါ Variable Value နေရာတွင် copy ကူးခဲ့သော link ကို ထည့်ပေးပါ။

ပြီးလျှင် OK ဟု နှိပ်ပေးပြီး ကျန်သော နေရာများတွင်လည်း OK ဟုသာ နှိပ်ပေးပြီးထွက်လိုက်ပါ။ ထို့နောက်တွင်မူ window key+R ကို နှိပ်ပြီး cmd သို့သွားပါ။ ထို့နောက် ပေါ်လာသောနေရာတွင် mongod ဟု ရေးပြီး enter ခေါက်ပေးပါ။

```
:\Users\User>mongod
020-05-27T21:33:14.319+0630 I  CONTROL  [main]
 --sslDisabledProtocols 'none'
020-05-27T21:33:14.321+0630 W  ASIO     [main]
```

ထို့နောက် စာသားများ အများကြီး ပေါ်လာပါက mongodb ကို မိမိတို့ computer မှ သိသွားပါပြီ။

```
ng TLS 1.0, to force-enable TLS 1.0 specify --sslDisa

figured during NetworkInterface startup
ting : pid=17456 port=27017 dbpath=C:\data\db\ 64-bit

Windows 7/Windows Server 2008 R2
```

နောက်တစ်ဆင့် အနေဖြင့် computer ထဲမှ C ထဲသို့ သွားပြီး data ဟူသည့် folder တစ်ခု ဆောက်ပါ ပြီးလျှင် data ဆိုသည့် folder ထဲကို ဝင်၍ db ဆိုသည့် folder တစ်ခု ထပ်ဆောက်ပါ ။

အထက်ပါ အဆင့်များပြီးပါက မိမိ တို့ ဖွင့်ထားသော cmd(command promt) ကိုပိတ်ပြီး window key +r ကို နှိပ်ကာ cmd ဟုရေးပြီး ပြန်ဖွင့်ပါ။ ထို့နောက် ပေါ်လာသော command promt တွင် mongod ဟု ရေးပြီး enter နှိပ်ပါ။

```
C:\WINDOWS\system32\cmd.exe - mongod
C:\Users\User>mongod
2020-05-27T21:36:36.433+0630 I  CONTROL  [main] Automatically disablin
edProtocols 'none'
2020-05-27T21:36:36.769+0630 W  ASIO     [main] No TransportLayer conf
2020-05-27T21:36:36.770+0630 I  CONTROL  [initandlisten] MongoDB start
ost=DESKTOP-T3K2A1L
2020-05-27T21:36:36.771+0630 I  CONTROL  [initandlisten] targetMinOS:
2020-05-27T21:36:36.771+0630 I  CONTROL  [initandlisten] db version v4
2020-05-27T21:36:36.771+0630 I  CONTROL  [initandlisten] git version:
2020-05-27T21:36:36.772+0630 I  CONTROL  [initandlisten] allocator: tc
2020-05-27T21:36:36.772+0630 I  CONTROL  [initandlisten] modules: none
2020-05-27T21:36:36.773+0630 I  CONTROL  [initandlisten] build environ
2020-05-27T21:36:36.773+0630 I  CONTROL  [initandlisten]     distmod:
2020-05-27T21:36:36.774+0630 I  CONTROL  [initandlisten]     distarch:
2020-05-27T21:36:36.774+0630 I  CONTROL  [initandlisten]     target_ar
2020-05-27T21:36:36.775+0630 I  CONTROL  [initandlisten] options: {}
2020-05-27T21:36:36.778+0630 I  STORAGE  [initandlisten] wiredtiger_op
e_max=0M),session_max=33000,eviction=(threads_min=4,threads_max=4),con
hive=true,path=journal,compressor=snappy),file_manager=(close_idle_tim
50),statistics_log=(wait=0),verbose=[recovery_progress,checkpoint_prog
2020-05-27T21:36:36.821+0630 I  STORAGE  [initandlisten] WiredTiger me
```

အထက်ပါ အတိုင်းစာသားများစွာ ပေါ်လာလျှင် Mongodb server စတင်ခြင်း အောင်မြင်ပါပြီ။ ထို mongodb server အား ချိတ်ရန် မိမိတို့ install လုပ်ထားသည့် mongodb application သို့ သွားပါ။ ဖွင့်ထားတဲ့ command promt(cmd) ကို ဖွင့်လက်စအတိုင်းသာ ထားခဲ့ပေးပါ။ ထို့နောက် ပေါ်လာသော နေရာများတွင် next များကိုသာနှိပ်ပြီး အောက်ပါအတိုင်း ပေါ်လာသော နေရာအထိသွားပေးပါ။

ပေါ်လာသော နေရာမှ Hostname တွင် localhost ဟု ရေးပေးပါ။ ထို့နောက် Port နေရာတွင် 27017 ဟု ရေးပြီး connect ကို နှိပ်ပေးပါ။ ထို့နောက် အောက်ပါအတိုင်းပေါ်လာလျှင် connect လုပ်ခြင်း အောင်မြင်ပါပြီ။

Connection MongoDB with Python

Python နှင့် mongodb အား ချိန်ဆက်ပေးရန် pymongo ဆိုသည့် library တစ်ခု လိုအပ်ပါသည်။ ထို library အား install လုပ်ရန် နောက်ထပ် command promt အသစ်အားဖွင့်ပြီး install လုပ်နိုင်သလို မိမိအသုံးပြုနေကြ vs code မှလည်း install လုပ်နိုင်ပါသည်။ စာရေးသူ အနေဖြင့် vs code မှ install လုပ်ထားပါသည်။

```
PS C:\Users\User\Documents\Python> pip install pymongo
Requirement already satisfied: pymongo in c:\users\user\appdata\local\programs\python\python37-32\lib
WARNING: You are using pip version 19.3.1; however, version 20.1.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
PS C:\Users\User\Documents\Python>
```

စာရေးသူအနေဖြင့် install လုပ်ထားပြီး ဖြစ်သောကြောင့် Requirement already satisfied ဟု ပေါ်ခြင်း ဖြစ်ပါသည်။ pymongo မိမိတို့ computer တွင် ရှိသည် မရှိသည်ကို သိနိုင်ရန် အောက်ပါ အတိုင်း python interpreter တွင် import pymongo ဟု ရေးကြည့်ပြီး error မတက်ပါက အဆင်ပြေပါပြီ။

```
You should consider upgrading via the 'python -m pip install --upgrade pip' co
PS C:\Users\User\Documents\Python> python
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 19:29:22) [MSC v.1916 32 bit
Type "help", "copyright", "credits" or "license" for more information.
>>> import pymongo
>>> 
```

ထို့နောက် Python File တစ်ခုဖွင့်ပြီး အောက်ပါတို့ကိုရေးကာ MongoDB နှင့် connect လုပ် နိုင်ပါပြီ။

Sample Program (317)

```
import pymongo
connection = pymongo.MongoClient("localhost",27017)
database = connection["my_db1"]
collection=database["my_col1"]
data={'WinHtut':'CrazyCoder','age':26,'Hobby':'BuildingTools'}
collection.insert_one(data)
```

အထက်ပါအတိုင်း program ရေးပြီး run လိုက်ပါက မိမိတို့ mongodb database server ထဲတွင် data ရောက်နေသည်ကို တွေ့နိုင်ပါသည်။ ပထမဆုံးအနေဖြင့် refresh ကို တစ်ချက် နှိပ်ပေးပါ။ ထို့နောက် my_db1 ကို တစ်ချက်နှိပ်ပါ။ ပြီးလျှင် my_col1 ကို တစ်ချက်ထပ်နှိပ်ပေးပါ။ ညာဘက်ဘေးဌ data များ ပေါ်လာသည်ကို မြင်ရပါမည်။

Program ရှင်းလင်းချက်

```
import pymongo
connection = pymongo.MongoClient("localhost",27017)
```

အထက်ပါ code line သည် mongodb localhost server ကိုချိတ်ဆက်ရန် ဖြစ်သည်။ mongodb local host server သည် default အနေဖြင့် localhost ဖြစ်ပြီး port သည် 27017 ဖြစ်ပါသည်။ ထိုသို့ ချိတ်ဆက်နိုင်ရန် MongoClient ဆိုသည့် Method တစ်ခုကို ခေါ်သုံးရပါသည်။ ထို့နောက် connection ဆိုသည့် object တစ်ခုတည်ဆောက်လိုက်ပါသည်။

```
database = connection["my_db1"]
```

ထိုconnection တွင် my_db1 ဆိုသည့် database တစ်ခု တည်ဆောက်လိုက်ပါသည်။ ထို database ထဲတွင် MySQL ၌ table တည်ဆောက်သော်လည်း

mongodb တွင်မူ collection ဟု သာ သုံးနှုန်းပြီး collection တစ်ခုကို တည်ဆောက်ပါမည်။ ထို့ကြောင့် database ဆိုသည့် object တစ်ခု တည်ဆောက်ပါသည်။ထို database object ကိုသုံးပြီး my_db1 ထဲ၌ collection များစွာ တည်ဆောက်နိုင်ပါသည်။

```
collection=database["my_col1"]
```

အထက်ပါ codeline သည် my_col1 ဆိုသည့် collection တစ်ခုကို my_db1 ဆိုသည့် database ထဲ၌ တည်ဆောက်ခြင်း ဖြစ်ပါသည်။ ထို collection ထဲတွင် မိမိတို့ ထည့်လိုသည့် data များ ထည့်နိုင်ပါသည်။

```
data={'WinHtut':'CrazyCoder','age':26,'Hobby':'BuildingTools'}
collection.insert_one(data)
```

Data ထည့်ရန် ပထမဆုံး အနေဖြင့် data များ စုထားသည့် python object တစ်ခု တည်ဆောက်လိုက်ပါသည်။ ပြီးနောက်မှ ထို data object ကို collection.insertone ဆိုသည့် method ကိုသုံးပြီး database ထဲသို့ လွှမ်းထည့်လိုက်ပါသည်။ MongoDB ကောင်းတဲ့ အချက်များတွင် နောက်တစ်ခေါက် အထက်ပါ code များကို ထပ် run လျှင် error တက်မည် မဟုတ်ပဲ my_db1 ထဲမှ my_col1 ထဲသို့ data များ နောက်တစ်ကြိမ် ထည့်ပေးသွားမှာ ဖြစ်ပါတယ်။

MongoDB Primary Key

Mongodb မှာ document or data တွေကို collection တွေထဲမှာ သီးခြားသတ်မှတ်ထားနိုင်ရန် " _id" ကိုအသုံးပြုပါတယ်။ User ကနေပြီး အသုံးပြုတဲ့အချိန်မှာ တခါတည်း id ကို ထည့်ပေးလိုက်နိုင်သလို အကယ်၍ မထည့်ပေးဘူး ဆိုလျှင်လည်း Mongodb ကနေ id တစ်ခုကို အလိုလျောက် သတ်မှတ်ပေးလိုက်မှာ ဖြစ်ပါတယ်။ အောက်မှာ ပြထားသည့်ပုံသည် ယခင် သင်ခန်းစာတွင် id မထည့်ပေးလိုက်သည့်အတွက် mongodb မှ အလိုလျောက်ထည့်ပေးလိုက်ခြင်း ကို ပြထားပါတယ်။

အကယ်၍ မိမိထည့်ချင်သော Id ဖြင့် ထည့်လိုပါက အောက်ပါအတိုင်း ရေးပြီး ထည့်နိုင်ပါတယ်။ "_id": မိမိထည့်ချင်သော number

```
data={"_id":1,'WinHtut':'CrazyCoder','age':26,'Hobby':'BuildingTools
'}
```

မိမိထည့်ထားသော id များကို ပြန်လိုချင်ပါကလည်း inserted_id ဆိုသည့် property ကို ပြန်သုံးပြီး ကြည့်နိုင်ပါသေးတယ်။ insert_one ဆိုသည့် method ကို အသုံး ပြုသောအခါ return အနေဖြင့် InsertOneResult object ကို return ပြန်ပေးပါတယ်။ ထို object ထဲတွင် inserted_id ဆိုသည့် Property တစ်ခုရှိပါသည်။

```
data={"_id":2,'WinHtut':'CrazyCoder','age':26,'Hobby':'BuildingTools
'}
obj=collection.insert_one(data)
print(obj.inserted_id)
```

အထက်ပါအတိုင်း program run ကြည့်သောအခါတွင် output အနေဖြင့် 2 ဆိုသည့် output ကို ပြန်လည် ရရှိပါမည်။ ထိုနည်းတူ မိမိတို့ database ထဲတွင်လည်း id number 2 ဖြင့် data များ ထပ်ရောက်နေမှာ ဖြစ်ပါတယ်။ သတိပြုရန်အချက်မှာ မိမိတို့ထည့်ပေးလိုက်သော id များသည် ထပ်နေလို့ မရပါဘူး။ အကယ်၍ ထပ်နေခဲ့မည် ဆိုလျှင် dup key error တက်မှာ ဖြစ်ပါတယ်။

```
File "C:\Users\winht\AppData\Local\Programs\Python\Python38-32\lib\site-packages\pymongo\collection.py", line 597, in _i
  _check_write_command_response(result)
File "C:\Users\winht\AppData\Local\Programs\Python\Python38-32\lib\site-packages\pymongo\helpers.py", line 221, in _chec
  _raise_last_write_error(write_errors)
File "C:\Users\winht\AppData\Local\Programs\Python\Python38-32\lib\site-packages\pymongo\helpers.py", line 202, in _rais
  raise DuplicateKeyError(error.get("errmsg"), 11000, error)
mongo.errors.DuplicateKeyError: E11000 duplicate key error collection: my_db1.my_col1 index: _id_ dup key: { _id: 2 }
```

Distinct

Collection ထဲမှ data တွေရဲ့ id တွေ အားလုံးရလိုသောအခါတွင် Distinct method ကို သုံး နိုင်ပါတယ်။ distinct method ထဲမှာ id ကိုတော့ ထည့်ပေးရပါမည်။

Sample Program (318)

```
import pymongo
connection = pymongo.MongoClient("localhost",27017)
database = connection["my_db1"]
collection=database["my_col1"]
_ids = collection.find().distinct('_id')
print(_ids)
```

```
Output:
[1, 2, ObjectId('5ede0b1a8cfd768acfa68080')]
```

Sample Program (318)သည် my_col1 ဆိုသည့် collection ထဲမှ id များအားလုံးကို ရယူရန် အတွက် ဖြစ်ပါတယ်။

insert_many()

Python မှာ database ထဲသို့ data များ တစ်ချိန်တည်းမှာ အများကြီးထည့်နိုင်ရန်

insert_many ကို အသုံးပြုပါတယ်။ ယခု program တွင် my_db1 ဆိုသည့် database ထဲ၌ my_col2 ဆိုသည့် collection အသစ်တစ်ခု တည်ဆောက်ပါမည်။ ထို့နောက် data များစွာကို စုပြီးထည့် ထားနိုင်ရန် list တစ်ခု တည်ဆောက်ထားပါမည်။ ထို့နောက် data များထည့်ရန် try ကို သုံးမည် ဖြစ်ပြီး အကယ်၍ error တစ်စုံတစ်ရာ ဖြစ်ပါက ပြန်ပြပေးနိုင်ရန် Exception ကိုလည်း ထည့်ထားပေးပါမည်။

Sample Program (319)

```
connection = pymongo.MongoClient("localhost",27017)
database = connection["my_db1"]
collection=database["my_col2"]
data=[
{"_id":1,'WinHtut':'CrazyCoder','age':26,'Hobby':'BuildingTools'},
{"_id":2,'Win':'CrazyCoder','age':26,'Hobby':'BuildingTools'},
{"_id":3,'Htut':'CrazyCoder','age':26,'Hobby':'BuildingTools'},
{"_id":4,'WinHtut1':'CrazyCoder','age':26,'Hobby':'BuildingTools'},
{"_id":5,'WinHtut2':'CrazyCoder','age':26,'Hobby':'BuildingTools'}
]
try:
    obj = collection.insert_many(data)
    print("Data are successfully inserted")
    print(obj.inserted_ids)
except Exception as error:
    print(error)
```

Output:

```
Data are successfully inserted
[1, 2, 3, 4, 5]
```

MongoDB Compass ထဲတွင်လည်း အောက်ပါပုံထဲကအတိုင်း မြင်ရမည်ဖြစ်သလို table ပုံစံဖြင့် ကြည့်လိုပါကလည်း မြှားပြထားသည့် table ကိုနှိပ်ပြီး ကြည့်နိုင်ပါသည်။

Fetching

Collection ထဲမှ data များကို ပြန်လည်ရယူနိုင်ဖို့အတွက် find_one() method ကို သုံးနိုင်ပါတယ်။

Sample Program (320)

```
database = connection["my_db1"]
collection=database["my_col2"]

try:
    data=collection.find_one()
    print(data)
except Exception as error:
    print(error)
```

Find_one() method ကိုသုံးမည်ဆိုလျှင် return အနေဖြင့် collection ထဲမှ ပထမဆုံး row မှာ ရှိသည့် data များအားလုံးကိုပြန်ပေးပါတယ်။ ထိုသို့ သုံးမည်ဆိုလျှင် အထက်ပါ program ထဲမှ အတိုင်း collection.find_one() ကို သုံးရမည် ဖြစ်သည်။ အကယ်၍ collection ထဲတွင်ရှိသော data များ အားလုံးကို ရယူလိုပါက find method ကို အောက်ပါအတိုင်း အသုံးပြုနိုင်ပါတယ်။

MySQL: SELECT * FROM collection (MySQL မှာ ယခု အတိုင်းဖြစ်သည်)

```
try:
    for i in collection.find():
        print(i)
```

```
except Exception as error:
    print(error)
```

အထက်ပါ program ကို run ပြီးချိန်၌ collection ထဲမှ data များအားလုံး output အနေဖြင့် ပြန်ရမှာပါ။

```
de/test.py
{'_id': 1, 'WinHtut': 'CrazyCoder', 'age': 26, 'Hobby': 'BuildingTools'}
{'_id': 2, 'Win': 'CrazyCoder', 'age': 26, 'Hobby': 'BuildingTools'}
{'_id': 3, 'Htut': 'CrazyCoder', 'age': 26, 'Hobby': 'BuildingTools'}
{'_id': 4, 'WinHtut1': 'CrazyCoder', 'age': 26, 'Hobby': 'BuildingTools'}
{'_id': 5, 'WinHtut2': 'CrazyCoder', 'age': 26, 'Hobby': 'BuildingTools'}
PS C:\Research>
```

Properties Zero

Collection ထဲမှ မိမိထုတ်ချင်သော data များကိုသာ သီးသန့် ဆွဲထုတ်နိုင်ရန် properties ကို 0 or 1 ပုံစံဖြင့် ရေးနိုင်ပါသည်။ ဥပမာ ကျွန်တော်တို့ my_col2 ဆိုသည့် collection ထဲမှ age မှ လွဲပြီး အခြားသော data များအားလုံးကို ဆွဲထုတ်လိုပါက “age”:0 ဟုရေးပြီး ချန်ထားခဲ့နိုင်ပါသည်။

Syntax:

collection.find({} , {“age”:0})

Find method ထဲမှ ပထမ parameter နေရာတွင် parentheses {} က လွဲပြီး ဘာမှ မထည့်ရပဲ ဒုတိယ parameter နေရာတွင် မိမိတို့လိုသလို 1 or 0 ဖြင့် ရေးနိုင်သည်။ 1 ဟုရေးလျှင် ထို data တစ်ခုတည်းကိုသာ id ဖြင့် ဖော်ပြပေးမည် ဖြစ်ပြီး 0 ဟုရေးလျှင် ထို data ကို လုံးဝဖော်ပြ မည် မဟုတ်ပါ။

```
try:
    for i in collection.find({},{"age":0}):
        print(i)
except Exception as error:
    print(error)
```

Output:

```
de/test.py
{'_id': 1, 'WinHtut': 'CrazyCoder', 'Hobby': 'BuildingTools'}
{'_id': 2, 'Win': 'CrazyCoder', 'Hobby': 'BuildingTools'}
{'_id': 3, 'Htut': 'CrazyCoder', 'Hobby': 'BuildingTools'}
{'_id': 4, 'WinHtut1': 'CrazyCoder', 'Hobby': 'BuildingTools'}
{'_id': 5, 'WinHtut2': 'CrazyCoder', 'Hobby': 'BuildingTools'}
PS C:\Research>
```

အထက်ပါ output ကိုကြည့်လျှင် age မပါဝင်တော့သည်ကို မြင်ရပါမည်။

Properties One

မိမိ တို့ collection ထဲမှ Hobby တစ်ခုတည်းကိုသာ လိုချင်ပါက အောက်ပါအတိုင်း ရေးနိုင်ပါသည်။

Sample Program (321)

```
try:
    for i in collection.find({},{"Hobby":1}):
```

```
        print(i)
except Exception as error:
    print(error)
```

Output :

```
de/test.py
{'_id': 1, 'Hobby': 'BuildingTools'}
{'_id': 2, 'Hobby': 'BuildingTools'}
{'_id': 3, 'Hobby': 'BuildingTools'}
{'_id': 4, 'Hobby': 'BuildingTools'}
{'_id': 5, 'Hobby': 'BuildingTools'}
PS C:\Research>
```

အကယ်၍ id ကို မပါစေလိုပဲ age နှင့် hobby ကိုပဲ ဖော်ပြစေချင်ပါက Id ကို 0 ပေးခဲ့ပြီး age နှင့် hobby တို့ကို 1 ပေးကာ အောက်ပါအတိုင်းလည်း ရေးသားနိုင်ပါသည်။

Sample Program (322)

```
try:
    for i in collection.find({},{"_id":0,"age":1,"Hobby":1}):
        print(i)
except Exception as error:
    print(error)
```

Output:

```
de/test.py
{'age': 26, 'Hobby': 'BuildingTools'}
{'age': 26, 'Hobby': 'BuildingTools'}
{'age': 26, 'Hobby': 'BuildingTools'}
{'age': 26, 'Hobby': 'BuildingTools'}
{'age': 26, 'Hobby': 'BuildingTools'}
PS C:\Research>
```

MongoDB Query

Key and value ကို ထည့်ပြီးတော့လည်း မိမိရှာချင်တဲ့ သီးခြား data တွေကိုလည်း ရှာနိုင်ပါ သေးတယ်။ ထိုသို့ ရှာနိုင်ရန် မိမိတို့ ရှာလိုသည့် key and value ကို ရေးပါ။ ပြီးလျှင် object တစ်ခု တည်ဆောက်ပါ။

```
query = {"WinHtut":"CrazyCoder"}
```

Sample Program (323)

```
query = {"WinHtut":"CrazyCoder"}
try:
    data=collection.find(query)
    for d in  data:
        print(d)
except Exception as error:
    print(error)
```

Output:

```
de/test.py
{'_id': 1, 'WinHtut': 'CrazyCoder', 'age': 26, 'Hobby': 'BuildingTools'}
PS C:\Research>
```

MongoDB Query With Operator

Find method ထဲတွင် operator များဖြင့်လည်း တွဲသုံးနိုင်ပါသေးတယ်။ ထိုသို့ တွဲသုံးနိုင်ရန် collection ထဲမှ မည်သည့် field ကိုရှာဖွေမည် ဖြစ်ကြောင်းကို ပထမဦးဆုံး ရေးပေးရမည်။ ထို့နောက် မိမိရှာလိုသော field ထဲမှ data ကို ထည့်ပေးရမည်။

field data

```
query = {"_id": {"$lt":3} }
```

operator

ပထမ နေရာတွင် id ကို ရှာလိုသောကြောင့် _id ကို ထည့်ပေးပါသည်။ ဒုတိယနေရာတွင် operator နှင့် မိမိတို့ သိလိုသော data ကို parentheses ထဲတွင် ထည့်ပေးရပါမည်။

Sample Program (324)

```
query = {"_id": {"$lt":3} }
try:
    data=collection.find(query)
    for d in  data:
        print(d)
except Exception as error:
    print(error)
```

Sample Program (324)အား runပြီးချိန်တွင် operator သည် (lt) less than ဖြစ်သောကြောင့် id 3 ထက် ငယ်သည့် 2 and 1 တွင် ရှိသော data များ အားလုံးကို ထုတ်ပေးမှာ ဖြစ်ပါတယ်။ ထို့ကြောင့် output အနေဖြင့် အောက်ပါအတိုင်း တွေ့ရပါမည်။

```
{'_id': 1, 'WinHtut': 'CrazyCoder', 'age': 26, 'Hobby': 'BuildingTools'}
{'_id': 2, 'Win': 'CrazyCoder', 'age': 26, 'Hobby': 'BuildingTools'}
PS C:\Research>
```

အကယ်၍ 3 ထက်ကြီးသည့် data များအားလုံးကို ထုတ်ယူလိုပါက gt ကိုသုံးနိုင်ပြီး အောက်ပါ အတိုင်းရေးနိုင်ပါသည်။

```
query = {"_id": {"$gt":3} }
```

Output:

```
{'_id': 4, 'WinHtut1': 'CrazyCoder', 'age': 26, 'Hobby': 'BuildingTools'}
{'_id': 5, 'WinHtut2': 'CrazyCoder', 'age': 26, 'Hobby': 'BuildingTools'}
```

Number တွေတင် မဟုတ်ပဲ character များကိုပါ သုံးပြီးတော့လည်း ရှာနိုင်ပါသေးသည်။ ဥပမာ name ကို ရှာချင်သည် ဆိုပါစို့ အောက်ပါ အတိုင်း ရေးရမည်။

```
query = {"name": {"$gt":"W"} }
```

နောက်တွင် Capital W ကို ရေးထားသည် ထို့ကြောင့် W ထက်ကြီးသည့် အစ စာလုံး

အားလုံးကို ရှာ ထုတ် ပေးမှာဖြစ်ပါတယ်။ Character အားလုံးတွင် ascii value များ ရှိကြသည်။ ဥပမာ a to z (အသေးသည်) 97 မှ 122 ထိ ဖြစ်ပြီး A to Z (အကြီးသည်) 65 မှ 90 ထိ ဖြစ်သည်။ ထို့ကြောင့် W value သည် 87 ဖြစ်ပြီး 87 နှင့် 87 ထက်ကြီးသော value များ အားလုံးကို ဖော်ပြ ပေးမှာ ဖြစ်ပါတယ်။ အောက်ပါပုံထဲမှအတိုင်း ရှိပြီးသား data များထဲမှ W ထက်ကြီးသော data များကို ရှာဖွေရန် အောက်ပါအတိုင်း ရေးနိုင်ပါသည်။

Sample Program (325)

```
query = {"name": {"$gt":"W"} }
try:
    data=collection.find(query)
    for d in  data:
        print(d)
except Exception as error:
    print(error)
```

Output:

```
de/test.py
{'_id': 1, 'name': 'Win', 'age': 26, 'Hobby': 'BuildingTools'}
{'_id': 5, 'name': 'Xexe', 'age': 26, 'Hobby': 'BuildingTools'}
PS C:\Research>
```

```
_id: 1
name: "Win"
age: 26
Hobby: "BuildingTools"
```

```
_id: 2
name: "Aung"
age: 26
Hobby: "BuildingTools"
```

```
_id: 3
name: "Naung"
age: 26
Hobby: "BuildingTools"
```

```
_id: 4
name: "Kaung"
age: 26
Hobby: "BuildingTools"
```

```
_id: 5
name: "Xexe"
age: 26
Hobby: "BuildingTools"
```

Collection ထဲမှ data များကို manual အားဖြင့်လည်း ပြင်နိုင်ပါသည်။ မိမိတို့ ပြုပြင်လိုသော Document ကို တစ်ချက်နှိပ်လိုက်ပါက အောက်ပါအတိုင်း edit လုပ်နိုင်ရန် Option များ ပေါ်လာပါမည်။

ထို့သို့ edit ကို နှိပ်လိုက်ပါ။ အောက်ပါအတိုင်း ပေါ်လာမည် ဖြစ်ပြီး cursor ချကာ key ရော value များပါ အားလုံးကို မိမိတို့ စိတ်ကြိုက် ပြုပြင်နိုင်ပါသည်။

မိမိတို့ လိုအပ်သည် အတိုင်း ပြုပြင်ပြီးလျှင် UPDATE ဆိုသည့် button ကို နှိပ်ပြီး save နိုင်ပါသည်။

MongoDB REGEX

Regex ဆိုတာ regular expression ကို ဆိုလိုတာပါ။ သူ့ကို စစ်ဆေးလိုတဲ့ နေရာတွေမှာသုံးပါတယ်။ mongodb မှာ regular expression ကို အသုံးပြုနိုင်ပါတယ်။ regex ကို အသုံးပြုရမည့် keyword ကတော့ regex ပါပဲ။ ထို keyword နောက်မှ မိမိတို့စစ်ဆေးလိုသော character ကိုထည့်ပြီး စစ်ဆေးနိုင်ပါတယ်။

```
query = {"name": {"$regex":"^W"} }
```

အထက်ပါအတိုင်း မိမိ query object ထဲတွင် ထည့်ရေးကြည့်မည်ဆိုလျှင် output အနေဖြင့် W ပါသော data ကိုသာ ဖော်ပြပေးမှာပါ။

```
de/test.py
{'_id': 1, 'name': 'Win', 'age': 26, 'Hobby': 'BuildingTools'}
```

Ascending and Descending Data

Collection ထဲမှာ ရှိတဲ့ data တွေကို Alphabetically အရ ငယ်စဉ်ကြီးလိုက် စီနိုင်သလို ကြီးစဉ်ငယ်လိုက်လည်း စီနိုင်ပါတယ်။ ထိုသို့ စီနိုင်ရန် sort ဆိုသည့် method ကို အသုံးပြုနိုင်ပါတယ်။ ထို sort method ထဲတွင် parameter အနေဖြင့် မိမိတို့ စီလိုသော field or key name ကို ထည့်ပေးရပါတယ်။

Sample Program (326)

```
try:
    data=collection.find().sort("name")
    for d in  data:
        print(d)
except Exception as error:
    print(error)
```

Output:

```
{'_id': 2, 'name': 'Aung', 'age': 26, 'Hobby': 'BuildingTools'}
{'_id': 4, 'name': 'Kaung', 'age': 26, 'Hobby': 'BuildingTools'}
{'_id': 3, 'name': 'Naung', 'age': 26, 'Hobby': 'BuildingTools'}
{'_id': 1, 'name': 'Win', 'age': 26, 'Hobby': 'BuildingTools'}
{'_id': 5, 'name': 'Xexe', 'age': 26, 'Hobby': 'BuildingTools'}
```

အထက်တွင် ရေးပြထားသော program သည် Ascending ကိုရေးပြထားခြင်း ဖြစ်ပြီး Descending လုပ်နိုင်ရန် sort method နောက်တွင် ဒုတိယ parameter အနေဖြင့် -1 ကို ထည့်ပေးရပါမည်။

```
data=collection.find().sort("name",-1)
```

Output:

```
{'_id': 5, 'name': 'Xexe', 'age': 26, 'Hobby': 'BuildingTools'}
{'_id': 1, 'name': 'Win', 'age': 26, 'Hobby': 'BuildingTools'}
{'_id': 3, 'name': 'Naung', 'age': 26, 'Hobby': 'BuildingTools'}
{'_id': 4, 'name': 'Kaung', 'age': 26, 'Hobby': 'BuildingTools'}
{'_id': 2, 'name': 'Aung', 'age': 26, 'Hobby': 'BuildingTools'}
```

Delete

Collection ထဲမှ data တွေကို ဖျက်နိုင်ဖို့ delete method ကို သုံးနိုင်ပါတယ်။ delete method မှာမှ delete_one() နှင့် delete_many() ဟု နှစ်မျိုးရှိပါတယ်။ အရင် သင်ခန်းစာများ အတိုင်းပင် မိမိတို့ ထည့်လိုသော (သို့မဟုတ်) စစ်ဆေးလိုသော (သို့မဟုတ်) ဖျက်ထုတ်လိုသော condition တစ်ခုကို Object တစ်ခုဆောက်ပြီး ထို object ကို delete_one or delete_many() ထဲသို့ ထည့်ပြီး အသုံးပြုနိုင်ပါတယ်။ ထို့ပြင် မိမိတို့ documents ဘယ်လောက်များများကို delete လုပ်လိုက်သလဲ ဆိုတာ သိလိုလျှင် deleted_count ကိုပါ သုံးနိုင်ပါသေးတယ်။

Sample Program (327)

```
myquery = {"name":"Win"}
try:
    data=collection.delete_one(myquery)
```

```
    print("Deleted Documents:",data.deleted_count)
except Exception as error:
    print(error)
```

အကယ်၍ မိမိတို့သတ်မှတ်ပေးထားတဲ့ condition documents မရှိဘူးဆိုလျှင်လည်း error တက်မည် မဟုတ်ပါ။ အကယ်၍ documents တွေအားလုံးကို ဖျက်ချင်ရင်တော့ delete_many method နောက်တွင် parentheses ကလွဲပြီး မည်သည့် parameters မျှမထည့်ပဲ ရေးနိုင်ပါသည်။

```
data=collection.delete_many({})
```

Sample Program (328)

```
try:
    data=collection.delete_many({})
    print("Deleted Documents:",data.deleted_count)
except Exception as error:
    print(error)
```

Output:

```
de/test.py
Deleted Documents: 4
PS C:\Research>
```

Sample Program (328) ကို run ပြီးချိန်တွင် မိမိတို့ collection ထဲ၌ မည်သည့် documents မျှ ရှိတော့မည် မဟုတ်ပါ။ ထို့ကြောင့် နောက် program များကို စမ်းရန်အတွက် documents များ ပြန်ထည့်ထားရန် လိုအပ်ပါသည်။ စာရေးသူအနေဖြင့် အောက်ပါ program အတိုင်း Data များ ပြန်ထည့်ထားပါသည်။

```
data=[
{"_id":1,'name':'Win','age':26,'Hobby':'BuildingTools'},
{"_id":2,'name':'Htut','age':30,'Hobby':'Crackers'},
{"_id":3,'name':'Xerox','age':40,'Hobby':'Gamers'},
{"_id":4,'name':'Zeero','age':50,'Hobby':'Programmers'},
{"_id":5,'name':'Anon','age':80,'Hobby':'Hackers'}
]
try:
    obj = collection.insert_many(data)
    print("Data are successfully inserted")
    print(obj.inserted_ids)
except Exception as error:
    print(error)
```

Delete_many() ဖြင့် delete လုပ်ရာမှာ regular expression ကိုသုံးပြီးတော့လည်း delete လုပ်နိုင်ပါသေးတယ်။ အသုံးပြုပုံမှာ mongoDB regex သင်ခန်းစာတိုင်းပင် ဖြစ်သည်။

```
query = {"name": {"$regex":"^W"} }
```

Sample Program (324)

```
query = {"name": {"$regex":"^W"} }
try:
    data=collection.delete_many(query)
```

```
    print("Deleted Documents",data.deleted_count)
except Exception as error:
    print(error)
```

အထက်ပါ program ကို run ပြီးချိန်မှာတော့ database ထဲမှ w နှင့် စသည့် data အားလုံးကို ဖျက်ပစ်မှာ ဖြစ်ပါတယ်။

Drop collection

Collection တစ်ခုလုံးကို ဖျက်ချလိုတဲ့အခါမျိုးမှာ ဆိုရင်တော့ drop ကို သုံးနိုင်ပါတယ်။

Sample Program (330)

```
try:
    collection.drop()
    print("Collection {} is droped".format(collection))
except Exception as error:
    print(error)
```

Sample Program (330) ကို run ပြီးချိန်တွင် မိမိတို့ ယခင်က တည်ဆောက်ခဲ့သော mycol2 ဆိုသည့် collection တစ်ခုလုံးကို ဖျက်လိုက်မှာ ဖြစ်ပါတယ်။

MongoDB CSV file

MongoDB ထဲသို့ CSV file တစ်ခုထည့်ပြီးထို file အား မိမိတို့ CRUD လုပ်ရန် လေ့ကျင့်နိုင်ပါသည်။

https://www.stats.govt.nz/large-datasets/csv-files-for-download/

Or

https://u.pcloud.link/publink/show?code=XZuMWEkZ3BlK2VyTkLXXeof0fj4Pq72Hon2y

အထက်ပါ link တစ်ခုခုမှ download ဆွဲပါ။ ထို့နောက် MongoDB Compass ကို ဖွင့်ပါ ။ ထို့နောက် မိမိတို့ ယခင်က တည်ဆောက်ခဲ့သည့် my_db1 ကို တစ်ချက်နှိပ်ပါ (Or) မရှိလျှင်လည်း မိမိတို့ဘာသာ တည်ဆောက်နိုင်သည်။ ထို့နောက် အပေါင်း ပုံစံလေးကိုနှိပ်ပြီး businessInfo ဆိုသည့် collection တစ်ခုကို တည်ဆောက်ပါမည်။

ထို့နောက် အောက်ပါ ပုံ အတိုင်းပေါ်လာမည် ဖြစ်ပြီး businessInfo ဆိုသည့် စာတန်းကို တစ်ချက် နှိပ်လိုက်ပါ။ ပြီးလျှင် Import Data ကို တစ်ချက်နှိပ်ပါ။

Import Data ကို နှိပ်ပြီးလျှင် CSV file ကို တစ်ချက် နှိပ်ပါ ထို့နောက် browse ကို နှိပ်ပြီး မိမိတို့ download ဆွဲထားသည့် နေရာသို့သွားပါ။

အထက်ပါ ပုံထဲမှ အတိုင်း ဘာမှပေါ်မလာလျှင် JSON ဆိုတာ တစ်ချက် နှိပ်ပြီး All files ကိုရွေးပေးလိုက်ပါ။ ထို့နောက် ပေါ်လာသော csv file ကို နှိပ်ပြီး Open ကို နှိပ်ပေးလိုက်ပါ။

အောက်ပါအတိုင်း ပေါ်လာခဲ့လျှင် IMPORT ကိုတစ်ချက် နှိပ်ပေးပါ။

ထို့နောက် Done ကို နှိပ်ပြီးလျှင် မိမိတို့ businessInfo ဆိုသည့် collection ထဲ၌ အောက်ပါ အတိုင်း dataများ အားလုံးရောက်နေ သည်ကို မြင်ရပါမည်။

```
_id: ObjectId("5edfaed3e586823914f8af64")
Series reference: "2019.09"
Period: "PPIQ.SQN800000"
Description: "All industries excl admin health educati, Inputs Indexes (Level 0)"
Revised: "1167"
Initially published: "1164"

_id: ObjectId("5edfaed3e586823914f8af65")
Series reference: "2019.12"
Period: "PPIQ.SQN800000"
Description: "All industries excl admin health educati, Inputs Indexes (Level 0)"
Revised: "1170"
Initially published: "1166"

_id: ObjectId("5edfaed3e586823914f8af66")
Series reference: "2019.09"
Period: "PPIQ.SQN800001"
Description: "All industries ex admin health edu OOD, Inputs Indexes (Level 0)"
Revised: "1167"
Initially published: "1164"

_id: ObjectId("5edfaed3e586823914f8af67")
Series reference: "2019.12"
Period: "PPIQ.SQN800001"
Description: "All industries ex admin health edu OOD, Inputs Indexes (Level 0)"
Revised: "1170"
Initially published: "1166"
```


Update

Mongodb ထဲသို့ documents များ updateလုပ်လိုသော အခါတွင် method နှစ်ခုကို အသုံး ပြုနိုင်ပါတယ်။ တစ်ခုတည်းကို update လုပ်လိုသောအခါတွင် update_one() ဆိုသည့် method ကို သုံးနိုင်သလို တစ်ချိန်တည်းမှာ အများကြီးကို Update လုပ်လိုသော အခါတွင် update_many() method ကိုသုံးနိုင်ပါတယ်။ Tables ပုံစံဖြင့် ကြည့်နိုင်ရန်အပေါ် ဆုံးက မြားလေးကို တစ်ချက်နှိပ်လိုက်ပါ။ ထို့နောက် Series reference ရဲ့ ပထမဆုံး document ဖြစ်တဲ့ 2019.09 နေရာကို 2020 . 10 ဟု ချိန်းပါမည်။

Update လုပ်ရန် update_one method ကို သုံးပါမည်။ update_one method သည် parameter နှစ်ခု ယူပါသည် ပထမ parameter သည် မိမိ update လုပ်လိုသော field name နှင့် value ဖြစ်ပြီး ဒုတိယ Parameter သည် document အသစ်နှင့် operator ဖြစ်ပါသည်။ Operator နေရာတွင် $set ကို အသုံးပြုရပါမည်။

Sample Program (331)

```
myquery = {"Series reference":"2019.09"}
newDoc = {"$set":{"Series reference":"2020.10"}}
try:
    collection.update_one(myquery,newDoc)
    print("Success...")

except Exception as error:
    print(error)
```

Sample Program (331)ကို run ပြီးချိန်တွင် mongodb ထဲ၌ ပထမဆုံး နေရာတွင် 2020.10 ဟု ပြောင်းလဲ သွားသည်ကို မြင်ရပါမည်။

Update Many

မိမိလိုချင်သော condition များကို သတ်မှတ်ပြီး တစ်ပြိုင်တည်းမှာပင် documents အားလုံးကို Update လုပ်နိုင်ပါတယ်။ထိုသို့ ပြုလုပ်နိုင်ရန် mongodb regex ကိုလည်းသုံးနိုင်ပါတယ်။ ယခင် သင်ခန်းစာ database ထဲမှ Description field တွင်ရှိသော ALL နဲ့ စသည့် document အားလုံးကို Updating...ဆိုသည့် new document ဖြင့် Update လုပ်ပါမည်။ ထိုသို့ Update ပြုလုပ်ရန် ပထမဆုံး myquery ဆိုသည့် object တစ်ခုကို တည်ဆောက်မည် ဖြစ်ပြီး ထို object ထဲသို့ မိမိတို့ Update လုပ်ချင်သည့် field နှင့် ထို field ထဲမှ ပြင်ချင်သည့် document အစ
စာလုံး များကို ရေးပေးပါမည်။ ထို့နောက် document များနေရာတွင် အသစ် အစားထိုးလိုသည့် data များကို newDoc ဆိုသည့် object တစ်ခု တည်ဆောက်ပါမည်။ ထို Object ထဲတွင် set ဆိုသည့် Operator ကိုသုံးပြီး data အသစ် ထည့်ပါမည်။

Sample Program (332)

```
myquery = {"Description":{"$regex":"All"}}
newDoc = {"$set":{"Description":"Updating..."}}
try:
    collection.update_many(myquery,newDoc)
    print("Success...")

except Exception as error:
    print(error)
```

အောက်ပါပုံသည် Program မ run ခင် ပုံ ဖြစ်ပါသည်။

	Description String	Revised String
	"All industries excl admin heal	"1167"
	"All industries excl admin heal	"1170"
	"All industries ex admin health	"1167"
	"All industries ex admin health	"1170"
	"All industries, Inputs Indexes	"1166"
	"All industries, Inputs Indexes	"1169"
	"All industries excl OOD, Input	"1166"
	"All industries excl OOD, Input	"1169"
	"Agriculture, Forestry and Fish	"1207"
	"Forestry and Logging, Inputs I	"1217"

အောက်ပါပုံသည် Program run ပြီးပုံ ဖြစ်ပါတယ်။ data တွေအားလုံး update ဖြစ်သွားသည်ကို မြင်ရမှာပါ။

Period String	Description String
"PPIQ.SQN800000"	"Updating..."
"PPIQ.SQN800000"	"Updating..."
"PPIQ.SQN800001"	"Updating..."
"PPIQ.SQN800001"	"Updating..."
"PPIQ.SQN900000"	"Updating..."
"PPIQ.SQN900000"	"Updating..."
"PPIQ.SQN900001"	"Updating..."
"PPIQ.SQN900001"	"Updating..."
"PPIQ.SQNAA0000"	"Agriculture, Forestry and Fish

limit()

Database ထဲမှ documents တွေကို မိမိလိုချင်သလောက်သာ ရယူလိုသောအခါတွင် limit method ကို သုံးပါတယ်။ limit method သည် parameter တစ်ခုသာ ရယူပြီး ထို parameter number အတိုင်း return ပြန် ပေးပါသည်။

Sample Program (333)

```
try:
    doc = collection.find().limit(10)
    for d in doc:
        print(d)

except Exception as error:
    print(error)
```

Sample Program (333)ကို run ပြီးချိန်တွင် output အနေဖြင့် database ထဲမှ documents 10 ခုကို မြင်ရမှာ ဖြစ်ပါတယ်။

Atlas Cloud

Mongo Atlas Cloud သည် Visa, paypal စသည့် မည်ိသို့သော ငွေ ပေးချေမှုမျိုးမှ လုပ်စရာ မလိုပဲ gmail တစ်ခုဖြင့် account ဖောက်ရုံဖြင့် 512MB အသုံးပြုနိုင်ပါသည်။ ပထမဆုံးအနေဖြင့် Atlas Cloud တွင် account ဖောက်ရန် https://www.mongodb.com/cloud/atlas ယခု ဖော်ပြပါ link သို့ သွားပြီး sign up လုပ်ပြီး အသုံးပြု နိုင်ပါသည်။

Signup ကို နှိပ်ပြီး မိမိတို့ gmail ကို add ပေးပါ။ ထို့နောက် အောက်ပါ အတိုင်းပေါ်လာသော page တွင် I accept ကို တစ်ချက်နှိပ်ပေးပြီး Submit ကို နှိပ်ပေးပါ။

မိမိတို့ အနေဖြင့် Free သုံးမှာ ဖြစ်သောကြောင့် Create a Cluster ကို နှိပ်ပေးပါ။

ထို့နောက် အောက်ပါအတိုင်း ပေါ်လာသော page တွင်လည်း Create a Cluster ကိုသာ ဆက်နှိပ်ပေးပါ။

ထို့နောက်တွင်မူ အောက်ပါအတိုင်း page တစ်ခု ပေါ်လာပါမည်။

ပေါ်လာသော page မှ Collection ကို နှိပ်ပါ ။ နောက်ထပ်ပေါ်လာသော page တွင် Add My Own Data ဆိုတာကို ဆက်နှိပ်ပေးပါ။

နောက်ထပ် ပေါ်လာသော page တွင် database name and collection တို့ကို ထည့်ပေးပါ။ စာရေးသူအနေဖြင့် myDB နှင့် myCollection ဟုသာ ပေးလိုက်ပါသည်။

အထက်ပါပုံ အတိုင်းပေါ်လာလျှင် Database နှင့် ထို Database ရဲ့ collection တို့ တည်ဆောက် ပြီးပါပြီ။ ထို့နောက်တွင်မူ အောက်ပါပုံထဲမှအတိုင်း Database Access ထဲသို့ သွားပါ။

ပြီးလျှင် add new database user ကို နှိပ်ပါ။

စာရေးသူ အနေဖြင့် root and toor ကိုသာ အသုံးပြုထားပါသည် ထို့နောက် Add user ကို နှိပ်ပေးပါ။ Database access အတွက် အကောင့် လုပ်ပေးရခြင်းမှာ ထို database ကို

အကောင့်လုပ်ထားသူမှသာ access လုပ်နိုင်ရန် ဖြစ်ပါသည်။ ထို့နောက် Network access ကို နှိပ်ပေးပါ။ Network access သည် မိမိတို့ သတ်မှတ်ထားသော ip များမှ သာလျှင် ထို database သို့ access လုပ်နိုင်မည် ဖြစ်ပါသည်။ username and password သိလျှင်တောင် ip မတူလျှင် ဝင်၍ မရပါ။

Network access ထဲတွင် Add IP Address ဆိုတာကို နှိပ်ပေးပါ နောက်ထပ်ပေါ်လာသော နေရာတွင် Allow Access From Anywhere ကို နှိပ်ပေးပြီး Confirm ကို နှိပ်ပေးပါ။ Add current ip address ကို သုံးနိုင်သော်လည်း မိမိတို့ စမ်းသပ်စဉ်ကာလတွင် ip error များ မတက်စေရန် Allow access from anywhere ကိုသုံးပါမည်။ မိမိတို့ Business or Personal database အတွက်ဆိုလျှင် မိမိ လိုသလို ip ကို ပေးနိုင်ပါသည်။

အထက်ပါ အဆင့်များပြီးလျှင် Clusters ထဲသို့သွားပါ။ ပေါ်လာသော page တွင် Connect ဆိုတာကို နှိပ်ပေးပါ။

Connect ထဲသို့ ရောက်သွားလျှင် Connect Your Application ဆိုတာကို နှိပ်ပေးပါ။

နောက်ထပ် ပေါ်လာသော page တွင် Driver ကို Python ကို ရွေးပေးပါ Version နေရာတွင် 3.6 or later ကို ရွေးပေးပါ။ ပြီးလျှင် Copy ကို နှိပ်ပေးပြီး link ကို copy ကူးခဲ့ပါ။

အထက်ပါအတိုင်း copy ကူးခဲ့ပြီးလျှင် မိမိတို့ program ၌ အောက်ပါအတိုင်းလုပ်ဆောင်ပါ။

Sample Program (334)

```
import pymongo
uri="mongodb+srv://root:toor@cluster0-85eda.mongodb.net/test?r
etryites=true&w=majority"
connection = pymongo.MongoClient(uri)
database = connection["myDB"]
collection=database["myCol"]
data={'WinHtut':'CrazyCoder','age':26,'Hobby':'BuildingTools'}
try:
    collection.insert_one(data)
    print("Data successfully inserted")
except Exception as err:
    print(err)
```

အစိမ်းရောင်ဖြင့် ရေးထားသော uri နေရာတွင် မိမိတို့ copy ကူးလာသော link ကို ထည့်ရပါမည်၊ ထိုထဲမှ root and toor နေရာတွင် မိမိတို့ database username and password များကို ထည့်ရပါမည်။ myDB နေရာတွင် မိမိတို့ ရဲ့ database name ကို ထည့်ပေးရမည် myCol နေရာတွင် မိမိတို့ တည်ဆောက်ခဲ့သော collection name ကို ထည့်ပေးရပါမည်။

```
import pymongo
                                Database User name
uri="mongodb+srv://root:toor@cluster0-85eda.mongodb.net/test?retryWrites=true&w=majority"
                                Database Password
                                                encode url to connect database
connection = pymongo.MongoClient(uri)
database = connection["myDB"]                   database name
collection=database["myCol"]                    database collection name

data={'WinHtut':'CrazyCoder','age':26,'Hobby':'BuildingTools'}
      python dictionary object
                                method to insert data
try:
    collection.insert_one(data)
    print("Data successfully inserted")
except Exception as err:        error from exception
    print(err)
```

ဒီနေရာမှာ exception ကိုပဲ သုံးထားပါတယ် ထိုကဲ့သို့ သုံးထားတဲ့ အတွက် ဖြစ်လာတဲ့ error အတိအကျကို ပြန်ထုတ် ပေးရန် အတွက်ပါ။ ဥပမာ username and password မှားနေခဲ့ရင် Authentication error ဆိုပြီး ပြပေးပါတယ်။

Django(Web Development)

Django သည် Python ရဲ့ web framework တစ်ခု ဖြစ်သလို ကမ္ဘာ တစ်လွှားမှာလည်း အသုံးပြုနေကြပြီး ဖြစ်ပါတယ်။ Django ကို စတင် အသုံး ပြုနိုင်ရန် Python 3 ကို ပထမဦးစွာ install လုပ်ထားဖို့ လိုအပ်ပါတယ်။ ထို့ပြင် pip ကိုလည်း install ထားရန် လိုအပ်ပါသည် pip သည် python ကို install လုပ်ကတည်း တစ်ပါတည်း ပါဝင်ပါသည်။ Python ရဲ့ package တွေကို management လုပ်ရန် အတွက် ဖြစ်ပါသည်။

```
C:\Users\winht>python
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC
Type "help", "copyright", "credits" or "license" for more info
>>> exit()

C:\Users\winht>pip --version
pip 19.2.3 from c:\users\winht\appdata\local\programs\python\p

C:\Users\winht>_
```

အထက်ပါ ပုံအတိုင်း python version များနှင့် pip version များ ပေါ်လာရန် လိုအပ်ပါသည်။ Django ကို စက်တစ်ခုလုံးမှာ install လုပ်တာမျိုး မဟုတ်ပဲ မိမိတို့ Project အလိုက်သာ လိုသလို install လုပ်မည် ဖြစ်ပါသည်။ ထို့ကဲ့သို့ ပြုလုပ်ခြင်း ဖြင့် နောက်ပိုင်းတွင် version မတူသည့် ပြဿနာများကို ရှောင်ရှား နိုင်ပါသည်။ ထို့ကြောင့် ပထမဆုံး အနေဖြင့် virtual envrionment တစ်ခု တည်ဆောက်ရန် လိုအပ်ပါသည်။ ထိုသို့ တည်ဆောက်ရန် virtualwrapper-win ကို install လုပ်ပေးရန် လိုအပ်ပါသည် virtualwrapper ကို install လုပ်ထားမှသာ virtual environment ကို ပြုလုပ်နိုင်မည် ဖြစ်သည်။ ထို virtual env ထဲမှာမှ django ကို မိမိတို့ လိုသလို install လုပ်မည် ဖြစ်သည်။

pip install virtualenvwrapper-win

```
C:\Users\winht>pip install virtualenvwrapper-win
Collecting virtualenvwrapper-win
  Downloading https://files.pythonhosted.org/packages/d3/07/7599a80e13e58e0bb561ed03c5
/virtualenvwrapper-win-1.2.6.tar.gz
Requirement already satisfied: virtualenv in c:\users\winht\appdata\local\programs\pyt
(from virtualenvwrapper-win) (16.7.9)
Installing collected packages: virtualenvwrapper-win
  Running setup.py install for virtualenvwrapper-win ... done
Successfully installed virtualenvwrapper-win-1.2.6
WARNING: You are using pip version 19.2.3, however version 20.1.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

C:\Users\winht>
```

အထက်ပါ အတိုင်း virtualenvwrapper ကို install လုပ်ပြီးပါက virtual environment တစ်ခုကို create လုပ်ပါမည်။ mkvirtualenv vm (နောက်က vm သည် virtual env ရဲ့ name ဖြစ်ပြီး မိမိ အဆင်ပြေသလို ပေးနိုင်သည်)

```
C:\Users\winht>mkvirtualenv vmdj
 C:\Users\winht\Envs is not a directory, creating
Using base prefix 'c:\\users\\winht\\appdata\\local\\programs\\python\\pythor
New python executable in C:\Users\winht\Envs\vmdj\Scripts\python.exe
Installing setuptools, pip, wheel...
done.

(vmdj) C:\Users\winht>ls
```

vmdj ထဲသို့ရောက်သွားသည်ကို တွေ့ရပါမည်။ အထက်ပါ အတိုင်းပေါ်လာပါက virtual environment ပြုလုပ်ခြင်း အောင်မြင်ပါသည်။ ထို့နောက် django ကို install ဆက်လုပ်ပါမည်။

pip install Django

```
(vmdj) C:\Users\winht>pip install django
Collecting django
  Downloading Django-3.0.7-py3-none-any.whl (7.5 MB)
     |████████████████████████████████| 7.5 MB 930 kB/s
Collecting pytz
  Downloading pytz-2020.1-py2.py3-none-any.whl (510 kB)
     |████████████████████████████████| 510 kB 1.1 MB/s
Collecting asgiref~=3.2
  Downloading asgiref-3.2.7-py2.py3-none-any.whl (19 kB)
Collecting sqlparse>=0.2.2
  Downloading sqlparse-0.3.1-py2.py3-none-any.whl (40 kB)
     |████████████████████████████████| 40 kB 1.3 MB/s
Installing collected packages: pytz, asgiref, sqlparse, django
Successfully installed asgiref-3.2.7 django-3.0.7 pytz-2020.1 sqlparse-0.3.1

(vmdj) C:\Users\winht>
```

First Project With Django

အထက်ပါ အဆင့်များ ပြီးသွားလျှင် project folder တစ်ခု စ ဆောက်ပါမည်

စာရေးသူ အနေဖြင့် djProjects ဟု နာမည် ပေးလိုက်ပါသည်။ ထို့နောက် ထို djProjects folder ထဲသို့ ဝင်ပြီး django application တစ်ခုကို စတင် လိုက်ပါသည် ထိုသို့ စတင်ရန် application name ကို ပေးရန်လည်း လိုအပ်ပါသည်။

```
mkdir djProjects
cd djProjects
django-admin startproject djapp
```

အထက်ပါ အဆင့်များ ပြီးပါက python ဖြင့် web application တစ်ခုပြုလုပ်ခြင်း အောင်မြင် သွားပါပြီ။ မိမိတို့ Pc သို့ သွားပြီး အောက်ပါ အတိုင်း သွားကြည့်ပါ C:\Users\winht\djProjects\djapp

ပုံပါ အတိုင်း djapp folder တစ်ခုနှင့် manage file တစ်ခုကို တွေ့ရပါမည်။ ထို့နောက် cmd သို့ပြန်သွားပြီး djapp folder ထဲသို့ဝင်ပါ။

Djapp folder ထဲတွင် djapp folder နှင့် manage.py တို့ကို တွေ့ရမည်။

```
(vmdj) C:\Users\winht\djProjects>cd djapp

(vmdj) C:\Users\winht\djProjects\djapp>dir
 Volume in drive C has no label.
 Volume Serial Number is AA6D-23FE

 Directory of C:\Users\winht\djProjects\djapp

06/15/2020  07:57 PM    <DIR>          .
06/15/2020  07:57 PM    <DIR>          ..
06/15/2020  08:19 PM    <DIR>          djapp
06/15/2020  07:57 PM               646 manage.py
               1 File(s)            646 bytes
               3 Dir(s)  126,782,296,064 bytes free
```

Application ကို စတင် run ရန် python manage.py runserver ဟုရေးပေးရမည်။ အခြားသော server များကို Install လုပ်ပေးရန် မလိုအပ်ပါ။ အဘယ်ကြောင့်ဆိုသော် django ကို install လုပ်ကတည်းက localhost မှာ အသုံးပြုနိုင်ရန် built-in server တစ်ပါတည်း ပါဝင်နေပါသည်။

```
(vmdj) C:\Users\winht\djProjects\djapp>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 17 unapplied migration(s). Your project may not work properly until you apply the m
 auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
June 15, 2020 - 20:27:28
Django version 3.0.7, using settings 'djapp.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

ထိုကဲ့သို့ run ပြီးပါက မိမိ နှစ်သက်ရာ browser တစ်ခုကို ဖွင့်၍ url တွင် 127.0.0.1:8000 ဟုရေးပြီး enterကိုနှိပ်လိုက်ပါက django app တစ်ခု ပေါ်လာသည်ကို မြင်ရပါမည်။ Localserver တွင် web application တစ်ခု ပြုလုပ်ခြင်း အောင်မြင်ပါပြီ။

အထက်တွင် run လိုက်သော web page သည် built-in ပါဝင်ပြီးသော Program ကို run ခြင်း ဖြစ်ပါသည်။ ဆက်လက်ပြီး မိမိတို့ ကိုယ်တိုင် project တစ်ခု ထပ်ဆောက်ပါမည် ထို project ထဲတွင် မိမိတို့ ပေါ်ချင်သော web page ပုံစံ များကို ရေးသားပါမည်။

ပထမဆုံး အနေဖြင့် Project folder ရှိသည့် နေရာကိုသွားပါ ထို့နောက် workon (virenv name) ကိုရေးသားပြီး မိမိတို့ယခင်က ဆောက်ခဲ့သော virtual envrionment ကို ပြန်သုံးပါမည်။ နောက်ဆုံး အနေဖြင့် မိမိပြုလုပ်ချင်သည့် project name ကိုရေးပေးရပါမည်။

workon vmdj
python managy.py startapp secondApp

စာရေးသူ အနေဖြင့် secondApp ဆိုသည့် project တစ်ခုကို ထပ်ဆောက်လိုက်ပါသည်။

```
C:\Users\winht>cd djProjects

C:\Users\winht\djProjects>cd djapp

C:\Users\winht\djProjects\djapp>workon vmdj
(vmdj) C:\Users\winht\djProjects\djapp>dir
 Volume in drive C has no label.
 Volume Serial Number is AA6D-23FE

 Directory of C:\Users\winht\djProjects\djapp

06/15/2020  10:05 PM    <DIR>          .
06/15/2020  10:05 PM    <DIR>          ..
06/15/2020  08:27 PM                 0 db.sqlite3
06/15/2020  08:19 PM    <DIR>          djapp
06/15/2020  07:57 PM               646 manage.py
               2 File(s)            646 bytes
               3 Dir(s)  126,562,914,304 bytes free

(vmdj) C:\Users\winht\djProjects\djapp>python manage.py startapp secondApp

(vmdj) C:\Users\winht\djProjects\djapp>
```

ထို့နောက် vscode ကို ဖွင့်ပြီး မိမိတို့ djapp folder ကို ဖွင့်ပါ။

ထို့နောက် အောက်ပါ အတိုင်း file directory များကို တွေ့ရပါမည်။

DJAPP folder ထဲ၌ djapp folder , secondApp folder နှင့် manage.py ဆိုသည့် file သုံးခု ရှိပါမည်။ manage.py သည် program run ရန် ဖြစ်ပြီး secondApp folder သည်မိမိတို့ ရေးသားလိုသည့် program များ ရေးရန်ဖြစ်သည်။ ထို secondApp folder ထဲမှ views ထဲတွင် မိမိတို့ web page တွင် ပေါ်လိုသည့် contents များကို ရေးသားရပါမည်။

- ပထမဆုံးအနေဖြင့် secondApp folder ထဲတွင် urls.py ဆိုသည့် Python file တစ်ခု ဆောက်ပေးရမည် ဖြစ်ပြီး ထို file သည် views.py file နှင့် Link ချိတ်ရန် ဖြစ်သည်။
- ဒုတိယအနေဖြင့် views.py ထဲတွင် မိမိတို့ ရေးသားလိုသည့် contents များကို ရေးသားရမည်။
- တတိယ အနေဖြင့် djapp folder ထဲမှ urls.py ထဲတွင် အဓိက ပေါ်လိုသည့် home page link ကို ချိတ်ပေးရပါမည်။

  အထက်ပါ သုံးခုထဲကို အဆင့်လိုက် ပြုလုပ်နိုင်လျှင် မိမိတို့ရေးသား လိုသည့် webpage တစ်ခု ကို ရေးသား နိုင်ပါပြီ။

First Step

` urls.py ဆိုသည့် python file တစ်ခု တည်ဆောက်ပါမည်။ ထို့ကြောင့် file directory သည် အောက်ပါ အတိုင်း ဖြစ်သွားပါမည်။

ထို urls.py file ထဲတွင် အောက်ပါ program များကို ရေးသားပါမည်။

```
from django.contrib import admin          #Line 1
from django.urls import path              #Line 2
```

```
from.import views                          #Line 3


urlpatterns = [                            #Line 4
    path('',views.home,name='home'),
]
```

Line number 2 မှ path ကို import လုပ်ရ ခြင်းသည် file တစ်ခုနှင့် တစ်ခု ချိတ်ဆက်နိုင်ရန် ဖြစ်သည်။ line 3 မှ views သည် views.py file ထဲမှ function များကို ယခု urls.py file ထဲမှ ခေါ်သုံးရန် ဖြစ်သည်။ အထက်ပါ program တွင် views.home ဟုရေးသားထားပြီး home function ကို ခေါ်သုံးထားပါသည်။

```
path('',views.home,name='home'),
```

path ဆိုသည့် function ထဲတွင် မိမိတို့ home page ထားလိုသည့် function name ကို ရေးပေးရပါမည်။

Second Step

Views.py file ထဲတွင် အောက်ပါ program တို့ကို ဖြည့်စွက် ရေးသားပါမည်။

```
from django.shortcuts import render                #Line 1
from django.http import HttpResponse               #Line 2

# Create your views here.                          #Line 3
def home(request):                                 #Line 4
    return HttpResponse("hello world")             #Line 5
```

Line 1 မှ render သည် မိမိ Page များကို user များ မြင်နိုင်ဖို့ ဖော်ပြရန် ဖြစ်ပြီး HttpResponse သည် contents များကို ဖော်ပြရာတွင် http ပုံစံဖြင့် ဖော်ပြပေးရန် ဖြစ်သည်။ ယခု program တွင် HttpResponse ကို မသုံးပဲ ပုံမှန် return “hello world” ဟုလည်း ရေးနိုင်သည်။

Third Step

တတိယ အနေဖြင့် djapp folder ထဲမှ urls.py ထဲတွင် မိမိတို့ ဖော်ပြ လိုသည့် Home page ကို ချိတ်ပေးရန် ဖြစ်သည်။ ထို့သို့ ချိတ်ပေးရန် ရေးသားရမည့် Program မှာ အောက်ပါ အတိုင်း ဖြစ်သည်။

```
from django.contrib import admin            #Line 1
from django.urls import path,include        #Line 2
    urlpatterns = [                         #Line 3
    path('',include('secondApp.urls')),
    path('admin/', admin.site.urls),
]
```

Line 2 တွင် path and include function ကို အသုံးပြုရန် Import လုပ်ပေးသည်။ include function သည် မိမိတို ချိတ်ဆက်ပေးလိုသည့် Url များကို ချိတ်ဆက်ပေးရန် သုံးပါသည်။ ထိုကဲ့သို့ရေးပေးမှ သာ program စ run သော အချိန်တွင် include function

နောက်မှ url အတိုင်းသွားပြီး မိမိတို့ ဖော်ပြ လိုသည့် page ကို ဖော်ပြ ပေးမှာ ဖြစ်ပါတယ်။ ထိုသို့ မဟုတ်လျှင် ယခင် သင်ခန်းစာမှ ပေါ်နေသည့် django home page ချည်းသာ ပေါ်နေမှာ ဖြစ်ပါတယ်။

Program run ရန် python manage.py runserver သာ ဖြစ်သည်။ ထို့နောက် မိမိနှစ်သက်သည့် browser တွင် 127.0.0.1:8000 ဟု ရေးပြီး enter လိုက်ပါက hello world ဆိုသည့် စာသား ပေါ်လာသည်ကို မြင်ရပါမည်။

Django Template Language

Template ဆိုတာ text file တစ်ခုဖြစ်ပါတယ်။ သို့သော် သာမန် text file မဟုတ်ပဲ text-based format တွေ ဖြစ်တဲ့ HTML , XML , CSV တို့ကို generate လုပ်နိုင်သလို Dynamic contents တွေထည့်ဖို့လည်း အသုံး ပြုပါတယ်။ Dynamic contents တွေဆိုတာ မိမိတို့ ဖော်ပြချင်တဲ့ data ကို လိုသလို ဖော်ပြ နိုင်တာမျိုးပါ static ကဲ့သို့ ပုံသေမဟုတ်ပဲ။ ဥပမာ web page တစ်ခုမှာ user ဆီက name ရယူပြီး ထို name ကို ပြန်လည် ဖော်ပြ ပေးနိုင်ဖို့ အောက်ပါ အတိုင်း ရေးနိုင်ပါတယ်။

```
<h1> Hello {{username}} </h1>
```

ယခု သင်ခန်းစာမှာတော့ Django Template language ကို သုံးပြီး dynamic web page တစ်ခု တည်ဆောက်သွားမှာ ဖြစ်ပါတယ်။ ထိုသို့ တည်ဆောက်ရန် ပထမဆုံး templates ဆိုသည့် folder တစ်ခုကို ယခင် project file ထဲတွင် ဖန်တီးပါမည်။

1. Templates folder တစ်ခု ဖန်တီးမည်
2. Templates folder ထဲတွင် home.html file တစ်ခုတည်ဆောက်မည်။ ထို html file အား template file အဖြစ် သုံးမှာ ဖြစ်သည့် အတွက် ထို html tag ထဲတွင် name ဆိုသည့် variable ကိုရေးသားခဲ့ပါမည်။
3. Settings.py ထဲတွင် templates folder path လမ်းကြောင်းပေးမည်။
4. Views.py ထဲတွင် Httpresponse မှ တစ်ဆင့် render method ကိုပြောင်းမည်။ page များကို ဖော်ပြလိုသော အခါတွင် render method ကိုသုံးပါသည်။
5. Render method ထဲတွင် parameter သုံးခု ထည့်ပေးရမည် ပထမ parameter သည် request , ဒုတိယ Parameter သည် html file name နှင့် တတိယ Parameter သည် မိမိတို့ ဖော်ပြချင်သည့် dynamic contents ဖြစ်ပါသည်။

Step 1

Step 2

Templates folder တစ်ခု ဖန်တီးပြီး ထို folder ထဲတွင် home.html ဆိုသည့် file တစ်ခု တည်ဆောက် ထားပါသည်။ ထို file ထဲတွင် အောက်ပါ အတိုင်း name ဆိုသည့် variable တစ်ခု ထည့်ထားပါသည်။ ထို variable နေရာသို့ value သည် views.py ဆိုသည့် file မှ လာမှာ ဖြစ်ပါတယ်။

Step 3

Settings.py file ထဲမှ Line number 57 တွင် အောက်ပါ အတိုင်း path လမ်းကြောင်း ပေးရမည်။

```
'DIRS': [os.path.join(BASE_DIR,'templates')],
```

Step 4

```
def home(request):
    return render(request,'home.html',{'name':'WinHtut'})
```

Step 5

```
return render(request,'home.html',{'name':'WinHtut'})
```

အထက်ပါ အဆင့်များ အားလုံးပြီးပါက File ထဲမှ save all ဆိုတာကို နှိပ်ပေးပြီး မိမိတို့ browser တွင် ပြန် run ကြည့်ပါက အောက်ပါ အတိုင်း dynamic webpage တစ်ခုကို အောင်မြင်စွာ ဖန်တီး နိုင်သည်ကို တွေ့မြင်ရပါမည်။

Hello from Dynamic Page

My name is WinHtut

MongoDB and Django

ယခု သင်ခန်းစာမှာတော့ mongodb and django တို့ပေါင်းပြီး data ရယူကာ web page တစ်ခုမှာ ပြန်လည်ဖော်ပြတဲ့ application တစ်ခု ရေးသွားမှာပါ။ ယခု သင်ခန်းစာမှာတော့ home.html နှင့် views.py file နှစ်ခုတည်းကိုသာ ကိုင်တွယ်ပြီး ရေးပြသွားမှာ ဖြစ်ပါတယ်။ ပထမဆုံး အနေဖြင့် home.html ထဲတွင် အောက်ပါ အတိုင်း ရေးသားပါမည်။

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        h1 {
          text-align: center;
        }
        </style>
</head>
<body>
        <h1>All Data Are </h1>
        <h1>id : {{id}}</h1>
        <h1>name : {{name}}</h1>
        <h1>age : {{age}}</h1>
        <h1>hobby : {{hobby}}</h1>

<h3>{{error}}</h3>
</body>
</html>
```

အထက်ပါ html file ထဲ၌ style tag တစ်ခု ထည့်ထားပါသည် ထို style tag သည် စာသားများအား web page အလယ်တွင် ပေါ်စေရန် ဖြစ်ပါသည်။ သတိပြုရန်မှာ h3 tag ဖြင့် error ဆိုသည့် variable ကို ဖော်ပြထားပါသည် ထို error သည် views.py မှ exception ဖြစ်ပြီး ဖြစ်လာသော error များကို တိတိ ကျကျ ဖော်ပြရန် ဖြစ်ပါသည်။

Mongodb ကို django နဲ့ တွဲ ရေးတဲ့ အခါ အထက်ပါ အတိုင်း pylint import error ဆိုပြီး တက်နိုင်ပါတယ်။ ထိုကဲ့သို့၌ ဖြစ်လာပါ က control+shift+p ကို နှိပ်လိုက်ပါ ထို့နောက် ပေါ်လာသော နေရာတွင် အောက်ပါ အတိုင်း python ဟု ရိုက်ပြီး python interpreter ကိုရွေးပေးပါ။

အောက်ပါအတိုင်း မိမိတို့ အသုံးပြုနေသည့် virtual environment များ ပေါ်လာပါမည် ပေါ်လာသည့် env များ ထဲမှ မိမိတို့ လက်ရှိ run နေသည့် env ကို select လုပ်ပေးလိုက်ပါ ခဏာ အကြာတွင် ညာဘက် အောက် ဒေါင့်မှ noti တစ်ခု တက်လာပါမည်။

တက်လာသော noti တွင် pylint ကို install လုပ်ရန် မေးခြင်း ဖြစ်ပြီး install လုပ်ပေးလိုက်ပါက မိမိတို့ mongodb ကို import လုပ်ခြင်း အဆင်ပြေ သွားမှာ ဖြစ်သလို အခြားသော modules များထည့်ရန် အတွက်လည်း အဆင်ပြေသွားမှာပါ။

ထိုကဲ့သို့ ပေါ်မလာပဲ error များ ဆက်တက်နေပါက မိမိတို့ ယခု project ရေးနေသည့် virtual env ထဲသို့ သွားပြီး လိုအပ်သော module များကို install လုပ်ပေးရမှာပါ။ ဥပမာ pymongo ကို မသိဘူးဆိုလျှင် pip3 install pymongo ဟု install ပြန်လုပ် ပေးရမှာပါ။

ပထမဆုံး အနေဖြင့် mongodb ကို သွားချိတ်ပါမည်။ mongodb သို့မချိတ်မီ database

name (djDB) နှင့် collection name (djCO) တို့ကို ပေးထားပြီး id , name ,age ,Hobby တို့ data များအရင် ဆုံး ထည့်ထားပေးပါ။ အကယ်၍ မထည့်တတ်ပါက mongodb insert data အခန်းကို ပြန်လည် ဖတ်ရှုနိုင်ပါသည်။

နောက်တစ်ဆင့် အနေဖြင့် views.py file ထဲတွင် အောက်ပါတိုင်း mongodb ကို ချိတ်ပါမည် ချိတ်ပြီးလျှင် home function တစ်ခု တည်ဆောက်ပြီး try and except တို့ကို သုံးကာ data render လုပ်ပါမည်။ try and except တို့ကို သုံးရခြင်းမှာ python ရဲ့ exception သည် error ကို အတိအကျ ဖော်ပြ ပေးနိုင်တဲ့ အတွက် ပုံမှန်အားဖြင့် error တစ်ခုခုဖြစ်လျှင် error log ပေါင်းများစွာကို ဖတ်စရာ မလိုတော့ပဲ တိကျသည့် error ကို သိရှိနိုင်ရန် ဖြစ်သည်။ ထို့ကြောင့် try ထဲတွင် render method ကိုပင်သုံးကာ data များကို ဖော်ပြပေးထားပြီး exception ထဲ၌လည်း render ကိုသုံးကာ error log ကို ဖော်ပြ ပေးထားပါသည်။

```
from django.shortcuts import render
from django.http import HttpResponse
import pymongo
connection = pymongo.MongoClient("localhost",27017)
database = connection["djDB"]
collection=database["djCO"]
def home(request):
    try:
        data=collection.find_one()
        for i in data.items():
            print(i)
        id,name,age,hobby=data['_id'],data['name'],data['age'],data['Hobby']
                                                                    return
render(request,'home.html',{'id':id,'name':name,'age':age,'hobby':hobby})
    except Exception as error:
        return render(request,'home.html',{'error':error})
```

အထက်ပါ အတိုင်းရေးပြီး save ကာ server ကို ပြန် run လိုက်ပါက အောက်ပါ အတိုင်း web page တစ်ခုကို မြင်ရပါမည်။

Warning

မိမိကိုယ်တိုင် python file တစ်ခုကို ပထမဆုံး ရေးကာ database ထဲသို့ data များ ထည့်ထားရပါမည်။

Template Inheritance

Template inheritance လုပ်တယ်ဆိုတာ template file တစ်ခုမှာ ရေးထားတဲ့ code တွေကို အခြား template file တစ်ခုကနေ inheritance လုပ်တာပါ ဆိုလိုသည်မှာ parent.html and home.html ဆိုသည့် template file နှစ်ခုတွင် parent.html ထဲတွင် ရေးထားသည့် template code တွေကို home.html ဆိုသည့် file ထဲမှ ယူသုံးတာမျိုးပါ။ ထိုသို့ ယူသုံးရန် အတွက်လည်း extends ကိုသာ အသုံးပြုရုံဖြင့် ရနိုင်ပါသည်။ ယခု Program အတွက် ပထမဆုံး templates folder ထဲတွင် parent.html ဆိုသည့် template file တစ်ခု တည်ဆောက်မည် ။

အထက်ပါ အတိုင်း ဆောက်ပြီးလျှင် parent.html ထဲ၌ အောက်ပါ တို့ကို ရေးသားပါမည်။

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
                <meta    name="viewport"    content="width=device-width,
initial-scale=1.0">
    <title>test</title>
</head>
<body><h1>
    <div id="content",name="content">
        {%block content%}
        {%endblock%}

    </div>
    </h1>
</body>
</html>
```

သတိပြုရန်မှာ parent.html ထဲတွင် block content အဖွင့်နှင့် အပိတ်ကို ထည့်ပေးရမည်။ ထိုသို့ ထည့်ပေးမှသာ ယခု file ထဲတွင် ရေးထားသော design or other elements များကို အခြားသော file မှ ယူသုံး နိုင်မည်ဖြစ်သည်။ ဥပမာ page တိုင်းတွင် home button လေး ပါချင်ပါက ထို home buttom အတွက် code ကိုယခု parent.html ထဲတွင် ထည့်ပေးရမည်။ ထို့သို့ ထည့်ပေးလျှင် ယခု parent.html အား inheritance လုပ်သည့် page တိုင်းတွင် home buttom လေး ပါဝင်နေမှာ ဖြစ်ပါတယ်။

home.html ထဲတွင် အောက်ပါတို့ကို ရေးသားပါမည်။

```
{% extends "parent.html" %}                    #Line 1
{% block title %}Template{% endblock%}         #Line 2

{% block content%}                             #Line 3

{% for key,value in data.items %}              #Line 4
<li>{{key}}{{value}}</li>                      #Line 5
{%endfor%}                                     #Line 6
{% endblock%}                                  #Line 7
```

ပထမဆုံး ရေးထားသည့် extends သည် parent.html ထဲမှ elements များကို ယူသုံးမည်ဟု ဆိုခြင်းဖြစ်ပြီး ဒုတိယ line title အတွက် ဖြစ်သလို block အစနှင့် အဆုံးကို

တစ်ပါတည်း ဖော်ပြ ပေးရသည်။ line 3 တွင် block content အစကို ရေးထားပြီး အဆုံးတွင် ပြန်ပိတ်ထားသည်။ ထို အထဲတွင် for statement ကိုပါ ထည့်သုံးထားပါသည်။ for ကို အဆုံးသတ်ပေးရန် အတွက်လည်း line 7 တွင် endfor ကို ထည့်ပေးရပါသည်။

Line 4 မှ data.items တွင် data သည် views.py မှ ပို့ပေးမည့် data object ဖြစ်ပါသည်။ ထို data object ထဲတွင် database မှ ရလာသော data များ ပါဝင်ပါသည်။ views.py ထဲတွင် အောက်ပါ အတိုင်းသာ ရေးထားပါသည်။

```
from django.shortcuts import render                          #Line 1
from django.http import HttpResponse                         #Line 2
import pymongo                                               #Line 3
connection = pymongo.MongoClient("localhost",27017)          #Line 4
database = connection["djDB"]                                #Line 5
collection=database["djCO"]                                  #Line 6
def home(request):                                           #Line 7
    try:                                                     #Line 8
        data=collection.find_one()                           #line 9
        return render(request,'home.html',{'data':data})     #Line 10
    except Exception as error:                               #Line 11
        return render(request,'test.html',{'error':error})   #Line 12
```

Line 9 တွင် ရလာသော data object အား home.html သို့ ပို့ပေးလိုက်ခြင်း ဖြစ်ပါသည်။

အထက်ပါ အတိုင်း program အားလုံးအား save ပြီး run လိုက်လျှင် home.html page ထဲတွင် အောက်ပါ အတိုင်း format များဖြင့် ပေါ်လာသည်ကို တွေ့ရပါမည်။

အထက်တွင် ပေါ်လာသော data များသည် mongodb ထဲမှ ယခင် သင်ခန်းစာများတွင် ရေးသားထားသော documents မှ ဖြစ်သည်။ ထိုသို့ ပေါ်လာရန်အတွက်လည်း home.html ထဲတွင် အောက်ပါ code လေးကို ထည့်ပေးထားခြင်း ဖြစ်သည်။

```
{% for key,value in data.items %}
<li>{{key}}{{value}}</li>
{%endfor%}
```

Django Form Action

ယခု သင်ခန်းစာမှာတော့ HTML form တစ်ခု တည်ဆောက်ပြီး ထို form ထဲမှ data များကို အခြားသော html form တစ်ခုဖြင့် ပြန်ပြမှာ ဖြစ်ပါတယ်။ နောက်သင်ခန်းစာများတွင်တော့ database ထဲသို့ ထည့်ခြင်း ပြန်ထုတ်ပြခြင်း တို့ကို ဖော်ပြသွားမှာပါ။ ယခင် သင်ခန်းစာမှာ သုံးထားသည့် templates folder ထဲမှ home.html နှင့်

parent.html တို့အပြင် result.html ဆိုသည့် file တစ်ခု ထပ်ထည့်ပါမည်။ ထို file သည် user ထည့်ပေးလိုက်သော data များကို ပြန်ထုတ်ပြ ရန်ဖြစ်သည်။ Templates file directory ထဲတွင် အောက်ပါ အတိုင်း files များရှိကြပါသည်။

Templates

- home.html
- parent.html
- result.html

ပထမဆုံး အနေဖြင့် result.html ထဲတွင် အောက်ပါ အတိုင်းရေးသားထားပါမည်။

For result.html

```
<h1>name: {{name}}</h1>
<h1>email: {{email}}</h1>
<h1>password: {{pass}}</h1>
```

အထက်ပါ အတိုင်းရေးသားထားခြင်းကို jinja ပုံစံဖြင့်ရေးသားခြင်းဟု ခေါ်ပြီး name , email , pass စသည့် data များကို views.py မှ လှမ်းပို့ပေးမည် ဖြစ်ပါသည်။

home.html ထဲတွင်လည်း အောက်ပါ အတိုင်း ပြင်ဆင်ရေးသားပါမည်။ input သုံးခုနှင့် submit button တစ်ခုကို form တစ်ခုထဲတွင် ထည့်ထားပါသည်။ သတိပြုရန်မှာ form ရဲ့ class name ကို box ဟုပေးထားပြီး action ကိုတော့ add ဟု ပေးထားပါသည်။ ထို form မှ submit button ကို နှိပ်လိုက်လျှင် input ထဲမှ data များကို ယူပြီး views ထဲမှ add function ကို သွားခေါ်ပါမည်။ ထိုသို့ သွားခေါ်နိုင်ရန် အတွက်လည်း ယခင် သင်ခန်းစာများ၍ settings.py ထဲတွင် templates folder ကို path လမ်းကြောင်း ပေးခဲ့ ခြင်း ဖြစ်သည်။

For home.html

```
{% extends "parent.html" %}
{%block title%}Template{%endblock%}
{%block content%}

<form class="box" action="add">
    <h1>User Info</h1>
    <input type="text" name="name" placeholder="Username">
    <input type="text" name="email" placeholder="Email Address">
    <input type="text" name="password" placeholder="Password"><br>
    <input type="submit" name="" value="Register">
    </form>

{%endblock%}
```

Parent.html ထဲတွင် css code အနည်းငယ်ကို head tag ထဲတွင် style tag ကိုသုံးပြီး ထည့်ပေးပါမည်။

```
<style>
    body{
     margin: 0;
     padding: 0;
     font-family: sans-serif;
     background: #f6f7f8;
    }
```

```
.box{
  width: 300px;
  padding: 40px;
  position: absolute;

}
    </style>
```

Home.html ထဲမှ body style ကို အနည်းငယ်ပြင်ရန် နှင့် box class ပေးထားသည့် form style ကို အနည်းငယ် ပြင်ရန် ဖြစ်သည်။ ထို့ကြောင့် parent.html ထဲတွင် code အားလုံး အောက်ပါ အတိုင်း ရှိနေပါမည်။

onlineApp > templates > parent.html > html

```
<!DOCTYPE html><html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Template</title>
    <style>
        body{
         margin: 0;
         padding: 0;
         font-family: sans-serif;
         background: #f6f7f8;  }
        .box{
         width: 300px;
         padding: 40px;
         position: absolute; }
           </style>
</head> <body>
    <h1>
        <div id="content ",name="content">
                {%block content%}
                {%endblock%}
        </div>
    </h1>
</body>  </html>
```

File အားလုံးအား save ပြီး run ကြည့်ပါက အောက်ပါအတိုင်း ပေါ်လာသည်ကို မြင်ရပါမည်။

Data များထည့်ပြီး register button ကို နှိပ်လိုက်လျှင်လည်း မည်သည့် အလုပ်မျှ လုပ်ဦးမှာ မဟုတ်ပါဘူး အဘယ်ကြောင့်ဆိုသော် add function ကို views.py ထဲတွင် မရေးရသေးသည့် အတွက် ဖြစ်ပါသည်။ ထို့ကြောင့် views.py ထဲတွင် add function ကို အောက်ပါ အတိုင်းရေးပါမည်။

```
def add(request):
    name=request.GET['name']
```

```
    email=request.GET['email']
    password=request.GET['password']
        return  render(request,'result.html',{'name':name  ,  'email':email  ,
'pass':password})
```

Form ကနေမှ တစ်ဆင့် data ထည့်ပြီး register ဆိုသည့် button ကို နှိပ်လိုက်သည်နှင့် တစ်ပြိုင်နက် add function ကို သွားခေါ်သောအခါတွင် add function ထဲ၌ GET method ကို သုံးပြီး data များကို ရယူပါမည်။ name = request.GET['name'] တွင် form မှာ တစ်ဆင့် name ဆိုသည့် နေရာမှ ရယူလာသော data ကို name ဆိုသည့် variable ထဲသို့ ထည့်ခြင်း ဖြစ်ပါသည်။

```
<h1>User Info</h1>
<input type="text" name="name" placeholder="Username">
<input type="text" name="email" placeholder="Email Address">
<input type="text" name="password" placeholder="Password"><br>
<input type="submit" name="" value="Register">
</form>
```

```
def add(request):
    name=request.GET['name']
    email=request.GET['email']
    password=request.GET['password']
    return render(request,'result.html',{'name':name , 'email':email , 'pass':password})
```

အထက်ပါ နည်းအတိုင်း name , email , password တို့အား ရယူပြီး render method ကို result.html သို့ လှမ်းပို့ပေးလိုက်ပါသည်။ Program အားလုံ်းအား save ပြီး register button ကို နှိပ်လိုက်သည်နှင့် အောက်ပါ အတိုင်း data များကို တွေ့ရပါမည်။

အထက်ပါ ပုံရဲ့ url bar ကို တစ်ချက် ကြည့်မည်ဆိုလျှင် မိမိတို့ ပို့လိုက်သည့် data များကို မြင်နေရပါသည် real world app တစ်ခုတွင် ထိုကဲ့သို့ မြင်နေရခြင်းသည် ပြဿနာပေါင်း များစွာ ရှိလှသည့်အတွက် html form ကနေမှ တစ်ဆင့် data များပို့သောအခါတွင် အသုံးပြုနိုင်ရန် POST ကိုသုံးနိုင်ပါသည်။ POST နည်းတူ GET , HEAD , PUT , DELETE , CONNECT , OPTIONS , TRACE စသည့် method များလည်း ရှိပါသေးသည်။ ထိုသို့သော method များအားလုံးသည် HTTP (hypertext transfer protocol ထဲတွင် ပါဝင်ပြီး clients and servers များကြားတွင် commutations ပြုလုပ်ရန် သုံးပါသည်။

GET

GET method သည် သတ်မှတ်ထားတဲ့ နေရာသို့မဟုတ် မိမိ ရယူလိုသော နေရာကနေ data များ ရယူ နိုင်ရန် အသုံးပြုပါသည်။ သို့သော် get method ကိုအသုံးပြုရာတွင် url ၌ name and value စသည့် data များ ဖော်ပြနေတဲ့အတွက် အရေးကြီးတဲ့ data တွေကိုရယူတဲ့အခါမှာ အသုံးပြုရန် မသင့်တော်ပါ။ အထူးသဖြင့် username and password ။

ယခင် သင်ခန်းစာတွင် views.py ထဲ၌ get method အသုံးပြုပုံကို ပြထားပါသည်။ GET သည် data length ကိုလည်း Limit လုပ်ထားပါသေးတယ်။ စုစုပေါင်း character 2048 ထိသာ လုပ်ဆောင် နိုင်ပါတယ်။

POST

POST method ကတော့ client ကနေ တစ်ဆင့် server ဆီသို့ data ပို့ရာတွင် အသုံးပြုပါတယ်။ get method ကိုသုံးရာတွင် html form ထဲ၌ method ကိုရေးစရာ မလိုသော်လည်းပဲ POST method ကိုသုံးရာတွင်မူ method="post" ဆိုတာကို ရေးပေးရန် လိုအပ်ပါသည်။ POST method သည် pass လုပ်တဲ့ parameter data တွေကို Url မှာ ပြမပေးပါဘူး။ ထို့ပြင် browser history မှာလည်း parameters တွေကို stored လုပ်ထားခြင်း မရှိသည့်အတွက် POST method သည် GET method ထက် အနည်းငယ် ပိုမို လုံခြုံပါသည်။

ပထမဆုံး အနေဖြင့် မိမိတို့ ယခင် project ထဲမှ home.html ထဲမှ form ထဲတွင် method="POST"ဟု သွားထည့်ပါမည်။ ထို့ပြင် add နောက်တွင်လည်း / ကို ထည့်ပေးရပါမည်။

```
    </style>
</head>
<body>
    <form class="box" action="add/" method="POST">
        <h1>User Info</h1>
        <input type="text" name="name" placeholder="Username">
        <input type="text" name="email" placeholder="Email Address">
        <input type="text" name="password" placeholder="Password"><br>
        <input type="submit" name="" value="Register">
        </form>
</body>
</html>
```

ထို့ပြင် views.py ထဲတွင်လည်း အောက်ပါအတိုင်း GET ကနေ POST သို့ ချိန်းပေးရပါမည်။

```
def add(request):
    name=request.POST['name']
    email=request.POST['email']
    password=request.POST['password']

    return render(request,'result.html',{'name':name , 'email':email , 'pass':password})
```

အထက်ပါ file နှစ်ခုအား ပြင်ကာ save ပြီး run မည်ဆိုလျှင် CSRF error တက်သည်ကို အောက်ပါ အတိုင်း မြင်ရပါမည်။

Forbidden (403)

CSRF verification failed. Request aborted.

Help

Reason given for failure:
CSRF token missing or incorrect.

In general, this can occur when there is a genuine Cross Site Request Forgery, or when Django's

- Your browser is accepting cookies.
- The view function passes a request to the template's render method.
- In the template, there is a {% csrf_token %} template tag inside each POST form that targe
- If you are not using CsrfViewMiddleware, then you must use csrf_protect on any views that u
- The form has a valid CSRF token. After logging in in another browser tab or hitting the b

You're seeing the help section of this page because you have DEBUG = True in your Django settinc

You can customize this page using the CSRF_FAILURE_VIEW setting.

ထို့ပြဿနာကို ဖြေရှင်းရန် html file ထဲမှ form tag ထဲတွင် အောက်ပါအတိုင်း csrf_token ကို jinja template ဖြင့် ထည့်ပေးရမည်။

```
<body>
    <form class="box" action="add/" method="POST">
      {% csrf_token %}
        <h1>User Info</h1>
        <input type="text" name="name" placeholder="Username">
        <input type="text" name="email" placeholder="Email Addres
```

အထက်ပါအတိုင်း ထည့်ပြီး save ကာ run ကြည့်ပါက ကောင်းမွန်စွာ အလုပ်လုပ်သွားသည်ကို မြင်ရပါမည်။

127.0.0.1:8000/add/

name: WinHtut

email: winhtutonline@gmail.com

password: pa$$w0rd

Application With PostgreSQL

ယခု သင်ခန်းစာမှာတော့ ToDo Application တစ်ခု ကို လက်တွေ့ရေးသားသွားမှာဖြစ်သလို PostgreSQL ကိုလည်း အသုံးပြုသွားမှာပါ။ ထို့ကြောင့် ပထမဆုံး အနေဖြင့် မိမိတို့ စက်တွင် postgreSQL ကို install လုပ်ထားရန် လိုအပ်ပါသည်။

https://www.postgresql.org/download/

စာရေးသူ အနေဖြင့် windows သုံးသည့်အတွက် Windows ကိုသာရွေးချယ်လိုက်ပါသည်။

ထို့နောက် အထက်ပါအတိုင်း Download the installer ကို နှိပ်ပြီး download လုပ်လိုက်ပါသည်။ Install လုပ်နည်းများသည် အရမ်းလွယ်ကူသည့်အတွက် မဖော်ပြလိုတော့ပါ။ သတိပြုရန်မှာ installation လုပ်သော အချိန်တွင် password ပေးရပါမည်။ ထို password ကို မှတ်သားထားရန် လိုအပ်ပြီး setting.py ထဲတွင် DATABASES object ထဲ၌

ထို့ပြင် ထို postgresql ကို management လုပ်နိုင်ရန် အတွက် pgadmin ကို download ဆွဲရန် လိုအပ်ပါသည်။ https://www.pgadmin.org/download/ ယခု link ကို သွားပြီး မိမိတို့ environment နှင့်ကိုက်ညီသည့် package ကိုသုံးနိုင်ပါသည်။ စာရေးသူအနေဖြင့် Windows သုံးသည့်အတွက် windows ကိုသာ နှိပ်ပြီး download ဆွဲ လိုက်ပါသည်။

ရလာသော File အား install လုပ်ပေးရုံသာ ဖြစ်ပြီး install ပြီးပါက မိမိတို့ browser တွင် database server ပေါ်လာမည် ဖြစ်ပါသည်။

postgreSQL database ကို မိမိတို့ Django မှ သိရန် pip3 install --user psycopg2 ဟုရေးကာ install လုပ်ပေးရန် လိုအပ်ပါသည်။ ထို့နောက် စာရေးသူအနေဖြင့်သင့်တင့်သော နေရာတွင် Online ToDO ဆိုသည့် folder တစ်ခု တည်ဆောက်လိုက်ပါသည်။ ထို folder ထဲတွင်

Online Todo ဆိုသည့် django project file တစ်ခု ကို တည်ဆောက် လိုက်ပါသည်။ ထိုသို့ တည်ဆောက်ရန် အတွက်လည်း django-admin startproject Online Todo ဟု terminal or cmd တွင်ရေးကာ တည်ဆောက်ပါသည်။

```
PS C:\Users\winht\djprojects\OnlineToDo> django-admin startproject OnlineTodo
PS C:\Users\winht\djprojects\OnlineToDo> dir


    Directory: C:\Users\winht\djprojects\OnlineToDo


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----        6/28/2020   8:27 PM                OnlineTodo


PS C:\Users\winht\djprojects\OnlineToDo> cd .\OnlineTodo\
PS C:\Users\winht\djprojects\OnlineToDo\OnlineTodo>
```

ထို့နောက် python manage.py runserver ဆိုပြီး အောက်ပါ အတိုင်း run လို့ အဆင်ပြေပါက django project file တစ်ခု ဆောက်လုပ်ခြင်းပြီးပါပြီ။

```
PS C:\Users\winht\djprojects\OnlineToDo\OnlineTodo> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 17 unapplied migration(s). Your project may not work properly until you
Run 'python manage.py migrate' to apply them.
June 28, 2020 - 21:12:05
Django version 3.0.7, using settings 'OnlineTodo.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Django application တစ်ခု စတင်ရေးသားဖို့ အတွက် todoapp ဆိုသည့် django application တစ်ခု တည်ဆောက်ပါမည်။ထိုသို့ တည်ဆောက်ရန် python manage.py startapp todoapp ဟု ရေးရမည်။

```
s\winht\djprojects\OnlineToDo\OnlineTodo> python manage.py startapp todoapp
s\winht\djprojects\OnlineToDo\OnlineTodo>
```

Migration

Django နှင့် PostgreSQL တို့ ချိတ်ဆက်မိသွားဖို့ အတွက် migration လုပ်ဖို့ လိုအပ်ပါတယ်။ ထိုသို့ ပြုလုပ်ရန်

1. postgreSQL ထဲမှာ database တစ်ခု သွားဆောက်ရပါမည်။
2. Settings.py ထဲတွင် DATABASES object ထဲ၌
   - Engine မှာ backends.postgresql
   - 'NAME':'မိမိဆောက်လိုက်သော database name'
   - 'USER': 'postgres'
   - 'PASSWORD':'မိမိတို့ ပေးခဲ့သော pw'
   - 'HOST': 'localhost'(အကယ်၍အခြားနေရာတွင်ထားပါက ထိုနေရာကို ထည့်ပေးရမည်။
3. Models.py ထဲတွင် မိမိတို့လိုချင်သည့် database model တစ်ခု တည်ဆောက်ပေးရမည်။ ထိုကဲ့သို့ ဆောက်ပေးလိုက်သည့် အတိုင်း postgreSQL ထဲတွင် database tables များ ရှိနေမည် ဖြစ်သည်။
4. Migration လုပ်ရန်ဖြစ်သည်။

1.Database တစ်ခု ဆောက်ရန်အတွက် postgreSQL pgAdminကိုဖွင့်ပါ။ ပထမဆုံး ပေါ်လာသည့် နေရာတွင် မိမိတို့ ပေးထား်းသည့် password ကိုထည့်ပြီး ဖွင့်ပါ။

ထို့နောက် ဘယ်ဘက် အပေါ်ဒေါင့် မြှားပြသည့် နေရာမှ servers ကို နှိပ်ပါ။ ထို့နောက် Databases ကို right click ထောက်ပြီး Databases ကို နှိပ်ပါ ထို့နောက် Create မှ တစ်ဆင့် Database ကို နှိပ်ပါ။

Database ကို မိမိတို့ နှစ်သက်သည့် အတိုင်း ပေးနိုင်သလို ထို Database name ကို

setting.py ထဲတွင် ပြန်လည် အသုံးပြုမည် ဖြစ်ပါသည်။ စာရေးသူ အနေဖြင့် OnlineTodo ဟု ပေးခဲ့ပါသည်။

OnlineTodo database သည် cross ပုံစံဖြင့် active မဖြစ်သေးတာကို မြင်ရပါမည်။

2.အနေဖြင့် Setting.py ထဲသို့ သွားပြီး DATABASES object ကို အောက်ပါ အတိုင်း သွားပြင်ပါမည်။

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'OnlineTodo',
        'USER': 'postgres',
        'PASSWORD':'toor',
        'HOST':'localhost'
    }

}
```

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'OnlineTodo',
        'USER': 'postgres',
        'PASSWORD':'toor',
        'HOST':'localhost'
    }
}
```

3. အနေဖြင့် migration ပြုလုပ်ရန် မိမိတို့ app ကို settings.py ထဲမှ INSTALLED_APPS ထဲတွင် ထည့်ပေးရမည်။ ထို့နောက် setting.py ထဲသို့သွားပြီး အောက်ပါ အတိုင်းရေးပါ။

```
31  # Application definition
32
33  INSTALLED_APPS = [
34      'django.contrib.admin',
35      'django.contrib.auth',
36      'django.contrib.contenttypes',
37      'django.contrib.sessions',
38      'django.contrib.messages',
39      'django.contrib.staticfiles',
40      'todoapp'
41  ]
```

အထက်ပါအတိုင်းရေးပြီးပါက terminal တွင် migrate စလုပ်နိုင်ပါပြီ။ ထိုသို့ ပြုလုပ်နိုင်ရန် python manage.py migrate ကိုရေးပေးရပါမည်။

```
PS C:\Users\winht\djprojects\OnlineToDo\OnlineTodo> python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying sessions.0001_initial... OK
PS C:\Users\winht\djprojects\OnlineToDo\OnlineTodo>
```

အထက်ပါ အတိုင်း ပေါ်လာပါက ပထမ အဆင့် migrate လုပ်ခြင်း ပြီးပါပြီ။ စစ်ဆေးရန် အတွက် postgreSQL သို့ ပြန်သွားပြီး အောက်ပါတို့ကို ကြည့်ပါက postgreSQL မှ default လုပ်ပေးသည့် tables များကို မြင်ရပါမည်။

နောက်ဆုံးအနေဖြင့် မိမိတို့ database model အား migration ပြုလုပ်ပေးရန် အတွက် todoapp ထဲမှ models.py ထဲသို့သွားပြီး class တစ်ခု ရေးသားပါမည်။

```
OnlineTodo > todoapp > models.py > ...
1   from django.db import models
2
3   # Create your models here.
4   class Todoapp(models.Model):
5       content = models.TextField()
6
```

အထက်ပါအတိုင်း ပြုလုပ်ပြီးပါက program save ကာ terminal တွင် python manage.py makemigrations todoapp ဟုရေးကာ မိမိတို့ model ကို migration လုပ်ပေးရမည်။ todoapp နေရာတွင် မိမိတို့ app name ကို ထည့်ပေးရပါမည်။

```
PS C:\Users\winht\djprojects\OnlineToDo\OnlineTodo> python manage.py makemigrations todoapp
Migrations for 'todoapp':
  todoapp\migrations\0001_initial.py
    - Create model Todoapp
PS C:\Users\winht\djprojects\OnlineToDo\OnlineTodo>
```

အထက်ပါ အဆင့်ပြီးပါက မိမိတို့ app ထဲတွင် 0001_initial.py ဆိုသည့် python file တစ်ခု ဖြစ်ပေါ်လာပါမည်။

တတိယအဆင့်အနေဖြင့် sqlmigrate လုပ်ရန်အတွက် terminal တွင် မိမိတို့ app name နှင့် file name ကို အောက်ပါ အတိုင်း ရေးပေးရပါ ဦးမည်။

>> python manage.py sqlmigrate todoapp 0001

```
PS C:\Users\winht\djprojects\OnlineToDo\OnlineTodo> python manage.py sqlmigrate todoapp 0001
BEGIN;
--
-- Create model Todoapp
--
CREATE TABLE "todoapp_todoapp" ("id" serial NOT NULL PRIMARY KEY, "content" text NOT NULL);
COMMIT;
PS C:\Users\winht\djprojects\OnlineToDo\OnlineTodo>
```

နောက်ဆုံးအဆင့် အနေဖြင့် python manage.py migrate ကို ရေးပေးရန် လိုအပ်ပါသေးသည်။

>> python manage.py migrate

```
PS C:\Users\winht\djprojects\OnlineToDo\OnlineTodo> python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions, todoapp
Running migrations:
  Applying todoapp.0001_initial... OK
PS C:\Users\winht\djprojects\OnlineToDo\OnlineTodo>
```

အထက်ပါ အဆင့်ပြီးသွားပါက postgreSQL သို့ ပြန်သွားပါ။ မိမိတို့ database မှ တစ်ဆင့် Schemas သို့သွားပြီး Tables ကို right click ထောက်ကာ refresh တစ်ချက်လုပ်ပေးပါ။ ထို့နောက် အောက်ဆုံးတွင် မိမိတို့ todoapp_todoapp ဆိုသည့် table ပေါ်လာသည်ကို မြင်ရပါမည်။

ပေါ်လာသော todoapp table အား right click ထောက်ပြီး View/Edit Data ကို နှိပ်ကာ All Rows ဆိုတာကို နှိပ်လိုက်လျှင် ညာဘက် ၌ မိမိတို့ တည်ဆောက်လိုက်သည့် table ပေါ်လာသည်ကို မြင်ရပါမည်။

အောက်ပါပုံအတိုင်း ပေါ်လာမည် id သည် auto generate လုပ်ပေးမည် ဖြစ်ပြီး content သည် မိမိတို့ models.py ထဲမှ todoapp ထဲတွင် တည်ဆောက်ခဲ့သည့် object name ဖြစ်သည်။

အထက်ပါ အဆင့်များပြီးပါက Migration လုပ်ခြင်း ပြီးပါပြီ။ Database ထဲသို့ CRUD operation များ လုပ်ဆောင် နိုင်ပါပြီ။

Project စတင် ရေးသားရန် အတွက် အောက်ပါအချက်တို့ကို ဆက်လက် လုပ်ဆောင်ရပါမည်။

1. မိမိတို့ application folder(todoapp) ထဲတွင် urls.py ဆိုသည့် python file တစ်ခု တည်ဆောက်မည်။ထို file သည် urls များအား လုံးရဲ့ path များကို ချိတ်ဆက်ပေးရန် ဖြစ်သည်။
2. Project Folder(OnlineTodo) ထဲတွင် ရှိသော urls.py ထဲ၌ မိမိတို့ application folder ထဲမှ urls.py file ရဲ့ path လမ်းကြောင်းကို ထည့်ပေးရန် ဖြစ်သည်။
3. Views.py ထဲတွင် မိမိတို့ ဖော်ပြလိုသည့် အချက်အလက်များကို function တစ်ခု ဖြင့် ဖော်ပြ ပေးရန် ဖြစ်သည်။

1.todoapp application folder ထဲတွင် urls.py ဆိုသည့် python file တစ်ခုကိုအောက်ပါ အတိုင်း တည်ဆောက်ပါ။

2.Project Folder ထဲမှ urls.py တွင် အောက်ပါ အတိုင်း application folder ထဲမှ urls.py အား သိအောင် path ချိတ်ပေးပါမည်။ include ကို import လုပ်ရန် သတိပြုပါ။

```
from django.contrib import admin
from django.urls import path,include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('todo/',include('todoapp.urls'))
]
```

3. Views.py ထဲတွင် အောက်ပါ အတိုင်း todo_list ဆိုသည့် function တစ်ခု သွားရေးပါမည်။ ယခင် အခန်းများတွင်လည်းဖော်ပြ ပြီးဖြစ်တာမို့ ယခု အခန်းတွင် အသေးစိတ်ထပ်ပြီး မဖော်ပြလိုတော့ပါ။

OnlineTodo > todoapp > views.py > ...

```
1  from django.shortcuts import render
2  from django.http import HttpResponse
3
4  # Create your views here.
5  def todo_list(request):
6      return HttpResponse("This is todo_list page")
7
```

နောက်ဆုံးအဆင့်အနေဖြင့် urls.py ထဲတွင် အောက်ပါအတိုင်း views.py မှ todo_list function အား ခေါ်ရန် ဖြစ်ပါသည်။

settings.py | models.py | urls.py | views.py

OnlineTodo > todoapp > urls.py > ...

```
1  from django.urls import path
2  from . import views
3
4  urlpatterns=[
5  path('list/',views.todo_list)
6
7  ]
```

အထက်ပါ အဆင့်များ အားလုံးပြီးပါက file အားလုံး အား save ပြီး browser တွင် အောက်ပါ အတိုင်း run ကြည့်ပါက This is todo_list page ဆိုတာကို မြင်ရပါမည်။

todo သည် project folder ထဲမှ urls.py ထဲတွင် မိမိတို့ ပေးထားခဲ့သည့် url ဖြစ်ပြီး list သည် application folder ထဲ၌ ရှိသော urls.py ရဲ့ url ဖြစ်ပါသည်။ ထို list url အတွက် အလုပ်လုပ်ရန် fuction သည် views.py ထဲမှ ဖြစ်ပါသည်။

Adding template in Application

Users များဆီမှ data များရယူခြင်း ပေးပို့ခြင်းများကို ဆောင်ရွက်နိုင်ရန် HTML file များကို စရေးပါမည်။ ထို html file များရှိသည့် template ကို ဖန်တီးရာတွင် django ရဲ့ pattern အချို့ကို လိုက်နာရပါမည်။

1. Application folder ထဲတွင် templates ဆိုသည့် folder တစ်ခု တည်ဆောက်ပါမည်။
2. Templates folder ထဲတွင် မိမိတို့ application folder နှင့် name တူသည့် folder တစ်ခု တည်ဆောက်ပါမည်။
3. ထို folder ထဲတွင် html file များ စတင်ဖန်တီးပါမည်။

Todo_list.html file ထဲတွင် html code များနှင့် boostrap တို့ကို add လုပ်ရန် todo_list.html ထဲသို့သွားပါ။ ထို့နောက် Abbreviation ကိုသုံးပြီး html format တစ်ခုကို ရေးသားပါမည်။ html file ထဲတွင် ! ကို တစ်ချက်ရေးပြီး tab နှိပ်လိုက်ရုံဖြင့် html format များ ကျလာပါမည်။ (in visual studio code)

အထက်ပါ အတိုင်းပေါ်လာလျှင် tab ကို နှိပ်လိုက်ပါ။ အောက်ပါ အတိုင်းပေါ်လာပါမည်။

စာရေးသူ အနေဖြင့် line 6 မှ Document နေရာတွင် Todos ဟုရေးခဲ့ပါသည်။ ထို့နောက် boostrap ကို add ရန် https://getbootstrap.com/ သို့သွားပြီး Get Started ကိုရွေးပါ။ အောက်ပါ page သို့ရောက်သွားလျှင် ပထမဆုံး CSS မှ copy ဆိုတာကို နှိပ်ခဲ့ပေးပါ။ ပြီးလျှင် todo_list.html file ထဲမှ head tag ထဲတွင် သွားထည့်ပေးပါ။ ထိုနည်းတူ JS ကိုလည်း copy လုပ်ပြီး body tag ထဲတွင် ထည့်ပေးပါ။

နောက်တစ်ဆင့်အနေဖြင့် font - awesome ကို မိမိတို့ stylesheet တွင် ထည့်ရန် https://cdnjs.com/libraries/font-awesome ယခု link သို့သွားပြီး all.min.css ဆိုသည့် link ကို copy လုပ်ရပါမည်။

Copy ကူးလာသော link ကို အောက်ပါပုံထဲကအတိုင်း head tag ထဲတွင် link tag ဖြင့် သွားရောက် add ပေးရပါမည်။ အထက်ပါ အဆင့်များ ပြီးသွားပါက အောက်ပါ ပုံစံ အတိုင်း ဖြစ်နေရပါမည်။

```
3   <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <title>Document</title>
7       <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css" integrity="sha384-9a
8
9       <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.13.1/css/all.min.css">
10  </head>
11  <body>
12
13
14
15      <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MV
16  <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js" integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWl
17  <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js" integrity="sha384-OgVRvuATP1z7JjHLkuOU7Xw7
18  </body>
19  </html>
```

from bootstrap

from bootstrap

from font-awesome

Application Design

ယခု သင်ခန်းစာမှာတော့ bootstrap မှ class များကို သုံးပြီး မိမိတို့ application ကို

design လုအောင် လုပ်သွားမှာပါ။ ယခု Topic သည် django ကိုသာ အဓိကထား သင်ကြားခြင်းကြောင့် စာရေးသူ အနေဖြင့် bootstarp အကြောင်းကို အသေးစိတ်ရှင်းပြမည်မဟုတ်ပါ။ ပထမဆုံး အနေဖြင့် အောက်ပါ အတိုင်း design ပေါ်အောင် အရင် ရေးပါမည်။

ပုံပါ အတိုင်း design ထွက်လာစေရန် အောက်ပါ အတိုင်းရေးရပါမည်။

```
10  </head>
11  <body class="bg-light">
12
13      <div class="container">
14          <div class="row mt-5">
15          <div class="col-md-8 offset-md-2">
16              <div class="card">
17                  <div class="card-header shadow-sm bg-white">
18
19                  </div>
20              </div>
21          </div>
22      </div>
23      </div>
24
25      <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity=
```

- Line 11 တွင် background colour ကို light ဖြစ်စေရန် class="bg-light" ကိုသုံးထားပါသည်။
- Line 13 တွင် container ကို သုံးထားပါသည် container တွင် နှစ်မျိုး ရှိပါသည် .container နှင့် container-fluid တို့ဖြစ်သည်။ .container သည် အပြည့်မပေါ် ပဲ သတ်မှတ်ထားသလို သာ ပေါ်ပေးပြီး .container-fluid သည် page တစ်ခုလုံး 100% အပြည့်ပေါ်ပါသည်။ ထို့ကြောင့် စာရေးသူ အနေဖြင့် screen ပေါ် လိုက်ပြီး responsive ဖြစ်စေရန် container ကို သုံးထားခြင်း ဖြစ်ပါသည်။ လေ့လာသူများ အနေဖြင့် container နေရာတွင် container-fluid ကိုလည်း ထည့်ပြီး စမ်းကြည့်နိုင်ပါသည်။
- Line 14 သည် row သည် row အတွက် ဖြစ်ပြီး m သည် margin ကို ဆိုလိုခြင်း ဖြစ်ကာ t သည် top ကို ဆိုလိုခြင်း ဖြစ်သည်။
- Line 15 မှ md သည် desktops အတွက်ကို ဆိုလိုခြင်း ဖြစ်ပြီး Bootstrap grid system မှာ class လေးခုရှိပါတယ် xs , sm,md , lg xs သည် phones များ အတွက်ဖြစ်ပြီး sm သည် tablets များအတွက် ဖြစ်သည် md သည် desktops များအတွက် ဖြစ်ပြီး lg သည် larger desktops များအတွက် ဖြစ်သည်။ dynamic and flexible ဖြစ်ဖို့အတွက် အထက်ပါ class တွေကို combined လုပ်ပြီး အသုံးပြုနိုင်ပါတယ်။
- Line 16 မှ card တွင် display အတွက် များပြားလှတဲ့ options တွေ ပါဝင်ပါတယ်။ ဥပမာ headers , footers , content , background colour တို့ကို မိမိစိတ်ကြိုက်ပြုပြင်နိုင်သလို အခြားသော Options များလည်း ရှိပါသေးသည်။ နောက်တစ်ဆင့် အနေဖြင့် မိမိတို့ ထည့်လိုသည့် စာသားကို အောက်ပါအတိုင်း code အနည်းငယ် ထပ်ရေးပြီး ထည့်နိုင်ပါသည်။ စာရေးသူ အနေဖြင့် h1 tag ဖြင့် My

Todo App

ဟုရေးထားပြီး i tag ဖြင့် double check လုပ်နိုင်မည့် icon လေးပါ ထည့်ထားပါသည်။

```
13      <div class="container">
14          <div class="row mt-5">
15          <div class="col-md-8 offset-md-2">
16              <div class="card">
17                  <div class="card-header shadow-sm bg-white">
18                      <h1 class="display-5 text-info">
19                          <i class="fas fa-check-double"></i>
20                          MyToDo App
21                      </h1>
22                  </div>
23              </div>
24          </div>
25      </div>
26      </div>
```

အထက်ပါ အတိုင်း ထပ်မံ ရေးသားပြီးသည့်အခါ မိမိတို့ web page ၌ အောက်ပါ အတိုင်း ပေါ်နေသည်ကို မြင်ရပါမည်။ views.py ထဲတွင် အောက်ပါအတိုင်း သွားရောက်ရေးပေးရန် လိုအပ်ပါသေးသည်။

```
OnlineTodo > todoapp > views.py > ...
1   from django.shortcuts import render
2   from django.http import HttpResponse
3
4   # Create your views here.
5   def todo_list(request):
6       return render(request,'todoapp/todo_list.html')
```

နောက်တစ်ဆင့် အနေဖြင့် container class ထဲမှာပဲ user မှ data များ ထည့်ရန် input form တစ်ခုနှင့် ထည့်လိုက်သော data များကို submit လုပ်ရန် submit button ကို calendar plus icon နှင့် ပြုလုပ်ပါမည်။ မိမိတို့ စိတ်ကြိုက် design များ ထပ်ထည့်လိုပါက https://www.w3schools.com/icons/icons_reference.asp ယခု Link တွင် သွားရောက်လေ့လာ နိုင်ပါသည်။program ကိုအောက်တွင် ဖော်ပြထားပါသည်။ program run ပြီးသော အခါ ထွက်လာမည့် output ကိုပါ တစ်ခါတည်း ပြထားပါသည်။

```
19                <i class="fas fa-check-double"></i>
20                MyToDo App
21            </h1>
22        </div>
23    <div class="card-body">
24        <form action="" autocomplete="off"">
25            <div class="input-group">
26                <input type="text" name="content" class="from-control">
27                <div class="input-group-append text-info">
28                    <span class="input-group-text bg-white py-0">
29                        <button type="submit" class="btn btn-sm text-info">
30                            <i class="fas fa-calendar-plus fa-lg"></i>
31                        </button>
32                    </span>
33
34                </div>
35            </div>
36        </form>
37    </div>
```

User ထည့်ပေးလိုက်သော data များကို ရယူ နိုင်ရန် အတွက် form action တွင် jinja template ကိုသုံးပြီး function တစ်ခုကို သွားခေါ်ပါမည်။ ထို့နောက် form ရဲ့ method ကိုလည်း post ဟု ပေးမည် ဖြစ်ပါသည်။ ထို့ပြင် django form တိုင်း တွင် **မသာသော နေရာများမှ** post လုပ်ခြင်းကို ကာကွယ်ရန် csrf_token ကိုလည်း ထည့်ထားပါမည်။

```
        </div>
    <div class="card-body">
        <form action="{% url 'insert_todo'%}" method="post" autocomplete="off"">
            {% csrf_token %}
            <div class="input-group">
                <input type="text" name="content" class="from-control">
                <div class="input-group-append text-info">
                    <span class="input-group-text bg-white py-0">
                        <button type="submit" class="btn btn-sm text-info">
                            <i class="fas fa-calendar-plus fa-lg"></i>
                        </button>
                    </span>
```

Adding Data To PostgreSQL

အထက်ပါ အဆင့်ပြီးပါက urls.py ထဲတွင် route(path) တစ်ခု သွားဆောက်ပေးရပါမည်။ ပြီးလျှင် ထို route အတွက် function ကို views.py ထဲတွင် သွားရောက် ရေးပေးရပါမည်။

1. ပထမ အဆင့်အနေဖြင့် urls.py ထဲတွင် insert_todo ဖြင့် path တစ်ခုသွားဆောက်ပါမည်။
2. Views.py ထဲတွင် insert_todo အတွက် function တစ်ခု ရေးပေးရပါမည်။
3. Insert_todo function သည် form မှ ရလာသော data များကို object တစ်ခု တည်ဆောက်ပြီး PostgreSQL ထဲသို့ ထည့်ပေးပါမည်။
4. ထို့နောက် မူရင်း todo/list/ url ဆီသို့ redirect method ကိုသုံးပြီး ပြန်ပို့ပေးလိုက်ပါမည်။

1. Urls.py ထဲတွင် အောက်ပါ အတိုင်း သွားရေးပါမည်။

OnlineTodo > todoapp > urls.py > ...

```
1   from django.urls import path
2   from . import views
3
4   urlpatterns=[
5   path('list/',views.todo_list),
6   path('insert_todo/',views.insert_todo,name='insert_todo')
7
8   ]
```

from views.py

for jinja template

route

2. Views.py ထဲတွင် insert_todo function ကို အောက်ပါအတိုင်း သွားရေးပါမည်။ ပထမဆုံး အနေဖြင့် HttpRequest နှင့် redirect method တို့ကို import လုပ်ပေးရန် လိုအပ်သလို models.py ထဲမှ Todoapp ဆိုသည့် class ကိုလည်း import လုပ် ပေးရန် လိုအပ်ပါသည်။ ထို့ကြောင့် views.py ထဲတွင် HttpRequest , redirect , Todoapp စသည့် သုံးခုကို ပထမ ဦးဆုံး import လုပ်ရပါမည်။

OnlineTodo > todoapp > views.py > ...

```
1   from django.shortcuts import render,redirect
2   from django.http import HttpResponse,HttpRequest
3   from .models import Todoapp
```

Form မှ name သည် content ဖြစ်သလို method သည်လည်း post method ဖြစ်သည့်အတွက် request.POST['content'] ကို ရေးထားခြင်း ဖြစ်သည်။ ရလာသော data များကို content ဆိုသည့် object ထဲသို့ ထည့်ထားပါသည်။ ထို object အား Todoapp ဆိုသည့် class ထဲသို့ pass လုပ်လိုက်ပါသည်။ Todoapp class သည် database model အတွက် class ဖြစ်သည့်အတွက် todo object သည် database နှင့် တိုက်ရိုက် သက်ဆိုင်နေတော့မည်။ ထို့ကြောင့် todo.save() ဟူ၍ save method ကို သုံးလိုက်သည့် အခါ data များကို database ပေါ်တွင် သွားရောက် save ပေးမည် ဖြစ်ပါသည်။

```
def insert_todo(request:HttpRequest):
    todo = Todoapp(content = request.POST['content'])
    todo.save()
    return redirect('/todo/list')
```

အထက်ပါ အဆင့်များပြီးပါက program အားလုံးကို save ပြီး ပြန်run ကာ အောက်ပါ အတိုင်း data ထည့်ပြီး calendar ပုံလေးကို နှိပ်လိုက်ပါက မိမိတို့ ထည့်ပေးလိုက်သည့် data များကို database ပေါ်တွင် သွားရောက် သိမ်းပေးနေသည်ကို တွေ့ရပါမည်။

Online Todo Database ထဲမှ သို့သွားပြီး ထိုမှတစ်ဆင့် todapp_todoapp သို့ သွားကာ right click ထောက်ပြီး view/edit data >> All rows သို့ အောက်ပါအတိုင်း ဖွင့်ကြည့်နိုင်ပါသည်။

မိမိတို့ database ထဲတွင် ထည့်လိုက်သော dddd ဆိုသည့် data များရှိနေသည်ကို အောက်ပါ အတိုင်း မြင်ရပါမည်။

Fetching Data From PostgreSQL

နောက်တစ်ဆင့် အနေဖြင့် postgreSQL မှ data များ ထုတ်ပြီး မိမိတို့ list page တွင် ပြန်လည် ဖော်ပြ ပေးပါမည်။

1. Views.py ထဲမှ todo_list function ထဲတွင် postgreSQL မှ data များ ရယူပြီး todo_list.html ထဲသို့ data များ ပြန်ပို့ပေးပါမည်။
2. Todo_list.html ထဲတွင် jinja template ဖြင့် for loop ကို သုံးကာ data များ ပြန်လည် ဖော်ပြ ပေးပါမည်။

1.todo_list function ထဲတွင် အောက်ပါ အတိုင်းသွားရေးပါမည်။

```
# Create your views here.
def todo_list(request):

    all_data = {'todo_list': Todoapp.objects.all() }
    return render(request,'todoapp/todo_list.html',all_data)

```

Todoapp model ထဲမှ data အားလုံး ရယူလိုသည့် အတွက် Todoapp.objects.all() ကို ရေးထားခြင်း ဖြစ်ပြီး data များ ပို့ပေးရန် အတွက် dictionary ပုံစံဖြင့် ရေးထားပါသည်။

2. ရလာသော data အားလုံးကို ပြန်ထုတ်ပြရန် အတွက် jinja for loop ကိုသုံးပြီး ထုတ်လိုက်ပါသည်။ အစဉ်လိုက်ဖော်ပြနိုင်ရန်အတွက် jinja for loop နှင့် မူရင်းရှိသော form အား ul tag ထဲတွင် ထည့်လိုက်ပါသည်။ ထို့ပြင် input form ကိုလည်း list ပုံစံဖြင့် ပြပေးနိုင်ရန် li tag ထဲသို့ ထည့်လိုက်ပါသည်။ အောက်ပါ ပုံထဲတွင် အသေးစိတ် ဖော်ပြထားပါသည်။

```
20                  MyToDo App
21                </h1>
22              </div>
23          <div class="card-body">
24              <ul class="list-group">
25              <li class="list-group-item">
26              <form action="{% url 'insert_todo' %}" method="post" autocomplete="off"">
27                  {% csrf_token %}
28                  <div class="input-group">
29                      <input type="text" name="content" class="from-control">
30                      <div class="input-group-append text-info">
31                          <span class="input-group-text bg-white py-0">
32                              <button type="submit" class="btn btn-sm text-info">
33                                  <i class="fas fa-calendar-plus fa-lg"></i>
34                              </button>
35                          </span>
36
37                      </div>
38                  </div>
39              </form>
40          </li>
41              {% for todo in todo_list%}
42              <li class="list-group-item">{{todo.content}}</li>
43              {% endfor %}
44          </ul>
45          </div>
46
```

အထက်ပါ ပုံထဲမှ line 29 from-control နေရာတွင် form control ဟုပြင်ပေးပါ။ ထိုသို့ ပြင်လိုက်လျှင် input form သည် ဘေးထပ် အပြည့် ပေါ်ပေးမှာ ဖြစ်ပါတယ်။ အဆင့်များ အားလုံး ပြီးပါက program အားလုံးအား save ပြီး မိမိ ထည့်လိုသည့် data များ ထည့်ကြည့်လျှင် အောက်ပါ အတိုင်း list အားလုံးကို ပြန်ပြပေးနေသည်ကို မြင်ရပါမည်။

ယခု အဆင့်ထိ program များအားလုံးကို အောက်တွင် ဖော်ပြထားပါသည်။ error များတက်ခဲ့ပါက အောက်ပါ code များနှင့် နှိုင်းယှဉ် စစ်ဆေးကြည့် နိုင်ပါသည်။

Views.py

```
from django.shortcuts import render,redirect
from django.http import HttpResponse,HttpRequest
from .models import Todoapp
```

```
# Create your views here.
def todo_list(request):

    all_data = {'todo_list': Todoapp.objects.all() }
    return render(request,'todoapp/todo_list.html',all_data)

def insert_todo(request:HttpRequest):

    todo = Todoapp(content = request.POST['content'])
    todo.save()
    return redirect('/todo/list')
```

Urls.py

```
from django.urls import path
from . import views

urlpatterns=[
path('list/',views.todo_list),
path('insert_todo/',views.insert_todo,name='insert_todo')

]
```

Todo_list.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css
"
integrity="sha384-9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKG
j7Sk" crossorigin="anonymous">
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.13.1/css/all.min.c
ss">
</head>
<body class="bg-light">

    <div class="container">
        <div class="row mt-5">
        <div class="col-md-8 offset-md-2">
            <div class="card">
                <div class="card-header shadow-sm bg-white">
                    <h1 class="display-5 text-info">
                        <i class="fas fa-check-double"></i>
                        MyToDo App
                    </h1>
```

```
                </div>
            <div class="card-body">
                <ul class="list-group">
                    <li class="list-group-item">
                        <form action="{% url 'insert_todo' %}" method="post"
autocomplete="off"">
                    {% csrf_token %}
                    <div class="input-group">
                                        <input type="text" name="content"
class="form-control">
                        <div class="input-group-append text-info">
                            <span class="input-group-text bg-white py-0">
                                        <button type="submit" class="btn btn-sm
text-info">
                                <i class="fas fa-calendar-plus fa-lg"></i>
                            </button>
                            </span>

                        </div>
                    </div>
                </form>
            </li>
                {% for todo in todo_list%}
                <li class="list-group-item">{{todo.content}}</li>
                {% endfor %}
            </ul>
            </div>

            </div>
        </div>
    </div>
    </div>

              <script   src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXa
Rkfj" crossorigin="anonymous"></script>
<script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMf
ooAo" crossorigin="anonymous"></script>
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"
integrity="sha384-OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR
0JKI" crossorigin="anonymous"></script>
</body>
</html>
```

မိမိတို့ web page UI တွင် ဖော်ပြပေးလာသော data များကို ပြန်ဖျက်ရန် အတွက်

button ခလုတ် တစ်ခု ထပ်ထည့်ပေးပါမည်။ ထို button အား နှိပ်လိုက်လျှင် ထို todo_list data များ စာရင်းမှ ပျောက်သွားမှာ ဖြစ်ပါတယ်။

Delete Items

Todo စာသား တစ်ခုချင်းစီတိုင်းဘေးတွင် delete button များကို ထည့်ပါမည်။ ထို့ပြင် item များမရှိပါက items များ မရှိကြောင်းကိုလည်း စာသား တစ်ခုဖြင့်ဖော်ပြပေးပါမည်။ ထိုသို့ ထည့်ရန် အတွက် todo_list.html ထဲမှ for loop ထဲတွင် အောက်ပါ အတိုင်းရေးပါမည်။

```
41                  {% for todo in todo_list%}
42                  <li class="list-group-item">{{todo.content}}</li>
43                  {% empty %}
44                  <li class="list-group-item">
45                      <span class="font-italic">No items todo.</span>
46                  </li>
47                  {% endfor %}
48              </ul>
49              </div>
```

Output အနေဖြင့် အောက်ပါ အတိုင်းထွက်လာပါမည်။

အထက်ပါ အတိုင်း ရေးလိုက်လျှင် todo_list ထဲ၌ data များ မရှိပါက No items todo ဆိုတဲ့ စာသားကို ဖော်ပြပေးမှာ ဖြစ်ပါတယ်။ နောက်တစ်ဆင့်အနေဖြင့် data များဘေးတွင် delete button တစ်ခုထားရန် အောက်ပါ အတိုင်းရေးပါမည်။

```
38                  </div>
39              </form>
40          </li>
41              {% for todo in todo_list%}
42              <li class="list-group-item">{{todo.content}}
43                  <form action="{% url 'delete_todo_item' todo.id%}" method="post" class="float-right d-inline">
44                      {% csrf_token %}
45                      <button type="submit" class="btn">
46                          <i class="far fa-trash-alt fa-log text-danger float-right"></i>
47                      </button>
48                  </form>
49              </li>
50              {% empty %}
51              <li class="list-group-item">
52                  <span class="font-italic">No items todo.</span>
53              </li>
54              {% endfor %}
55          </ul>
```

Action မှ အလုပ်လုပ်မည့် function name အား delete_todo_item ဟုပေးခဲ့ပါသည်။ ထို့ကြောင့် views.py ထဲတွင် ထို function ကို အောက်ပါ အတိုင်း သွားရောက် ရေးသားပါမည်။

```
def delete_todo_item(request,todo_id):
    todo_delete = Todoapp.objects.get(id=todo_id)
    todo_delete.delete()
    return redirect('/todo/list')
```

Views.py ထဲမှာ function ရေးပြီး ဖြစ်သည့်အတွက် urls.py ထဲတွင် ထို path အား ထည့်ပေးရန် လိုအပ်ပါသေးသည်။ အောက်ပါ အတိုင်းရေးရပါမည်။ ထိုသို့ ရေးသားရာတွင် int:todo_id ကိုပါ ထည့်ရေးပေးရပါမည်။ အဘယ်ကြောင့်ဆိုသော် delete button ကို နှိပ်လိုက်သည့် နှင့် တစ်ပြိုင်နက် delete_todo_item function ကို သွားခေါ်ရာတွင် တစ်ပါတည်း ထို data ရဲ့ id ကို ပါ ရယူ နိုင်ရန် ဖြစ်သည်။

```
OnlineTodo > todoapp > urls.py > ...
from django.urls import path
from . import views

urlpatterns=[
path('list/',views.todo_list),
path('insert_todo/',views.insert_todo,name='insert_todo'),
path('delete/<int:todo_id>',views.delete_todo_item,name='delete_todo_item')

]
```

အထက်ပါ အဆင့်များပြီးပါက file အားလုံးကို save ပြီး run ပါ။ ထို့နောက် data ထည့်ခြင်း ဖျက်ခြင်းများကို ဆောင်ရွက်နိုင်ပါပြီ။

Practical RESTful API

ယခုအခန်းမှာတော့ Django နဲ့ REST APIတစ်ခုရေးသားသွားမှာပါ။ REST(representational state transfer) လို့ခေါ်သလို web service လို့လည်း ခေါ်ပါတယ်။ ယနေ့ ခေတ်မှာတော့ အဓိက အနေနဲ့ web services တွေကို develop လုပ်တဲ့ နေရာမှာ သုံးပါတယ်။ ယခင် ကာလတွေမှာတော့ SOAP(Simple Object Access Protocol) ကို web services တွေမှာ သုံးကြပြီး 2000 ခုနှစ်နောက်ပိုင်းမှ စတင်ပြီး REST API ကို အသုံးပြုလာရခြင်း အကြောင်းတွေထဲက တစ်ခုကတော့ bandwidth အရမ်းနည်းလို့ပါ။ ယခု အခန်းမှာ local အတွက် Django REST ကိုလည်း သုံးမှာ ဖြစ်သလို Postman ကိုလည်း အသုံးပြုသွားမှာပါ။

1. ပထမဆုံး အနေဖြင့် djangoenv ဆိုသည့် virtual environment တစ်ခု ဖန်တီးပါမည်
2. Django ကို version ဖြင့် အတိအကျ install လုပ်ပါမည်။

   Django ကို version အတိအကျ ဖြင့် install လုပ်ရခြင်းမှာ မိမိတို့ Project တွင် version နှင့်ပတ်သက်သော ပြဿနာများကိုရှောင်ရှားနိုင်ရန်နှင့် application တစ်ခုကို စုပေါင်းရေးသည့် အခါ version အတူတူရေးနိုင်သလို server များပေါ်တွင် တင်သည့် အချိန်တွင်လည်း version ပြဿနာများကို ပြေလည် စေရန် ဖြစ်သည်။

   စာရေးသူ အနေဖြင့် django version 3.0.7 ကို အသုံးပြုထားပါသည်။ django ကို version ဖြင့် install လုပ်နိုင်ရန် pip install django==3.0.7 စသဖြင့် ရေးပြီး ပြုလုပ်နိုင်ပါသည်။ ထို့ပြင် မိမိတို့ ယခု လက်ရှိ အသုံးပြုနေသည့် django version ကို သိလိုသောအခါတွင်လည်း python -m django --version စသည့် command ကို အသုံးပြုပြီး သိနိုင်ပါသည်။

```
C:\Users\winht>python -m django --version
3.0.7

C:\Users\winht>
```

```
C:\Users\winht\djprojects\REST_API>mkvirtualenv djangoenv
Using base prefix 'c:\\users\\winht\\appdata\\local\\programs\\python\\python38-32'
New python executable in C:\Users\winht\Envs\djangoenv\Scripts\python.exe
Installing setuptools, pip, wheel...
done.

(djangoenv) C:\Users\winht\djprojects\REST_API>workon djangoenv
(djangoenv) C:\Users\winht\djprojects\REST_API>pip install django==3.0.7
Collecting django==3.0.7
  Using cached Django-3.0.7-py3-none-any.whl (7.5 MB)
Collecting pytz
  Using cached pytz-2020.1-py2.py3-none-any.whl (510 kB)
Collecting sqlparse>=0.2.2
  Using cached sqlparse-0.3.1-py2.py3-none-any.whl (40 kB)
Collecting asgiref~=3.2
  Using cached asgiref-3.2.10-py3-none-any.whl (19 kB)
Installing collected packages: pytz, sqlparse, asgiref, django
Successfully installed asgiref-3.2.10 django-3.0.7 pytz-2020.1 sqlparse-0.3.1

(djangoenv) C:\Users\winht\djprojects\REST_API>django-admin startproject djapp
```

စာရေးသူ အနေဖြင့် djangoenv ဆိုသည့် virtual environment တစ်ခုကို တည်ဆောက်လိုက်ပြီး ထို ထဲတွင် djapp ဆိုသည့် project တစ်ခုကို တည်ဆောက်ထားပါသည်။ ထို့အတူ infoAPI ဆိုသည့် application တစ်ခုကိုလည်း တည်ဆောက်ထားပါသည်။

```
(djangoenv) C:\Users\winht\djprojects\REST_API\djapp>python manage.py startapp infoAPI

(djangoenv) C:\Users\winht\djprojects\REST_API\djapp>dir
 Volume in drive C has no label.
 Volume Serial Number is AA6D-23FE

 Directory of C:\Users\winht\djprojects\REST_API\djapp

07/02/2020  10:42 PM    <DIR>          .
07/02/2020  10:42 PM    <DIR>          ..
07/02/2020  10:42 PM    <DIR>          djapp
07/02/2020  10:42 PM    <DIR>          infoAPI
07/02/2020  10:18 PM               646 manage.py
               1 File(s)            646 bytes
               4 Dir(s)  107,268,866,048 bytes free
```

Django REST Framework

ယခု သင်ခန်းစာမှာ Django REST framework ကိုသုံးမှာ ဖြစ်တဲ့အတွက် djangorestframework ဆိုတဲ့ package ကို Install လုပ်ပေးရန် လိုအပ်ပါသည်။ ထို့ပြင် postgreSQL ကိုလည်းသုံးမှာ ဖြစ်တဲ့အတွက် psycopg2 ကိုလည်း install လုပ်ပေးရန် လိုအပ်ပါသည်။

>> pip install djangorestframework

>>pip install psycopg2

```
(djangoenv) C:\Users\winht\djprojects\REST_API>pip install djangorestframework
Collecting djangorestframework
  Downloading djangorestframework-3.11.0-py3-none-any.whl (911 kB)
     |████████████████████████████████| 911 kB 409 kB/s
Requirement already satisfied: django>=1.11 in c:\users\winht\envs\djangoenv\lib\site-packages (from dja
) (3.0.7)
Requirement already satisfied: asgiref~=3.2 in c:\users\winht\envs\djangoenv\lib\site-packages (from dja
orestframework) (3.2.10)
Requirement already satisfied: pytz in c:\users\winht\envs\djangoenv\lib\site-packages (from django>=1.1
mework) (2020.1)
Requirement already satisfied: sqlparse>=0.2.2 in c:\users\winht\envs\djangoenv\lib\site-packages (from
angorestframework) (0.3.1)
Installing collected packages: djangorestframework
Successfully installed djangorestframework-3.11.0

(djangoenv) C:\Users\winht\djprojects\REST_API>pip install psycopg2
Collecting psycopg2
  Downloading psycopg2-2.8.5-cp38-cp38-win32.whl (986 kB)
     |████████████████████████████████| 986 kB 409 kB/s
Installing collected packages: psycopg2
Successfully installed psycopg2-2.8.5

(djangoenv) C:\Users\winht\djprojects\REST_API>
```

အထက်ပါအတိုင်း install လုပ်ပြီးပါက settings.py ထဲမှ DATABASES ကို postgresql ထည့်ပေးရန် လိုအပ်သလို application မှာ infoAPI ကိုထည့်ပေးရန် လိုအပ်ပါသည်။ rest framework ကို install လုပ်သည့်အတွက် rest_framework ကိုလည်း ထည့်ပေးထားပါသည်။ DATABSES ထဲသို့ database name ထည့်ရန် လွန်ခဲ့သည့် သင်ခန်းစာမှာလို database တစ်ခုသွားဆောက်ပေးရန် လိုအပ်ပါသည်။ ထို့ကြောင့် postgresql pgadmin4 သို့ သွား၍ infoAPI ဆိုသည့် database တစ်ခု တည်ဆောက်ပါမည်။

infoAPI Database

Databases ထဲတွင်လည်း အောက်ပါ အတိုင်း ပြင်ရပါမည်။

```
76
77  DATABASES = {
78      'default': {
79          'ENGINE': 'django.db.backends.postgresql',
80          'NAME': 'infoAPI',
81          'USER':'postgres',
82          'PASSWORD':'toor',
83          'HOST':'localhost'
84      }
85  }
```

INSTALLED_APPS ထဲတွင်အောက်ပါ ပုံ အတိုင်းရေးထားပါသည်။

```
33  INSTALLED_APPS = [
34      'django.contrib.admin',
35      'django.contrib.auth',
36      'django.contrib.contenttypes',
37      'django.contrib.sessions',
38      'django.contrib.messages',
39      'django.contrib.staticfiles',
40      'infoAPI',
41      'rest_framework',
42  ]
```

Migration

Database migration ပြုလုပ်ရန်အတွက် ယခင် သင်ခန်းစာအတိုင်းပင် models.py ထဲ၌ info ဆိုသည့် class တစ်ခုတည်ဆောက် ပေးပါမည်။ ထို class ထဲတွင် name , email , phone number ,degree စသည့်တို့ကို အောက်ပါ အတိုင်း ရေးထားပါမည်။

1. ပထမဆုံး အနေဖြင့် အောက်ပါ command ကိုသုံးပြီး ယခင်သင်ခန်းစာ အတိုင်းပင် migration စလုပ်ပါမည်။
   >> python manage.py migrate
2. Models.py ထဲတွင် class သွားဆောက်ပါမည်။
3. infoAPI ဆိုသည့် app အား migration ပြုလုပ် ပေးပါမည်။
4. နောက်တစ်ကြိမ် migrate ထပ်မံ ပြုလုပ်ရပါမည်။

1. Migrate

```
(djangoenv) C:\Users\winht\djprojects\REST_API\djapp>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
```

2. Class

```
djapp > infoAPI > models.py > ...
1  from django.db import models
2
3  # Create your models here.
4  class Info(models.Model):
5      name = models.CharField(max_length=50)
6      email=models.CharField(max_length=50)
7      phnumber=models.CharField(max_length=15)
8      degree = models.CharField(max_length=20)
9
```

3.app migration

```
(djangoenv) C:\Users\winht\djprojects\REST_API\djapp>python manage.py makemigrations infoAPI
Migrations for 'infoAPI':
  infoAPI\migrations\0001_initial.py
    - Create model Info
```

4.last migrate

```
(djangoenv) C:\Users\winht\djprojects\REST_API\djapp>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, infoAPI, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
```

အထက်ပါ အဆင့်များပြီးပါက မိမိတို့ postgresql ထဲမှ tables များထဲတွင် infoAPI_info ဆိုသည့် table ကိုမြင်ရမှာ ဖြစ်သလို ထို table ထဲတွင်လည်း အောက်ပါ အတိုင်း column များရှိနေသည်ကို မြင်ရပါမည်။

Data Output Explain Messages Notifications

id [PK] integer	name character varying (50)	email character varying (50)	phnumber character varying (15)	degree character varying (20)

Serializers

ရှုပ်ထွေးတဲ့ data format တွေ querysets တွေနဲ့ models တွေကို native python datatypes များသို့ ပြောင်းရာတွင် serialzers ကိုသုံးပါတယ်။ ယခင် JSON formatသင်ခန်းစာများတွင်လည်း ယခု serialization အကြောင်းကို ပြောခဲ့ပြီးဖြစ်ပါတယ်။ ထို့ပြင် python ရဲ့ datatypes တွေမှ တစ်ဆင့် အခြားသော complex ဖြစ်တဲ့ data format တွေကိုပြောင်းဖို့ အတွက်လည်း deserialization ကိုလည်း သုံးနိုင်ပါသေးတယ်။

ပထမဆုံးအနေဖြင့် မိမိတို့ infoAPI ထဲတွင် serializers.py ဆိုသည့် python file တစ်ခု တည်ဆောက်ပါမည်။ထို file ထဲတွင် infoSerializer ဆိုသည့် class တစ်ခု တည်ဆောက်ပါမည်။ ထို class ထဲတွင် models.py ထဲမှ Info ဆိုသည့် class နှင့် ထို class ထဲမှ fields or column အားလုံး ယူပါမည်။ rest_framework ထဲမှ serializers ကို သုံးမှာ ဖြစ်တဲ့ အတွက် import လုပ်ပေးရန်လည်း လိုအပ်ပါသည်။

djapp > infoAPI > serializers.py > ...

```
1  from rest_framework import serializers
2  from .models import Info
3
4  class InfoSerializer(serializers.ModelSerializer):
5      class Meta:
6          model = Info
7          fields = '__all__'
8
```

Viewsets

Django REST framework သည် Viewset ဆိုသည့် class တစ်ခုတည်းတွင် data များ အားလုံး ရယူခြင်း list() , id ဖြင့် သီးသန့် ရယူခြင်း retrieve() , အသစ် ဖန်တီးခြင်း create() , ရှိပြီးသားများ ကို ထပ်မံပြုပြင်ခြင်း update() , ဖျက်ခြင်း destroy() စသည့် method များ viewset class မှ support လုပ်ပေးပါသည်။ ထို့ကြောင့် ပထမဆုံး infoViewset ဆိုသည့် class တစ်ခု ဆောက်ပေးမည်။ ထို class ထဲတွင် data များ အားလုံးရယူပြီး queryset ထဲသို့ ထည့်ပေးရမည်။ queryset ကို queryset ဟုသာ အတိအကျ ရေးပေးရမည်။ ထို့နောက် serialization and deserializingလုပ်ရန်အတွက်serializer_classကြေငြာခြင်းတို့ကို

အောက်ပါအတိုင်းရေးပေးရမည်။ infoAPI application folder ထဲတွင် viewsets.py ကို အရင်ဆုံး ဆောက်ပေးရန် လိုအပ်ပါသည်။ မြားနှင့်ပြထားသော အရာများသည် အရေးကြီးသည့် အတွက် အတိအကျ ရေးပေးရပါမည်။

```
djapp > infoAPI > viewsets.py > ...
1   from rest_framework import viewsets
2   from .import models
3   from . import serializers
4
5   class InfoViewset(viewsets.ModelViewSet):
6       queryset = models.Info.objects.all()
7       serializer_class = serializers.InfoSerializer
8
9
```

Routers and DefaultRouter

Django REST framework မှာ routers လည်းပါဝင်ပါတယ်။ routers ကို သုံးခြင်းဖြင့် (post ,get ,etc) တို့အတွက် route တွေ အများကြီးရေးစရာမလိုပဲ router ထဲမှ defaultRouter ကိုသုံးခြင်းဖြင့် single line of code နဲ့ တင်ပြီးပါတယ်။ ထို့နောက် router ထဲမှ register ဆိုသည့် method ကိုအသုံးပြုပေးရပါမည်။ ထို method သည် parameter နှစ်ခု ယူပြီး ပထမ တစ်ခုသည် prefix and viewset ဖြစ်ပါသည်။ prefix သည် မိမိတို့ url တွင် ပေါ်လိုသည့် url address ဖြစ်ပြီး viewset သည် viewset class ဖြစ်သည်။ project folder ဖြစ်သည့် djapp ထဲတွင် routers.py ဆိုသည့် python file ကို create လုပ်ပေးရပါမည်။သတိပြုရန် အချက်မှာ infoAPI ဆိုသည့် application folder ထဲတွင် မဟုတ်ပါ။

```
djapp > djapp > router.py > ...
1   from infoAPI.viewsets import InfoViewset
2   from rest_framework import routers
3
4   router = routers.DefaultRouter()
5   router.register('info',InfoViewset)
6
```

အထက်ပါ အဆင့်များပြီးပါက project folder (djapp) ထဲမှ Urls.py ထဲတွင် path ထည့်ပေးရန်လို အပ်ပါသေးသည်။

```
16  from django.contrib import admin
17  from django.urls import path,include
18  from .router import router
19  urlpatterns = [
20      path('admin/', admin.site.urls),
21      path('api/',include(router.urls))
22  ]
```

အထက်ပါ အဆင့်များအားလုံး ပြီးပါက file အားလုံးကို save ပြီး server ကို runပါ url တွင် အောက်ပါ အတိုင်း သွားပေးရပါမည်။

http://127.0.0.1:8000/api/info/

စာရေးသူသည် id:1 ကို စမ်းထားသည့် အတွက် data အချို့ ပေါ်နေခြင်း ဖြစ်ပါသည်။ စာဖတ်သူများ အနေဖြင့်တော့ တွေ့ရမည်မဟုတ်ပါ။ အောက်ပါ ပုံထဲမှ အတိုင်း မိမိတို့ ထည့်လိုသည့် data များထည့်ကာ post ကို နှိပ်လိုက်ပါက မိမိတို့ ထည့်လိုက်သည့် data များ ပေါ်လာသည်ကို တွေ့ရပါမည်။

Data များကို Database ထဲတွင်လည်း သွားရောက် စစ်ဆေး နိုင်ပါသည်။ အောက်ပါ အတိုင်း Data များ မြင်နေရပါက REST API တစ်ခု တည်ဆောက်ခြင်း အောင်မြင်ပါပြီ။

Data Output Explain Messages Notifications

	id [PK] integer	name character varying (50)	email character varying (50)	phnumber character varying (15)	degree character varying (20)
1	1	WinHtut	winhtutonline@gmail.com	09453741976	LLB
2	2	MgMgkaung	mgmgkaung@gmail.com	09453741976	GHI

Postman

မိမိတို့ API အား အခြားသော application , web service များမှ ဆက်သွယ်လိုသောအခါတွင် အဆင်ပြေ မပြေကို သိနိုင်ရန် Postman ကို သုံးနိုင်ပါတယ်။ Postman ကို အောက်ပါ link မှာ download ရယူ နိုင်ပါတယ်။

https://www.postman.com/downloads/

Postman ကို installation ပြီးပါက အောက်ပါအတိုင်း (+) new ကို နှိပ်ပါ။

ထို့နောက် ဘေးတွေ GET ဖြင့် url ထည့်ရန် အောက်ပါ အတိုင်းပေါ်လာပါမည်။

Enter request URL နေရာတွင် url ထည့်ရန် မိမိတို့ web page မှာ အောက်ပါ url အားသွားရောက် copy ကူးပါ။

ထို့နောက် မိမိတို့ postman url နေရာတွင် copy ကူးလာသော Url အားထည့်ပါ။ ပြီးလျှင် send ကိုနှိပ် ပါ။ ပုံထဲက အတိုင်း data များ ရလာသည်ကို မြင်ရပါမည်။ အထက်ပါ အဆင့်ပြီးလျှင် မိမိတို့ server မှ data ရယူခြင်း အောင်မြင်ပါပြီ။ Status မှာလည်း 200OK ဆိုသည့် စာသား ကို တွေ့မြင်ရပါမည်။ 200 သည် success ဖြစ်ကြောင်း ဆိုလိုခြင်း ဖြစ်သည်။ အကယ်၍ 400 ပေါ်လာပါက bad syntax ဖြစ်ပြီး request လုပ်ခြင်း မအောင်မြင်ကြောင်းကို ပြခြင်း ဖြစ်သည်။ 404 ဆိုရင်တော့ server not found error ကိုဆိုလိုခြင်း ဖြစ်ပြီး မိမိတို့ ထည့်ပေးလိုက်သည့် url နှင့် မည်သည့် server မှ မရှိကြောင်းကို ဆိုလိုခြင်း ဖြစ်သည်။

နောက်တစ်ဆင့် အနေဖြင့် POST request ကို သုံးပြီး data များ ပို့ပေးပါမည်။ ထို့ကြောင့် မိမိတို့ postman မှ Url အား right click နှိပ်ပြီး Duplicate Tab ကို နှိပ်ပါ။

1. **အသစ်ပေါ်လာသော နေရာတွင် GET နေရာတွင် POST ကို ပြောင်းပါ။**
2. **ထို့နောက် Headers ကို တစ်ချက် နှိပ်ပါ ၊**
3. **KEY အောက်က နေရာတွင် Content-Type ဟုရေးပါ**
4. **VALUE အောက်က နေရာတွင် application/json ဟုရေးပါ**
5. **Body ကို တစ်ချက် နှိပ်ပါ**
6. **ထို့နောက် raw ကို နှိပ်ပါ**

Body ထဲတွင်

7. **Raw ကို နှိပ်ပါ**
8. **ထို့နောက် JSON ကို ထပ်နှိပ်ပါ**
9. **မိမိတို့ ရေးလိုသည့် content များကိုရေးပါ။ သို့သော် key နေရာတွင် name, email, phnumber, degree တို့ကိုသာ ထည့်ပေးရမည်။ အဘယ်ကြောင့် ဆိုသော် မိမိတို့ database ထဲတွင် ထိုနည်းအတိုင်း ဆောက်ပေးထားသောကြောင့် ဖြစ်သည်။**
10. **ထို့နောက် send ကို နှိပ်ပါ။**
11. **Status 201 Created ကိုတွေ့ရပါက မိမိတို့ data ပို့ခြင်း အောင်မြင်ပါပြီ။**

```
{
"name":"SaNayMaungMaung",
"email":"SaNayMaungMaung@gmail.com",
"phnumber":"09453741976",
"degree":"CS"
}
```

မိမိတို့ Database ထဲသို့ သွားရောက် စစ်ဆေးကြည့်ပါကလည်း ထည့်ပေးလိုက်သည့်

data အသစ် တိုးလာသည်ကို မြင်ရပါမည်။

Data Output Explain Messages Notifications

	id [PK] integer	name character varying (50)	email character varying (50)	phnumber character varying (15)	degree character varying (20)
1	1	WinHtut	winhtutonline@gmail.com	09453741976	LLB
2	2	MgMgkaung	mgmgkaung@gmail.com	09453741976	GHI
3	3	MgMgkaung	mgmgkaung@gmail.com	09453741976	GHI
4	4	SaNayMaungMaung	SaNayMaungMaung@gmail.com	09453741976	CS

Postman ဖြင့် GET request ကို သုံးကာ မိမိလိုချင်သည့် data များကိုလည်းသီးသန့် ရယူ နိုင်ပါတယ်။ ထိုသို့ ရယူရန် မိမိလို ချင်သည့် data ရဲ့ id ကို သိရန် လိုအပ်ပါသည် ဥပမာ စာရေးသူ အနေဖြင့် WinHtut ရဲ့ အချက်လက်တွေကို လိုချင်ပါက WinHtut ရဲ့ id သည် 1 ဖြစ်သောကြောင့် 1 ကိုအောက်ပါ အတိုင်း Url တွင် ထည့်ပေးပြီး data ကို ရယူ နိုင်ပါသည်။ GET request အသစ် ပြုလုပ်ရန် ယခင်ရှိပြီးသား တစ်ခုအား duplicate ပြုလုပ်ပါ။

1. GET ဟု ပြောင်းထားပါ
2. Info url နောက်တွင် info/1 ဟု ရေးပါ
3. ပြီးလျှင် send ကို နှိပ်ပါ
4. အောက်ပါ အတိုင်း 1 တွင်ရှိသော data များ ရလာမှာ ဖြစ်သလို
5. Status 200 OK ဆိုတာကိုလည်း မြင်ရပါမှာပါ။

Untitled Request

GET (1) http://127.0.0.1:8000/api/info/1 (2) Send (3)

Params Authorization Headers (9) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (8) Test Results Status: 200 OK Time: 39 ms Size: 370 B Sa

Pretty Raw Preview Visualize JSON

```
{
    "id": 1,
    "name": "WinHtut",
    "email": "winhtutonline@gmail.com",
    "phnumber": "09453741976",
    "degree": "LLB"
}
```

မိမိ ပြုပြင် (update) လုပ်လိုသည့် data များကိုလည်း update ပြုလုပ်နိုင်ပါသေးသည်။

1. Url တစ်ခုကို duplicate ပြုလုပ်ပါ
2. PUT ကို ချိန်းပါ
3. Url နောက်ဆုံးတွင် မိမိပြင်လိုသည့် id ကို info/4/ ဟု ထည့်ပါ(စာရေးသူ အနေဖြင့် id 4 ကို ပြင်ပါမည်)
4. Headers ထဲတွင်လည်း Content-type, Application/json တို့ကို ထားပေးပါ
5. Body ကိုနှိပ်ပြီး raw ကို သွားပါ JSON ဟုပြောင်းပါ
6. မိမိ update လုပ်လိုသည့် content များကို ထည့်ပါ
7. Send ကို နှိပ်ပါ။
8. Data အသစ် ပြောင်းလဲ သွားသည်ကို အောက်တွင် မြင်ရမည်ဖြစ်သလို

9. Status 200 OK ဆိုတာကိုလည်း မြင်ရပါမည်။

Postman ကိုသုံးပြီး data များကို delete လည်း လုပ်နိုင်ပါသေးသည်။

1. Url တစ်ခုကို duplicate ပြုလုပ်ပါ
2. DELETE ကို ချိန်းပါ
3. ထို့နောက် send ကို နှိပ်ပါ
4. Status တွင် 204 No Content ကို မြင်လိုက်ရပါက delete လုပ်ခြင်း အောင်မြင်ပါပြီ

Django Form Login/out

ယခု အခန်းတွင် Django ရဲ့ Form ကိုသုံးပြီး login/out လုပ်တဲ့ program တစ်ပုဒ်ကို ရေးသွားမှာပါ။

1. LoginAndOut ဆိုသည့် folder တစ်ခု တည်ဆောက်ပါမည်
2. ထို folder ထဲတွင် virtualenv တစ်ခုတည်ဆောက်ပြီး django version 3.0.3 ကို install လုပ်ထားပါမည်။
3. Virtualenv ထဲတွင် LoginAndOut ဆိုသည့် django project တစ်ခု

တည်ဆောက်ပါမည်။

4. ထို LoginAndOut project ထဲတွင် login ဆိုသည့် django app တစ်ခု တည်ဆောက်ပါမည်။

```
C:\Users\winht\djprojects\LoginAndOut>mkvirtualenv LoginAndOut
Using base prefix 'c:\\users\\winht\\appdata\\local\\programs\\python\\python38-32'
New python executable in C:\Users\winht\Envs\LoginAndOut\Scripts\python.exe
Installing setuptools, pip, wheel...
done.

(LoginAndOut) C:\Users\winht\djprojects\LoginAndOut>pip install django==3.0.3
Collecting django==3.0.3
  Using cached Django-3.0.3-py3-none-any.whl (7.5 MB)
```

```
(LoginAndOut) C:\Users\winht\djprojects\LoginAndOut>django-admin startproject LoginAndOut
```

```
(LoginAndOut) C:\Users\winht\djprojects\LoginAndOut\LoginAndOut>python manage.py startapp login
```

```
 Directory of C:\Users\winht\djprojects\LoginAndOut\LoginAndOut

07/04/2020  04:01 PM    <DIR>          .
07/04/2020  04:01 PM    <DIR>          ..
07/04/2020  04:01 PM    <DIR>          login
07/04/2020  04:01 PM    <DIR>          LoginAndOut
07/04/2020  03:59 PM               652 manage.py
               1 File(s)            652 bytes
               4 Dir(s)  105,986,174,976 bytes free
```

LoginAndOut ဆိုသည့် app တစ်ခုကို ဖန်တီးလိုက်သည့်အတွက် settings.py file ထဲတွင် ထို app name အားသွားရောက် ရေးပေးထားရပါမည်(အရင်သင်ခန်းစာတွင်လည်း ဖော်ပြခဲ့ပြီး ဖြစ်သည်) နောက် တစ်ဆင့်အနေဖြင့်project folder ထဲမှ urls.py ထဲမှာပင် app ရဲ့ urls.py ကို ချိတ်ပေးရန် လိုအပ်ပါသေးသည်။ ထို့ကြောင့်urls.py ထဲတွင် အောက်ပါ အတိုင်းချိတ်ပေးရမည်။ include ကိုလည်း Import လုပ်ပေးရန် လိုအပ်ပါသည်။

```
16  from django.contrib import admin
17  from django.urls import path,include
18
19  urlpatterns = [
20      path('admin/', admin.site.urls),
21      path('login/',include('login.urls')),
22  ]
```

Login app folder ထဲမှ urls.py ကို သွားရောက်ချိတ်ပေးခဲ့သည့်အတွက် urls.py အား app folder ထဲတွင် သွားဆောက်ပေးရန် လိုအပ်ပါသည်။ ထို urls.py ထဲတွင် home , dashboard , register , login , logout စသည့် html page ငါး ခုကို တစ်ခါတည်း အောက်ပါ အတိုင်း ချိတ်ပေးထားပါမည်။

LoginAndOut > login > urls.py > ...

```python
from django.urls import path
from .import views
urlpatterns = [
    path('',views.home,name='home'),
    path('dashboard/',views.dashboard,name='dashboard'),
    path('login/',views.login,name='login'),
    path('register',views.register,name='register'),
    path('logout/',views.logout,name='logout'),
]
```

Html page များ တည်ဆောက်ရန် app folder ထဲတွင် templates ဆိုသည့် folder တစ်ခု တည်ဆောက်ပါမည်။ ထို templates folder ထဲတွင် html page များ တည်ဆောက်ပါမည်။

နောက် တစ်ဆင့် အနေဖြင့် home.html ထဲတွင် အောက်ပါ အတိုင်း template ကို ရေးထားပါမည်။ ထို့နောက် register.html ထဲတွင် home.html အား inheritance လုပ်ပြီး အောက်ပါအတိုင်း ရေးပါမည်။ POST method ကိုသုံးထားသည့် အတွက် csrf_token ကိုလည်း ထည့်ပေးရမည်။

LoginAndOut > login > templates > home.html > ...

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <title>Home</title>
7   </head>
8   <body>
9       {% block content %}
10      <h1>Welcome , This is Home Page</h1>
11      {% endblock %}
12  </body>
13  </html>
```

LoginAndOut > login > templates > register.html > ...

```
1   {% extends 'home.html' %}
2   {%block content%}
3       <h1>Create New Account </h1>
4       <form method="POST">
5           {% csrf_token %}
6           <button> Register </button>
7           {% endblock %}
8       </form>
```

အထက်ပါ အဆင့်များပြီးပါက views.py ထဲတွင် ထိုfile နှစ်ခုအားချိတ်ပြီး server ကို run ကြည့်ရပါမည်။ အခြားသော function များအား pass လုပ်ထားပါသည်။

LoginAndOut > login > views.py > ...

```python
from django.shortcuts import render

# Create your views here.
def home(request):
    return render(request ,'home.html')

def dashboard(request):
        pass
def register(request):
    return render(request,'register.html')

def login(request):
    pass
def logout(request):
    pass

```

Server run ပြီး http://127.0.0.1:8000/login/register ယခု url အတိုင်းသွားကြည့်ပါက register page ကိုရောက်သွားပါမည်။ အကယ်၍ စာဖတ်သူတို့ အနေဖြင့်လည်း ရောက်သွားပါက file များ ချိတ်ဆက်ခြင်း အောင်မြင်ပါပြီ။ page မပေါ်ပ် error တက်နေပါက အစကနေ step by step ပြန်လည် စစ်ဆေးရန် လိုအပ်ပါသည်။

နောက်တစ်ဆင့်အနေဖြင့် django ရဲ့ form အား အသုံးပြုပြီး registration form တစ်ခု တည်ဆောက်ပါမည်။ django သည် registration ပြုလုပ်ရန် form အား support လုပ်ပေးထားပါသည်။ form.as_p ဟု သုံးလျှင် registration ပြုလုပ်ရန် form တစ်ခု ရလာမှာ ဖြစ်ပါတယ်။

1. Views.py မှ register function သို့သွားမည်
2. User မှ request လုပ်လာသော method အား POST method ဖြစ်သလားဟု စစ်ဆေးမည်
3. POST method ဖြစ်လျှင် UserCreationFrom function အား အသုံးပြုပါမည်။ UserCreationFrom သည် ModelForm Class မှ inheritance လုပ်ထားခြင်း ဖြစ်သည်။
4. Form အချက်အလက် အားလုံး မှန်ကန်လျှင် form ထဲမှ ရှိသော data များအား save ပြီး login page သို့ redirect လုပ် ပေးပါမည်။
5. ထို့ပြင် account ဖန်တီးလို့ ရကြောင်းကိုလည်း စာကြောင်း တစ်ကြောင်း ထုတ်ပေးပါမည်။
6. နောက်ဆုံး အနေဖြင့် else statement ထည့်ပြီး register.html ဆီသို့ form တစ်ခု ပို့ပေးလိုက်ပါမည်။ POST method မဟုတ်ခဲ့ရင် ဆိုတဲ့ အခြေနေမှာ ပို့ပေးမှာ ဖြစ်ပါတယ်။

7. ထို့ပြင် register.html ထဲ၌ လည်း {{form.as_p}} ဆိုသည်ကို ထည့်ပေးရန် လိုအပ်ပါသည်။ as_p ဆိုသည့် html paragraph ပုံစံဖြင့် ဖော်ပြ ပေးရန် ဖြစ်သည်။ Views.py ထဲတွင် ရှိသော program မှာ အောက်ပါ အတိုင်း ဖြစ်ပါတယ်။

LoginAndOut > login > views.py > ...

```
1   from django.shortcuts import render,redirect
2   from django.contrib.auth.forms import UserCreationForm
3   from django.contrib import messages
4
5   # Create your views here.
6   def home(request):
7       return render(request ,'home.html')
8
9   def dashboard(request):
10          pass
11  def register(request):
12      if request.method == "POST":
13          form = UserCreationForm(request.POST)
14          if form.is_valid():
15              form.save()
16              messages.success(request, 'Account created successfully')
17              return redirect('login')
18      else:
19          form = UserCreationForm()
20      return render(request,'register.html',{'form':form})
21
22  def login(request):
23      pass
24  def logout(request):
25      pass
```

register.html

```
1   {% extends 'home.html' %}
2   {%block content%}
3       <h1>Create New Account </h1>
4       <form method="POST">
5           {% csrf_token %}
6           {{form.as_p}}
7           <button> Register </button>
8           {% endblock %}
9       </form>
10
```

အထက်ပါ အတိုင်း file များအား save ပြီး run လိုက်ပါက အောက်ပါ ပုံ အတိုင်း Registration form တစ်ခုကို တွေ့ရပါမည်။

Django Admin

Django Admin သို့ ဝင်ရောက်ရန် superuser အကောင့် တစ်ခု ဖန်တီးပါမည်။ python manage.py createsuperuser ဟု ရေးပေးရမည်။ ထို့နောက် username , email , password တို့အား အောက်ပါ အတိုင်း ထည့်ပေးရပါမည်။

```
(LoginAndOut) C:\Users\winht\djprojects\LoginAndOut\LoginAndOut>python manage.py createsuperuser
Username (leave blank to use 'winht'): root
Email address: root@gmail.com
Password:
Password (again):
The password is too similar to the username.
This password is too short. It must contain at least 8 characters.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.

(LoginAndOut) C:\Users\winht\djprojects\LoginAndOut\LoginAndOut>
```

အထက်ပါအတိုင်း ရေးပြီးပါက မိမိတို့ server အား ပြန် run ပြီး browser တွင် new tag တစ်ခုဖွင့်ပါ ။ ထို့နောက် http://127.0.0.1:8000/admin/login/?next=/admin/ ယခု url အတိုင်း သွားပါ။

မိမိတို့ superuser အဖြစ် create လုပ်လိုက်သည့် username and password ကို ထည့်ပါ။ ထို့နောက် login လုပ်ပါ။ အောက်ပါအတိုင်း မြင်ရပါမည်။ user ကို တစ်ချက် နှိပ်ကြည့်ပါ မိမိတို့ superuser account တစ်ခုတည်းသာ မြင်ရမည် ဖြစ်ပြီး မည်သည့်

account မှ မရှိသေးသည်ကို တွေ့ရမည်။

LoginView

User အတွက် login page ဖန်တီးရန် django ထဲမှ LoginView ကို အသုံးပြုပါမည်။ ထိုသို့ အသုံးပြုရန် urls.py ထဲသို့ သွားပါ။ ထို့နောက် login url အတွက် အောက်ပါ အတိုင်း LoginView.as_view ဟုရေးပေးပါ။

urls.py • views.py • register.html home.html settings.py

LoginAndOut > login > urls.py > ...

```
1   from django.urls import path
2   from .import views
3   from django.contrib.auth.views import LoginView
4   urlpatterns = [
5       path('',views.home,name='home'),
6       path('dashboard/',views.dashboard,name='dashboard'),
7
8       path('login/',LoginView.as_view(),name='login'),
9
10      path('register',views.register,name='register'),
11      path('logout/',views.logout,name='logout'),
12  ]
```

နောက်တစ်ဆင့် အနေဖြင့် login.html သို့သွားပြီး အောက်ပါ အတိုင်း ရေးပေးပါ။

LoginAndOut > login > templates > login.html > form

```
1  {% extends 'home.html' %}
2  {%block content%}
3      <h1>Please Login</h1>
4      <form method="POST">
5          {% csrf_token %}
6          {{form.as_p}}
7          <button> Login</button>
8          {% endblock %}
9      </form>
```

Warning: templates folder ထဲတွင် registration folder တစ်ခု တည်ဆောက်ပါ။ ထို registration folder ထဲသို့ login.html ဆိုသည့် file အား ပြောင်းထည့်လိုက်ပါ။ အဘယ်ကြောင့်ဆိုသော် LoginView ကိုသုံးလျှင် django သည် templates folder ထဲတွင် registration folder ကို ထပ်ရှာပါသည်။ ထို folder ထဲကမှ login.html ကို အသုံးပြုပါသည်။ ထို့ကြောင့် မိမိတို့ file directory ပုံစံသည် အောက်ပါ အတိုင်း ဖြစ်နေရပါမည်။

Registration လုပ်ခြင်းနှင့် login လုပ်ခြင်းများ အဆင် ပြေမပြေ သိနိုင်ရန် မိမိတို့ server အား ပြန် run ပြီး http://127.0.0.1:8000/login/register url သို့သွားကာ register ပြုလုပ်ပါ။ username နေရာတွင် number အနည်းကိုပါ ထည့်ပေးပါ။ ဥပမာ winhtut99 password ကိုတော့ မိမိတို့ အဆင်ပြေတာ ပေးနိုင်သလို password ကို နှစ်ကြိမ်ရိုက်ပေးရန် လိုအပ်ပါသည်။

အထက်ပါပုံ အတိုင်းပြုလုပ်ပြီး register ကို နှိပ်လိုက်ပါက လုပ်ဆောင်မှု အားလုံးမှန်ကန်လျှင် Login page သို့အောက်ပါ အတိုင်း ရောက်သွားပါမည်။

ထို့နောက် username and password အားဖြည့်ပြီး Login နှိပ်လိုက်ပါ။ အောက်ပါ page သို့ ရောက်သွားပါမည်။

မိမိတို့ url တွင် admin သို့သွားပြီး superuser အကောင့်ဖြင့် login ဝင်ပါ။

ထို့နောက်တွင် users ကို တစ်ချက်နှိပ်ပြီး ကြည့်လိုက်ပါက မိမိတို့ register လုပ်ထားသည့် username များကို မြင်ရပါမည်။

မှတ်ချက်။ Login ဝင်လိုက်သည် နှင့်တစ်ပြိုင်နက် profile သို့ရောက်သွားရခြင်းမှာ LOGIN_REDIRECT_URL ကို default အနေဖြင့် '/account/profile/' ဟု ပေးထားသည့် အတွက်ကြောင့် ဖြစ်သည်။ ထို့ကြောင့် settings.py ထဲတွင် အောက်ဆုံးစာကြောင်းအား အောက်ပါ အတိုင်း ပြောင်းပေးရမည်။

```
118
119   # Static files (CSS, JavaScript, Images)
120   # https://docs.djangoproject.com/en/3.0/howto/static-files/
121
122   STATIC_URL = '/static/'
123   LOGIN_REDIRECT_URL='/login/dashboard/'
```

Path အသစ် dashboard ကို ပေးလိုက်သည့် အတွက် dashboard အတွက် views.py နှင့် dashboard.html တို့တွင် source code များ သွားရေးပေးရမည်။ views.py ထဲမှ dashboard function အတွက် login_required decorator ကို သုံးပါမည်။ ထို decorator သည် django.contrib.auth.decorators ထဲတွင် ရှိပါသည်။ ထို့ကြောင့် Import လုပ်ပေးရန် လိုအပ်သည်။ views.py ထဲတွင် dashboard function အတွက် ရေးသည့် code မှာ အောက်ပါ အတိုင်း ဖြစ်သည်။

LoginAndOut > login > views.py > register

```
from django.shortcuts import render,redirect
from django.contrib.auth.forms import UserCreationForm
from django.contrib import messages
from django.contrib.auth.decorators import login_required

# Create your views here.
def home(request):
    return render(request ,'home.html')

@login_required
def dashboard(request):
        return render(request,'dashboard.html')

```

dashboard.html ထဲတွင်လည်း အောက်ပါ အတိုင်းရေးရန် လိုအပ်ပါသည်။ username အား ပြန်လည် ဖော်ပြပေးလိုသည့်အတွက် user.username ဟု jinja template ဖြင့်ရေးသားထားခြင်း ဖြစ်သည်။

LoginAndOut > login > templates > dashboard.html > ...

```
{% extends 'home.html' %}
{%block content%}
    <h1>Hello ,{{user.username}}</h1>

{% endblock %}

```

အထက်ပါ အဆင့်များ ရေးပြီးပါက File များအား save ပြီး run ပါ ထို့နောက် login သို့ပြန်သွားပြီး login ဝင်ပါ။ ပြီးလျှင် http://127.0.0.1:8000/login/dashboard/ ယခု link သို့ သွား ကြည့်ပါက မိမိတို့ username အားတွေ့ရပါမည်။

is_authenticated

တကယ်လို့ အသုံးပြုနေသည့် သူသည် home.html page သို့ရောက်သောအခါ user authenticated ဖြစ်တယ် register လုပ်ပြီး login ဝင်ထားတယ်ဆိုရင် logout ဆိုတဲ့ link လေး တစ်ခု ထပ်ထည့်ပေးပါမည်။ ထိုကဲ့သို့ ပြုလုပ်ရန် jinja template ဖြင့် if user.is_authenticated ကို အသုံးပြုပါမည်။ ထို့ပြင် login ဝင်ထားသူ မဟုတ်ပါကလည်း

Login လုပ်ရန်နှင့် register လုပ်ရန်တို့ကို ပြပေးရန် login and register လုပ်ရန် link နှစ်ခု ပြပေးထားပါမည်။ homt.html ထဲတွင် ရေးသားထားပုံတို့အား အောက်တွင် ကြည့်နိုင်သည်။

```
8   <body>
9       {% block content %}
10      <h1>Welcome , This is Home Page</h1>
11      {% if user.is_authenticated %}
12          <a href="{% url 'logout' %}">Logout</a>
13          {% else%}
14          <a href="{% url 'login' %}">Login</a>
15          <a href="{% url 'register' %}">Register</a>
16      {% endif %}
17      {% endblock %}
18  </body>
19  </html>
```

Link ထဲတွင် login page နှင့် register page တို့အား ထည့်ထားသည်ကို သတိပြုပါ။

Logout path အတွက် urls.py ထဲတွင် LogoutView ကို ထည့်ပေးရန် လိုအပ်ပါသေးသည်။ Django views ထဲမှပင် LogoutView ကိုလည်း သုံးနိုင်ပါသည်။

```
LoginAndOut > login > urls.py > ...
1   from django.urls import path
2   from .import views
3   from django.contrib.auth.views import LoginView,LogoutView
4   urlpatterns = [
5       path('',views.home,name='home'),
6       path('dashboard/',views.dashboard,name='dashboard'),
7
8       path('login/',LoginView.as_view(),name='login'),
9
10      path('register',views.register,name='register'),
11      path('logout/',LogoutView.as_view(),name="logout"),
12  ]
```

LogoutView ကိုသုံးလိုက်၍ logout.html file သည် မလိုတော့သည့် အတွက် ဖျက်ပစ်လိုက်နိုင်ပါသည်။ အကယ်၍ logoutView မသုံးပဲ မိမိတို့ ဘာသာ စမ်းရေးလိုသူ များအတွက်မဖျက်ပစ်ပဲ မိမိတို့ ရေးချင်သည့် html code များရေးကာ urls.py ထဲတွင် logout အား path လုပ်ပေးပြီး views.py ထဲတွင်လည်း ဖျက်စရာ မလိုပဲ render လုပ်ပေးနိုင်ပါသည်။ ထို့ပြင် views.py ထဲမှလည်း login function နှင့် logout function တို့အား ဖျက်ပြစ်လိုက် နိုင်ပါသည်(မိမိတို့ဘာသာ စမ်းရေးလိုသူများအတွက် မဖျက်ပဲ ရေးကြည့် နိုင်ပါသည်)

အထက်ပါ အဆင့်များ ပြီးပါက file အားလုံးအား save ပြီး run လိုက်ပါ။ ထို့နောက် Home page သို့သွားလျှင်အောက်ပါ အတိုင်း မြင်ရမည်။

Home page တွင် login and register တို့ကို မြင်ရမည် ဖြစ်ပြီး ထို နှစ်ခုမှာလည်း အလုပ် လုပ်နေသည်ကို မြင်ရမည်။ ထို့နောက် login သို့သွားပါ။

Username and password များထည့်ပြီး Login ဝင်ပါက dashboard သို့ရောက်သွားပါမည်။ ထို့နောက် Url မှ တစ်ဆင့် home page သို့ပြန်သွားပါ။ Url မှသွားလျှင် user သည် login ဝင်ထားပြီးသား authenticated ဖြစ်သည့်အတွက် Logout ကို ပြပေးပါလိမ့်မည်။

အထက်ပါ ပုံတွင် ပေါ်နေသော Logout ကို တစ်ချက် နှိပ်လိုက်ပါက django ရဲ့ logout View သို့ ရောက်ရှိသွားပါလိမ့်မည်။

NumPy

NumPy
Numerical Python

NumPy or numerical python သည် python ရဲ့ library တစ်ခု ဖြစ်ပြီး scientific computing တွင်အသုံးပြုပါသည်။ NumPy ထဲတွင် multidimensional array , matrices , mathematical functions , logical , shape manipulation , sorting , selecting, I/O , discrete Fourier transforms , basic linear algebra , basic statistical operations , random simulation နှင့် တစ်ခြားသော functions များစွာ ပါဝင်ပါသေးတယ်။ အားလုံးနီးပါးကို C programming language ဖြင့် ရေးထားတာ ဖြစ်တဲ့အတွက် မြောက်များစွာသော အသေးစိတ် ဂဏန်းတွက်ချက်မှုတွေကို လျှင်မြန်စွာ လုပ်ဆောင် နိုင်ပါတယ်။ ယနေ့ခေတ်မှာ scientific/mathematical နဲ့ သက်ဆိုင်တဲ့ python base software တွေမှာ ဆိုရင် Python ရဲ့ built-in ပါဝင်တဲ့ sequence types တွေ ဖြစ်တဲ့ list, tuples တို့ကို အသုံး ပြုတတ်ရုံဖြင့် လုံလောက်တော့မည် မဟုတ်ပဲ NumPy arrays တွေကို အသုံးပြုတတ်ဖို့လည်း လိုအပ်လာပါပြီ။ အဘယ်ကြောင့်ဆိုသော များပြားလှတဲ့ data တွေကို ကိုင်တွယ် တွက်ချက်ဖို့ mathematical ဆိုင်ရာနှင့် အခြားသော operations တွေကို လုပ်ဆောင်ဖို့ ဆိုရင် python ရဲ့ built-in sequences တွေဟာ NumPy arrays ထက် code အရေအတွက်လည်း ပိုများသလို efficiency လည်း ပိုနည်းပါတယ်။

NumPy arrays ရဲ့ size ဟာ စတင် create လုပ်ကတည်း က အသေသတ်မှတ် ပြီးသား ဖြစ်ပါတယ် သို့သော် python ရဲ့ list ကတော့ dynamic allocation လုပ်နိုင်တဲ့ အတွက် list ရဲ့ size ကို ပြောင်းလဲ နိုင်ပါတယ်။

NumPy Ndarray

Ndarray သည် NumPy ရဲ့ အရေးကြီးတဲ့ အစိတ်အပိုင်းတစ်ခု ဖြစ်ပါတယ်။ Ndarray ထဲတွင် data type တူညီတဲ့ elements တွေကိုသာ stores လုပ်နိုင်သလို access လုပ်တဲ့ အခါမှာလည်း ပထမဆုံး နေရာကို 0 (zero) ကနေစတင်ပြီး ရယူ နိုင်ပါတယ်။ data type တွေ တူညီတဲ့အတွက် element တစ်ခုချင်းစီရဲ့ memory ပေါ်မှာ နေရာယူတဲ့ size ကလည်း အတူတူဘဲ ဖြစ်ပါတယ်။ ထိုသို့ data types တွေ တူတဲ့ အတွက် Homogeneous data လို့လည်း ခေါ်ပါတယ် python ရဲ့ lists ကိုတော့ heterogenous data လို့ ခေါ် ဆိုပါတယ်။ ndarray ကို ရှေ့ပိုင်း သင်ခန်းစာများတွင် sample program များနှင့် တစ်ကွ အသေးစိတ် ဖော်ပြသွားပါမည်။
၈၅

Creating a ndarray function and object

NumPy ရဲ့ array class တွေကို ndarray(n-dimensional array) ဟုခေါ်ပါတယ်။

ထို့ကြောင့် array object တစ်ခုကို create လုပ်နိုင်သလို ပြုလုပ်ရန် ပထမဆုံး numpy ကိုတော့ import လုပ်ရန် လိုအပ်ပါတယ်။ numpy ကို install လုပ်ရန်အောက်ပါ command ကိုသုံးနိုင်ပါသည်။

```
>> pip3 install --user numpy
```

Sample Program 335

```
import numpy as np
arr = np.array
print(arr)
```

ပထမဆုံး code line ကတော့ numpy library ကို အတိုကောက် np အနေဖြင့် သုံးမည်ဟုကြေငြာ ပေးထားခြင်း ဖြစ်ပါသည်။ ဒုတိယ code line သည် Numpy(np) ထဲမှ array ဆိုသည့် function တစ်ခုကို လှမ်းခေါ်ပြီး arr ဆိုသည့် array object တစ်ခုကို တည်ဆောက်လိုက်ပါသည်။ ထို arr သည် built in သို့မဟုတ် မိမိတို့ ဖန်တီးထားသည့် array ဖြစ်နေသလားဆိုတာကို စစ်ဆေးရန် print ထုတ်ကြည့်သော အခါတွင် built- in function array ဆိုသည့် output ကို မြင်ရမှာ ဖြစ်ပါတယ်။

<built-in function array>

သို့သော် ဘယ်လို type မျိုးလဲ ဆိုတာ သိနိုင်ဖို့ အတွက်တော့ type function ကိုသုံးပြီး အောက်ပါ အတိုင်း စစ်ဆေးကြည့်နိုင်ပါတယ်။

```
import numpy as np
arr = np.array
print("Type of arr",type(arr))
```

အထက်ပါအတိုင်း စစ်ဆေးကြည့်မည် ဆိုလျှင် builtin_function_or_method ဆိုသည့် output ကိုတွေ့ရမှာ ဖြစ်သလို အကယ်၍ array function ထဲတွင် data elements များထည့်ပြီး စစ်ဆေးကြည့်သော အခါတွင်မူ class numpy.ndarray ဆိုသည့် output ကို တွေ့ရမှာ ဖြစ်ပါတယ်။

```
import numpy as np
arr = np.array([[ 1, 2, 3],
                [ 4, 2, 5],
                [7,8,9]])
print("Type of arr",type(arr))
```

Output

Type of arr <class 'numpy.ndarray'>

Dimensions

Array တစ်ခုရဲ့ axes အရေအတွက်ကို dimension ဟု ခေါ်ဆိုခြင်း ဖြစ်ပါတယ်။ axes တွေအကြောင်းကို အတိအကျ နားလည်ဖို့ ဆိုရင် 2-diemnsional array တစ်ခု တည်ဆောက်ပြီး ထို array ထဲတွင် data များထည့်ခြင်း သို့မဟုတ် အသစ်ထည့်ခြင်းဖြင့် n-dimensional array တွေကို ချည်းကပ်နိုင်ပါတယ်။ ပထမဆုံးအနေဖြင့် numpy ကို သုံးပြီး empty array တစ်ခု တည်ဆောက်ပါမည်။ ထိုသို့ တည်ဆောက်ရန် အတွက်လည်း numpy ထဲမှ empty ဆိုသည့် method ကို သုံးနိုင်ပါသည်။

Empty

Empty function သည် ပုံမှန်အားဖြင့် parameter သုံးခု ယူပါတယ်။ shape , data type , order တို့ ဖြစ်ပါတယ်။ shape သည် မိမိတို့ ပြုလုပ်လိုသည့် array ရဲ့ ပုံစံ ဖြစ်ပြီး data type သည် ထို array ထဲမှ elements များရဲ့ data type ဖြစ်ပါတယ်။ order သည် array ထဲမှ element များကို မည်သည့် memory layout ဖြင့် သုံးမည်ကို ဆိုလိုခြင်း ဖြစ်ပါတယ်။ အကယ်၍ empty array တစ်ခု ပြုလုပ်ရာတွင် order ကို ထည့်မရေးလျှင် default အားဖြင့် C ကို သုံးပါတယ်။ အခြေခံအားဖြင့် order လေး မျိုးရှိပါတယ်။

- C - Contiguous layout(row - major order)
- F - Fortran layout (column-major order)
- A - any order
- K - keep order

ယခု မိမိတို့ ဖန်တီးမည့် empty array မှာတော့ Contiguous layout ကို သုံးပါမည်။

C - Contiguous layout

Contiguous layout ဆိုတာက elements တွေရဲ့ memory address တွေဟာ အစဉ်လိုက် ရှိနေတာပါ။ element တစ်ခုနဲ့ တစ်ခုကြားမှာ ခြားနေတာမျိုး၊ ကွာဟ နေတာမျိုး မရှိတဲ့ အတွက် index ကိုသုံးပြီးတော့လည်း elements များကို access ရယူ နိုင်ပါတယ်။ Contiguous layout ကို row major order လို့လည်း ခေါ်ဆိုပါသေးတယ်။ Fortran layout ကိုတော့ column-major order လို့ ခေါ်ဆိုပါတယ်။ row-major order နှင့် column-major order မှ ခြားနားချက်ကို သိနိုင်ရန် အောက်ပါ ပုံကို ကြည့်ပါ။

Empty Function syntax

numpy.empty(shape , dtype = float , order='C')

Sample Program - 336

```
num.py > ...
import numpy as np
arr = np.array(
            [[ 1, 2, 3],
             [ 4, 2, 5],
             [ 7, 8, 9]]
            )
print("Type of arr",type(arr))
print("Number of dimensions of arr :",arr.ndim)
```

Sample Program – 336 ကို ကြည့်မည် ဆိုလျှင် output အနေဖြင့် 2 dimension ဆိုတာကိုပဲ ရရှိမှာပါ။ အဘယ်ကြောင့် ဆိုသော row and column ပါဝင်တဲ့ array တစ်ခုတည်း ပါဝင်ပါတယ်။ row and column ကို x ,y ဟုလည်း သတ်မှတ်ထားနိုင်ပါတယ်။ ထိုကဲ့သို့ array တစ်ခုတည်း ပါဝင်ခြင်းကို 1 depth လို့လည်း ခေါ်ဆိုပါတယ်။ array တစ်ခု ရဲ့ dimensions များကို လိုချင်သည့် အခါ ndim ကို သုံးပါတယ်။

အထက်ပါ ပုံများကို ကြည့်ခြင်း အားဖြင့် 2 dimension ကို နားလည်သွားပါလိမ့်မည် ထိုကဲ့သို့ 2 dimensions ပါဝင်သည့် array တစ်ခု ထဲတွင် နောက်ထပ် 2 dimension ပါဝင်သည့် array တစ်ခု ထပ်ရောက်လာခြင်းကို 3 dimensions or n-dimensions ဟု ခေါ်ဆိုခြင်း ဖြစ်ပါသည်။ ဥပမာ အနေဖြင့် array တစ်ခုအား စာအုပ် တစ်အုပ်ဟု ယူဆသည် ဆိုပါစို့။ ထိုစာအုပ်ပေါ်သို့ နောက်ထပ် စာအုပ် တစ်အုပ် ထပ်မံ တင်ကြည့်လိုက်ပါက စာအုပ် နှစ်အုပ် ဖြစ်သွားမှာ ဖြစ်သလို array တစ်ခု တည်းမှာ array နှစ်ခု ဖြစ်သွားတဲ့ အတွက် 2 depth ဟု ခေါ်ဆို ပါတယ်။ ထိုကဲ့သို့ 2 depth ဟို ခေါ်ဆိုလိုက်သည့် အတွက် x , y အပြင် z ဆိုသည့် နောက်ထပ် axis တစ်ခုလည်း ထပ်တိုးလာမှာပါ။

အထက်ပါ သဘောတရားများ အား နားမလည်သေးလျှင် အောက်ပါအတိုင်း programming နည်း ဖြင့် step by step ရှင်းလင်း ပေးသွားပါမည်။

1. ပထမဆုံး အနေဖြင့် numpy ထဲမှ arrange ဆိုသည့် method ကိုသုံးပြီး elements 8 ခုပါဝင်သည့် array တစ်ခုကို တည်ဆောက်ပါမည်။
2. ထို array ကို 2 rows and 4 columns (x , y) သာ ပါဝင်သည့် 2d array တစ်ခု သို့ ပြောင်းလဲ ပါမည်။ ထိုသို့ ပြောင်းလဲ နိုင်ရန် numpy ထဲမှ reshape method ကို သုံးနိုင်သည်။
3. နောက် တစ်ဆင့် အနေဖြင့် stage 2 မှ ရလာသော array ကို 4 rows and 2 columns အဖြစ်သို့ ပြန်လည် ပြောင်းလဲပါမည်။
4. နောက်ဆုံးအဆင့် အနေဖြင့် stage 3 မှ array အား 2 rows and 2 columns နှင့် 2 depth ပါဝင်သည့် 3 dimensional array သို့ ပြောင်းလဲ ပါမည်။

Sample Program 337

```
import numpy as np
array  = np.arange(8)
print("First Stage Array ",array)

array =np.arange(8).reshape(2,4)
print("Second Stage Array",array)
```

Output

First Stage Array [0 1 2 3 4 5 6 7]

Second Stage Array [[0 1 2 3]

[4 5 6 7]]

Sample Program 337 ကို run ကြည့်ပါက ပထမဆုံး output တွင် 0 to 7 ထိရှိသည့် ပုံမှန် array တစ်ခုကို ဖန်တီးထားခြင်း ဖြစ်ပြီး ဒုတိယ output တွင် 0 to 7 ထိ ရှိသည့် elements များအား 2 dimensions array ပုံစံဖြင့် ပြန်လည် ဖန်တီးထားခြင်း ဖြစ်သည့်အတွက် row and column (x ,y) ပုံစံ output တို့ကို တွေ့ရမှာပါ။ ထိုကဲ့သို့ ပြုလုပ်နိုင်ရန် reshape method ကို သုံးထားပြီး reshape method သည် ပထမ parameter ကို row အဖြစ် လုပ်ဆောင်ပြီး ဒုတိယ parameter ကို columns အဖြစ် လုပ်ဆောင်ပါသည်။ နောက်ထပ် parameter ထပ်ထည့်လျှင် ထို parameter ကို depth အဖြစ် 3 dimensions ပုံစံသို့ လုပ်ဆောင်ပါသည်။ အောက်တွင် sample program ကို ထပ်မံ ရေးပြထားပါသည်။

Complete Sample Program - 338

```
import numpy as np
array  = np.arange(8)
```

```
print("First Stage Array ",array)

array =np.arange(8).reshape(2,4)
print("Second Stage Array",array)

array =np.arange(8).reshape(2,4)
print("Third Stage 2d Array",array)

array =np.arange(8).reshape(2,2,2)
print("Fourth Stage 3d  Array",array)
```

Output

First Stage Array [0 1 2 3 4 5 6 7]

Second Stage Array [[0 1 2 3]

[4 5 6 7]]

Third Stage 2d Array [[0 1 2 3]

[4 5 6 7]]

Fourth Stage 3d Array [[[0 1]

[2 3]]

[[4 5]

[6 7]]]

Fourth Stage ရဲ့ program ရေးပုံနှင့် output များကို သေချာ ကြည့်ပါက 3 dimensional array ဆိုတာကို ကောင်းမွန်စွာ သဘောပေါက်သွားပါမည်။

Jupyter Notebook

Jupyter Notebook သည် web-based interactive environment တစ်ခု ဖြစ်ပါတယ်။ web-based interactive ဆိုတာကတော့ desktop applications တွေမှာ ရတဲ့ လုပ်ဆောင်ချက်တွေ ကို web application ပုံစံနဲ့ လုပ်ဆောင်နိုင်တာကို ပြောတာပါ။ Jupyter

Notebook က data analysis and computational problem-solving တွေကို ပြုလုပ်ရာမှာများစွာ အသုံးပြုကြပါတယ်။ ဘာကြောင့်လည်းဆိုတော့ jupyter notebook ဟာ display လုပ်တဲ့ အပိုင်းမှာ program တွေကိုတင် ဖော်ပြနိုင်တာမျိုး မဟုတ်ပဲ သခ္ခာ equation တွေ figures တွေ နဲ့ videos and image တွေကိုပါ output မှာ တွဲပြီး ဖော်ပြပေးနိုင်ပါတယ်။ ထို့ပြင် HTML and JavaScript တို့ကို သုံးပြီးတော့လည်း user interface တွေပါ ဖန်တီးနိုင်ပါသေးတယ်။

Jupyter notebook ကိုသုံးဖို့ installation နည်းလမ်းများစွာ ရှိပါတယ်။ စာရေးသူ အနေနဲ့ကတော့ anaconda ကို recommend ပေးထားကြတဲ့ အတွက် anaconda ကို install လုပ်ပြီးမှ anaconda navigator မှ တစ်ဆင့် jupyter notebook ကို ကိုသုံးပါမယ်။ ထို့ကြောင့် ပထမဆုံး အနေဖြင့် အောက်ပါ link မှာ anaconda ကို download ဆွဲပြီး install လုပ်ပေးရန် လိုအပ်ပါသည်။ စာရေးသူအနေဖြင့် individual version ကိုသာ သုံးထားပါသည်။

https://www.anaconda.com/products/individual

Windows , MacOS and Linux အားလုံးကို support လုပ်ထားတဲ့ အတွက် မိမိတို့ environment နဲ့ ကိုက်ညီတာကို download ဆွဲပြီး install လုပ်နိုင်ပါတယ်။ anaconda ကို install လုပ်ပြီးသည်နှင့် တစ်ပြိုင်နက် anaconda navigator ကို အောက်ပါ အတိုင်းဖွင့် လိုက်ပါ။ Anaconda Navigator ကို ဖွင့်လိုက်သည်နှင့် တစ်ပြိုင်နက် JupyterLab , Jupyter Notebook စတဲ့ အခြားသောအရာများလည်း ပေါ်နေပါလိမ့်မယ်။ ထို့နောက် Jupyter Notebook အောက်က launch ဆိုတာကို တစ်ချက်နှိပ်လိုက်သည်နှင့် တစ်ပြိုင်နက် webpage တစ်ခု ပေါ်လာပါမည်။ ထိုသို့ webpage ပေါ်မလာခင် မိမိ နှစ်သက်ရာ browser ကို လည်း ရွေးချယ်နိုင်ပါသေးတယ်။ web page သည် localhost webserver တစ်ခုအနေနဲ့ port 8888 မှာ ပေါ်လာမှာ ဖြစ်ပါတယ်။အကယ်၍ port 8888 မိမိ computer ရဲ့ အခြား webserver port တူနေမယ်ဆိုရင်တော့ ဖွင့်မရတာမျိုး ဖြစ်တတ်ပါတယ်။ထိုကဲ့သို့အခြေနေ ဖြစ်လာလျှင် အခြားသော webserver ရဲ့ Port အား ချိန်းပေးရန် လိုအပ်ပါတယ်။

ပထမဆုံး အနေဖြင့် project folder တစ်ခုဆောက်ရန် new ထဲကို သွားပါ။ ထို့နောက် Python3 ကို တစ်ချက် နှိပ်ပါ။ အောက်ပါ အတိုင်း ပေါ်လာပါမည်။

နာမည်ချိန်းရန် အတွက် ယခုမြှားပြထားသော နေရာအား တစ်ချက်နှိပ်ပြီး မိမိတို့ အဆင်ပြေသည့် နာမည်ကို ချိန်းနိုင်ပါတယ်။ စာရေးသူအနေဖြင့် 1-numpy ဟုသာ နာမည် ပေးထားပါတယ်။

နောက်တစ်ဆင့်အနေဖြင့် Python program စ run ရန် print("Hello Jupyter") ဟု အောက်ပါ အတိုင်းရေးပြီး control+enter ကို နှိပ်ကာ run ကြည့်ပါ။ output ပါ တစ်ခါတည်း ဖော်ပြ နေသည်ကို မြင်ရပါမည်။

Jupyter Notebook မှာ file တွေကို JSON (JavaScript Object Notation) ပုံစံနဲ့ stored လုပ်ပါတယ် ။file ရဲ့ extension ကိုတော့ ipynb(ipython notebook) နဲ့သိမ်းပါတယ်။ ထို့ပြင် Jupyter Notebook မှာ ကျွန်တော်တို့ ရေးလိုက်တဲ့ python code တွေကို pure python code ပုံစံနဲ့ မရရှိနိုင်ပါဘူး။ သို့သော် Notebook မှာ မိမိတို့ လိုချင်တဲ့ ပုံစံဖြင့် ပြန်လည် download ရယူ နိုင်ပါတယ်။ Jupyter Notebook မှာ အောက်ပါ တို့ကို support လုပ်ပေးထားပါတယ်။

Cell Types

Jupyter Notebook မှာ menu bar and toolbar ဆိုတဲ့ content နှစ်ခု ပါဝင်ပါတယ်။ toolbar ထဲကမှ drop-down menu မှတစ်ဆင့် cell ကိုဘယ်လိုပုံစံဖြင့် run ချင်လဲဆိုတာကို လုပ်ဆောင် ပေးပါတယ်။ cell ဆိုတာ မိမိတို့ ရေးထားတဲ့ codes တွေကို ပြောတာပါ။

Drop-down menu မှာ အောက်ပါ အတိုင်း cell လေးခု ပါဝင်ပါတယ်။

Code

Code cell အနေနဲ့ run မယ် ဆိုရင် shift+enter I control+enter ဟူ၍ run သည့်ပုံစံ နှစ်ခု ရှိပါတယ် ။ shift+enter နှင့် run မည်ဆိုလျှင် run လိုက်သည့် code ရဲ့ output ကိုလည်း ဖော်ပြ ပေးသလို နောက်ထပ် cell အသစ် တစ်ခုလည်း ထပ်ပေါ်လာပေးပါတယ်။ control+enter နှင့်သာ run မည် ဆိုလျှင်တော့ ယခုလက်ရှိ cell ကိုသာ run ပေးပြီး output ပြပေးပါတယ်။ နောက်ထပ် cell အသစ်တစ်ခု ပေါ်မလာ ပေးပါ။ မိမိတို့ ကြိုက်တဲ့ cell တွေကို ဖြတ်ပြီး run လိုတဲ့အခါမှာ အသုံးပြုနိုင်ပါတယ်။ အောက်မှာတော့ shift+enter နှင့် run ထားသည့် ပုံကို ပြပေးထားပါတယ်။

Shift+enter ကို နှိပ်လိုက်တဲ့ အခါမှာ cell ထဲမှ code တွေကို ယူပြီးတော့ kernel ထဲကို ပို့ပေးပါတယ်။ kernel သည် python interpreter ကို အသုံးပြုပြီး output ထုတ် ပေးပါတယ်။

ထွက်လာတဲ့ output ကို browser ရဲ့ သက်ဆိုင်ရာ cell များအောက်မှာ ပြန်ပြ ပေးပါတယ်။ Jupyter notebooks မှာ programming language တစ်ခုတည်းကို ရေးသားနိုင်တာ မဟုတ်ပါဘူး။ အခြားသော programming language များကိုလည်း ရေးသားနိုင်ပါတယ်။ ဥပမာ R , Julia ကျွန်တော်တို့ ရေးလိုက်တဲ့ cell တွေထဲမှာ ရှိတဲ့ code တွေကို သက်ဆိုင်ရာ compiler or interpreter တွေနဲ့ process လုပ်နိုင်အောင် လုပ်ဆောင်ပေးတာကို Kernal လို့ခေါ်ပါတယ်။ ထို့ပြင် မိမိတို့ကိုယ်တိုင် kernel တွေကို ပြုပြင်နိုင်သလို အသစ်လည်း ဖန်တီးနိုင်ပါသေးတယ်။ Jupyter notebook က မိမိတို့ လက်ရှိ ဘယ် kernel ကို အသုံးပြုနေလဲဆိုတာကို page ရဲ့ ညာဘက် အပေါ်မှာ သို့မဟုတ် logo နဲ့ ဖော်ပြ ပေးထားပါတယ်။ ကျွန်တော်တို့အနေနဲ့ Jupyter notebook ကို စတင်လိုက်တဲ့ အချိန်မှာ ယခင်က မိမိတို့ အသုံးပြုခဲ့သော kernel နဲ့ ပြန်လည်ဖော်ပြ ပေးပါတယ်။ သို့သော်လည်းပဲ မိမိတို့ အသုံးပြုလိုတဲ့ kernel ကို ပြန်ချိန်းပြီး အသုံးပြုနိုင်ပါတယ်။ အကယ်၍ jupyter lab ကို အသုံးပြုနေခဲ့တယ် ဆိုရင်တော့ Kernel menu မှ တစ်ဆင့် Change Kernel ကိုသွားပြီး မိမိတို့ လိုချင်တဲ့ kernel ကို ချိန်း နိုင်ပါတယ်။

Markdown Cells

Text, Equation, HTML, Headings, Embedded code, LaTeX equations တွေကို markdown cell ကိုသုံးပြီး မိမိတို့ program မှာ add နိုင်ပါတယ်။ ထို့ပြင် HTML5 video tag တွေကိုလည်း ထည့်နိုင်ပါသေးတယ်။ အကယ်၍ italic ပုံစံနဲ့ စာသားတွေ ထည့်ချင်တယ်ဆိုရင် ပုံမှန် cell ရဲ့ ပထမဆုံး character နေရာတွင် asterisk ကိုသုံးပြီး အောက်ပါအတိုင်း ရေးသားနိုင်ပါတယ်။

Output

အကယ်၍ bold ဖြင့်အသုံးပြုချင်ရင်တော့ ** နှစ်ခုကို အသုံးပြု နိုင်ပါတယ်။ cell တစ်ခုကို heading အနေဖြင့် အသုံးပြုလိုလျှင်တော့ ပထမဆုံး character နေရာမှာ # ကို

ထည့်ပေးရပါမည်။ # နောက်တွင် space တစ်ချက် ခြားပေးရပါမည်။

Output

အကယ်၍ Heading ကို အနည်းငယ် ပိုသေးချင်ရင်တော့ ## နှစ်ခုကို အသုံးပြုနိုင်ပါတယ်။ ## နှစ်ခုနောက်တွင် space တစ်ခု ပါဝင်သည်ကိုတော့ သတိပြုပါ။

LaTeX equations

Mathematical ဆိုင်ရာ ဖော်ပြချက်တွေ အတွက်ဆိုလည်း အောက်ပါ အတိုင်း အသုံးပြု နိုင်ပါသေးတယ်။

$e^x=\sum_{i=0}^\infty \frac{1}{i!}x^i$

Output

Adding File or Image

Jupyter notebook သည် မိမိတို့ရဲ့ cell ထဲမှာ markdown ကိုသုံးပြီး image or files တွေကိုထည့်ဖို့လည်း ခွင့်ပြုထားပါသေးတယ်။ ထိုသို့ထည့်ဖို့အတွက် HTML tag တွေကိုသုံးနိုင်ပါတယ်။ Markdown သည် HTML ရဲ့ superset ဖြစ်တဲ့အတွက် HTML tag တွေကို အသုံးပြုနိုင်ခြင်းဖြစ်ပါတယ်။ အောက်ပါ cell ထဲမှာတော့ ယခုလက်ရှိရေးနေတဲ့ python file ရှိတဲ့ same location မှာပဲ မိမိတို့ ထည့်ချင်တဲ့ file ကို ထားပြီး အောက်ပါအတိုင်း

ဖော်ပြနိုင်ပါတယ်။ စာရေးသူ အနေဖြင့်တော့ ပုံတစ်ပုံ ကို အသုံးပြုထားပါတယ်။

```
<img src="winhtut.jpg" />
```

In [ ]:

Output

$e^x = \sum_{i=0}^{\infty} \frac{1}{i!} x^i$

အကယ်၍ မိမိတို့ cell ထဲမှာ ထည့်ချင်တဲ့ file က same location မဟုတ်ခဲ့လျှင်လည်း အောက်ပါ အတိုင်း location path ကို အတိအကျ ပေးပြီး ထည့်နိုင်ပါတယ်။

```
<img src="../photo/winhtut.jpg" />
```

Images တွေတွင်မကပဲ videos file တွေကိုပါ add လို့ ရနိုင်ပါသေးတယ်။ စာရေးသူ အနေဖြင့် ယခုလက်ရှိ same location မှာ video file တစ်ခုကို cell ထဲတွင် ထည့်ချင်တဲ့ အတွက် အောက်ပါ အတိုင်း ရေးသားထားပါတယ်။

```
<img src="winhtut.jpg" />
<video controls src="2020 Py Intern Guide by WinHtut.mp4">animation</video>
```

In [ ]:

Output

In [ ]:

စာရေးသူ add လိုက်သော video file လေး ပေါ်လာခြင်း ဖြစ်ပါတယ်။

Raw NBconvert

Raw ကိုသုံးပြီးတော့ cell တစ်ခုကို run မည် ဆိုလျှင် မည်သည့် Processing မှ လုပ်မည် မဟုတ်ပဲ မိမိတို့ဖော်ပြပေးသည့် အတိုင်းသာ ပြန်လည်ဖော်ပြ ပေးမှာပါ။ သို့သော် nbconvert ကိုတော့ မိမိတို့ လိုအပ်သလို များစွာ အသုံးပြုနိုင်ပါတယ်။ nbconvert ကို အသုံးပြုဖို့ ဆိုလျှင်တော့ ပထမဦးစွာ pip install convert ဟု ရေးပြီး Install လုပ်ပေးရန် လိုအပ်ပါတယ်။ nbconvert ကို သုံးခြင်းဖြင့် ဥပမာ .ipynb ဆိုသည့် jupyter file ကနေ HTML , LaTeX , PDF ,

Markdown and reStructuredText တွေ ဆီသို့ အလွယ်တကူ ပြောင်းလဲ ပေးနိုင်ပါတယ်။ jupyter မှာ ပြောင်းလိုသော အခါတွင် အောက်ပါနည်းလမ်းများ အတိုင်းရေးပြီး ပြောင်းလဲနိုင်ပါတယ်။ html and pdf ကို example အနေဖြင့် ဖော်ပြ ပေးထားပါတယ်။

jupyter nbconvert --to html myfile.ipynb (ipynb file မှ html file သို့ ပြောင်းရန်)

jupyter nbconvert --to pdf myfile.ipynb (ipynb file မှ pdf file သို့ ပြောင်းရန်)

သို့သော် Jupyter notebook မှ File menu ထဲမှ Download ကိုသွားပြီး မိမိလိုချင်သော format ဖြင့်လည်း download ရယူနိုင်ပါတယ်။ ယခု သင်ခန်းစာမှာတော့ nbconvert ကို အဓိက ထားပြောချင်တာ မဟုတ်တဲ့အတွက် ဤနေရာ မှာပဲ အဆုံးသတ်လိုက်ပါမည်။

Jupyter Notebook ကို အမြန်ဆုံးနည်းလမ်းနဲ့ အသုံးပြုနိုင်ဖို့ကတော့ shortcuts များကို အသုံးပြုခြင်းဖြစ်ပါတယ်။ Jupyter Notebook menu bar မှ Help ကို သွားပြီး Keyboard Shortcuts ကို နှိပ်လိုက်ခြင်းဖြင့် အောက်ပါအတိုင်း shortcuts list များကျလာမှာ ဖြစ်ပါတယ်။

Nbextensions

Jupyter မှာလည်း visual studio code ကဲ့သို့ extensions များစွာ ရှိပါတယ်။ ထို extensions များကို သုံးနိုင်ဖို့ nbextensionsကို အရင်ဆုံး install လုပ်ပေးဖို့ လိုအပ်ပါတယ်။ မိမိတို့အနေနဲ့ jupyter notebook ကို စဖွင့်တဲ့အချိန်မှာ Files , Running , Clusters စတဲ့ Options သုံးခုကိုပဲ မြင်ရမှာပါ Nbextensions ကို မြင်ရမှာ မဟုတ်ပါဘူး။

Nbextensions ကို install လုပ်ဖို့ အတွက်နည်းလမ်း နှစ်ခုကို အသုံးပြုနိုင်ပါတယ်။ conda and pip package manager ဖြစ်ပါတယ်။ စာရေးသူ အနေဖြင့် conda ကိုသာ အသုံးပြုသွားပါမည်။ ပထမဆုံးအနေဖြင့် command promt သို့သွားပြီး အောက်ပါ အတိုင်းလုပ်ဆောင်ပေးပါ။

```
conda install -c conda-forge jupyter_contrib_nbextensions
```

အထက်ပါ အဆင့်တွင် y/n ကို တောင်းပါလိမ့်မယ်။ y ကိုနှိပ်ပြီး install လုပ်ပေးပါ။ အကုန်လုံး ပြီးသွားလျှင် အောက်ပါ command တစ်ကြောင်းကို ထပ် run ပေးပါ။

```
conda install -c conda-forge jupyter_nbextensions_configurator
```

အထက်ပါ command ကိုလည်း လုပ်ဆောင်ပြီးသွားပါက jupyter notebook ကို ဖွင့်ပြီး ပြန် run ကြည့်ပါ။ ထို့နောက် Nbentensions ပါလာတာကို မြင်ရပါမည်။

Nbextensions ထဲတွင် အထက်ပါအတိုင်း မိမိတို့ အသုံးပြုလိုသည့် extensions များကို အသုံးပြုနိုင်ပါပြီ ။ ပထမဆုံး အနေဖြင့် disable ဆိုသည့် checkbox ကို uncheck လုပ်ပေးပါ။ သို့မှသာ အသုံးပြုနိုင်မှာ ဖြစ်ပါတယ်။ စာရေးသူအနေဖြင့် Collapsible Headings , Notify , Hinterland , Autopep8 , Code Font Size စတဲ့ extensions 5 ခုကို ထပ်ပြီး အမှန်ခြစ် ပေးထားပါတယ်။တစ်ခုခြင်းစီရဲ့ အသုံးဝင်ပုံကို အောက်မှာပြောပြပေးပါ့မယ်။

Collapsible Heading

Collapsible headings က အရမ်းကို အသုံးဝင်ပါတယ်။ သူ့ကို မိမိတို့ ရေးထားတဲ့ program တွေကို အစဉ်လိုက် သတ်မှတ်ပေးပြီး သိမ်းထားလို့ ရပါတယ်။ ဆိုလိုချင်တာက Heading တစ်ခုစီ ခွဲပြီး code တွေကိုဖွက်ထားလို့ရပါတယ်။ ထို့ကြောင့် ကြည့်ရတာ အရမ်းကိုရိုးရှင်းနေမှာပါ။

ပထမဆုံးအနေဖြင့် markdown ကိုသုံးပြီး Heading တစ်ခု ဖန်တီးလိုက်ပါ။ မိမိ ကြိုက်တဲ့ နာမည်ပေးပါ။ ထို Heading အောက်တွင် မိမိ ရေးသားလိုသည့် program များကိုရေးပါ။ Collapsible ကို အသုံးပြုထားပြီ ဖြစ်တဲ့ အတွက် ဘေးမှာ မြားအသေးလေး တစ်ခု ပါနေပါမည်

ထိုမြားလေးကို နှိပ်လိုက်ပါ မိမိတို့ ရေးထားသည့် codes များကို ပြန်မြင်ရပါမည်။

Autopep8

PEP ဆိုတာကတော့ Python Enhancement Proposal ကို ဆိုလိုတာပါ။ autopep8 ကတော့ မိမိရေးထားသော code တွေ ဖတ်ရ လွယ်ကူစေရန်နှင့် format မကျသော code များအား format ကျစေရန် ကူညီ ပေးပါတယ်။ ဥပမာ list တစ်ခုကို ကြေငြာထားရင် space ခြားသင့်တဲ့ နေရာတွေမှာ space မခြား ထားဘူးဆိုရင် autopep8 ကိုသုံးခြင်းဖြင့် space ခြားပေးပါတယ်။ autopep8 ကို အသုံးပြုရန် မိမိရေးထားသော codes များအား select မှတ်ပြီး တူးပုံ icon လေးကို နှိပ်ပေးပါ။

Hinterland

Program ရေးတဲ့နေရာမှာ အရေးကြီးဆုံး လိုအပ်ချက် ဖြစ်တဲ့ auto complete ဖြစ်နိုင်ဖို့ Hinterland ကို သုံးနိုင်ပါတယ်။ ဥပမာ im လို့ရေးလိုက်တာနဲ့ import ဆိုတာကို ဖော်ပြပေးနေတာပါ။ programmer အနေဖြင့် code ရေးရတာ အရမ်းကို မြန်သွားပါတယ်။

NumPy.Zeros

သတ်မှတ်ထား်းသည့် shape အတိုင်း numpy array တစ်ခု ဖန်တီးရာတွင် array ထဲမှာ ရှိသည့် elements များကို 0 နဲ့ initialized လုပ်ချင်သောအခါတွင် numpy.zeros ကိုသုံးပါတယ်။ zeros သည်လည်း parameter သုံးခု ယူပါတယ်။ shape , dtype and order တို့ဖြစ်ပါတယ်။ order မှာ C style ဖြစ်တဲ့ row-major order နဲ့ ပြုလုပ်နိုင်သလို FORTRAN-style column-major နဲ့လည်း order စီစဉ်ထားနိုင်ပါတယ်။ ဥပမာ row သုံးခု columns သုံ်းခု ရှိတဲ့ array တစ်ခုကို တည်ဆောက်၍ ထို array ထဲတွင် elements တွေကို value 0 များနှင့် သတ်မှတ်ပေးထားမည် ဆိုလျှင် အောက်ပါ အတိုင်း ရေးနိုင်ပါတယ်။ order ကိုတော့ ထည့်မရေးထားပါဘူး default C ကိုသာ သုံးထားပါတယ်။

```
In [ ]: #NumPy.zeros

In [3]: import numpy as np
        arr =np.zeros( (3,3) ,dtype = int)
        print(arr)

        [[0 0 0]
         [0 0 0]
         [0 0 0]]
```

NumPy.ones

Zeros routine မှာ elements value တွေကို 0 များဖြင့် သတ်မှတ် ပေးထားခဲ့သည်။ ထိုနည်းတူ values များအားလုံးကို 1 ဖြင့် သတ်မှတ်လိုပါက numpy.ones ကိုသုံးနိုင်ပါတယ်။ အခြား parameters များ အားလုံးသည် zeros နှင့် အတူတူသာပင် ဖြစ်ပါသည်။

```
In [ ]: #NumPy.ones

In [4]: import numpy as np
        arr =np.ones( (3,3) ,dtype = int)
        print(arr)

        [[1 1 1]
         [1 1 1]
         [1 1 1]]
```

Create NumPy Array With data

Numpy array တွေကို ဖန်တီးတဲ့ နေရာမှာ zeros or ones ဖြင့်မဟုတ်ပဲ အခြားသော list or tuple စတဲ့နေရာတွေက data တွေကို ယူပြီးတော့လည်း create လုပ်နိုင်ပါတယ်။ ထိုသို့

create လုပ်နိုင်ဖို့ numpy မှာ asarray ဆိုတဲ့ routine ကို support လုပ်ပေးထားပါတယ်။ asarray သည် လည်း parameter သုံးခု ယူပါတယ်။ sequence , dytpe and order တို့ဖြစ်ပါတယ်။ ပထမဆုံး parameter ဖြစ်တဲ့ sequence နေရာမှာ မိမိတို့ အသုံးပြုချင်တဲ့ list or tuple စတဲ့ sequence ဖြစ်တဲ့ data များကို ထည့်ပေးရပါတယ်။ dtype and order တို့သည့် အထက်တွင် ဖော်ပြခဲ့ပြီး ဖြစ်သည်။ အောက်မှာ sample program နှစ်ခု ဖော်ပြ ထားပါတယ် ပထမတစ်ခုသည် list sequence ကိုသုံးပြီး ဖော်ပြထားပြီး ဒုတိယ တစ်ခုသည် tuple sequence ကိုသုံးထားပါတယ်။

Using List

```
In [8]: #NumPy.asarray

In [7]: import numpy as np
        myList=[1,2,3,4,5,6,7,8,9]  #sequence
        a = np.asarray(myList);
        print(type(a))
        print(a)

        <class 'numpy.ndarray'>
        [1 2 3 4 5 6 7 8 9]
```

Using Tuple

```
In [8]: #NumPy.asarray

In [10]: import numpy as np
         mytuple=(1,2,3,4,5,6,7,8,9)  #sequence
         a = np.asarray(mytuple);
         print(type(a))
         print(a)

         <class 'numpy.ndarray'>
         [1 2 3 4 5 6 7 8 9]
```

Asarray ထဲတွင် list တစ်ခုတည်း ဖြတ်လို့ရတာမဟုတ်ပဲ list တစ်ခုတည်းမှာ list တစ်ထပ်ပိုပြီးတော့လည်း ဖြတ်လို့ရသလို tuple တစ်ခုတည်းမှာ list တစ်ခုထပ်ပိုပြီးတော့ဖြတ်နိုင်ပါတယ်။ sample program ကိုအောက်တွင် ဖော်ပြထားပါတယ်။

```
In [8]: #NumPy.asarray

In [13]: import numpy as np
         mylist=[[1,2,3,4,5,6,7,8,9],[10,11,12]]  #sequence
         mytuple=([1,2,3,4,5,6,7,8,9],[10,11,12])  #sequence
         a = np.asarray(mylist);
         t = np.asarray(mytuple)
         print(type(a))
         print(a)
         print(mytuple)

         <class 'numpy.ndarray'>
         [list([1, 2, 3, 4, 5, 6, 7, 8, 9]) list([10, 11, 12])]
         ([1, 2, 3, 4, 5, 6, 7, 8, 9], [10, 11, 12])
```

NumPy.frombuffer

Ones ,zeros , list or tuple များနှင့် မဟုတ်ပဲ array တစ်ခုဖန်တီးတဲ့နေရာမှာ buffer data တွေကို elements များအနေဖြင့် အသုံးပြုလိုသောအခါတွင် frombuffer ဆိုတာကို သုံးနိုင်ပါတယ်။ Numpy သည် buffer data တွေကို array အဖြစ် အသုံးပြုနိုင်ရန် support လုပ်ပေးထားပါတယ်။ frombuffer function ဟာ paramter လေးခု ယူပါတယ်။ ပထမ တစ်ခုသည် buffer data ဖြစ်ပြီး dtype , count နှင့် offset တို့ ဖြစ်ပါတယ်။

numpy.frombuffer(bufferData , dtype=float , count = -1 , offset = 0)

```
In [8]: #NumPy.asarray

In [34]: import numpy as np
         bData = b'green hackers'
         print(type(l))
         a=np.frombuffer(bData, dtype ='S1', count = -1, offset = 0)
         print(a)
         print(type(a))

         <class 'bytes'>
         [b'g' b'r' b'e' b'e' b'n' b' ' b'h' b'a' b'c' b'k' b'e' b'r' b's']
         <class 'numpy.ndarray'>
```

Dtype မှာ S1 လို့ရေးထားခြင်းက string of length 1 ကို ဆိုလိုချင်တာပါ။ count ကို -1 ထားရခြင်းကတော့ buffer ထဲမှာ ရှိသမျှ data အားလုံးကို ထုတ်ယူခ္ပျင်လို့ပါ။ အကယ်၍ buffer data ၅ လုံးထိသာ ထုတ်ယူချင်တယ်ဆိုရင် count=5 ဟုရေးသားနိုင်ပါတယ်။ -1 ဆိုတာကတော့ အဆုံးထိ သွားမယ်လို့ ပြောချင်တာပါ။ offset ကို 0 ထားရခြင်းမှာ buffer data တွေကို ဘယ်နေရာက စထုတ်မယ်ဆိုတာကို သတ်မှတ် ပေးလိုက်တာပါ။ 0 ဟုရေးထားတာ ဖြစ်တဲ့အတွက် အစကနေ စပြီး ထုတ်မှာပါ။ အကယ်၍ ဒုတိယ နေရာကနေ စထုတ်ချင်တယ်ဆိုရင်တော့ offset=1 ဟုရေးပေး နိုင်ပါတယ်။

```
In [36]: import numpy as np
         bData = b'green hackers'
         print(type(I))
         a=np.frombuffer(bData, dtype ='S1', count = 5, offset = 1)
         print(a)
         print(type(a))

         <class 'bytes'>
         [b'r' b'e' b'e' b'n' b' ']
         <class 'numpy.ndarray'>
```

NumPy.fromiter

Fromiter ကတော့ iterable ဖြစ်တဲ့ object တွေကနေတစ်ဆင့် array တစ်ခု ဖန်တီးလိုတဲ့ အခါမှာ သုံးပါတယ်။ ဥပမာ list တစ်ခုကနေ iter function ကိုသုံးပြီး iterable ဖြစ်အောင် ပြုလုပ်ကာ ရလာသော iterable object ကို fromiter ထဲမှာ ထည့်ခြင်းဖြင့် array တစ်ခုကို ဖန်တီး နိုင်ပါတယ်(list သည် iterable ဖြစ်တဲ့အတွက် iter function ကိုမသုံးပဲနှင့်လည်း တိုက်ရိုက် အသုံးပြု နိုင်ပါတယ်)။ iterable object ကို ပထမဆုံး parameter အဖြစ် fromiter function မှ ရယူပါတယ်။

numpy.fromiter(iterable , dtype , count= -1)

```
In [42]: import numpy as np
         data = [0,2,4,6]
         arr = np.fromiter(data, dtype = int)
         print(arr)
         print(type(arr))

         [0 2 4 6]
         <class 'numpy.ndarray'>
```

Incremental Sequences

Numerical computing မှာ အစနဲ့ အဆုံးကြားမှာ နေရာယူထားတဲ့ value တွေကို ကိုင်တွယ် နိုင်ဖို့ကလည်း အရေးကြီးပါတယ်။ ထိုကဲ့သို့ သော array တွေ တည်ဆောက်နိုင်ဖို့ NumPy မှ function နှစ်ခုကို အသုံးပြုနိုင်ပါတယ်။ ပထမ တစ်ခုသည် arange ဖြစ်ပြီး ဒုတိယ တစ်ခုသည် linspace ဖြစ်ပါတယ်။ fuction နှစ်ခုလုံးက parameter ၃ ခု ယူပါတယ်။ ပထမ နှစ်ခု သည် starting point and ending point ဖြစ်ပါတယ်။ arange ရဲ့ တတိယ parameter ကတော့ increment ဖြစ်ပြီး ဥပမာ 1 ဟု ရေးထားမည် ဆိုလျှင် elements တစ်ခုခြင်းစီ တိုင်းကို 1 တိုးပေးသွားမှာပါ။ linspace function ရဲ့ တတိယ Parameter ကတော့ array တစ်ခုတည်းမှာ ပါဝင်မယ့် ဂဏန်းအရေအတွက် ဖြစ်ပါတယ်။ ဥပမာ 10 ဟုရေးထားမည် ဆိုလျှင် စုစုပေါင်း elements 11 ခု ပါဝင်မှာပါ။ ထို function နှစ်ခုသည် ပုံမှန်အားဖြင့် တူညီသည်ဟု ထင်ရသော်လည်း arange သည် second parameter ဖြစ်တဲ့ ending point သည် array ရဲ့ elements တွေထဲမှာ ပါဝင်လာခြင်း မရှိပါဘူး။ သို့သော် linspace မှာတော့ ပါဝင်ပါတယ်။ အကယ်၍ မပါလိုပါကလည်း parameter များကို မိမိတို့ လိုသလို ကစားပြီး ထိန်းညှိနိုင်ပါတယ်။ noninteger မဟုတ်တဲ့ increment elements တွေအတွက်ဆိုရင်တော့ linspace ကို အသုံးပြုဖို့ recommend ပေးထားကြပါတယ်။

```
In [46]: import numpy as np
         arrObj = np.arange(0 , 20 , 2) # 2 သည် increment အတွက် ဖြစ်သည်

         linObj = np.linspace(0.0 , 20 , 3 ) # 3 သည် ရေအတွက် ဖြစ်သည်
         print('For integer sequence ',arrObj)
         print('For noninteger sequence',linObj)

         For integer sequence  [ 0  2  4  6  8 10 12 14 16 18]
         For noninteger sequence [ 0. 10. 20.]
```

Logarithmic Sequences

Elements တွေ တစ်ခုနဲ့ တစ်ခုကြားမှာ logarithmically အရ distributed လုပ်နိုင်ဖို့ numpy မှာ logspace ဆိုသည့် function ပါဝင်နေပါသေးတယ်။ ပုံမှန်အားဖြင့် linspace နှင့် တူသော်လည်း logspace ကတော့ logarithmically နည်းအရ increment လုပ်သွားပါတယ်။ logspace ကို data တစ်ခုနဲ့ တစ်ခု ကြားမှာ ရှိတဲ့ data points များကို logarithmically နည်းအရ သိလိုတဲ့အခါမှာ သုံးပါတယ်။ အောက်ပါ program မှာ ဆိုလျှင် 0 and 2 ကြားမှာ ရှိတဲ့ data points တွေကို ဖော်ပြ ပေးထားပါတယ်။ တတိယ parameter ဖြစ်တဲ့ 5 ကတော့ အရေအတွက် ဖြစ်ပါတယ်။ logspace function ရဲ့ default ဟာ 10 ဖြစ်တဲ့အတွက် 10 to the power ဖြင့် ဖော်ပြ ထားခြင်း ဖြစ်ပါသည်။

100 = 1 to 10**2=100**

```
In [48]: import numpy as np
         arrObj = np.logspace(0 , 2 ,5)
         print(arrObj)

         [  1.          3.16227766  10.          31.6227766  100.        ]
```

Meshgrid Arrays

One-dimensional coordinate arrays တွေမှ တစ်ဆင့် multidimensional coordinate grids တွေအဖြစ် ပြောင်းလဲလိုတဲ့အခါမှာ meshgrid ကိုသုံးပါတယ်။ ဆိုလိုချင်တာကတော့ ထောင့်မှန် စတုဂံပုံ (rectangular grid) တွေ တည်ဆောက်လိုတဲ့အခါ x values နှင့် y values များ လိုအပ်ပါတယ်။ array များဘက်မှ ပြန်စဉ်းစားကြည့်မည် ဆိုလျှင် x one-dimensional နှင့် y one-dimensional array တို့ပေါင်းစပ်ပြီး x and y နှစ်ခု ပါဝင်သည့် multidimensional array ကို ပေါင်းစပ်ဖန်တီးနိုင်သည့် meshgrid ကို အသုံးပြု နိုင်ပါတယ်။

```
In [9]: import numpy as np
        x = np.array([-1,0,1])
        y = np.array([-2,0,2])
        X,Y = np.meshgrid(x,y)
        print(X)
        print('Shape Of X',X.shape)

[[-1  0  1]
 [-1  0  1]
 [-1  0  1]]
Shape Of X (3, 3)
```

အထက်ပါ program မှာ ဆိုလျှင် x and y array နှစ်ခု တည်ဆောက်ထားပါတယ်။ ထို array နှစ်ခုအား meshgrid ကိုသုံးပြီး ပေါင်းလိုက်သည့် အခါမှာတော့ X array အတွက် row and column ပါဝင်သည့် two D array တစ်ခု ရရှိလာပါသည်။ x သည် row ပုံစံဖြင့် သွားသည့် အတွက် -1 0 1 ဟူသည့် သုံးခု ပေါ်လာခြင်းဖြစ်သည်။ ထိုနည်းတူ y တွင်လည်း အောက်ပါအတိုင်း row နှင့် column ဖြစ်လာပါမည်။

```
In [11]: import numpy as np
         x = np.array([-1,0,1])
         y = np.array([-2,0,2])
         X,Y = np.meshgrid(x,y)
         print(Y)
         print('Shape Of Y',Y.shape)

[[-2 -2 -2]
 [ 0  0  0]
 [ 2  2  2]]
Shape Of Y (3, 3)
```

အများဆုံး အသုံးပြုသည့် ပုံစံကတော့ x and y variables နှစ်ခုကို မိမိတို့ အသုံးပြုလိုတဲ့ function များထဲတွင် parameters များအဖြစ် ထည့်ချင်တာပါ။ ဥပမာ x and y values များကို ပေါင်းပြီး ရလာတဲ့ value ရဲ့ 2 power ကို လိုချင်သည့် function တစ်ခု ဆိုပါစို့ အောက်ပါ အတိုင်း ရေးသားနိုင်ပါတယ်။

Z = (x + y) **2

```
In [16]: import numpy as np
         x = np.array([-1,0,1])
         y = np.array([-2,0,2])
         X,Y = np.meshgrid(x,y)
         print('X',X)
         print('Y',Y)
         fun = (X+Y)**2
         print('Fun',fun)

X [[-1  0  1]
 [-1  0  1]
 [-1  0  1]]
Y [[-2 -2 -2]
 [ 0  0  0]
 [ 2  2  2]]
Fun [[9 4 1]
 [1 0 1]
 [1 4 9]]
```

အကယ်၍ 3-D array များ ဖန်တီးလိုသော အခါတွင်လည်း အောက်ပါ အတိုင်း ဖန်တီးနိုင်ပါ သေးတယ်။

```
In [20]: import numpy as np
         x = np.array([-1,0,1])
         y = np.array([-2,0,2])
         z = np.array([3 ,0 ,3])
         X,Y ,Z= np.meshgrid(x,y,z)
         print('X',X)

X [[[-1 -1 -1]
  [ 0  0  0]
  [ 1  1  1]]

 [[-1 -1 -1]
  [ 0  0  0]
  [ 1  1  1]]

 [[-1 -1 -1]
  [ 0  0  0]
  [ 1  1  1]]]
```

Properties of other Arrays

အခြားသော array တစ်ခုရဲ့ properties , shape and datatype တွေကို တစ်ပုံစံတည်း ထပ်တူပွားလိုသောအခါတွင် ones_like ကိုသုံးနိုင်ပါတယ်။ ပထမဆုံး အနေဖြင့် function တစ်ခု ရှိပါမည်။ ထို function ထဲသို့ parameter အနေဖြင့် array တစ်ခု ဖြတ်ပါမည်။ ဒုတိယ တစ်ဆင့် အနေဖြင့် ဖြတ်လာသော array ကို ones_like function ထဲသို့ parameter အနေဖြင့် ပြန်သုံးပါမည်။ ထို့နောက် ones_like function သည် return အနေဖြင့် ဖြတ်လာသော array နှင့် တစ်ပုံစံတည်း တူသော array တစ်ခုကို ပြန်ပေးပါမည်။ မှတ်သားရန် တစ်ခုမှာ ones_like function သည် array အသစ်ထဲမှ elements များအားလုံးကို 1 ဖြင့် ဖြည့်ထားမည် ဖြစ်ပါသည်။ ထိုနည်းတူ zeros_like , full_like , empty_like စသည့် function များကိုလည်း မိမိတို့လိုအပ်သလို သုံးနိုင်ပါတယ်။

Full_like

အခြားသော array တစ်ခုရဲ့ size and shape တို့အတိုင်း array တစ်ခု ဖန်တီးပြီးသော်လည်း ထို array ရဲ့ elements များကို မိမိတို့ ထည့်ချင်သည့် data values များ

ထည့်လိုသောအခါတွင် full_like function ကို သုံးနိုင်ပါတယ်။ full_like function အတွက် အနည်းဆုံး parameter နှစ်ခု လိုအပ်ပါတယ်။ ပထမ တစ်ခုသည် array ဖြစ်ပြီး ဒုတိယ တစ်ခုသည် မိမိတို့ ထည့်လိုသော data value ဖြစ်ပါတယ်။ sample program ကို အောက်တွင် ဖော်ပြထားပါတယ်။

np.full_like(xarray , myData)

```
In [23]: import numpy as np
         x = np.array([-1,0,1])
         y = np.array([-2,0,2])
         z = np.array([3 ,0 ,3])
         X,Y ,Z= np.meshgrid(x,y,z)
         def fun(x):
            newarr=np.full_like(x,11)
            return newarr
         fun(X)

Out[23]: array([[[11, 11, 11],
                 [11, 11, 11],
                 [11, 11, 11]],

                [[11, 11, 11],
                 [11, 11, 11],
                 [11, 11, 11]],

                [[11, 11, 11],
                 [11, 11, 11],
                 [11, 11, 11]]])
```

Matrix Arrays

Matrices or two-dimensional arrays တွေဟာ numerical computing မှာ အလွန် အရေးကြီးပါတယ်။ ထို့ကြောင့် numpy မှာ matrices array တွေ ဖန်တီးဖို့ အတွက် function တွေ ထောက်ပံ့ပေးထားပါတယ်။ ဥပမာ identity and eye ဆိုသည့် function များကို သုံးပြီး matrix array တွေ ဖန်တီးနိုင်ပါတယ်။ identity function သည် parameter အနည်းဆုံး တစ်ခုယူပါသည်။ ဥပမာ 4 ဆိုသည့် parameter တစ်ခုကို ထည့်ပေးလိုက်သည် ဆိုပါစို့ row 4 ခု နှင့် column 4 ခု ရှိသည့် matrix တစ်ခုကို ဖန်တီးပေးသည့် အပြင် diagonal(ထောင့်ဖြတ်မျဉ်း) value အနေဖြင့် 1 ကိုလည်း ထည့်ပေးပါတယ်။ dtype ကိုထည့်ပေးနိုင်သလို default အနေဖြင့် float ဖြင့် ဖော်ပြ ပေးပါတယ်။

```
In [25]: import numpy as np
         x = np.identity(4)
         print(x)

         [[1. 0. 0. 0.]
          [0. 1. 0. 0.]
          [0. 0. 1. 0.]
          [0. 0. 0. 1.]]
```

Diagonals value ကို မိမိတို့ ဖြတ်စေချင်တဲ့ အတိုင်း ဖြတ်လိုလျှင် eye ဆိုသည့် function ကို သုံးနိုင်ပါတယ်။ ဥပမာ row 6 ခုနှင့် column 6 ခုရှိသည့် array တစ်ခုတွင် diagonal value ကို 3 ခုမြောက် column ကနေ စဖြတ်စေချင်လျှင် eye(6 , k=3) ဟု

ရေးနိုင်ပါတယ်။ စ မှတ်ကို zero ကနေ စတင် ရေတွက်ပေးရမည် ဖြစ်ကြောင်းကိုတော့ သတိပြုပါ။

```
In [27]: import numpy as np
         x = np.eye(6 , k=3)
         print(x)

[[0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]]
```

အကယ်၍ diagonals value ကို ဒုတိယမြောက် row ကနေ စတင် ဖြတ်လိုသော အခါတွင်လည်း အောက်ပါအတိုင်း ရေးသားနိုင်ပါသေးတယ်။

```
In [28]: import numpy as np
         x = np.eye(6 , k= -2 )
         print(x)

[[0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0.]]
```

Diagonals value များနေရာတွင် မိမိထည့်လိုသည့် data များ ထည့်လိုသောအခါတွင်လည်း diag function ကိုသုံးနိုင်ပါတယ်။ ပထမဆုံး အနေဖြင့် arange ဖြင့် array တစ်ခုကို ဖန်တီးမည်။ ထို arange function ထဲတွင် elements value နှင့် elements count တို့အပြင် step size ကိုပါ ထည့်ပေးလိုက်ပါမည်။ ထို့ကြောင့် output အနေဖြင့် 0,5,10,15 တို့ပါဝင်သော array တစ်ခုကို ရရှိပါမည်။ အဘယ်ကြောင့်ဆိုသော် 0 နှင့် 20 ကြားမှာရှိသည့် 5 ခြားနားသည့် values များကို ထုတ်ခြင်းကြောင့် ဖြစ်သည်။ ဒုတိယ အနေဖြင့် ထိုကဲ့သို့ ရလာသော array ကို diag function ထဲသို့ ဖြတ်စေပါမည်။ array ထဲတွင် ရှိသော elements အရေအတွက်သည် 4 ခု ဖြစ်သည့် အတွက် row 4 ခုနှင့် columns 4 ခု ပါဝင်သော array တစ်ခုကို ဖန်တီးပေးမည်။ ထို array ရဲ့ diagonals value တွေကိုတော့ 0 ,5,10,15 စသည်ဖြင့် အသီးသီး ထားပေးသွားပါမည်။ row 4 ခုနှင့် columns 4 ခုပါဝင်သည့် array တစ်ခုကို ဖန်တီးပေးမှ သာလျှင် arange ထဲမှ ရလာသော elements 4 ခုကို diagonals values များ အဖြစ် အပြည့် ထည့်နိုင်မည် ဖြစ်ပါသည်။

```
In [34]: import numpy as np
         x = np.arange(0,20,5)
         x

Out[34]: array([ 0,  5, 10, 15])

In [35]: di = np.diag(x)
         di

Out[35]: array([[ 0,  0,  0,  0],
                [ 0,  5,  0,  0],
                [ 0,  0, 10,  0],
                [ 0,  0,  0, 15]])
```

အထက်ပါ program တွင် example အနေဖြင့် 0 , 20 , 5 ကို သုံးထားပါသည်။ နောက်ထပ် sample program အနေဖြင့် 0 , 30 , 5 ကို အောက်ပါ program တွင် သုံးပြထားပါသည်။

```
In [36]: import numpy as np
         x = np.arange(0,30,5)
         x

Out[36]: array([ 0,  5, 10, 15, 20, 25])

In [37]: di = np.diag(x)
         di

Out[37]: array([[ 0,  0,  0,  0,  0,  0],
                [ 0,  5,  0,  0,  0,  0],
                [ 0,  0, 10,  0,  0,  0],
                [ 0,  0,  0, 15,  0,  0],
                [ 0,  0,  0,  0, 20,  0],
                [ 0,  0,  0,  0,  0, 25]])
```

Indexing and Slicing

Numpy array တစ်ခုရဲ့ elements and subarrays တို့ကို standard square bracket [ ]ကိုသုံးပြီး access လုပ်နိုင်ပါတယ်။ထို square bracket ထဲမှာ မိမိတို့လိုချင်သည့် elements များကို index formats များသုံးပြီး လိုသလို access လုပ်နိုင်ပါတယ်။ sample program များရေးနိုင်ရန် ပထမဆုံး အနေဖြင့် arange ကို သုံးပြီး array တစ်ခု ဖန်တီးပါမည်။ ထို array အား index formats များသုံးပြီး access လုပ်ပါမည်။

```
In [39]: import numpy as np
         a = np.arange(0,10)
         a
Out[39]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [40]: a[0]
Out[40]: 0

In [41]: a[-1]
Out[41]: 9

In [43]: a[3]
Out[43]: 3

In [44]: a[1:-1]
Out[44]: array([1, 2, 3, 4, 5, 6, 7, 8])

In [45]: a[1 : -1 : 2 ]
Out[45]: array([1, 3, 5, 7])

In [46]: a[ :5 ]
Out[46]: array([0, 1, 2, 3, 4])

In [48]: a[-5:]
Out[48]: array([5, 6, 7, 8, 9])

In [49]: a[ :: -2 ]
Out[49]: array([9, 7, 5, 3, 1])
```

A[0] သည် array ရဲ့ ပထမဆုံး index 0 ကို access လုပ်လိုတဲ့ အခါမှာ သုံးပါတယ်။
A[-1] သည် array ရဲ့ နောက်ဆုံး element ကို access လုပ်လိုတဲ့ အခါမှာ သုံးပါတယ်။
A[3] သည် array ရဲ့ 3 ခုမြောက်မှ element ကို access လုပ်လိုတဲ့ အခါမှာ သုံးပါတယ်။
A[1 : -1] သည် array ရဲ့ index 1 မှ စတင်ပြီး နောက်ဆုံးထိ element များကို access လုပ်လိုတဲ့ အခါမှာ သုံးပါတယ်။
A[1 : -1 : 2] သည် array ရဲ့ index 1 မှ စတင်ပြီး နောက်ဆုံးထိ element တိုင်းရဲ့ second element ကို access လုပ်လိုတဲ့ အခါမှာ သုံးပါတယ်။
A[:5] သည် array ရဲ့ index 0 မှ စတင်ပြီး index 4 ထိ element ကို access လုပ်လိုတဲ့ အခါမှာ သုံးပါတယ်။
A[-5 :] သည် array ရဲ့ 5 index မှ စတင်ပြီး အဆုံးထိ element ကို access လုပ်လိုတဲ့ အခါမှာ သုံးပါတယ်။
A[::-2] သည် array အား ပြောင်ပြန်လှန်ပြီး second value များကို access လုပ်လိုတဲ့ အခါမှာ သုံးပါတယ်။

Reshaping and Resizing

flatten()

flatten() function ဟာ multidimensional array တွေကို flattened one-dimensional array များ အဖြစ် ပြန်ပြောင်းပေးပါတယ်။ multidimensional array မှာပါသည့် elements အရေ အတွက် အတိုင်း flattened array ရဲ့ length ကို ပြန်ပေးပါတယ်။ ဆိုလိုချင်တာက row နှစ်ခု column နှစ်ခုရှိတဲ့ array တစ်ခုမှာ elements အားဖြင့် 4 ခု ရှိပါတယ်။ ထို array အား flatten လုပ်မည်ဆိုလျှင် အသစ်ရလာသော array ရဲ့ length ဟာလည်း 4 ခု ရှိနေမှာ ဖြစ်ပါတယ်။

```
In [1]:  import numpy as np
         data = np.array([ [1,2],[3,4]])
         data
Out[1]:  array([[1, 2],
                [3, 4]])   ← multidimensional

In [3]:  flat=data.flatten()
         flat
Out[3]:  array([1, 2, 3, 4])   ← one dimensional

In [6]:  data.flatten().shape
Out[6]:  (4,)   ←
```

Newaxis

One-dimensional array မှ တစ်ဆင့် မိမိ လိုသလို row vector or column vector array များ ဖန်တီးလိုတဲ့အခါမှာ newaxis ဆိုသည့် function ကိုသုံးနိုင်ပါတယ်။ newaxis သည် ရှိပြီးသား data များထဲသို့ နောက်ထပ် axis တစ်ခု ထပ်ထည့်ပေးမှာပါ။ ပထမဆုံး အနေဖြင့် arange ကိုသုံးပြီး elements ၅ ခု ပါဝင်သည့် array တစ်ခုကို တည်ဆောက်ပါမည်။ ထို array သည် ပုံမှန် one-dimension array သာ ဖြစ်သည်ကို သတိပြုပါ။

```
In [17]:  data = np.arange(5)
          data
Out[17]:  array([0, 1, 2, 3, 4])
```

ပိုမို နားလည်ရန် ထို array ရဲ့ shape ကိုကြည့်ပါ။

```
In [23]:  data = np.arange(5)
          print(data)
          data.shape

          [0 1 2 3 4]

Out[23]:  (5,)   ←
```

အထက်ပါ array အား row vector array ပုံစံဖြင့် ရရှိလိုပါက အောက်ပါအတိုင်း slicing and newaxis ကို သုံးပြီး ပြောင်းလဲ နိုင်သည်။

```
In [24]: row_vec = data[np.newaxis , : ]
         row_vec

Out[24]: array([[0, 1, 2, 3, 4]])
```

Column vector array ပုံစံဖြင့် ရရှိလိုပါကလည်း အောက်ပါ အတိုင်း slicing and newaxis ကိုသုံးပြီး ပြောင်းလဲ နိုင်ပါတယ်။ ပုံမှ နားလည်စေရန် row_vec + col_vec တို့ကို ပေါင်းကြည့်ပါက multi dimensional ပုံစံဖြင့် array တစ်ခု ရရှိလာပါမည်။

```
In [26]: col_vec = data[ : , np.newaxis]
         print(col_vec)
         col_vec.shape

         [[0]
          [1]
          [2]
          [3]
          [4]]

Out[26]: (5, 1)

In [27]: new = row_vec + col_vec
         new

Out[27]: array([[0, 1, 2, 3, 4],
                [1, 2, 3, 4, 5],
                [2, 3, 4, 5, 6],
                [3, 4, 5, 6, 7],
                [4, 5, 6, 7, 8]])
```

Expand_dims

Array တစ်ခုထဲမှာ နောက်ထပ် dimensions တွေ add ဖို့ newaxis တင်မကပဲ အခြားသော function တွေလည်း ရှိနေပါသေးတယ်။ expand_dims function သည် array တစ်ခုထဲသို့ မိမိတို့ add ချင်သည့် dimension များ ထပ်မံ addနိုင်ပါတယ်။ expand_dims သည် parameters နှစ်ခု ယူပါသည်> ပထမ တစ်ခုသည် array ဖြစ်ပြီး ဒုတိယ တစ်ခုသည် dimension ဖြစ်သည်။ data[: , np.newaxis] သည် np.expand_dims(data , axis =1) နှင့် တူပါသည်။ ယခင် သင်ခန်းစာမှ column vector နှင့် တူညီပါသည်။

```
In [4]: import numpy as np
        data = np.arange(5)
        exp = np.expand_dims(data , axis  = 1)
        exp

Out[4]: array([[0],
               [1],
               [2],
               [3],
               [4]])
```

Row vector အတွက်ဆိုရင်တော့ axis value ကို အထက်ပါ ပုံအတိုင်း 0 လို့ သတ်မှတ်ပေးလိုက်ခြင်း ဖြင့် ရနိုင်ပါတယ်။ np.expand_dims(data , axis = 0)

```
In [5]: import numpy as np
        data = np.arange(5)
        exp = np.expand_dims(data , axis  = 0)
        exp

Out[5]: array([[0, 1, 2, 3, 4]])
```

Vstack , hstack

ယခင် သင်ခန်းစာများတွင် array data series ထဲမှ data များကို ထုတ်ခြင်း၊ ထည့်ခြင်း၊ ပုံစံ ပြောင်းခြင်း၊ slicing များကို သုံးပြီး မိမိတို့လိုသလို access လုပ်ခဲ့ကြသည်။ array data series များကို ကိုင်တွယ်ရာတွင် တစ်ခါတစ်ရံ array များကို ပေါင်းပြီး ပိုကြီးမားသည့် higher-dimensional array များ ဖန်တီးရန်လည်း လိုအပ်ပါသေးသည်။ ထို့ကြောင့် numpy သည် vstack နှင့် hstack ဆိုသည့် fuctions နှစ်ခုကို ထည့်ပေးထားပါသေးသည်။ vstack function သည် parameter အနေဖြင့် ထည့်ပေးလိုက်သော array များကို vertical or row အတိုင်းပေါင်းပြီး ထည့်ပေးလိုက်ပါသည်။ ထိုနည်းတူ hstack ကတော့ ထည့်ပေးလိုက်သော array များကို horizontal or column အတိုင်းပေါင်းပေးပါသည်။ sample program နှစ်ပုဒ်ကို အောက်တွင် ရေးပြထားပါသည်။

```
In [6]: import numpy as np
        data = np.arange(5)
        data

Out[6]: array([0, 1, 2, 3, 4])

In [7]: vs = np.vstack((data, data , data))
        vs

Out[7]: array([[0, 1, 2, 3, 4],
               [0, 1, 2, 3, 4],
               [0, 1, 2, 3, 4]])

In [8]: hs = np.hstack( (data , data , data ))
        hs

Out[8]: array([0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0, 1, 2, 3, 4])
```

Concatenate

Concatenate သည်လည်း vstack and hstack တို့အတိုင်း array တွေကို ပေါင်းဖို့ သုံးပါတယ်။ သို့သော် concatenate သည် axis 1 or 0 မိမိတို့ ပေါင်းချင်သည့် အတိုင်းပေါင်းနိုင်ရန် အတွက် parameter တစ်ခု ထပ်ထည့်နိုင်ပါတယ်။ သတိပြုရန် အချက်မှာ concatenate ကို သုံးမည် ဆိုလျှင် array များသည် dimensions များတူညီရပါမည်။

```
In [12]: import numpy as np
         d1 = np.array([[2, 4], [1,3]])
         d2= np.array([[6,7], [8,9]])
         d2

Out[12]: array([[6, 7],
                [8, 9]])

In [13]: d1

Out[13]: array([[2, 4],
                [1, 3]])

In [14]: con = np.concatenate((d1,d2),axis=1)
         con

Out[14]: array([[2, 4, 6, 7],
                [1, 3, 8, 9]])

In [16]: con0 = np.concatenate( (d1,d2),axis=0)
         con0

Out[16]: array([[2, 4],
                [1, 3],
                [6, 7],
                [8, 9]])
```

Arithmetic Operations

Numpy array တွေကို addition , subtraction , multiplication and division များလုပ်ဆောင်နိုင်ပါတယ်။ ထိုသို့ လုပ်ဆောင်ရာမှာ mathematical operators များဖြင့် လုပ်ဆောင် နိုင်သလို numpy ထဲမှ function များ ဖြင့်လည်း လုပ်ဆောင် နိုင်ပါတယ်။

```
In [18]: import numpy as np
         x = np.array([ [1,2]])
         y = np.array( [ [5,6],[7,8]])
         x+y

Out[18]: array([[ 6,  8],
                [ 8, 10]])

In [19]: x-y

Out[19]: array([[-4, -4],
                [-6, -6]])

In [20]: x/y

Out[20]: array([[0.2       , 0.33333333],
                [0.14285714, 0.25      ]])

In [21]: x*y

Out[21]: array([[ 5, 12],
                [ 7, 16]])
```

Element Wise Elementary Mathematical Functions

1. Trigonometric functions အတွက် np.cos ,np,sin , np.tan တို့ပါဝင်ပါတယ်။
2. Inverse Trigonometric functions အတွက် np.arccos , np.arcsin , np.arctan
3. Hyperbolic trigonometric functions အတွက် np.cosh , np.sinh , np.tanh တို့ပါဝင်ပါတယ်။
4. Inverse hyperbolic trigonometric functions အတွက် np.arccosh , np.arcsinh , np.arctanh တို့ပါဝင်ပါတယ်။
5. Square root အတွက် np.sqrt ပါဝင်ပါတယ်။
6. Exponential အတွက် np.exp ပါဝင်ပါတယ်။
7. Logarithms of base e ,2 , and 10 တို့အတွက်လည်း np.log , np.log2 , np.log10 တို့ပါဝင်ပါတယ်။

Trigonometric Functions

Trigonometric Functions အကြောင်းကို သိဖို့ဆိုရင် radians တွေအကြောင်းကို အကြောင်းကို ဦးစွာ သိဖို့ လိုအပ်ပါတယ်။ Radians တွေဆိုတာ ဘာလဲ။

စက်ဝိုင်းတစ်ခုရဲ့ အလယ် central မှ တိုင်းသော 1 radian ရှိသည့် arc length နှင့် radius သည် အတူတူပင် ဖြစ်ပါသည်။

အထက်ပါ ပုံတွင် မြှားနှင့် ပြထားသော r သည် arc length ဖြစ်သည်။ ထို arc length တန်ဖိုးသည် radius r တန်ဖိုးနှင့် အတူတူပင် ဖြစ်ပါသည်။ အဘယ်ကြောင့် ထိုသို့ တူနေရလည်း ဆိုလျှင် 1 radian ကြောင့် ဖြစ်ပါသည်။ 1 radian ကို degree နှင့် ဖော်ပြမည်ဆိုလျှင် 57.29577951231 ကို ရရှိပါမည်။

1 radian = 180/π (3.1415)

1 radian = 57.29577951231 degree

အထက်ပါ တွက်ချက်ပုံကို ကြည့်ပြီး degree တွေကနေ radians တွေဆီသို့ ပြောင်းလဲနိုင်ပါပြီ။ degree ကနေ randian သို့အောက်ပါ formula ကို အသုံးပြုပြီးပြောင်းလဲနိုင်ပါတယ်။

Formula 1* x π/180* = 0.01745 radian

အထက်ပါ အတိုင်းကြည့်မည် ဆိုလျှင် 1 degree သည် 0.01745 radian ဖြစ်သည်

ဤသို့ ဆိုလျှင် 70 degree သည် ဘယ်လောက် radian ရှိမလဲပေါ့။

1* x π/180* = 0.01745

70* x π/180* = ?

70* x 0.01745329251 = 1.2217304764 radian

ကျွန်တော်တို့ မျက်လုံးထဲမှာ တစ်ခုကို မြင်ကြည့်ရအောင် numpy array တစ်ခုထဲမှာ degree တွေ အများကြီး ပေးထားပြီး ထိုdegree တွေကနေတစ်ဆင့် radian value တွေကို တွက်ထုတ်လို့ရပါမယ်။ ထို radian value တွေမှ တစ်ဆင့် sine တန်ဘိုးတွေကို ရရှိနိုင်ပါတယ်။ sin တန်ဘိုးတွေ ဘယ်လိုရသလဲ သိဖို့ဆိုရင်တော့ sine , cos , tan တို့အကြောင်းကို သိရန်လိုအပ် လာပါတယ်။ Numpy ရဲ့ trigonometric function များထဲမှာ sin function ပါဝင်ပါတယ် 1.2217304 ဆိုတဲ့ radian value ကို sin function ထဲသို့ ထည့်လိုက်မည်ဆိုလျှင် 0.939692457 ဆိုတဲ့ output ကို ရရှိလာပါမည်။ sin function ထဲကို radain တန်ဘိုး ထည့်လိုက်လျှင် အဘယ်ကြောင့် ထို output ထွက်လာရသနည်း ဆိုလျှင် trigo အချိုး များကြောင့် ဖြစ်ပါတယ်။ ထို ထွက်လာသော output value သည်လည်း အချိုးပင် ဖြစ်ပါတယ်။ Trigo အချိုးများကို သိနိုင်ရန် Unit Circle အကြောင်းကို သိရှိဖို့ လိုအပ်ပါသည်။

Unit Circle in Trigonometric

စက်ဝိုင်းတစ်ခုအား degree ဖြင့် မှတ်မည်ဆိုလျှင် 360* degree ရှိပြီး radian အနေဖြင့် ဆိုလျှင် 2π radian ဖြစ်သည်။ ပထမဆုံးအနေနဲ့ zero နေရာကနေ စကြည့်ပါ။ unit circle က counter - clockwise အတိုင်း သွားပါတယ်။ စက်ဝိုင်း တစ်ခုလုံးရဲ့ တစ်ပတ် ပြည့်သွားတဲ့ တန်ဘိုးဟာ 2π radian ဖြစ်ပါတယ်။ ထို့ကြောင့် အစ သည် zero ဖြစ်ပြီး အဆုံးသည် 2π radian ဖြစ်သည်။ အထက်ပါပုံတွင် အနီရောင်မြှားဖြင့် ဖော်ပြထားပါသည်။ x ဝင်ရိုးတွင် ရှိသော အပြာရောင်နှင့် ပြထားသော နေရာရဲ့ radian တန်ဘိုးသည် π ဖြစ်သည်။ အဘယ်ကြောင့်ဆိုသော် 2pi ရဲ့ တစ်ဝက်သည် pi ဖြစ်သည်။ ထိုနည်းတူ အပြာရောင်နှင့် ပြထားသော y ဝင်ရိုးရှိ radain value သည် π/2 ဖြစ်သည်။ အဘယ်ကြောင့်ဆိုသော် π တစ်ခုရဲ့ တစ်ဝက်သည် π/2 ဖြစ်သည်။ အစိမ်းရောင်နှင့် ပြထားသော Y ဝင်ရိုးရှိ value သည် 3π/2 ဖြစ်သည်။ အဘယ်ကြောင့်ဆိုသော် အစိမ်းရောင်နှင့် ပြထားသည့် Y ဝင်ရိုးနေရာသည် 2 သို့ ရောက်ရင် 0.5 သာ လိုတော့သည့်အတွက် 1.5π ဖြစ်သည့် တစ်နည်းအားဖြင့် 3π/2 ဖြစ်သည်။

45* , 135* , 225* နှင့် 315* တို့အတွက် radian value တွေကို ဆက်လက်ပြီး သတ်မှတ်ပါမည်။ ပထမဆုံး 45* degree နေရာသည် π တစ်ခုရဲ့ လေးပုံ ပုံလျှင် တစ်ပုံဖြစ်တဲ့ အတွက် π over 4 (π/4) ဟုတ် မှတ်ယူနိုင်သလို တစ်နည်းအားဖြင့် ပြောမည် ဆိုလျှင် 1π/4 ဖြစ်သည်။ ထို့ကြောင့် 135* degree ဖြစ်တဲ့ သုံးစိတ်မြောက် နေရာကိုတော့ 3π/4 ဟုတ် မှတ်ယူ နိုင်ပါသည်။ ထိုနည်းတူ 5 စိတ်မြောက်နေရာ ဖြစ်တဲ့ 225* ရှိတဲ့ နေရာကိုတော့ 5π/4 ဟု မှတ်ယူ နိုင်ပြီး 315* degree ဖြစ်တဲ့ 7 စိတ်မြောက်နေရာကိုတော့ 7π /4 ဟု မှတ်ယူ နိုင်ပါသည်။

အထက်ပါ ပုံမှာတော့ Degree အနေဖြင့် ကြည့်မည်ဆိုလျှင် 360 degree ကို 12 ပုံ ပိုင်းထားတာ ဖြစ်ပါတယ်။ အနီရောင်ဖြင့် ပြထားသော မျဉ်းများကို ကြည့်ပါ။ radian အနေဖြင့် ကြည့်မည်ဆိုလျှင် 2π ရဖို့ အတွက်ဆိုလျှင် 12π ကို 6 ဖြင့် စားလျှင် နောက်ဆုံးအနေဖြင့် 2π သေချာပေါက် ရပါမည်။ ထို့ကြောင့် ပထမ radianအစိတ်တိုင်းအား 6 ဖြင့် စားပေးခြင်းဖြင့် radian value အတိအကျကို သေချာပေါက် ရပါမည်။ ပထမ 30 degree တွင် π/6 ထိုနည်းတူ ဒုတိယ 60 degree တွင် 2π/6 (တစ်နည်း အားဖြင့်

အထက်အောက် ချေလိုက်လျှင် 60 degree အတွက် ᾳ/3) ကို ရပေလိမ့်မည်။ ကျန်သည့် အစိတ်များ အားလုံးလည်း ထိုနည်း အတိုင်း ပင်ဖြစ်သည်။

အထက်တွင် ဖော်ပြထားသော တွက်ချက်နည်းများသည် 30 degree တစ်နည်းအားဖြင့် ᾳ/6 အတွက် ဖြစ်သည်။ 15 degree အတွက်ဆိုရင်ရော ဘယ်လိုပုံစံဖြင့် အလွယ်ဆုံး မှတ်နိုင်မည်နည်း။ ဘယ်လောက် degree ပဲ စိတ်ပိုင်းသည် ဖြစ်စေ စက်ဝိုင်း တစ်ပတ်ပြည့်သွားသောအဆုံးတွင် radian value 2ᾳ သေချာ ပေါက် ရ ရမည်ဖြစ်သည်။ 15 degree အတွက်ဆိုလျှင် 15*12 = 180 ဖြစ်သည့် အတွက် စက်ဝိုင်း တစ်ပိုင်းလျှင် 12 စိတ်ရှိမည် ဖြစ်သည်။ စက်ဝိုင်း တစ်ခုလုံး အတွက်ဆိုလျှင် 12*2=24 ရှိမည် ဖြစ်သည်။ ထို့ကြောင့် 24/2 = 2 ဖြစ်သည်။ ပထမဆုံး တစ်စိတ် 15 degree အတွက် ᾳ/12 ဒုတိယ 30 degree အတွက် 2ᾳ/12 or ᾳ/6 ဖြစ်မည်။ ကျန်အစိတ်ပိုင်းများ အားလုံးသည်လည်း ထိုနည်း အတိုင်းပင် အလွယ်တကူ မှတ်သား တွက်ယူ နိုင်ပါသည်။ အောက်တွင် ပုံနှင့် ဖော်ပြထားပါသည်။

Sine , Cosine , Tangent

Degree များတွင် x , y value များ သတ်မှတ်ချင်းများကို ဖော်ပြ ပေးသွားပါမည်။ ပထမဆုံး အနေဖြင့် sine , cos , tan များ ရဲ့ တွက်နည်းများကို ဦးစွာ သိရန် လိုအပ်ပါသည်။

ထောင့်မှန်တြိဂံ တစ်ခုမှာ Hypothesis , Adjance , Opposite ဆိုတဲ့ အနားသုံးခု ရှိပါတယ်။ ထိုထောင့်မှန် တြိဂံရဲ့ sine , cos ,tan value တွေကို ရှာတဲ့ အခါမှာ SOH ,CAH , TOA ကို သုံးပြီး ရှာနိုင်ပါတယ်။ ဆိုလိုချင်တာက sin value ကိုလိုချင်ရင် opposite ဆိုတဲ့ y ကို

hypotenuse ဆိုတဲ့ z နဲ့ စားပေးရမှာပါ။ cos ကိုလိုချင်ရင်လည်း adjacent ကို hypotenuse နဲ့စားရမှာ ဖြစ်သလို tan ကို လိုချင်ရင်လည်း opposite ကို adjacent နဲ့ စားပေးရမှာပါ။

ထောင့်မှန်တြိဂံ တစ်ခုမှာ $x^2+y^2=r^2$ (x square and y square) ပေါင်းခြင်းဟာ r(radius) square နဲ့ တူတယ်လို့ ဆိုပါတယ်။ Pythagoras က ပြောတာပါ။ ထိုနည်းတူ Unit Circle မှာ r(radius) value သည် 1 ဖြစ်ပါတယ်လို့ ပြောထားပါတယ် ထို့ကြောင့် $x^2+y^2=1$ ဖြစ်ပါတယ်။ x နှင့် y ဆိုတာက unit circle မှာ ရှိတဲ့ x and y ကြားက point လေးတွေပါ။ အောက်ပါ ပုံတွင် ကြည့်ပါ။

UNIT CIRCLE: $X^2 + Y^2 = 1$

မှတ်သားထားရန် တစ်ခုမှာ ထောင့်မှန်တြိဂံ တစ်ခုမှာ ကျန် နှစ်ထောင့် သည် 45* degree စီ အသီးသီး ရှိမည် ဆိုလျှင် x နှင့် y သည် အတူတူသာပင် ဖြစ်သည်။ ထို့ကြောင့် အောက်ပါအတိုင်း မှတ်ယူနိုင်ပါတယ်။ အောက်ပါတွက်ချက်ပုံအား ကြည့်ချင်းအားဖြင့် x နှင့် y value သည် √(½) ဖြစ်သည်။ ထို √(½) အား ရှင်းလိုက်မည်ဆိုလျှင် √2/2 ကို ရရှိမှာ ဖြစ်ပါတယ်။ ထို့ကြောင့် 45* degree မှာ ဆိုလျှင် x နှင့် y value များသည် အောက်ပါ ပုံထဲမှ အတိုင်း ဖြစ်သည်။

အထက်တွင် ရရှိလာသော √2/2 အား နောက်ဆုံး အဖြေရသည် အထိ ရှင်းကြည့်မည် ဆိုလျှင် 0.707 ကို ရရှိပါမည်။ တစ်နည်းအားဖြင့် x နှင့် y value များ ကို အောက်ပါ ပုံထဲမှ အတိုင်း မှတ်ယူ နိုင်ပါသည်။

မိမိတို့ရဲ့ NumPy သို့ ပြန်သွားပြီး 45* degree ရဲ့ radian ကို ရှာပါ ထိုမှ ရလာသော radian value ကို sine function ထဲသို့ ထည့်ကြည့်လျှင် output အနေဖြင့် 0.707 ကို သေချာပေါက် ပြန်ရပါမည်။ 45* degree ရဲ့ radian value သည် 0.78540 ဖြစ်သည်(အထက်တွင် radian value ရှာနည်း ဖော်ပြခဲ့ပြီးဖြစ်သည်)

```
In [2]: import numpy as np
        radian = 0.78540
        np.sin(radian)

Out[2]: 0.7071080798594735

In [ ]:
```

ဘာကြောင့် x တန်ဘိုး အနေဖြင့် 0.707 ဟု ယူလိုက်ရသနည်း?

SOH , CAH , TOA ကို ပြန်သွားမည် ဆိုလျှင် sine ကိုလိုချင်လျှင် opposite ကို hypotenuse နဲ့ စားပေးရမည် ဖြစ်သည်။

အထက်ပါ နှစ်ပုံကို ယှည်ကြည့်လိုက်မည်ဆိုလျှင် sine ကို လိုချင်လျှင် y ကို z နဲ့ စားပေးရမှာပါ။ ထို့ကြောင့် y တန်ဘိုး သည် y/z သည် 0.0707/1 ဖြစ်သည့် အတွက် 0.707 ကိုသာ ရ ရှိခြင်း ဖြစ်ပါသည်။ ထို့ကြောင့် 45* degree သည် radian value အနေဖြင့် 0.78540 ဖြစ်ပြီး sine value အနေဖြင့် 0.707 ဖြစ်သည်။

Degree = 45*
Radian = 0.78540
Sine = 0.707

ရှေ့ပိုင်း သင်ခန်းစာများ တွင် 45* အတွက် တွက်ချက်ပြီးသွားပြီ ဖြစ်သည့် အတွက် ယခု ဆက်ပြီး 60* degree အတွက်ကို တွက်ပါမည်။ Triangle တစ်ခုရဲ့အတွင်းထောင့်များ အားလုံးပေါင်းခြင်း သည် 180 degree ဖြစ်ပါတယ်။ ထိုနည်းတူ sides တွေ အားလုံးသာ တူညီ နေခဲ့မည် ဆိုလျှင် အတွင်းထောင့်များသည် 60 degree စီ ကိုယ်စီ ရှိနေကြမှာ ဖြစ်ပါတယ်။ ထို 60 degree စီ ကိုယ်စီရှိနေကြတဲ့ unit circle အတွင်းမှာ triangle တစ်ခုအား အလယ်တည့်တည့်မှ မျဉ်းဆွဲချလိုက်မည်ဆိုလျှင် အောက်ပါ ပုံအတိုင်း တွေ့ရပါမည်။

$x^2 + y^2 = 1$ ဖြစ်တဲ့ အတွက် x value သည် ½ ဖြစ်ပါတယ်။ အဘယ်ကြောင့် ½ ဟု ပြောရလဲ ဆိုလျှင် r value သည် 1 ပါ ထို 1 ကို တစ်ဝက် ဝက်တာ ဖြစ်တဲ့ အတွက် ½ ဟု ဆိုခြင်း ဖြစ်ပါသည်။

$(½)^2 + y^2 = 1$

$¼ + y^2 = 1$

$y^2 = 1 - ¼ = 1 - 0.25$

$y^2 = 0.75 = ¾$

$y = \sqrt{3/4}$

$y = \sqrt{3}/2$

Y တန်ဖိုးကိုပါ ရပြီဆိုတော့ ယခင် 45 degree value ကို ရှာခဲ့သည့် အတိုင်း sin , cos , tan တို့ကို ရှာလို့ ရသွားပါပြီ။ အောက်တွင်လည်း ပုံနှင့် တကွ အသေးစိတ် တွက်ပြထားပါသည်။

30 degree ကို ရှာရန် အထက်ပါ ပုံကို ကြည့်ပါ။ r, y ,x ဆိုပြီး အနီရောင် နောက်ခံ ဖြင့် ရေးထားသော triangle သည် 30 degree အတွက် ဖြစ်ပြီး R , Y , X ဆိုပြီး ပန်းရောင်နောက်ခံဖြင့် ရေးထားသော triangle သည် 60 degree အတွက် ဖြစ်သည်။ ပထမ triangle သည် လည်း 90 , 60 , 30 ဖြင့် 180 degree ဖြစ်သလို ဒုတိယ triangle သည်လည်း 90 ,60,30 ဖြင့် 180 degree ဖြစ်ပါသည်။ တစ်နည်း အားဖြင့် ဆိုလျှင် ထို triangle နှစ်ခုသည် အတူတူပင် ဖြစ်သည်။ ထို့ကြောင့် x , y ,r တန်ဖိုးများသည်လည်း အတူတူပင် ဖြစ်ပါသည်။ ထို့ကြောင့် 30 degree triangle ရဲ့ y တန်ဖိုးသည် ½ ဖြစ်ပြီး x တန်ဖိုးသည် √3/4 ဖြစ်သည် ။ x နှင့် y တန်ဖိုးအား ပြောင်းပြန်လှန်လိုက် ရုံသာ ဖြစ်သည်။ x နှင့် y တန်ဖိုးများ ရသွားပြီ ဖြစ်သည့် အတွက် sine value ကိုလည်း တွက်ထုတ်လို့ ရနိုင်ပါပြီ။ ယခု တစ်ခေါက် 30 degree အတွက် sine value ကို မိမိတို့ဘာသာ အစဉ်လိုက် တွက်ကြည့်စေချင်ပါသည် အကယ်၍ အခက်ခဲများ ရှိခဲ့ပါက ယခင် သင်ခန်းစာ တွက်နည်းများကို ကြည့်ရှုနိုင်ပါသည်။ 30 degree ရဲ့ sine value သည် 0.50000 ဖြစ်ပါသည်။

Rounding Functions

Numpy မှာ decimal float number တွေကို မိမိလိုသလို ဖြတ်ထုတ်ရန် function များစွာကို support လုပ်ထားပါတယ်။ ဥပမာ numpy array တစ်ခုထဲတွင်ရှိသော decimal float number တွေကို ဒသမ နောက် 2 နေရာထိပဲ ယူပြီး ဒသမနောက် 2 နောက်မှာ ရှိနေတဲ့ value သည် 5 ထက် ကျော်နေခဲ့လျှင် တစ်ခါတည်း ရှေ့ကို တစ်တိုးပေးပြီး ထုတ်ပြပေးနိုင်သည်။ ထိုသို့ ပြုလုပ်နိုင်ရန် အတွက် around ဆိုသည့် function ကို အသုံးပြု နိုင်ပါသည်။ around function သည် parameter နှစ်ခု ယူပါသည်။ ပထမ တစ်ခုသည် number or array ဖြစ်ပြီး ဒုတိယ တစ်ခုသည် မိမိလိုချင်သည့် ဒသမ နောက်မှ number အရေအတွက် ဖြစ်သည်။

အကယ်၍ ဒုတိယ parameter နေရာတွင် မည်သည့် value မှ မထည့်ပေးလျှင် default အနေဖြင့် zero ဖြစ်ပါသည် zero ဆိုလျှင် ဒသမ ရှေ့မှ value များကိုသာ ယူပါသည်။ ထိုနည်းတူ ဒုတိယ parameter နေရာတွင် -1 ဟု ရေးမည် ဆိုလျှင်လည်း ဒသမ ရှေ့မှ value များကိုသာ ယူပါသည်။ sample program ကို အောက်တွင် ဖော်ပြထားပါသည်။

```
In [10]: import numpy as np
         arr = np.array([12.205, 90.23120])
         print("Original array values:",end = " ")
         print(arr)
         print("Rounding by 2 decimal position",np.around(arr,2))
         print("Rounding by 0 - default decimal position",np.around(arr))
         print("Rounding by -1 decimal position",np.around(arr, -1))

         Original array values: [12.205  90.2312]
         Rounding by 2 decimal position [12.2  90.23]
         Rounding by 0 - default decimal position [12. 90.]
         Rounding by -1 decimal position [10. 90.]
```

အကယ်၍ ဒသမ ရှေ့မှ value များသာ လိုချင်သော အခါတွင်လည်း floor() function ကို အသုံးပြုလို့ ရပါသေးတယ်။ floor function သည် parameter အဖြစ် number or array ကို ယူပြီး return အနေဖြင့် ဒသမ ရှေ့မှ value များကိုပြန်ပေးပါတယ်။ သတိပြုရန် အချက်မှာ output value များသည် input ပေးလိုက်သော value များနှင့် တူညီသော ဒသမ ရှေ့မှ value များကို ပြန် ပေးပါသည်။

```
In [12]: import numpy as np
         arr = np.array([11.202, 09.23120, 55.020, 22.209])
         print(np.floor(arr))

         [11.  9. 55. 22.]
```

Ceil function သည် floor function နဲ့ လုပ်ဆောင်ပုံ တူညီသော်လည်း output value သည် input value ထက်ပိုကြီးသော ဒသမရှေ့မှ value များကိုသာ ပြန်ပေးပါသည်။ Ceil function အသုံးပြုပုံကို အောက်တွင် ဖော်ပြထားပါသည် output value များသည် မူရင်း input value ထက် 1 ကြီးနေသည်ကို သတိပြုပါ။

```
In [14]: import numpy as np
         arr = np.array([11.902, 09.23120, 55.020, 22.209])
         print(np.ceil(arr))

         [12. 10. 56. 23.]
```

Quadrants of the coordinate plane

ယခု lesson မှာတော့ x ,y အမှတ်နှစ်ခုဟာ ဘယ် Quadrant ထဲမှာ ပါသလဲ ဆိုတာကို ရှာကြည့်မှာ ဖြစ်ပါတယ်။ (1 , 5) , (-7 , 5) , (-5,-1) , (3 , -6)

အထက်မှာ ဖော်ပြထားတဲ့ ပုံကတော့ quadrant လေးခုကို ဖော်ပြပေးထားတာ ဖြစ်ပါတယ်။ ပထမဆုံး (1 , 5) သည် ဘယ် quadrant ထဲမှာ ပါသလဲ ဆိုတာကို ရှာရမှာ ဖြစ်ပါတယ်။ အောက်ပါ ပုံကို ကြည့်မည် ဆိုလျှင် (1 ,5) သည် quadrant I ထဲမှာ ပါသည်ကို မြင်ရပါမည်။

အောက်ပါပုံထဲမှာ အမှတ်လေးခုလုံးကို ဖော်ပြထားပါတယ်။ B , E , G , J အမှတ်များကို ကြည့်မည် ဆိုလျှင် ထိုအမှတ်များသည် ဘယ် quadrant ထဲတွင် ကျရောက်နေသလဲ ဆိုတာကို အလွယ်တကူ တွေ့နိုင်မှာ ဖြစ်ပါတယ်။

(1 , 5) Quadrant I

(-7 , 5) Quadrant II

(3 , -6) Quadrant IV

(-5 , -1) Quadrant III

Two - variable linear equation introduction

Linear ဆိုတာက အမှတ် သို့မဟုတ် data အချက်လက်တွေကို graphical နည်းလမ်းဖြင့် straight line ပုံစံ ဖော်ဆောင်နိုင်ခြင်း ဖြစ်သည်။ linear equation ဆိုတာကတော့ equation တစ်ကြောင်းမှာ ပါဝင်တဲ့ x and y အမှတ်တွေကို မျဉ်းဆွဲကြည့်မည့် ဆိုလျှင် အဖြောင့်တိုင်း straight line သာ ရရှိမှာ ဖြစ်ပါတယ်။ အကယ်၍ straight line မဖြစ်ခဲ့ဘူးဆိုလျှင် non-linear ဖြစ်သွားမှာ ဖြစ်ပါတယ်။ ဥပမာ အနေဖြင့် y = 2x -3 သည် linear equation တစ်ခု ဖြစ်ပါသည်။ အဘယ်ကြောင့် linear equation ဖြစ်ရသည် ဆိုတာကို အောက်တွင် step by step ရှင်းပြ ပေးသွားပါမည်။

y=2x - 3 ဆိုသည့် equation ၌ x တန်ဘိုးသည် 0 ဖြစ်ခဲ့မည်ဆိုလျှင် y တန်ဘိုးသည် -3 ဖြစ်မှာပါ။

y = (2*0) - 3

y = -3

အကယ်၍ x တန်ဘိုးသည် 1 ဖြစ်ခဲ့မည် ဆိုလျှင် y တန်ဘိုးသည် -1 ဖြစ်မှာပါ။

y= 2x - 3

y= (2*1) -3

y= 2 - 3

y= -1

အကယ်၍ x တန်ဘိုး သည် 2 ,3 ဖြစ်ခဲ့မည်ဆိုလျှင်လည်း y တန်ဘိုးသည် 1 , 3 တို့ကို ရမှာပါ။ တွက်နည်းသည် အထက်တွင် ဖော်ပြထားပြီး အတိုင်းသာ ဖြစ်သည် ။

အထက်တွင် ဖော်ပြထားသော ဇယား အား graph ဆွဲကြည့်မည် ဆိုလျှင် အောက်ပါ အတိုင်း ရရှိမှာ ဖြစ်ပါတယ်။ အောက်ပါ link တွင် မိမိ ကိုယ်တိုင် graph ဆွဲ နိုင်ပါတယ်။

https://www.geogebra.org/classic

x,y အမှတ်များကို ဖြတ်ပြီး ဆွဲထားသော အစိမ်းရောင်ဖြင့် မျဉ်းကြောင်းသည် straight line သာ ဖြစ်သည့် အတွက် y= 2x - 3 သည် linear equation တစ်ခု ဖြစ်ပါသည်။ အကယ် x သည် 1 ဖြစ်ခဲ့မည် ဆိုလျှင် graph ကို ကြည့်ပါက y တန်ဘိုးသည် -y ဖြစ်နေသည်ကို မြင်ရပါမည် အောက်ပါ graph တွင် ကြည့်ပါ။

အထက်တွင် ဖော်ပြထားသော graph ၌ A , D , G , J အမှတ်အားလုံး မျဉ်းဆွဲမည် ဆိုလျှင် straight line တစ်ခုကို ရရှိမှာ ဖြစ်ပါတယ်။ ထို့ကြောင့် y=2x - 3 သည် linear equaiton တစ်ခု ဖြစ်ပါတယ်။

နောက်ထပ် ပုစ္ဆာ တစ်ခုအနေဖြင့် 4x-3y = 12 ဆိုတဲ့ equation တစ်ခုကို တွက်ပြီး graph ချ ကြည့်ပါမယ်။ ပထမဆုံး x နေရာမှာ 0 နှင့် အစားထားပြီး y တန်ဖိုးကို ရှာကြည့်မှာ ဖြစ်ပါတယ်။

အထက်ပါ အတိုင်း တွက်ချက်ပြီး output များကို ကြည့်မည်ဆိုလျှင် x တန်ဘိုး 0 ဖြစ်ခဲ့လျှင် y တန်ဘိုးသည် - 4 ဖြစ်ပါတယ်။ ထိုနည်းတူ x တန်ဖိုး 1 ဖြစ်ခဲ့လျှင် y သည် -2.6 ဖြစ်ပါတယ်။ ထို x , y အမှတ်များအား graph ဆွဲကြည့်မည်ဆိုလျှင် အောက်ပါ အတိုင်း ရရှိပါမည်။ အောက်ပါ ပုံတွင် B , C , E , H အမှတ်များကို ကြည့်လျှင် 4x-3y=12 သည် Linear ဖြစ်သည်ကို တွေ့ရပါမည်။

Intercepts

x ဝင်ရိုးကို ဖြတ်သွားတဲ့ graph line ကို x intercept ဟု ခေါ်ပြီး y ဝင်ရိုးကို ဖြတ်သွားတဲ့ graph line ကို y intercept လို့ ခေါ်ပါတယ်။ ပထမဆုံးအနေဖြင့် y = ½ x -3 ဆိုတဲ့ Linear equation မှာ x တန်ဘိုး 1 ဖြစ်ရင် y တန်ဘိုး ဘယ်လောက် ဖြစ်မလဲ ဆိုတာကို တွက်ပါမည်။

$y = \frac{1}{2}x - 3$

if $x = 0$,

$y = (\frac{1}{2} \times 0) - 3$

$y = -3$

if $x = 2$,

$y = (\frac{1}{2} \times 2) - 3$

$= 1 - 3$

$= -2$

x	y
0	-3
2	-2
4	-1
6	0

if $x = 4$,

$y = (\frac{1}{2} \times 4) - 3$

$= 2 - 3 = -1$

if $x = 6$,

$y = (\frac{1}{2} \times 6) - 3$

$= 3 - 3 = 0$

အောက်ပါ ပုံတွင် ဖော်ပြထားသော C အမှတ်သည် x intercept ဖြစ်ပြီး B အမှတ် သည် y intercept ဖြစ်ပါတယ်။

ဆက်လက်ပြီး linear equation တစ်ခုကနေ x and y intercepts များ ရှာကြည့်ကြပါစို့။

-5x + 4y = 20

$-5x + 4y = 20$

if $x = 0$,

$4y = 20$

$y = 5$

if $y = 0$, $x = ?$

$-5x + 4y = 20$

$-5x = 20$

$x = -4$

ထို့ကြောင့် x =0 ဖြစ်ခဲ့လျှင် y = 5 ဖြစ်ပြီး y =0 ဖြစ်ခဲ့လျှင် x = -4 ဖြစ်ပါတယ်။ ထိုအမှတ် နှစ်ခုသည် x and y intercept များ ဖြစ်ပါတယ်။

Intro to slope

Slope ဆိုတာ vertical change ဆိုတဲ့ Δy နဲ့ horizontal change ဆိုတဲ့ Δx တို့ရဲ့ အချိုးဖြစ်ပါတယ် သို့မဟုတ် လိုင်းတစ်ခုရဲ့ direction နဲ့ steepness(မတ်စောက်) ခြင်းတို့ကို ဖော်ပြတဲ့ number တစ်ခု ဖြစ်ပါတယ်။ ဥပမာအားဖြင့်ဆိုရလျှင် line တစ်ခုသည် အောက်ကနေ အထက်သို့ ၄င်း၊ အထက်ကနေ အောက်သို့ ၄င်း၊ တိုးတာနဲ့ လျော့တာမျိုးဖြစ်နိုင်သလို horizontal or vertical လိုမျိုးလည်း ဖြစ်နိုင်ပါတယ်။ ထိုကဲ့သို့ အခြေနေမျိုးများတွင် vertical and horizontal ရဲ့ ratio value သည် slop ဖြစ်ပါတယ်။ y ကို vertical change လို့ခေါ်သလို x ကိုလည်း horizontal change လို့ ခေါ်ပါတယ်။ (Δ ဒီသကေ်လေးကိုတော့ delta လို့ခေါ်ပါတယ်) အောက်ပါပုံထဲမှာဆိုလျှင် slope နှစ်ခု ဆွဲပြထားပါတယ်။ ပထမပုံ အသေးထဲတွင် CD ဆိုတဲ့ မျဉ်းလေးသည် အကွက် နှစ်ခုစာ အကွာ အဝေးရှိသလို DE ဆိုတဲ့ မျဉ်းသည်လည်း အမှတ်နှစ်ခုစာ အကွာအဝေးရှိပါတယ်။ CD သည် x ဖြစ်ပြီး DE သည် y ဖြစ်သည့်အတွက် ထိုနှစ်ခုအား စားလျှင် 1 သာ ရပါသည်။ ထို့ကြောင့် slope တန်ဘိုးသည် 1 ဖြစ်ပါတယ်။

slope(m) = Δy/Δx

Negative Slope

ယခု သင်ခန်းစာမှာတော့ negative slope ကို ဖော်ပြပေးပါ့မယ် x တန်ဘိုး တိုးလားတိုင်း y တန်ဘိုး ဘယ်လောက်ထိ ကျသွားသလဲ ဆိုတာကို တွက်ပြီး slope ရဲ့ တန်ဘိုး ကို ဖော်ပြမှာပါ။

ပထမ ပုံ အသေးတွင် x တန်ဘိုးသည် 1 ဖြစ်ပြီး y တန်ဘိုးသည် decree ဖြစ်သွားတဲ့ အတွက် -2 ဖြစ်ပါတယ် ။

Slope = -2/1

Slope = -2

ထို့ကြောင့် ပထမပုံတွင် slope တန်ဘိုးသည် -2 ဖြစ်ပါတယ်။

ဒုတိယ ပုံတွင် x သည် 2 ဖြစ်ပြီး y သည် -4 ဖြစ်ပါတယ်။ ထို့ကြောင့် ဒုတိယပုံတွင် ဆွဲထားသော graph ရဲ့ slope သည် -2 ဖြစ်ပါတယ်။

အထက်တွင် ဖော်ပြခဲ့သော slope များသည် point တစ်ခုတည်း အပေါ်တွင် ယူခြင်းဖြစ်ပြီး အကယ်၍ point နှစ်ခု ဖြစ်နေခဲ့သည် ဆိုလျှင် slope တန်ဘိုးကို မည်ကဲ့သို့ တွက်မည်နည်း။ ထိုကဲ့သို့ point နှစ်ခု အတွက်ဆိုလျှင် y2-y2 / x2-x1 ဆိုသည့် formula ကိုသုံးနိုင်ပါတယ် ထို့ကြောင့် y2-y1 သည် Δy ကို ရမည်ဖြစ်ပြီး x2-x1 သည် Δx ကို ရမှာ ဖြစ်ပါတယ်။

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

$$\Delta y = y_2 - y_1$$

$$\Delta x = x_2 - x_1$$

$$2 \text{ points} = \underset{x_1 \quad y_1}{(4, 2)}, \underset{x_2 \quad y_2}{(-3, 16)}$$

$$= \frac{y_2 - y_1}{x_2 - x_1}$$

$$= \frac{16 - 2}{-3 - 4} = \frac{14}{-7}$$

$$m = -2$$

အထက်တွင် တွက်ပြထားသည့် ပုစ္ဆာတွင် (4,2) , (-3,16) စသည့် Point နှစ်ခုအား သုံးပြီး slope တန်ဖိုးကို တွက်ပြထားခြင်း ဖြစ်ပါသည်။ ယခု ဆက်လက်ပြီး ထို point နှစ်ခုအား visualize ဖြင့် graph ဆွဲပြီး ဖော်ပြပါမည်။

Point နှစ်ခုရဲ့ အမှတ်များအား အထက်ပါ slope ကိုကြည့်ပြီး သိနိုင်သည်။ဆက်လက်ပြီး slope တန်ဖိုး -2 အား visualize လုပ်ပြပါမည်။

အထက်ပါ ပုံတွင် အစိမ်းရောင်မျဉ်းနှင့်ပြထားသော FG(vertical x) သည် 7 ဖြစ်ပြီး GB (horizontal y)သည် အောက်ကို decree သွားတာ ဖြစ်တဲ့ အတွက် -14 ဖြစ်ပါသည်။ ထို့ကြောင့် -14/7 သည် slope တန်ဘိုး -2 ကို ရရှိစေပါသည်။

Linear Regression Using Least Squares Method (Best Fit Line)

Least Squares သည် ရရှိထားသော data set များအပေါ်မှာ regression analysis ကို အသုံးပြုပြီး line of best fit ကို ဆွဲတဲ့ နေရာမှာ အသုံးပြုတဲ့ method တစ်ခု ဖြစ်ပါတယ်။ Data point တစ်ခုခြင်းစီဟာလည်း dependent variable and independent variable ရဲ့ ဆက်စပ်မှု ကိုဖော်ပြ ပါတယ်။ ဥပမာ - ဆိုင် တစ်ဆိုင်မှာ ပန်းသီး တစ်သောင်းဖိုး ဝယ်ရင် ပန်းသီး ၁၀ လုံးပြန်ရပါတယ်။ ပိုက်ဆံ တစ်သောင်းသည် x (independent) ဖြစ်ပြီး ပန်းသီးသည် x များရင် များသလို လိုက်ပြောင်းနေမှာ ဖြစ်တဲ့အတွက် y (depended) ဖြစ်ပါတယ်။ ထို့ကြောင့် Regression Analysis မှာ dependent variables ကို vertical y-axis မှာ ဖော်ပြပြီး independent variables ကို horizontal x-axis မှာ ဖော်ပြပါတယ်။ ထိုနှစ်ခုကြားမှ best fit line ကိုတော့ least squares method မှ ရရှိမှာပါ။ အနှစ်ချုပ် အနေဖြင့် မှတ်သားရမည် ဆိုလျှင် least squares regression သည် dependent variable ရဲ့ အတိုးအလျော့ ဖြစ်ပျက်မှုကို predict(ကြိုတင်ခန့်မှန်းခြင်း) ဖြစ်ပါတယ်။

X	1	2	3	4	5	6	7
Y	1.5	3.8	6.7	9	11.2	13.6	16

အထက်မှာ ဖော်ပြထားတာကတော့ x and y example data set တစ်ခုဖြစ်ပါတယ်။

ပုံထဲမှာ ဖော်ပြထားတာကတော့ data point လေးတွေမှာ မျဉ်းလိုင်းတွေ ဆွဲပြထားတာ ဖြစ်ပါတယ်။ သို့သော် ထိုမျဉ်းလိုင်းများသည် အမှန် မဟုတ်သလို အကောင်းဆုံး မျဉ်းလိုင်းလည်း မဟုတ်ပါဘူး။ အကောင်းဆုံး မျဉ်းလိုင်း ရဖို့ ဆိုရင် တွက်ထုတ်ရမှာ ဖြစ်ပါတယ်။ ထိုသို့ တွက်ထုတ်ဖို့ အတွက် linear equation ကတော့ y = mx +b ဖြစ်ပါတယ်။ Statistics မှာ ဆိုရင်တော့ y = b0 + b1x ကိုသုံးပါတယ်။ ထို့ကြောင့် m ဆိုတဲ့ slope သည် b1 နှင့် အတူတူပင် ဖြစ်ပါတယ်။

ထိုနည်းတူ algebra မှ y - intercept ဖြစ်သည့် b နှင့် statistics မှ b1 သည်လည်း အတူတူပင် ဖြစ်ပါတယ်။

ပထမဆုံးအနေဖြင့် data set ထဲမှ slope (m) ကို တွက်ထုတ်ဖို့ လိုအပ်ပါတယ်။ slope ကိုတွက်ထုတ်ရာမှာ အသုံးပြုရမည့် equation သည် အောက်ပါအတိုင်း ဖြစ်ပါတယ်။

Equation ရဲ့ အပေါ်ပိုင်းကို စကြည့်မယ် ဆိုရင် n∑xy ကို စမြင်ရမှာပါ။ n သည် number of dataset ကို ဆိုလိုတာပါ။ ကျွန်တော်တို့ အသုံးပြုမယ့် data set မှာဆိုရင် data အနေဖြင့် x သည်လည်း 7 ခုရှိသလို y သည်လည်း 7 ခုပါပဲ။ ထို့ကြောင့် n သည် 7 ဖြစ်ပါတယ်။ နောက်တစ်ခု အနေဖြင့် ∑xy သည် x နှင့် y နှစ်ခု မြှောက်ပြီး ရလို့ရတဲ့ ဟာ အားလုံးရဲ့ ပေါင်းခြင်းကို ဆိုလိုတာပါ။ ∑x သည် x value အားလုံးရဲ့ ပေါင်းခြင်းဖြစ်ပြီး ∑y သည် y value အားလုံးရဲ့ ပေါင်းခြင်း ဖြစ်ပါတယ်။ equation ရဲ့ အောက်ဘက်ခြမ်းကို ကြည့်မည်ဆိုလျှင်လည်း n∑x^2 နှင့် (∑x)^2 တို့ကို မြင်ရမှာပါ။ ဤနေရာမှာ သတိပြုရမည့်အချက်သည် ∑x^2 နှင့် (∑x)^2 သည် မတူပါဘူး။ ∑x^2 သည် x^2 value အားလုံးရဲ့ ပေါင်းခြင်းကို ဆိုလိုခြင်းဖြစ်ပြီး (∑x)^2 သည် x value အားလုံး ∑ ပေါင်းခြင်း ရဲ့ နှစ်ထပ်

ဖြစ်ပါသည်။ ထို့ကြောင့် equation ရဲ့ လိုအပ်သော parameters အားလုံးကို အောက်ပါအတိုင်း ဇယား ဖြင့် တွက်ချက်ပြီး ဖော်ပြထားပါသည် ။

ယခုပုံတွင် ဖော်ပြထားသော value များသည် ကျွန်ုပ်တို့ equation အတွက် လိုအပ်သော parameters များကို တွက်ထားခြင်း ဖြစ်ပါတယ်။

X	Y	XY	X^2	Y^2
1	1.5	1.5	1	2.25
2	3.8	7.6	4	14.44
3	6.7	20.1	9	44.89
4	9.0	36	16	81
5	11.2	56	25	125.44
6	13.6	81.6	36	184.96
7	16	112	49	256
28	61.8	314.8	140	708.98

ယခု ဆိုလျှင် equation အတွက် လိုအပ်သော value များကို အောက်ပါ အတိုင်း ထည့်နိုင်ပါပြီ။

ပထမဆုံး အနေဖြင့် အောက်ပါ အတိုင်း value များကို အစားထိုးပါမည်။

$$m = \frac{7(314.8) - (28)(61.8)}{7(140) - (28)^2}$$

$$\Rightarrow m = \frac{473.2}{196} = 2.4142857$$

ယခုဆိုလျှင် slope (m) သည် 2.4142857 ကို ရရှိပါမည်။ နောက်တစ်ဆင့် အနေဖြင့် m တန်ဘိုးကို ရပြီ ဖြစ်သည့် အတွက် b တန်ဘိုးကို ရှာရန်လိုအပ်နေပါသေးသည်။ သို့မှသာ x တန်ဘိုး တစ်ခု အစားသွင်းလိုက်တိုင်း y တန်ဘိုးကို တွက်ထုတ်နိုင်မှာ ဖြစ်ပါတယ်။ b တန်ဘိုး ရှာရမည့် equation သည် အောက်ပါအတိုင်း ဖြစ်ပါတယ်။

$$b = \frac{\sum y - m \sum x}{n}$$

ထို equation ထဲ၌ သက်ဆိုင်ရာ value များကို အစားထိုးမည်ဆိုလျှင် အောက်ပါအတိုင်း b တန်ဘိုးကို ရရှိပါမည်။

$$= \frac{61.8 - (2.4142857)(28)}{7}$$

$$b = -0.82857$$

Equation တွင် y = mx +b ၌ b တန်ဘိုးနှင့် m တန်ဘိုးတို့ကိုရရှိပြီ ဖြစ်သည့်အတွက် independent variable ဖြစ်သည့် x တန်ဘိုး နေရာတွင် မိမိတို့ ကြိုက်နှစ်သက်သည့် value တစ်ခုကို အစားထိုးကြည့်မည်ဆိုလျှင် Least Square Method မှ Predict လုပ်ပေးလိုက်သည့် y တန်ဘိုးကို ရရှိပါပြီ။ ထိုသို့ မပြုလုပ်ခင် မိမိတို့ ရရှိလာသော value များကို မှန်/မမှန်ကို data set နှင့် တိုက်စစ်ကြည့်ရန် လိုအပ်ပါသေးသည် ။ ထိုသို့ တိုက်စစ်ရန် y = mx +b equation ထဲသို့ မိမိ တို့ data set ထဲမှ x တန်ဘိုး တစ်ခုထည့်ကြည့်ရန် လိုအပ်ပါသည်။ ထိုသို့ ထည့်လိုက်ပြီးသည့်နောက် ရရှိလာသော value သည် y တန်ဘိုးနှင့် ကိုက်ညီနေသည် သို့မဟုတ် အနီးစပ်ဆုံးတူညီနေမည် ဆိုလျှင် ကျွန်ုပ်တို့ တွက်ချက်မှု အားလုံး မှန်ကန်ပါပြီ။ ထို့သို့ တိုက်စစ်နိုင်ရန် အောက်ပါတို့ကိုဆက်လက် လေ့လာပါ။

m = 2.41

b = - 0.83

y = mx +b

If x = 2 and y=?

y = 2.41(2) - 0.83

= 3.99

အထက်ပါ တွက်ချက်နည်းများကို ကြည့်မည်ဆိုလျှင် x သည် 2 ဖြစ်သည့် အချိန်၌ y တန်ဘိုး 3.99 ကို ရရှိပါသည် ။ မိမိတို့ data set ဇယားကိုပြန်ကြည့်ပြီး တိုက်ကြည့်ပါမည်။

X	Y	XY	X^2	
1	1.5	1.5	1	
2	3.8	7.6	4	

ဇယားတွင် x=2 ဖြစ်ပါက y သည် 3.8 ဖြစ်သည့်အတွက် ကျွန်ုပ်တို့ တွက်ချက်မှုသည် point နောက်မှ 1 သာ ကွာသည့်အတွက် အနီးစပ်ဆုံး မှန်ကန်မှု ရှိပါသည်။ ပိုမို နားလည်စေရန် အခြားသော value ဖြစ်သည့် x=5 ဖြစ်ခဲ့လျှင် y သည် ဘယ်လောက်ဖြစ်မည် ဆိုတာကို အောက်ပါအတိုင်း တိုက်ကြည့် နိုင်ပါသေးသည်။ y သည် 11.22 ရရှိသည့် data set ဇယားနှင့် ပြန်တိုက်ကြည့်မည်ဆိုလျှင် မှန်ကန် နေပါသည်။

if $x = 5 \Rightarrow y = 2.41(5) - 0.83$
$= 11.22$

မိမိတို့ တွက်ချက်နည်းများမှာ မှန်ကန်နေပြီဖြစ်သည့် အတွက် x တန်ဘိုး ၁၀ ဖြစ်လျှင် y တန်ဘိုး မည်မျှ ရှိမည်ကို Predict လုပ်ကြည့်ပါမည်။

if $x = 10$, $y = 2.41(10) - 0.83$
$= 23.27$

အကယ်၍ x သာ 10 ဖြစ်နေခဲ့မည် ဆိုလျှင် y တန်ဘိုးသည် 23.27 ကို ရရှိပါသည်။ 23.27
အထက်ပါ output များနှင့် data set များကို google sheet ထဲသို့ထည့်ပြီး အောက်ပါအတိုင်း trends line ဆွဲကြည့်မည်ဆိုလျှင် မိမိတို့ တွက်ချက်ခဲ့သော value အားလုံးမှန်ကန် နေသည်ကို တွေ့ရပါမည်။

X သည် 10 ဖြစ်သောအခါတွင် Y သည် 23.27 အတိအကျရသည်ကိုတွေ့ရပါမည်။ ထို့ကြောင့် မိမိတို့ တွက်ချက်မှုအားလုံး မှန်ကန်ပါသည်။ သို့သော် Correlation Coefficient ကိုတွက်ထုတ်ရန် ကျန်ရှိနေသေးသည်။ အထက်ပါပုံ၌ $R^2 = 1$ ဟု ဖော်ပြထားသည်။ ထို value ကို အဘယ်ကြောင့် ဖော်ပြထားခြင်းနှင့် မည်သို့အသေးစိတ်တွက်ချက်ရမည်ကို အောက်တွင်ဆက်လက် ဖော်ပြသွားပါမည်။

Correlation Coefficient ®

Correlation Coefficient ဆိုတာ တိုင်းတာခြင်း ဖြစ်ပါတယ်။ ဘာကို တိုင်းတာတာလဲဆိုလျှင် linear regression based line နှင့် data point များကြား အကွာအဝေးကို တိုင်းတာခြင်း ဖြစ်ပါတယ်။ အချို့ data point များသည် trends နှင့် ကွာဝေးနေမျိုး ဖြစ်နိုင်ပါတယ်။ ထိုကဲ့သို့ များပြားလှတဲ့ data points တွေဟာ trend line နှင့် ဘယ်လောက် များများ ဝေးကွာ နေကြသလဲ ဆိုတာ သိနိုင်ဖို့ Correlation Coefficient (r) ကို တွက်ထုတ်ရပါတယ်။ ထို r ကို တွက်ထုတ်ရန် equation မှာအောက်ပါ အတိုင်း ဖြစ်ပါတယ်။

Correlation Coefficient (r)

$$r = \frac{n\sum xy - \sum x \sum y}{\sqrt{[n\sum x^2 - (\sum x)^2][n\sum y^2 - (\sum y)^2]}}$$

အထက်ပါ equation ထဲတွင် မိမိတို့ data ဇယားမှ parameters များကို ထည့်ပြီး တွက်နိုင်ပါတယ်။ အောက်တွင် တွက်ချက်ပုံ အသေးစိတ်ကို ဖော်ပြထားပါတယ်။

$$r = \frac{n\sum xy - \sum x \sum y}{\sqrt{[n\sum x^2 - (\sum x)^2][n\sum y^2 - (\sum y)^2]}}$$

$$= \frac{2203.6 - 1730.4}{\sqrt{(980 - 784)(4962.86 - 3819.24)}}$$

$$= \frac{473.2}{\sqrt{(196)(1143.62)}}$$

$$= \frac{473.2}{473.44}$$

$$\boxed{r = 0.999 \quad \text{near } 1}$$

အထက်ပါ တွက်ချက်နည်းများအရ r တန်ဘိုးသည် 0.999 ကို ရရှိပါသည်။ ထို့ကြောင့် အနီးစပ်ဆုံး ကြည့်မည် ဆိုလျှင် 1 ဖြစ်ပါသည်။ ထို့ကြောင့် graph တွင် r တန်ဘိုးသည် 1 ဖြစ်သည်ဟု ဖော်ပြထားခြင်း ဖြစ်ပါသည်။

r တန်ဘိုးများပေါ် မူတည်ပြီး မိမိတို့ data set များသည် linear relation ရှိမရှိ ကိုသိနိုင်သလို perfect , strong , moderate , weak , no linear relation စသည်တို့ကို သိနိုင်ပါသည်။ အကယ်၍ r တန်ဘိုးသည်

- - 1 ဆိုလျှင် negative linear relation ဖြစ်ပါသည်။ x value လည်း ကျသွားသလို y value လည်း ကျသွားမှာပါ။ Perfect downhill(negative) လို့လည်း ခေါ်ပါတယ်။
- -0.70 ခန့် ရခဲ့လျှင် strong downhill လို့ခေါ်ပါသည်။
- 0 သာရခဲ့လျှင် ထို data set များကြား linear relation မရှိဘူးဟု ဆိုလိုပါသည်။
- 1 ရခဲ့လျှင် perfect uphill (positive) x value တက်လာတာနဲ့ အမျှလည်း y တန်ဘိုးလည်း လုံးဝ လိုက်တက်လာတာ ဖြစ်ပါတယ်။

- 0.70 ရခဲ့လျှင် data set နှစ်ခုကြား သိသိသာသာ positive relation ရှိနေတာကို သွန်ပြတာပါ။
- 0.50 ဆိုလျှင်တော့ အလယ်အလတ်ပါပဲ။ moderate uphill(positive) လို့လည်း ခေါ်ပါတယ်။
- 0.30 ဝန်းကျင်ခန့် ရခဲ့မည် ဆိုလျှင်တော့ data set နှစ်ခုကြား Linear relation သည် weak ဖြစ်နေပါသည်။

အထက်ပါ Equation များနှင့် တွက်ချက်နည်းများကို သုံးပြီး အောက်တွင် python program ကို ရေးပြထားပါတယ်။ math library တစ်ခု square root ကိုရှာဖို့သုံးထားပါတယ်။ Python pure code နဲ့သာ ရေးထားတာ ဖြစ်တဲ့အတွက် ဖော်ပြခဲ့ပြီးသား equation များကို နားလည်လျှင် မိမိတို့ ကြိုက်နှစ်သက်သည့် programming ဖြင့် ပြန်လည်ရေးသားနိုင်ပါသည်။

Python Program (LSR)

```
#This Program was Designed By Win Htut
# Using Machine Learning Algorithm(LSR)
# If you wanna to all of mathematical explained pls read my LSR article
# Version 1.0
# Next version will add Plotting
import math
import matplotlib.pyplot as plt
def                 correlation_cof(size_of_data,sum_of_xy,sum_of_x,sum_of_y,
sum_of_xS,sum_of_yS):
    r1 = size_of_data*sum_of_xy - sum_of_x*sum_of_y
    r2 = size_of_data*sum_of_xS - sum_of_x*sum_of_x
    r3 = size_of_data*sum_of_yS - sum_of_y*sum_of_y
    r2 =r2 * r3
    r2 = math.sqrt(r2)
    corre_cof = r1/r2
    if corre_cof > 0.97:
        print("Correlation Coefficient is near 1 ",corre_cof)
    else:
        print(corre_cof)
def b(sum_of_y , slope_m , sum_of_x ,size_of_data):
    b1 = sum_of_y - slope_m*sum_of_x
    b = b1/size_of_data
    print("value of b",b)
x =[1,2,3,4,5,6,7]
y =[1.5,3.8,6.7,9.0,11.2,13.6,16]
sum_of_x = 28
sum_of_y =61.8
sum_of_xy=314.8
sum_of_xS=140
sum_of_yS=708.98
size_of_data=7

# slope , y=mx+b
```

```
m1 = size_of_data*sum_of_xy- sum_of_x*sum_of_y
m2 = size_of_data*sum_of_xS - sum_of_x*sum_of_x
slope_m=m1/m2
print("*********************************************************")
print("Slope Value is " ,slope_m)
b(sum_of_y , slope_m , sum_of_x ,size_of_data)
correlation_cof(size_of_data ,sum_of_xy,sum_of_x,sum_of_y,sum_of_xS,sum_of_yS)
print("*********************************************************")
```

Output

```
*********************************************************
Slope Value is  2.4142857142857146
value of b -0.8285714285714302
Correlation Cofficient is near 1  0.999483961218998
*********************************************************
```

Data Science Project Using Linear Regression

ယခုသင်ခန်းစာမှာတော့ Linear Regression ကို Data Science Movie Project တစ်ခုမှာ အသုံးပြုသွားမှာ ဖြစ်ပါတယ်။ ယခု project မှာတော့ ရုပ်ရှင်ကားတစ်ကားအတွက် budget ဘယ်လောက် သုံးရင် revenue အမြတ်ငွေ ဘယ်လောက် ရနိုင်သလဲ ဆိုတာကို predict လုပ်မှာ ဖြစ်ပါတယ်။ Data Science project တစ်ခုမှာ ပထမဆုံး သိရမယ့် တစ်ချက်ကတော့ Question ပါ။ ဘယ်လို question လဲဆိုရင် ဒီ project ရဲ့ ရည်ရွယ်ချက်ကဘာလဲ ဆိုတာပါ။ အဲ့ဒီတော့ ယခု Project မှာဆိုရင် movie တစ်ခုမှာ budget ငွေ ဘယ်လောက် သုံးရင် အမြတ်ငွေ ဘယ်လောက် ပြန်ရ နိုင်သလဲဆိုတဲ့အချက်ကိုသိဖို့ ဆိုရင် ဘယ်လို data တွေပေါ်မှာ အခြေခံရမလဲ ဆိုတာဖြစ်ပါတယ်။ Data Science မှာဆိုရင်တော့ Independent variable လို့ခေါ်ပြီး Machine learning မှာ ဆိုရင်တော့ Featureလို့ခေါ်ပါတယ်။

ဒုတိယ တစ်ချက်ကတော့ Data စုဆောင်းခြင်း နှင့် Data Cleaning ဖြစ်ပြီး မိမိတို့ algorithm အတွက် လိုအပ်တဲ့ data တွေကို ရှာဖွေ စုဆောင်းကာ မိမိတို့ prediction လုပ်ရာတွင် ပိုမို တိကျရန် အတွက် cleaning လုပ်ရမှာ ဖြစ်ပါတယ်။

တတိယ အချက်မှာတော့ Data Visualization and Data Exploration လုပ်ခြင်း တို့ ဖြစ်ပါ တယ်။ ဘာကြောင့် Visualization လုပ်ရလဲ ဆိုတော့ များပြား လှတဲ့ data တွေကို graph and chart နဲ့ပြမှာသာ သိသာ ထင်ရှား မှာ ဖြစ်ပါတယ်။

နောက်ဆုံး တစ်ချက်ကတော့ ရရှိလာတဲ့ အဖြေကို Analysis ပြန်လုပ်ခြင်း ဖြစ်ပါတယ်။ Analysis လုပ်တယ်ဆိုတာက result သည် မှန်နိုင်ချေများသလားဆိုတဲ့ အဖြေကိုပြန်လည် ဆန်းစစ်တာ ဖြစ်ပါတယ်။ လာမည့် သင်ခန်းစာများတွင် အသေးစိတ် ဆက်လက် ဖော်ပြသွားပါမည်။

Data Gathering and Cleaning

ယခု projects အတွက် data တွေ ရယူနိုင်ဖို့ the-numbers.com မှ ယခု Link ကိုသွားပါ။ https://www.the-numbers.com/movie/budgets ထို့နောက် Movie data တွေနဲ့ budget data တွေကို ရယူ နိုင်ပါတယ်။ ထိုကဲ့သို့ မယူလိုပါက အောက်ပါ Link များမှ မိမိတို့ အဆင်ပြေရာကို download ရယူ နိုင်ပါတယ်။

Link 1 or
https://yadi.sk/i/BuOi9Nsz-z3xYQ
Line 2 or
https://u.pcloud.link/publink/show?code=XZcutLXZMv75u2Cij6m4GJyFMcBprQY94qYy
Link 3 or
http://www.mediafire.com/file/yrhm7s9nqyjr1um/moviedata.csv/file
Link 4 or
https://drive.google.com/file/d/1DSWKo4HdFcFCydrGQVsKYoeyzsshBQav/view?usp=sharing

အထက်ပါ file အား download ရယူပြီးလျှင် excel or google sheet တွင် ဖွင့် ပါ။ စာရေးသူ အနေဖြင့် google sheet ကို အသုံးပြုထားပါသည်။ Google sheet သို့သွားပြီး file> open > upload ကို သွားပြီး မိမိတို့ download ဆွဲထားသည့် file ကို select လုပ် ပေးပါ။
https://docs.google.com/spreadsheets(googlesheetsသို့သွားရန်)

အထက်ပါ အတိုင်း google sheets သို့သွားပြီး data file ကို ဖွင့်ပြီးသည်နှင့် တစ်ပြိုင်နက် အောက်ပါပုံထဲကအတိုင်း $0 ဖြစ်နေသည့် data များကို တွေ့ရပါမည်။ထို data များသည် တွက်ချက်ရာတွင် တိကျမှု(accurate) ကို ထိခိုက်မှု ရှိစေနိုင်သည့်အတွက် ထို $0 ဖြစ်နေသည့် data အားလုံးကိုcleaningလုပ်ပါမည်။

Rank	Release Date	Movie Title	Production Budget ($)	Worldwide Gross ($)	Domestic Gross ($)
5293	8/2/1915	The Birth of a Nation	$110,000	$11,000,000	$10,000,000
5140	5/9/1916	Intolerance	$385,907	$0	$0
5230	12/24/1916	20,000 Leagues Und	$200,000	$8,000,000	$8,000,000
5299	9/17/1920	Over the Hill to the P	$100,000	$3,000,000	$3,000,000
5222	1/1/1925	The Big Parade	$245,000	$22,000,000	$11,000,000
4250	12/30/1925	Ben-Hur	$3,900,000	$9,000,000	$9,000,000
4630	12/8/1927	Wings	$2,000,000	$0	$0
5141	1/2/1929	The Broadway Meloc	$379,000	$4,358,000	$2,800,000
4240	1/1/1930	Hell's Angels	$4,000,000	$0	$0
5043	12/31/1931	Mata Hari	$558,000	$900,000	$900,000
5017	7/4/1933	King Kong	$672,000	$10,000,650	$10,000,000
5234	9/2/1933	She Done Him Wron	$200,000	$2,000,000	$2,000,000
5120	9/3/1933	42nd Street	$439,000	$2,281,000	$1,438,000
5154	1/1/1934	It Happened One Nig	$325,000	$2,500,000	$2,500,000
5025	6/9/1935	Top Hat	$609,000	$3,202,000	$1,782,000
4738	5/2/1936	Modern Times	$1,500,000	$165,049	$163,245
4768	6/26/1936	San Francisco	$1,300,000	$5,273,000	$2,868,000
4814	10/20/1936	Charge of the Light E	$1,200,000	$0	$0

Data cleaning လုပ်ရန် fx အောက်တွင်ရှိသော အကွက်အလွတ် အား တစ်ချက် နှိပ်လိုက်ပါ။

fx Rank

	A	B
1	Rank	Release
2	5293	8/
3	5140	5/

ထို့နောက် Data ထဲမှ sort range ကိုဆက်သွားပါ။

အထက်ပါ အတိုင်း သွားပြီးလျှင် Column E ကို ရွေးချယ်ပါ။

ပြီးလျှင် sort ကိုနှိပ်လိုက်ပါက အောက်ပါ ပုံထဲက အတိုင်း data များ ကွဲပြား သွားသည်ကို မြင်ရပါမည်။ ထို့နောက် $0 ဖြစ်နေသည့် data များအားလုံးကို ဖျက်ထုတ်ပါမည်။

A	B	C	D	E	F
5140	5/9/1916	Intolerance	$385,907	$0	$0
4630	12/8/1927	Wings	$2,000,000	$0	$0
4240	1/1/1930	Hell's Angels	$4,000,000	$0	$0
4814	10/20/1936	Charge of the Light E	$1,200,000	$0	$0
4789	10/28/1941	How Green Was My	$1,250,000	$0	$0
4419	7/28/1951	Alice in Wonderland	$3,000,000	$0	$0
4705	12/31/1955	The King's Thief	$1,577,000	$0	$0
4455	12/1/1956	Diane	$2,660,000	$0	$0
5152	4/13/1957	12 Angry Men	$340,000	$0	$0
3559	10/10/1968	Barbarella	$9,000,000	$0	$0
2159	1/1/1970	Waterloo	$25,000,000	$0	$0
4626	4/28/1971	Bananas	$2,000,000	$0	$0
4239	9/11/1972	1776	$4,000,000	$0	$0
5176	1/4/1975	Death Race 2000	$300,000	$0	$0
2713	10/21/1977	Damnation Alley	$17,000,000	$0	$0
5109	12/31/1984	The Toxic Avenger	$475,000	$0	$0
5136	1/4/1986	My Beautiful Laundre	$400,000	$0	$0
4523	6/19/1987	The Brave Little Toas	$2,300,000	$0	$0
5101	12/31/1990	Going Under	$500,000	$0	$0
3899	12/31/1995	Men of War	$6,000,000	$0	$0
2948	6/29/2001	Pandaemonium	$15,000,000	$0	$0
4765	8/21/2001	Batman - The Movie	$1,377,800	$0	$0
4638	12/31/2001	Pendulum	$2,000,000	$0	$0

$0 value များသည် 356 roll ထိ ဖြစ်တဲ့ အတွက် 356 roll ထိ သွားပါ။ ထို့နောက် 356 နေရာကို တစ်ချက်နှိပ်ပြီး shift+up key နှစ်ခုကို တွဲပြီး နှိပ်ထားလိုက်ပါ roll 1 ထိ select

လုပ်တာ ရောက်မှ ရပ်လိုက်ပါ ထိုနောက် delete roll 1-365 ကိုနှိပ်လိုက်ပါ(367 ထိ 0 ဖြစ်နေလျှင် 367 ကိုပါ ဖျက်ပါ)

အထက်ပါ အတိုင်း ဖျက်ပြီးပါက Cloumn A ,B, C , F တို့ကိုလည်း ဖျက်ရန် လိုပါသေးသည်။ A,B,C,F တို့အား control ဖြင့် select မှတ်ပြီး Delete selected columns ကို နှိပ်ကာ ဖျက်နိုင်ပါသည်။

Column D and E နှစ်ခုကိုသာ ချန်ထားပါမည်။ အထက်ပါအဆင့် ပြီးပါက Number များ ရှေ့တွင် ပါဝင်နေသော $ sign များကို ဖြုတ်ရန် ကျန်နေပါသေးသည်။ ထို့ကြောင့် fx အောက်တွင် ရှိသော လွတ်နေသော နေရာတွင် တစ်ချက် နှိပ်လိုက်ပါက column နှစ်ခုလုံးကို တစ်ပြိုင်တည်း select မှတ်ပြီး ဖြစ်နေပါမည်။ ထို့နောက် Control+F ကို နှိပ်ပါ

အထက်ပါ ပုံထဲမှ အတိုင်း More options ကိုသွားလိုက်ပါ။ ထို့နောက် ပေါ်လာသော အကွက်ထဲမှ Find နေရာတွင် $ ကိုရေးပါ။ Replace with နေရာတွင် ဘာမှ မရေးပါနှင့်။ ထို့နောက် Replaceallကိုနှိပ်ပါ။ပြီးလျှင်Done။

အထက်ပါအဆင့်ပြီးပါက မိမိတို့ data ထဲတွင် $ များ ပါဝင်မနေတော့ပါ။ နောက်တစ်ဆင့် အနေဖြင့် Column A and B အတွက် title များ ထည့်ပေးပါမည်။

	A	B
1	7000000.00	1000000.00
2	14000000.00	1000000.00
3	800000.00	1001437.00
4	185000000.00	1002572859.00
5	1300000.00	1005840.00
6	10000000.00	1011054.00

Tiles တွေ ထည့်ဖို့ အတွက် A ရဲ့ အောက်မှာ ထည့်ချင်တာ ဖြစ်တဲ့ အတွက် A အောက်က row ကို select လုပ်ပြီး menu မှ Insert ကို နှိပ်ကာ Row above ကို နှိပ်ပေးလိုက်ပါ။

အထက်ပါ အဆင့် ပြီးသွားပါက Column A and B အောက်တွင် row အလွတ် တစ်ခု ပေါ်လာပါမည်။ ထို ROW တွင် စာရေးသူအနေဖြင့် ProBudget and Gross ဟု နာမည်ပေးလိုက် ပါသည်။

	A	B
1	ProBudget	Gross
2	7000000.00	1000000.00
3	14000000.00	1000000.00

Exploratory and Visualise the Data

Data တွေကို Gathering and clean လုပ်ပြီးပြီဆိုတော့ Exploratory and Visualise ဆက်လုပ်ပါမည်။ ပထမဆုံးအနေဖြင့် google sheet မှ data များအား csv file(comma separated value) ပုံစံဖြင့် download ပြုလုပ်လိုက်ပါ။ ထိုသို့ ပြုလုပ်ရန် File > download > csv ကို ရွေးချယ်ပေးလိုက်ပါ။ Download ဆွဲရာတွင် name တောင်းပါက မိမိကြိုက်နှစ်သက်သည့် name ဖြင့် save နိုင်ပါသည်။ထို name သည် program ထဲတွင် ပြန်သုံးမှာဖြစ်သည့် အတွက် အရေးကြီးပါသည်။ စာရေးသူအနေဖြင့် movieCleaned ဟုနာမည် ပေးလိုက်ပါသည်။

ထို ဆင့်ပြီးသွားပါက jupyter notebook ကို ဖွင့်၍ download ဆွဲလို့ရလာသော csv file အား Uploadတင်ပေးပါ။

Upload ကိုနှိပ်ပြီး မိမိတို့ download ဆွဲပြီးရလာသော csv file အား ရွေးချယ်ပေးပါ။

Cleaned movie data ကိုအောက်ပါ Link များတွင် download ရယူ နိုင်ပါသည်။

Link-1

https://drive.google.com/file/d/1ClI9BLK1UegZ5ejesrbeTMc9oJv7xlAN/view?usp=sharing

Link-2

https://www.mediafire.com/file/yrhm7s9nqyjr1um/moviedata.csv/file

link-3

https://u.pcloud.link/publink/show?code=XZqvi4XZiOhAWxq5Gh5QmHQ4s1LAjLEFbdM7

link-4

https://yadi.sk/d/5rqe3V2caZdGgg?w=1

Upload တင်ပြီးသည်နှင့် တစ်ပြိုင်နက် မိမိတို့ အနေဖြင့် program တွင် ထို file ကို အလွယ် တကူ ယူသုံးနိုင်ပြီ ဖြစ်ပါတယ်။ အောက်ပါအတိုင်း program ရေးပြီး data များကို ထုတ်ယူ ကြည့်နိုင်ပါသည်။ ယခုသင်ခန်းစာမှ စတင်ပြီး Data တွေကို ကိုင်တွယ်တော့မှာဖြစ်တဲ့အတွက် pandas library ကို အသုံးပြုမှာဖြစ်ပါတယ်။ Pandas သည် Data Analysis and Manipulation လုပ်ဖို့အတွက် အမြန်ဆုံး နဲ့ အကောင်းဆုံး tool တစ်ခု ဖြစ်ပါသည်။ ထို့ပြင် pandas library ထဲမှ DataFrame ကိုလည်း အသုံးပြုမည်ဖြစ်ပါသည်။ Data များကို plot ဖြင့် ပြန်ပြရန် အတွက်လည်း matplotlibထဲမှpyplotကိုလည်းအသုံးပြုပါမည်။

In [14]:
```
import pandas as pd
from pandas import DataFrame
import matplotlib.pyplot as plt
data =pd.read_csv('movieCleaned.csv')
data
```

Out[14]:

	proBudget	gross
0	1000000	26
1	10000	401
2	400000	423
3	750000	450
4	10000	527

...

အထက်ပါ အဆင့်ပြီးပါက DataFrame Function ကိုသုံးပြီး Data များအား column အလိုက် ခွဲထုတ်ပါမည်။ DataFrame function သည် parameter 2 ခု ယူပါသည် ပထမ တစ်ခုသည် csv file မှ ဖတ်လို့ရလာသော data object ဖြစ်ပြီး ဒုတိယ တစ်ခုသည် column name ဖြစ်ပါသည်။ထို့ပြင် plt ထဲမှ scatter function ကိုသုံးပြီး X and y data များကို ထည့်ပေးကာ plot ဖြင့် ထုတ်ကြည့်ပါသည်။

In [15]:
```
import pandas as pd
from pandas import DataFrame
import matplotlib.pyplot as plt
data =pd.read_csv('movieCleaned.csv')

X = DataFrame(data, columns=['proBudget'])
y = DataFrame(data, columns=['gross'])
plt.scatter(X, y , alpha=0.5)
plt.show( )
```

အထက်ပါ program ကို run ကြည့်ပါက အောက်ပါအတိုင်း output များ

ထွက်လာသည်ကို မြင်ရပါမည်။

နောက်တစ်ဆင့်အနေဖြင့် Figure ရဲ့ size ကို ပိုကြီးရန်နှင့် Figure အား title , x and y title များ ထည့်ပေးပါမည်။ Matplotlib ကို နောက်ပိုင်းသင်ခန်းစာမှာ အသေးစိတ် ထပ်ဆွေးနွေးဦးမှာ ဖြစ်ပါတယ်။

```
import pandas
from pandas import DataFrame
import matplotlib.pyplot as plt

data = pandas.read_csv('movieCleaned.csv')
X = DataFrame(data, columns=['proBudget'])
y = DataFrame(data, columns=['gross'])

plt.figure(figsize=(10,6))
plt.scatter(X, y, alpha=0.5)
plt.title('Title: Cost and Gross')

plt.xlabel('ProBudget')
plt.ylabel('Gross $')
plt.show()
```

Evaluation

ယခု အပိုင်းမှာတော့ coefficient , intercept and score တို့ကို ရှာသွားမှာ ဖြစ်ပြီး sklearn library မှ linear_model ကို စပြီး သုံးသွားမှာ ဖြစ်ပါတယ်။ program အပြည့်အစုံကို အောက်မှာဖော်ပြထားပါတယ်။

```
In [19]: import pandas
         from pandas import DataFrame
         import matplotlib.pyplot as plt
         from sklearn.linear_model import LinearRegression

         data = pandas.read_csv('movieCleaned.csv')
         X = DataFrame(data, columns=['proBudget'])
         y = DataFrame(data, columns=['gross'])

         plt.figure(figsize=(10,6))
         plt.scatter(X, y, alpha=0.5)
         plt.title('Title: Cost and Gross')

         plt.xlabel('ProBudget')
         plt.ylabel('Gross $')
         plt.ylim(0, 3000000000)
         plt.xlim(0, 450000000)

         regression = LinearRegression()
         regression.fit(X, y)

         print(regression.coef_)
         print(regression.intercept_)
         print(regression.score(X, y))

         plt.plot(X, regression.predict(X), color='red', linewidth=3)
         plt.show()
```

Ylim အတွက်ဆိုလျှင် အထဲမှာ parameter နှစ်ခု ပါ ပါတယ် ပထမ တစ်ခုသည် min value ဖြစ်ပြီး ဒုတိယ တစ်ခုသည် max value ဖြစ်ပါတယ်။ စာရေးသူရဲ့ အမြတ်အစွန်း gross budget သည် 3 billion နောက်ဆုံးဖြစ်သည့် အတွက် 3000000000 ဟုရေးပေးထားခြင်းဖြစ်သည်။ xlim သည်လည်း ထို နည်းအတိုင်းပင် ဖြစ်ပါသည်။ xlim,ylim function များကို ပိုမို နားလည်စေရန် အောက်ပါprogramကိုစမ်းရေးကြည့်နိုင်ပါသည်။

```
In [4]: import matplotlib.pyplot as plt
        x = [1, 2, 4, 3, 0]
        plt.xlim(0, 10)
        plt.plot(x)

Out[4]: [<matplotlib.lines.Line2D at 0x284c4ebb970>]
```

နောက်တစ်ဆင့်အနေဖြင့် LinearRegression() method ကို သုံးပြီး regression ဆိုသည့် object တစ်ခုကို တည်ဆောက်လိုက်ပါသည်။ regression = LinearRegression() နောက်တစ်ဆင့် အနေဖြင့် fit method ကိုသုံးပြီး X and y parameter များထည့်ပေးလိုက်ပါသည်။ အဘယ်ကြောင့်ဆိုသော် sklearn ထဲမှ linearRegression() model

ကို စတင်အသုံးပြုမယ် ဆိုလျှင် fit method ကိုအရင်ခေါ်ပြီး လိုအပ်သည့် parameter ကိုထည့်ပေးရပါသည်။

```
[[3.11150918]]
[-7236192.72913963]
0.5496485356985727
```

print(regression.coef_) coefficient ကိုရှာရန်အတွက်ဖြစ်ပြီး Output အနေဖြင့် 3.11150918 ကို ရရှိပါသည်။ ထိုသို့ ရရှိတဲ့ အတွက် အသုံးပြုလိုက်ရတဲ့ budget နဲ့ အမြတ်ငွေ gross တို့ကြားမှာ positive relation ရှိတယ်ဆိုသည့် အဓိပ္ပာယ် ဖြစ်ပါတယ်။ ဆိုလိုချင်တာကတော့ 1 dollar ကို budget အနေဖြင့် သုံးလိုက်တိုင်း အမြတ်ငွေ အနေဖြင့် 3.1 dollar ရရှိနိုင်ပါတယ်။

print(regression.intercept_) y=mx+b နဲ့ ကြည့်မည်ဆိုလျှင် b တန်ဘိုးကို ရှာလိုက်တာပါ။ အကယ်၍သာ x ဆိုတဲ့ budget သည် zero ဖြစ်နေခဲ့မည် ဆိုလျှင် y ဆိုတဲ့ gross သည် -7236192.729 ဖြစ်နေမှာပါ။ ဒါဆိုရင် budget 20,000,000 (20 million) သာ သုံးခဲ့မည် ဆိုလျှင် gross ဘယ်လောက် ဖြစ်မလဲပေါ့။ y = 3.11150918*20,000,000 + (-7236192.729) တွက်ကြည့်မည်ဆိုလျှင် y = 54,993,990.871 ကို ရရှိပါတယ်။ထို့ကြောင့် အမြတ်ငွေ 54,993,990.871 dollar ကို ရရှိနိုင်တယ်ပေါ့။ ဒါဟာ ကျွန်တော်တို့ အသုံးပြုထားတဲ့ model ကနေ ထွက်လာတဲ့ ကိန်းဂဏန်းတွေ ဖြစ်ပါတယ်။ ထိုကိန်းဂဏန်းတွေသည် မှန်နိုင်လား/ မမှန်နိုင်ဘူးလားဆိုတာကို ပြန်လည် စစ်ဆေးနိုင်ပါတယ်။ အဲ့ဒါကတော့ ရှေ့ပိုင်း သင်ခန်းစာမှာ တွက်ခဲ့ ဖူးတဲ့ r square ဖြစ်ပါတယ်။

print(regression.score(X, y)) .score() ဆိုသည့် method ကို အသုံးပြုမည်ဆိုလျှင် r square value ကို ရရှိနိုင်ပါတယ်။ စာရေးသူအနေဖြင့် r square value သည် 0.5496485356985727 ကို ရရှိခဲ့ပါတယ်။ 0.549 ဖြစ်တဲ့ အတွက် 0.55 ဟု ယူမည် ဆိုလျှင် 55% မှန်နိုင်သည့် ဆိုသည့် အဖြေကို ရရှိပါတယ်။

ယခု ဆိုလျှင် Data Science Project တစ်ခုလုံးမှာ ပါရှိတဲ့ step အစအဆုံး ကို ပြုလုပ်ပြီး သွားပြီ ဖြစ်ပါတယ်။ ပြဿနာကို သိအောင် လုပ်ခြင်း , data စုဆောင်းခြင်း(Gathering) , data clean လုပ်ခြင်း , machine learning algorithm ကို အသုံးပြုခြင်း , evaluate and visualization ပြုလုပ်ခြင်း နှင့် ပြန်လည် စစ်ဆေးခြင်း တို့ဖြစ်ပါတယ်။

55% သာ ဖြစ်တဲ့ အတွက် စာရေးသူတို့ အနေဖြင့် မေးခွန်းများစွာ ကျန်ရှိနေပါသေးတယ် နောက်ထပ် data တွေ ထပ်ထည့် ရင် ဘယ်လို ပြောင်းလဲ သွား

နိုင်သေးသလဲ ? Linear Regression ဟာ အချို့သောအခြေနေအတွက် အဆင်မပြေနိုင်ပါဘူး။ အထူးသဖြင့် non-linear models တွေ အတွက်ပါ။ အကယ်၍ linear regression နှင့် အဆင်မပြေခဲ့ဘူး ဆိုလျှင် Support Vector Machines (SVM) , Decision Trees , Random Forest နှင့် Neural networks algorithms တို့ကို အသုံးပြုနိုင်ပါသေးတယ်။

Matplotlib For Visualization

ယခု သင်ခန်းစာမှာတော့ IDE အနေဖြင့် pycharm ကို အသုံးပြုသွားမှာ ဖြစ်ပါတယ်။ Professional project တွေပြုလုပ်တဲ့အခါမှာ library ပြဿနာများကို ဖြေရှင်းပေးနိုင်ဖို့ pycharm ကို အသုံးပြုနိုင်ပါတယ်။ vs code မှာလည်း ဖြေရှင်းပေးနိုင်သော်လည်း pycharm မှာ project စတည်ဆောက်ကတည်းက virtual enviroment တစ်ခုကို ဖန်တီးပေးပါတယ်။ထို့ကြောင့် programmer အနေဖြင့် step များစွာကို လုပ်နေစရာ မလိုပဲ project တစ်ခုကို virtual env မှာ မြန်မြန် ဖန်တီး ပြီးသား ဖြစ်သလို အလုပ် မြန်မြန် ပြီးမြောက်စေပါတယ်။ pycharm ကို download လုပ်နိုင်ဖို့ အောက်ပါ Link ကိုသွားလိုက်ပါ။ မိမိတို့ အသုံးပြုသည် operating system ကို အလိုအလျောက် detect လုပ်ပြီး download page ကို ပြပေးပါလိမ့်မယ်။

https://www.jetbrains.com/pycharm/download/

စာရေးသူတို့ အနေဖြင့် လူတိုင်း အသုံးပြုနိုင်မည့် Community Version ကိုသာ အသုံးပြုမှာ ဖြစ်သည့် အတွက် community version ကို download ဆွဲရန် လိုအပ်ပါသည်။

Pycharm ကို download လုပ်ပြီးသည်နှင့် တစ်ပြိုင်နက် install လုပ်ပြီးပါက အောက်ပါ ပုံ အတိုင်း file >> new project သို့သွားပြီး မိမိတို့ ပေးချင်သည့် project name ကို ပေးပါ။ စာရေးသူ အနေဖြင့် matplotlibVM ဟုပေးလိုက်ပါသည်။

ထို့နောက် matplotlibVM ဆိုသည့် project တစ်ခုကို ဖန်တီးပေးပါလိမ့်မယ်။ ထို matplolibVM ထဲသို့ ဝင်ကြည့်လျှင် အောက်ပါအတိုင်း venv ဆိုသည့် virutal envrioment တစ်ခု တည်ဆောက်ထားတာကို တွေ့ရပါမည်။

ထို venv အား right click ထောက်ပြီး new မှ တစ်ဆင့် File သို့သွားပြီး 1-mat.py ဆိုသည့် python file တစ်ခုကို တည်ဆောက်လိုက်ပါသည်။

အထက်ပါ အဆင့်များ ပြီးပါက venv အား right click တစ်ခါ ထပ်ထောက်ပြီး open in terminal ဆိုတာကို နှိပ်လိုက်ပါ။ ထို့နောက် terminal ပေါ်လာမည် ဖြစ်ပါသည်။ ယခုနည်းအတိုင်း အဆင်မပြေပါက ဘယ်ဘက် အောက်ဒေါင့်တွင် Terminal ကို နှိပ်ပြီးတော့ terminal ကို အသုံးပြု နိုင်ပါသည်။

Terminal သို့ ရောက်သွားပါက အောက်ပါ ပုံအတိုင်း venv ထဲတွင် ရောက်နေသည်ကို မြင်ရပါမည်။ ထို virtual environment ထဲတွင် မိမိတို့ အသုံးပြုမည့် library များကို သီးသန့် install လုပ်သွားမှာ ဖြစ်ပါတယ်။

ပထမဆုံးအနေဖြင့် matplotlib ကို install လုပ်ပေးရန် လိုအပ်ပါသည်။ pip install matplotlib ဆိုလျှင် matplotlib ရဲ့ နောက်ဆုံး version ကို install လုပ်ပေးမှာ ဖြစ်ပါတယ်။ install လုပ်ပြီးတဲ့ အခါမှာ မိမိတို့ install လုပ်ထားသည့် package များကို ပြန်ကြည့် လိုလျှင် pip list နှင့် ကြည့်နိုင်ပါသည်။ pip သည် Pip installs Packages , Pip install Python သို့မဟုတ် preferred installer program ကို ဆိုလိုခြင်း ဖြစ်သည်။

Packages Management

စာရေးသူတို့ လက်ရှိ ရှိနေသည့် matplotlibVM project ထဲမှ venv ထဲတွင် အောက်ပါ packages များ ရှိနေပါပြီ။ စာရေးသူ အနေဖြင့် matplolib version 3.3.3 နှင့် numpy version 1.19.4 တို့ကို install လုပ်ထားပါသည်။

```
(venv) C:\Users\winht\PycharmProjects\matplotlibVM\venv>pip list
Package         Version
--------------- -------
cycler          0.10.0
kiwisolver      1.3.1
matplotlib      3.3.3
numpy           1.19.4
Pillow          8.0.1
pip             20.3.3
pyparsing       2.4.7
python-dateutil 2.8.1
setuptools      51.1.0
six             1.15.0
```

အကယ်၍ package version အလိုက် အတိအကျ install လုပ်လို ပါကအောက်ပါအတိုင်း ပြုလုပ် နိုင်ပါသည်။ pip install matplotlib==3.3.3

```
(venv) C:\Users\winht\PycharmProjects\matplotlibVM\venv>pip install matplotlib==3.3.3
Requirement already satisfied: matplotlib==3.3.3 in c:\users\winht\pycharmprojects\matplot
```

ထို့ပြင် မိမိတို့ install လုပ်ထားပြီးသား package ကို upgrade လုပ်လိုပါကလည်း pip install --upgrade Package name ဟုရေးပြီး ပြုလုပ် နိုင်ပါသည်။

```
(venv) C:\Users\winht\PycharmProjects\matplotlibVM\venv>pip install --upgrade matplotlib
Requirement already satisfied: matplotlib in c:\users\winht\pycharmprojects\matplotlibvm\venv\
```

Install လုပ်ထားပြီးသား packages များကို uninstall လုပ်လိုပါကလည်း pip uninstall packagename ဟုရေးပြီး ဥပမာ pip uninstall matplotlib ယခု အတိုင်း ရေးပြီး ပြုလုပ် နိုင်ပါသည်။ pip နဲ့ ပတ်သက်ပြီး နောက်ဆုံး မှတ်ထားဖို့ လိုတဲ့ တစ်ခုကတော့ show ဖြစ်ပါတယ်။ သူကတော့ package တစ်ခုရဲ့ detail အချက် အလက် အားလုံးကို ပြပေးနိုင်ပါတယ်။

```
(venv) C:\Users\winht\PycharmProjects\matplotlibVM\venv>pip show matplotlib
Name: matplotlib
Version: 3.3.3
Summary: Python plotting package
Home-page: https://matplotlib.org
Author: John D. Hunter, Michael Droettboom
Author-email: matplotlib-users@python.org
License: PSF
Location: c:\users\winht\pycharmprojects\matplotlibvm\venv\lib\site-packages
Requires: python-dateutil, cycler, numpy, pillow, kiwisolver, pyparsing
Required-by:
```

စာရေးသူတို့ matplotlibVM project ထဲမှ 1-mat.py ဆိုသည့် python file ထဲကို ပြန်သွားပြီး matplotlib သင်ခန်းစာများ စရေးကြည့်ကြပါစို့။ ပထမဆုံး အနေဖြင့် matplotlib ထဲမှ pyplot ကို import လုပ်ပါမည်။ ထို့ပြင် list နှစ်ခု တည်ဆောက်ပြီး ထို list

နှစ်ခုထဲမှ အချက်လက်များကို plot ဖြင့် ပြန် ပြပါမည်။ program ရေးသား ပုံမှာ အောက်ပါ အတိုင်း ဖြစ်သည်။

```
1  import matplotlib.pyplot as plt
2
3  xData = [10,20,21,24,26,29,30,34,39,50]
4  yData = [100,200,210,240,260,290,300,340,390,500]
5  plt.plot(xData,yData)
6  plt.show()
```

Data များကို visualization လုပ်ပြနိုင်ဖို့ method နှစ်ခုကို အသုံးပြုရပါသည်။ ပထမ တစ်ခုသည် plot ဖြစ်ပြီး သူကတော့ မိမိတို့ plot လုပ်လိုသည့် data များကို parameter အနေဖြင့် ထည့်ပေးရပါတယ်။ ဒုတိယ တစ်ခုကတော့ show ဖြစ်ပါတယ် သူကတော့ plot လုပ်လိုတဲ့အခါတိုင်း အောက်ဆုံးတွင် မပါ မဖြစ် ပါရမယ့် function တစ်ခု ဖြစ်ပါတယ်။ အထက်ပါ program ကို run ကြည့်မည် ဆိုလျှင် အောက်ပါ အတိုင်း output များကို တွေ့ရမှာပါ။ program run ဖို့ အတွက် line1 ဘေးမှ အစိမ်းရောင်လေးကို နှိပ်ပြီး run နိုင်သည်။

Program run လိုက်သည်နှင့် figure ကို ဖော်ပြဖို့ windows form တစ်ခု ကျလာပါသည်။ ထို windows form တွင် အကျယ်ချဲ့ပြီး ကြည့်ရှု နိုင်ရန် left button pans and right button zoom ဆိုသည့် ခလုပ်ကို နှိပ်ပြီးကြည့်ရှု နိုင်သည်။ ကြည့်ရှု ပြီးသည့်အခါ မူလ နေရာသို့ ပြန်ရောက် နိုင်ရန် အတွက်လည်း Home button တစ်ခု ပါဝင်နေပါသေးသည်။ ထို့ပြင် မိမိတို့ အသေးစိတ် ကြည့်လိုသည့် နေရာကို ကြည့်ရှု နိုင် ဖို့ အတွက် zoom to rectangle ဆိုသည့် search button လေးကို နှိပ်ပြီး မိမိတို့ အသေးစိတ် ကြည့်ရှုလိုသည့် နေရာကို ကြည့်နိုင်သည်။ Figure ကို မိမိတို့ အဆင်ပြေသလို ကြည့်ရှုနိုင်ဖို့ အတွက်လည်း Configure subplots ဆိုသည့် button တစ်ခု ပါဝင်သည့်အတွက် ထို button ကို နှိပ်လိုက်လျှင် အောက်ပါ အတိုင်း Figure တစ်ခု ပေါ်လာမည် ဖြစ်ပြီး မိမိတို့ စိတ်ကြိုက် ပုံစံ ကို ချိန်းနိုင်သည်။

နောက်ဆုံးတစ်ခုအနေနဲ့ကတော့ data visualize လုပ်ပြီးရလာတဲ့ အနေအထားကို image တစ်ခု အနေဖြင့် သိမ်းလိုပါက save ဆိုတဲ့ button တစ်ခု ပါဝင်နေပါသေးတယ်။ ထို save button ကို နှိပ်ပြီး png image file type အနေဖြင့် save နိုင်ပါသေးတယ်။ နောက်တစ်ဆင့် အနေဖြင့် figure ကို x axis အတွက် xlable နှင့် y axis အတွက် ylable ထို့ပြင် figure အတွက် title တို့ကို ထည့်ပေးပါမည်။

```
5  plt.plot(xData,yData)
6  plt.xlabel("Budget")
7  plt.ylabel("Gross")
8  plt.title("Budget and Gross Figure")
9  plt.show()
```

အထက်တွင် ဖော်ပြထားသော line 6 , 7 , 8 တို့ကို ထည့်လိုက်ပါက output တွင် label များပါလာသည်ကို မြင်ရပါမည်။

Matplotlib Example Project

Matplotlib ကို ပိုမို နားလည်နိုင်ဖို့ အတွက် project တစ်ခု ရေးကြည့်ပါမည်။ ယခု project တွင် x and y နှစ်ခုတည်းသာ ရှိမည် မဟုတ်ပဲ x တစ်ခုနှင့် y အများကြီး ထားပါမည်။

ယခု project တွင် လအလိုက် လုပ်ငန်းများရဲ့ ဝင်ငွေများ အကြောင်းကို Figure နဲ့ ဖော်ပြသွားမှာ ဖြစ်ပါတယ်။ နှစ်တစ်နှစ်တွင် လများသည် တစ်ခုသာ ရှိသည့်အတွက် x ဝင်ရိုးတွင် ထားပါမည်။ ဝင်ငွေဝင်ရာ လုပ်ငန်း များသည် တစ်ခုထက် ပိုနိုင်တဲ့ အတွက် y ဝင်ရိုးတွင် ထားပါမည်။ စားရေးသူ အနေဖြင့် လုပ်ငန်း သုံးခုကို ထည့်ထားမှာ ဖြစ်တဲ့အတွက် y ဝင်ရိုးတွင် လုပ်ငန်း သုံးခုစာ data များကို ဖော်ပြသွားမှာဖြစ်ပါတယ်။

လုပ်ငန်းသုံးခုမှ ဝင်ငွေ အတွက် list 3 ခုနှင့် month များ အတွက် list တစ်ခု စုစုပေါင်း list 4 ခု ရှိပါမည်။

```
import matplotlib.pyplot as plt
month = [1,2,3,4,5,6,7,8,9,10,11,12]

income_job1=[1500000,2000000,1000000,3000000,4000000,500000,1000000,1500000,3500000,5000000,2000000,10000000]
income_job2=[1000000,2000000,2500000,2000000,5000000,500000,2000000,1000000,2000000,3000000,5000000,4000000]
income_job3=[100000,50000,80000,1000000,150000,200000,2500000,3000000,3400000,4000000,4500000,5000000]
```

ထို Data လေးခု အတွက် Plot လုပ်ရမည့် ပုံမှာ အောက်ပါ အတိုင်း ဖြစ်သည်။

```
1  import matplotlib.pyplot as plt
2  month = [1,2,3,4,5,6,7,8,9,10,11,12]
3  income_job1=[1500000,2000000,1000000,3000000,4000000,500000,1000000,1500000,3500000,5000000,2000000,10000000]
4  plt.plot(month , income_job1,color='red',linestyle='--',label='JOB1')
5
6  income_job2=[1000000,2000000,2500000,2000000,5000000,500000,2000000,1000000,2000000,3000000,5000000,4000000]
7  plt.plot(month , income_job2,label='JOB2')
8
9  income_job3=[100000,50000,80000,1000000,150000,200000,2500000,3000000,3400000,4000000,4500000,5000000]
10 plt.plot(month , income_job3,color='g',linestyle='-.',label='JOB3')
```

Line 4 တွင် month data နှင့် income_job1 အား plot တစ်ခု လုပ်ပြီး line 7 တွင်လည်း month data နှင့် income_job2 တို့အား plot ထပ်လုပ်ထားပါသည်။ ထို့ပြင် line 10 တွင်လည်း month data နှင့် income_job3 တို့ကို plot လုပ်ထားပါသည်။ ရှေ့မှ ရေးသည့် data သည် X axis အတွက် ဖြစ်သောကြောင့် Plot လုပ်တိုင်းတွင်လည်း x axis အတွက် month data ကို ရှေ့မှ ထည့်ထားပါသည်။ ထို့ကြောင့် Output အနေဖြင့် x axis တွင် month data တစ်ခုတည်းနှင့် y axis အတွက်income_job dataသုံးခုလုံးကိုရရှိမှာဖြစ်ပါတယ်။

Figure ထဲမှ data လိုင်းများကို မိမိတို့ အသုံးပြုလိုသည့် color , style နှင့် label များ ထိုးပြီး အသုံးပြုနိုင်ပါသည်။ line 7 တွင် color အတွက် red ကို သုံးထားပြီး linestyle အတွက် ' -- ' ကို သုံးထားပါသည်။ label properties သည် figure ထဲတွင် ဘယ် လိုင်း အမျိုးအစားသည် ဘယ် data အမျိုးအစား ဖြစ်သလဲ ဆိုတာကို ဖော်ပြဖို့ အတွက် အသုံးပြုပါသည်။ စာရေးသူ အနေဖြင့် income_job1 နှင့် income_job3 အတွက်သာ color နှင့် style တို့ကို သုံးထားပါသည်။ income_job2 တွင်ပေါ်နေသော color နှင့် style သည် default သာ ဖြစ်ပါသည်။ အထက်ပါ figure ထဲတွင် label များ မပေါ်သေးပါ။ ထိုသို့ ပေါ်ဖို့ အတွက်ဆိုလျှင် legend() ဆိုသည့် method ကို အသုံးပြု ပေးရမှာ ဖြစ်ပါတယ်။ ထို့ပြင် grid အကွက်များ ထည့်လို ပါကလည်း grid() method ကို သုံးနိုင်သလို xlable , ylable များ အတိအကျ ပြည့်စုံစွာ figure တွင် ပေါ်နိုင်ရန် အတွက်လည်း tight_layout() ဆိုသည့် method ကို အသုံးပြုရပါမည်။ ထို method များ အသုံးပြုပုံကိုအောက်ပါ ပုံတွင် ဖော်ပြထားသည်။

```
month = [1,2,3,4,5,6,7,8,9,10,11,12]
income_job1=[1500000,2000000,1000000,3000000,4000000,500000,1000000,1500000,3500000,5000000,2000000,10000000]
plt.plot(month , income_job1,color='red',linestyle='--',label='JOB1')

income_job2=[1000000,2000000,2500000,2000000,5000000,500000,2000000,1000000,2000000,3000000,5000000,4000000]
plt.plot(month , income_job2,label='JOB2')

income_job3=[100000,50000,80000,1000000,150000,200000,2500000,3000000,3400000,4000000,4500000,5000000]
plt.plot(month , income_job3,color='g',linestyle='-.',label='JOB3')

plt.xlabel('Monthly In 2020')
plt.ylabel('Income From Jobs')
plt.title('Income From All Jobs in 2020')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

Trend lines များရဲ့ styles များကိုတော့ မိမိတို့ လိုအပ်သလို အသုံးပြုနိုင်ရန် အောက်တွင်ဖော်ပြထားသည်။

Character Description

'-' solid line style
'--' dashed line style
'-.' dash-dot line style
':' dotted line style
'.' point marker
',' pixel marker
'o' circle marker
'v' triangle_down marker
'^' triangle_up marker
'<' triangle_left marker
'>' triangle_right marker
'1' tri_down marker
'2' tri_up marker

'3' tri_left marker
'4' tri_right marker
's' square marker
'p' pentagon marker
'*' star marker
'h' hexagon1 marker
'H' hexagon2 marker
'+' plus marker
'x' x marker
'D' diamond marker
'd' thin_diamond marker
'|' vline marker
'_' hline marker

Trend line များရဲ့ color များ အတွက် ဆို အောက်ပါ colors များကို အသုံးပြုနိုင်သည်။

Character	Colour
'b'	blue
'g'	green
'r'	red
'c'	cyan
'm'	magenta
'y'	yellow
'k'	black
'w'	white

ထို့ပြင် figure ရဲ့ style ကိုလည်း မိမိတို့ လိုအပ်သလို ပြောင်းလဲချင်ရင် style များစာရှိပါသေးသည်။ styles များကို သိနိုင်ရန် print(plt.style.available) ဟုရေးပြီး run ကြည့်ပါက သိနိုင်သည်။အောက်တွင် ဖော်ပြထားသော style များ ဖြင့် မိမိတို့ရဲ့ figure style ကို အသုံးပြုနိုင်ပါသည်။

['Solarize_Light2', '_classic_test_patch', 'bmh', 'classic', 'dark_background', 'fast', 'fivethirtyeight', 'ggplot', 'grayscale', 'seaborn', 'seaborn-bright', 'seaborn-colorblind', 'seaborn-dark', 'seaborn-dark-palette', 'seaborn-darkgrid', 'seaborn-deep', 'seaborn-muted', 'seaborn-notebook', 'seaborn-paper', 'seaborn-pastel', 'seaborn-poster', 'seaborn-talk', 'seaborn-ticks', 'seaborn-white', 'seaborn-whitegrid', 'tableau-colorblind10']

စာရေးသူ အနေဖြင့် အောက်ပါ program တွင် fivethirtyeight ဆိုသည့် style ကို သုံးထားပါသည်။ အောက်ပါ program ၌ line 2 တွင်ကြည့်ပါ။

Bar Charts

ရှေ့က သင်ခန်းစာမှာတော့ plot ကို သုံးခဲ့ကြပြီး ယခု သင်ခန်းစာမှာတော့ matplotlib ထဲက bar chart ကို အသုံးပြုသွားပါမည်။ Bar chart သည် X axis data တစ်ခုတည်းနှင့် Y axis data အမျိုးအစား များစွာတို့ကို အတူနှိုင်းယှဉ်ပြဖို့ အဆင်ပြေပါတယ်။ ဥပမာ December 1 ရက်နေ့မှာ Temperature ဘယ်လောက်၊ Humidity ဘယ်လောက်၊ လေတိုက်နှုန်း ဘယ်လောက် ရှိတယ် စတဲ့ အချက်တွေကို နှိုင်းယှဉ်ကြည့်လို့ ရပါတယ်။ X axis ကတော့ December 1 ဖြစ်ပြီးကျန်သည့် data များမှာတော့ Y axis ဖြစ်ပါတယ်။ Bar Chart ကို အသုံးပြုပုံမှာ plt.plot မှ plot နေရာတွင် bar ကို အသုံးပြုရုံသာ ဖြစ်ပါတယ်။ ရေးသား ထားပုံကို အောက်ပါ program တွင် လေ့လာ ကြည့်ပါ။

```
import  matplotlib.pyplot as plt

plt.style.use("fivethirtyeight")
#Day for december 2020 in Pyin Oo Lwin
decDays =[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
#temp in C
Temp=[16,15,10,9,14,20,25,16,15,10,9,14,20,25,17]
#Humidity is the concentration of water vapor present in the air
Humidity=[39,37,32,34,33,27,29,30,35,36,29,40,33,36,37]
#wind speed in mile per hour/mph
WinSpeed=[6,7,4,5,7,8,5,10,7,9,6,5,8,9,12]

plt.bar(decDays,Temp,color='b',label='Temp')
plt.bar(decDays,Humidity,color='g',label='Humidity')
plt.bar(decDays,WinSpeed,color='y',label='WinSpeed')

plt.xlabel('15 Days in December2020')
plt.ylabel('Temp,Humidity,WinSpeed')
plt.title('Weather Forecast')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

အထက်ပါ program တွင် Y axis ၌ Temp , Humidity and WinSpeed စသည့် Data သုံးခုကို သုံးထားသော်လည်း output တွင် Data နှစ်မျိုးတည်းရဲ့ bar ကို သာပြပေးပါ့လိမ့်မယ်။ ဘာကြောင့်လဲဆိုတော့ bar method သည် parameter တွင် bar(x , height , width , bottom , align) ထိုပုံစံ အတိုင်း သွားပါတယ်။

X parameter နေရာသည် x axis အတွက်ဖြစ်ပြီး ထို x axis မှာ data များစွာကို ပြချင်လျှင် bar ရဲ့ အထူ width နဲ့ တည်နေရာကို သတ်မှတ် ပေးလို့ရပါတယ်။ အောက်ပါ program တွင် bar တစ်ခုစီ အတွက် 0.25 စီ သတ်မှတ် ပေးပြီး နေရာကိုပါ သတ်မှတ် ပေးထားပါတယ်။ ပထမဆုံး မြှားကတော့ bar တစ်ခုစီရဲ့ width အထူကို သတ်မှတ် ပေးတာ ဖြစ်ပြီး အောက်က မြှားသုံးခုကတော့ position ကို သတ်မှတ် ပေးတာပါ။ ပထမဆုံး တစ်ခုတွင် + 0.0 ကို ရေးထားသည့်အတွက် အစတွင် ဖော်ပြပေးပြီး ဒုတိယ တစ်ခုတွင် +0.25 ဟုရေးထား်းသည့်အတွက် 0.25 နေရာတွင် ဖော်ပြ ပေးပါတယ်။ နောက်ဆုံး တစ်ခုသည် +0.50 ဖြစ်သည့် အတွက် 0.50 နေရာတွင် ဖော်ပြ ပေးပါတယ်။

```
1  import  matplotlib.pyplot as plt
2  import  numpy as np
3  plt.style.use("fivethirtyeight")
4  #Day for december 2020 in Pyin Oo Lwin
5  decDays =[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
6  xData = np.arange(len(decDays))
7  width = 0.50
8  #temp in C
9  Temp=[16,15,10,9,14,20,25,16,15,10,9,14,20,25,17]
10 #Humidity is the concentration of water vapor present in the air
11 Humidity=[39,37,32,34,33,27,29,30,35,36,29,40,33,36,37]
12 #wind speed in mile per hour/mph
13 WinSpeed=[6,7,4,5,7,8,5,10,7,9,6,5,8,9,12]
14
15 plt.bar(xData+0.0,Temp,width=width,color='r',label='Temp')
16 plt.bar(xData+0.25,Humidity,width=width,label='Humidity')
17 plt.bar(xData+0.50,WinSpeed,width=width,color='y',label='WinSpeed')
18 plt.xticks(ticks=xData,labels=decDays)
19 plt.xlabel('15 Days in December2020')
20 plt.ylabel('Temp,Humidity,WinSpeed')
21 plt.title('Weather Forecast')
22 plt.legend()
23 plt.grid(True)
24 plt.tight_layout()
25 plt.show()
```

Line 18 မှ xticks သည် DecDays ထဲမှ data များကို x axis တွင် ပြပေးချင်သည့်အတွက် ရေးပေးထားခြင်း ဖြစ်ပါတယ်။ Details ပြောရမည် ဆိုလျှင် xticks method သည် အနည်းဆုံး parameter နှစ်ခု ယူပါတယ်။ ပထမ တစ်ခုသည် location ဖြစ်ပြီး ဒုတိယ တစ်ခုသည် ထို location နေရာတွင် ပေါ်ပေးမည့် Data ဖြစ်ပါတယ်။ ဥပမာ ပြုရမည် ဆိုလျှင် x=[1,2,3] , data=[a,b,c] ဟု သတ်မှတ်ပေးမည်ဆိုလျှင် x axis ရဲ့ 1,2,3 နေရာတွင် a,b,c စသည့် data များ ဖော်ပြနေမှာ ဖြစ်ပါတယ်။ xticks([0, 1, 2], ['January', 'February', 'March'])ယခုကဲ့သို့ ရေးမည်ဆိုလည်း ရပါတယ်။ Program output ကိုတော့အောက်မှာ ဖော်ပြထားပါတယ်။

Bar Charts and CSV

ယခုသင်ခန်းစာတွင် CSV မှ data တွေကို ရယူပြီး bar chart နဲ့ ပြန်ပြ ပေးမှာ ဖြစ်ပါတယ်။ libraries များအနေဖြင့် pandas , collections , matplotlib တို့ကို အသုံးပြုသွားပါမည်။ စာရေးသူတို့ အနေဖြင့် ယခင် matplotlibVM ဆိုသည့် project ထဲတွင်ပင် ရေးသားသွားမှာ ဖြစ်တဲ့ အတွက် ထပ်မံလိုအပ် နေသေးသည့် pandas and collections library များကို install ပေးရန် လိုအပ်ပါသည်။

```
(venv) C:\Users\winht\PycharmProjects\matplotlibVM\venv>pip list
Package         Version
--------------- -------
collection      0.1.6
cycler          0.10.0
kiwisolver      1.3.1
matplotlib      3.3.3
numpy           1.19.4
pandas          1.2.0
Pillow          8.0.1
pip             20.3.3
pyparsing       2.4.7
python-dateutil 2.8.1
pytz            2020.5
setuptools      51.1.0
six             1.15.0
```

Libraries များကို install လုပ်ပြီးနောက် လိုအပ်သည့် library များကို import လုပ်ပါမည်။ ထို့နောက် csv file အား ဖတ်ပါမည်။ ထို့ကြောင့် အောက်ပါ program အား စတင် ရေးသား ပါသည်။

To Download Data

Link 1

https://yadi.sk/d/dO4Jk0OfQEp76g

Or

Link 2

https://u.pcloud.link/publink/show?code=XZLnebXZcbOtKS50sohVY4bIlGnahfTtVo1V

Or

Link 3

https://drive.google.com/file/d/1e8FO7JSEc_E5FRXRCVhVUTj_WNGnrnkH/view?usp=sharing

```
import pandas as pd
from collections import Counter
from matplotlib import pyplot as plt

plt.style.use("fivethirtyeight")

data = pd.read_csv('ProgrammingUsed.csv')
ids = data['Id']
lang_responses = data['UsedLanguages']
```

နောက်တစ်ဆင့်အနေဖြင့် counter function ကိုအသုံးပြုမှာ ဖြစ်တဲ့အတွက် counter function အကြောင်းကို အသေးစိတ်ဦးစွာရှင်းလိုပါသည်။ Counter သည် input အနေဖြင့် list,

tuple, dictionary , string စသည့် input များဖြစ်တဲ့ iterable objects များကို ရယူပြီး output အနေဖြင့် ထို iterable object တွေထဲမှာ ပါဝင်တဲ့ elements အရေအတွက် တို့ကို count လုပ်ပြီး ပြန်ပေးပါတယ်။ စာရေးသူတို့ အနေဖြင့်လည်း ယခု projects မှာ csv ထဲမှ data တွေကို count လုပ်ချင်ပါတယ်။ ဥပမာ javascript အသုံးပြုသူ ဘယ်နှစ်ယောက် ရှိလဲ ဆိုတာမျိုးပါ။ sample data နဲ့ ပြရမည် ဆိုလျှင် ['x','y','z','x','x','x','y', 'z','x','b','w'] data ယခု ဖော်ပြထားတဲ့ data တွေထဲမှာ x ကဘယ်နှစ်ခုပါသလဲ y က ဘယ်နှစ်ခု ပါသလဲ စသဖြင့် ရေတွက်ပြီး ခွဲထုတ် ပေးပါတယ်။ ထိုသို့ ခွဲထုတ်ပြီး ရလာတဲ့ data တွေကိုလည်း key and value ပုံစံဖြင့် ပြန်ပြ ပေးပါတယ်။

```
from collections import  Counter
myIter= ['x','y','z','x','x','x','y', 'z','x','b','w']
print(Counter(myIter))
#output
#Counter({'x': 5, 'y': 2, 'z': 2, 'b': 1, 'w': 1})
```

ထို့ပြင် lang_responses ထဲမှ data များကို ထုတ်ကြည့်သည့်အခါ အောက်ပါအတိုင်း data များစွာကို ရရှိပါတယ်။ ထို data များရဲ့ တစ်ခုနှင့်တစ်ခု ကြားတွင် semicolon များ ခြားနေသည်ကို တွေ့ရပါမည်။ ထို semicolon များကို ဖျက်ထုတ်ရန် လိုအပ်နေပါသေးသည်။

```
lang_responses = data['UsedLanguages']

language_counter = Counter()

for response in lang_responses:
    print(response)
```

Output

```
Bash/Shell/PowerShell;C++;Python;Ruby;Other(s):
C;C++;Other(s):
Bash/Shell/PowerShell;C++;HTML/CSS;JavaScript
Bash/Shell/PowerShell;HTML/CSS;Python;Other(s):
Assembly;C++;HTML/CSS;VBA
C;C++;Java
Assembly;Bash/Shell/PowerShell;C;C#;HTML/CSS;Java;JavaScript;PHP;SQL;Other(s):
C;C++;HTML/CSS;JavaScript;PHP;Python;SQL
HTML/CSS;Java;JavaScript
HTML/CSS;JavaScript;Python
Bash/Shell/PowerShell;Go;HTML/CSS;JavaScript;WebAssembly
HTML/CSS;JavaScript;Other(s):
Bash/Shell/PowerShell;HTML/CSS;Java;JavaScript;PHP;SQL;Swift
```

ထို data များ ကြားတွင်ခြားနေသော semicolon များကို ဖျက်ထုတ်နိုင်ရန် split function ကို အသုံးပြု နိုင်ပါသည်။ split function ထဲတွင် parameter အနေဖြင့် ; ကို ထည့်ပေးလိုက်မည် ဆိုလျှင် ထို ; များ ကို ဖျက်ထုတ်ပေးမည် ဖြစ်ပြီး output အနေဖြင့် list ပုံစံကို ပြန်လည် ရရှိမှာ ဖြစ်ပါတယ်။

```
for response in lang_responses:
    print(response.split(';'))
```

```
['Bash/Shell/PowerShell', 'Go', 'HTML/CSS', 'JavaScript', 'WebAssembly']
['HTML/CSS', 'JavaScript', 'Other(s):']
['Bash/Shell/PowerShell', 'HTML/CSS', 'Java', 'JavaScript', 'PHP', 'SQL', 'Swift']
```

နောက်တစ်ဆင့် အနေဖြင့် List ပုံစံနဲ့ ရရှိလာသော data များအား ရေတွက်ရန် လိုအပ်ပါသေးသည်။ ထို့ကြောင့် counter function ကို အသုံးပြုပါမည်။ အထက်တွင် ဖော်ပြခဲ့သည့် အတိုင်း counter function သည် သူ့ထဲသို့ ထည့်ပေးလိုက်သော data များအား ရေတွက်ပြီး ပြန်လည် ခွဲထုတ် ပေးပါတယ်။ ထိုသို့ ခွဲထုတ်ပြီး ရလာတဲ့ data တွေကို update လုပ်ဖို့ လိုပါတယ်။ ဥပမာ ပထမ Looping ပတ်လိုက်တယ်၊ html တစ်ခါ ပါတယ် ၊ ဒုတိယ တစ်ခါ ထပ်ပတ်တယ်၊ နောက် တစ်ခါ ထပ်ပါတယ်၊ html က ဒါဆိုရင် html ချည်းပဲ နှစ်ခုရှိပြီပေါ့။ ထိုနည်းတူ နောက်တစ်ခေါက် Looping ထပ်ပတ်ပြီး ပါလာရင်လည်း နောက်တစ်ကြိမ် ထပ်ပြီး Update ထပ်လုပ်ရမှာပါ။ ထိုသို့ လုပ်ဆောင် ပေးဖို့ update function ကို အသုံးပြုနိုင်ပါတယ်။ sample program ကို အောက်မှာ ဖော်ပြ ပေးထားပါတယ်။

```
13  for response in lang_responses:
14      language_counter.update(response.split(';'))
15  print(language_counter)
```

```
Counter({'JavaScript': 59219, 'HTML/CSS': 55466, 'SQL': 47544, 'Python': 36443, 'Java': 35917, 'Ba
```

အထက်ပါအတိုင်း languages တွေ အားလုံးကို ရေတွက်ပြီးသား ရပါလိမ့်မယ်။ ရလာတဲ့ data တွေကို ကြည့်မည်ဆိုလျှင် ရှေ့က languages တွေဖြစ်ပြီး နောက်ကတော့ languages တွေကို အသုံးပြုတဲ့ နှုန်း ဖြစ်ပါတယ်။ ထို့ပြင် languages အမျိုးအစားပေါင်း များစွာရှိတဲ့ထဲက top 10 ဆယ်ခုကိုလည်း ခွဲထုတ်ဖို့ လိုနေပါသေးတယ်။ အကယ်၍ ထိုသို့ top 10 မပြလိုပဲ ရှိသမျှ languages တွေ အားလုံးရဲ့ အသုံးပြုနှုန်းကို ပြမည်ဆိုလည်း ရပါတယ်။

```
13  for response in lang_responses:
14      language_counter.update(response.split(';'))
15
16  languages =[] # to store languages
17  rate =[] # to store using times of languages
18
19  for i in language_counter.most_common(10):
20      languages.append(i[0])
21      rate.append(i[1])
22
23  print(languages,rate)
```

Line number 16 and 17 ကတော့ languages တွေနဲ့ ထို languages တွေကို အသုံးပြုတဲ့ နှုန်းတွေကို ခွဲပြီး သိမ်းထားဖို့ list နှစ်ခု တည်ဆောက်လိုက်ခြင်း ဖြစ်ပါတယ်။ line 19 မှာတော့ အသုံးအများဆုံး 10 ခုကိုသာ လိုချင်သည့်အတွက် most_common(10) ကို အသုံးပြုလိုက်ခြင်း ဖြစ်ပါတယ်။ line 19 မှာ Looping တစ်ကြိမ် ပတ်လိုက်တိုင်း ထွက်လာတဲ့ data ထဲက language ကို languages ဆိုတဲ့ list ထဲကို ထည့်ပြီး ဒုတိယ တစ်ခုဖြစ်တဲ့ အကြိမ်အရေ အတွက်ကိုတော့ rate ဆိုတဲ့ list ထဲ ထည့်ပါတယ်။ ထို Looping ပြီးသွားတဲ့အချိန် အသုံးအများဆုံး ဆယ်ခုရဲ့ language နဲ့ rate ကို ခွဲထားပြီးသား ဖြစ်နေပါပြီ။ မိမိ လိုချင်သည့် data နှစ်ခုရပြီဆိုတော့ ထိုdata တွေကို plot လုပ်လို့ရပါပြီ။ program အပြည့်အစုံကိုတော့ အောက်မှာ ဖော်ပြထားပါတယ်။

```python
import pandas as pd
from collections import Counter
from matplotlib import pyplot as plt

plt.style.use("fivethirtyeight")

data = pd.read_csv('ProgrammingUsed.csv')
ids = data['Id']
lang_responses = data['UsedLanguages']

language_counter = Counter()

for response in lang_responses:
    language_counter.update(response.split(';'))

languages =[] # to store languages
rate =[] # to store using times of languages

for i in language_counter.most_common(10):
    languages.append(i[0])
    rate.append(i[1])

plt.barh(languages,rate)
plt.title("Developer Favourite Languages")
plt.ylabel("Languages")
plt.xlabel("Rate")
plt.tight_layout()
plt.show()
```

အကယ်၍ JavaScript ကိုအပေါ်ဆုံးမှာ ပေါ်စေချင်တယ်ဆိုရင် reverse function ကို အသုံးပြုနိုင်ပါတယ်။reverse function ကို line 22 တွင် languages.reverse() နှင့် line 23 တွင် rate.reverse() စသည့် code နှစ်ကြောင်းထပ်မံရေးသားပေးရမှာဖြစ်ပါတယ်။ ထို့ပြင် ရှိသမျှ languages အားလုံးရဲ့ အသုံးပြုမှုကို လိုချင်ရင်တော့ line 19 မှ most_common method ကို ဖြုတ်လိုက် နိုင်ပါတယ်။

Pie Chart

Pie Chart ဟာ စက်ဝိုင်းပုံ graph တစ်ခု ဖြစ်ပြီး မတူညီတဲ့ အချက်အလက်တွေကို စုပေါင်းပြီး ပြပေးပါတယ်။ အချက်လက်တွေရဲ့ ပါဝင်တဲ့ ပမာဏပေါ် မူတည်ပြီး data တစ်ခုနဲ့တစ်ခုကြား ခြားနားစွာ ပြသပေးပါတယ်။ Pie chart ကို အသုံးပြုမယ်ဆိုရင် data 6 ခု သို့မဟုတ် 6 ခုထက် ပိုနည်းတာကို ပြသသင့်ပါတယ်။ 6 ခုထက် ပိုများသွားမယ်ဆိုရင်

ကျွန်တော်တို့မျက်လုံးကို data တစ်ခုနဲ့ တစ်ခုကြားမှန်ကန်စွာ ခွဲခြားနိုင်မှု အားနည်းလာတဲ့ အတွက်ကြောင့်ပါ။ Pie Chart ဟာလည်း အခြား chart များနည်းတူ အနည်းဆုံး list နှစ်ခု ယူပါတယ်။ Pie chart ကို အသုံးပြုဖို့အတွက် pie ဆိုတဲ့ method အကို အသုံးပြုပါတယ်။

```
import matplotlib.pyplot as plt
Monthly_payment = [2000,3000,1500,2500,4000]
Payment_labels = ['Phone','Internet','Foods','School','Cycle']
plt.pie(Monthly_payment,labels=Payment_labels)
plt.show()
```

နောက်တစ်ဆင့်အနေဖြင့် မိမိတို့ pie chart ရဲ့ size ကိုလည်း ကြီးလို့ရပါသေးတယ်။ ထိုသို့ ကြီးဖို့အတွက်ဆိုရင် radius ဆိုတဲ့ property ကိုသုံးရပါမည်။ ထို့ပြင် pie chart ထဲမှာပါဝင်တဲ့ data တွေရဲ့ ပါဝင်မှုအတိအကျကို percentage နဲ့ ပြလိုပါက autopct property ကို သုံးနိုင်ပါတယ်။ ထို့ပြင် မိမိတို့ ပြချင်တဲ့ data ကို သီးသန့် ခွဲပြလိုပါကလည်း explode property ကို သုံးနိုင်သလို data တွေကိုလည်း မိမိတို့ပြချင်တဲ့ နေရာအတိုင်းချိန်းလိုက်လို့ရပါသေးတယ်။ ဥပမာအထက်ပါ Pie chart မှာဆိုလျှင် phone သည် 0 degree ကနေ စပါတယ်။ အကယ်၍ phone ကို 45 degree ကနေ စလိုပါက startangle ဆိုတဲ့ property ကိုသုံးပြီး မိမိတို့ စစေချင်တဲ့ degree ကို ထည့်ပေးနိုင်ပါတယ်။

အထက်ပါ program မှာဆိုလျှင် radius အနေဖြင့် 1.5 ကို သုံးထားသည့်တွက် ယခင် program size ထက် .5 ပိုကြီးမှာ ဖြစ်ပါတယ်။ ဘာကြောင့်လဲဆိုတော့ radius ရဲ့ default size ဟာ 1 ဖြစ်ပါတယ်။ ထို့ပြင် data များကို percentage အလိုက် ပြထားပြီး ဒသမ နောက် သုည နှစ်လုံးကို ပြထားပါတယ်။ 0.2f သည် point နှစ်ခုကို ဆိုလိုခြင်း ဖြစ်ပါတယ်။ ထို့ပြင် shadow ကိုသုံးထားသည့် အတွက် pie chart ရဲ့ အောက်ဘက်ခြမ်းမှာ shadow ပါဝင် နေတာကို မြင်ရမှာပါ။ explode ကို နှစ်ခု သုံးထားပါသည်။ explode ကိုသုံးရာတွင် list ပေးသောအခါ ပါဝင်သော အစိတ်အပိုင်းအရေအတွက် အတိုင်း အတိအကျထည့်ပေးရပါမည်။ သို့မှသာ ဘယ်အပိုင်းကို explode လုပ်ချင်လဲဆိုတာ အတိအကျ သိနိုင်မှာပါ။ အထက်ပါ program မှာ ဆိုလျှင် list ထဲမှာ data 6 ခု ထည့်ထားပြီး ဒုတိယ data နှင့် နောက်ဆုံး data ကို explode လုပ်ထားပါသည်။ ဒီနေရာမှာ တစ်ခု ရှုပ်ထွေးနိုင်တာက ဒုတိယ နဲ့ နောက်ဆုံးဆိုတာကို ဘယ်လိုခွဲခြားနိုင်သလဲ။ နာရီ လက်တံ ပြောင်းပြန် ပုံစံဖြင့် ခွဲခြားနိုင်ပါတယ်။ ပထမဆုံး ပုံတွင် Phone သည် ရှေ့ဆုံးမှ ဖြစ်ပါတယ်။ ထို phone data သည် 0 degree ကနေ စပါတယ်။ ဒုတိယပုံကို ကြည့်လျှင် Phone ရဲ့ အစသည် 90 degree သို့ပြောင်းသွားပါတယ်။ အဘယ်ကြောင့်ဆိုသော် startangle property ကို 90 degree သို့ သတ်မှတ် ပေးလိုက်သောကြောင့် ဖြစ်သည်။ ထို့ကြောင့် Phone သည် ပထမ ဖြစ်ပြီး Internet သည် ဒုတိယ ဖြစ်သည်။ ထိုနည်းတူ Cycle သည် နောက်ဆုံးဖြစ်သည်။ ယခု ဖော်ပြချက်သည် Pie chart ကို ကြည့်ပြီးခွဲခြားခြင်းဖြစ်သည်။ ထိုပြင် မိမိတို့ ပေးထားသော list ကို ကြည့်ပြီးတော့လည်း ခွဲခြား နိုင်ပါသည်။

Matplotlib Circle (Donut Chart)

ယခင်သင်ခန်းစာမှာတော့ pie chart ကို ဖန်တီးခဲ့ပြီး ယခု သင်ခန်းစာမှာတော့ donut chart ကို ဖန်တီးကြည့်ပါမည်။ Donut Chart ဆိုတော့အလယ်မှာ စက်ဝိုင်းပုံ အပေါက်နဲ့ ဖြစ်ပါတယ်။ ထို့ကြောင့် ထိုစက်ဝိုင်းပုံ အပေါက် တစ်ခုကိုဖန်တီးပါမည်။ Matplotlib တွင် Circle တစ်ခုကို ဖန်တီးရာ၌ အပိုင်းနှစ်ပိုင်း ရှိပါသည်။ ပထမ တစ်ပိုင်းသည် circle size , color တို့ကို သတ်မှတ် ပေးသည့် အပိုင်းဖြစ်ပြီး ဒုတိယ အပိုင်းသည် ထို circle ကို ဆွဲသည့်

အပိုင်းဖြစ်ပါသည်။ Circle ရဲ့ အစိတ်အပိုင်းများကို ဖန်တီးရာတွင် Matplotlib မှ circle ဆိုသည့် method ကို အသုံးပြုပြီး ဆွဲရာတွင် add_artist ဆိုသည့် method ကို သုံးပါသည်။

Circle method သည် circle အတွက် parameter သုံးခု ယူပါသည်။ ပထမ တစ်ခုသည် circle ရဲ့ centre အတွက် ဆိုလိုခြင်း ဖြစ်ပြီး ဒုတိယ တစ်ခုသည် စက်ဝိုင်းရဲ့ size r(radius) ဖြစ်သည်။ နောက်ဆုံး တစ်ခုသည် ထို စက်ဝိုင်းထဲတွင် ဖြည့်မည့် color ဖြစ်ပါသည်။ ထိုကဲ့သို့ သတ်မှတ်ပြီးသော circle ရဲ့ object တစ်ခုလုံးကို add_artist ဆိုသည့် method ထဲသို့ parameter တစ်ခု အနေဖြင့် ထည့်ပေးလိုက်ရပါသည်။ gcf သည် get current figure ဖြစ်ပြီး လက်ရှိ figure ကို ရယူပါတယ်။ အကယ်၍ current figure မရှိရင် အသစ် တစ်ခု ဖန်တီးပေးပါတယ်။

```
import matplotlib.pyplot as plt
Monthly_payment = [2000,3000,1500,2500,4000]
Payment_labels = ['Phone','Internet','Foods','School','Cycle']
plt.pie(Monthly_payment,labels=Payment_labels,radius=1.5,autopct='%0.2f%%'
        shadow=True,
        explode=[0.05,0.05,0.05,0.05,0.05],startangle=90)
centre_circle= plt.Circle( (0,0),1.2,fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)

plt.axis('equal')
plt.tight_layout()
plt.show()
```

အထက်ပါ program ကို ကြည့်မည် ဆိုလျှင် percentage တန်ဘိုးတွေဟာ နေရာ အတည်တကျမဖြစ်တော့ပဲ သက်ဆိုင်ရာ data များနဲ့ အနည်းငယ် ကွာဟ နေသည်ကို မြင်ရပါမည်။ data များနှင့် တစ်ထပ်တည်း ကျနေစေရန် pctdistance ဆိုသည့် method ကို သုံးရပါမည်။ ထို function သည် autopct ဆိုသည့် attribute နှင့် အလုပ် အတူလုပ်ပါသည်။ အကယ်၍ autopct ကို မသုံးဘူး ဆိုလျှင်တော့ pctdistance ကို သုံးစရာမလိုပါဘူး။ pctdistance နဲ့ autopct က ထုတ်ပေးလိုက်တဲ့ value တွေကို သူနဲ့ သက်ဆိုင်ရာ နေရာတွေမှာ ပေါ်ပေးနိုင်ဖို့ ကူညီပေးပါတယ်။ သူ့ရဲ့ default value က 0.6 ဖြစ်ပါတယ်။

```python
import matplotlib.pyplot as plt
Monthly_payment = [2000,3000,1500,2500,4000]
Payment_labels = ['Phone','Internet','Foods','School','Cycle']
plt.pie(Monthly_payment,labels=Payment_labels,radius=1.5,autopct='%0.2f%%',
        shadow=True,explode=[0.05,0.05,0.05,0.05,0.05],
        startangle=90,pctdistance=0.9)
centre_circle= plt.Circle( (0,0),1.2,fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)

plt.axis('equal')
plt.tight_layout()
plt.show()
```

နောက်ထပ် program တစ်ပုဒ်အနေဖြင့် ရှေ့တွင် ရေးဆွဲခဲ့သော Donut Chart အလည်တွင် နောက်ထပ် cricle တစ်ခု ထပ်ဆွဲပါမည်။ ပထမဆုံးအနေဖြင့် problem တစ်ခုကို စတင်စဉ်းစား ကြည့်ကြပါစို့။ Programming languages 4 ခုရှိပြီး အသုံးပြုသည့် လူဦးရေ 500,400,379,330 တွင် အမျိုးသမီး/ အမျိုးသား များ ပါဝင်နေပါသည်။ ထိုအချက်လက်များပေါ်တွင် မူတည်ပြီး Pie chart နှစ်ခုကိုဆွဲကာ ပထမ တစ်ခုသည် data အချက်အလက်များကို ဖော်ပြမည်ဖြစ်ပြီး ဒုတိယ တစ်ခုသည် အမျိုးသမီး /အမျိုးသား များကိုခွဲခြားကာ ဖော်ပြပေးမည့် pie chart ဖြစ်သည်။

```
1   import matplotlib.pyplot as plt
2
3   Programming = ['Python', 'C++', 'Ruby', 'Java']
4   Users = [504, 337, 415, 280]
5   labels_gender = ['Man','Woman','Man','Woman','Man','Woman','Man','Woman']
6   man_woman = [315,189,125,212,270,145,190,90]
7   pro_colors = ['#ff6666', '#C64DD5', '#D1A831', '#87A513']
8   m_w_colors = ['#20D9D6','#985FD6', '#20D9D6','#985FD6', '#20D9D6','#985FD6',
9                 '#20D9D6','#985FD6']
10
11  plt.pie(Users, labels=Programming, colors=pro_colors, startangle=45)
12  plt.pie(man_woman,colors=m_w_colors,radius=0.75,startangle=45)
13
14  donut_chart = plt.Circle((0,0),0.5,color='black', fc='white',linewidth=0)
15
16  fig = plt.gcf()
17  fig.gca().add_artist(donut_chart)
18
19  plt.axis('equal')
20  plt.tight_layout()
21  plt.show()
```

အထက်ပါ program တွင် line 3 and 4 ၌ programming languages တွေနှင့် ထို programming languages တွေကို အသုံးပြုတဲ့ users တွေအတွက် list နှစ်ခု ဆောက်ထားပါတယ်။ line 4 and 5 တွင် အသုံးပြုသူ man and woman နှင့် man ဘယ်နှစ်ယောက် woman ဘယ်နှစ် ယောက်သုံးလဲ ဆိုတာ သိနိုင်စေရန် list နှစ်ခု ထပ်ဆောက်လိုက်ပါသည်။ line 7 and 8 တွင်တော့ programming languages တွေအတွက် color နှင့် man and woman အတွက် colors များကို ခွဲပေးထားပါသည်။ line 11 and 12 တွင် pie chart နှစ်ခုအတွက် data များကို ထည့်ပေးပါသည်။ ပထမ Pie chart အတွက် radius ကို 0.5 အနေဖြင့်သာပေးထားပြီး ဒုတိယ တွင်တော့ 0.75 ကို ပေးထားပါသည်။ ထို့ပြင် နှစ်ခုလုံးရဲ့ startangle ကိုလည်း 45 degree စီ ပေးထားပါသည်။

အကယ်၍ အထက်ပါ ပုံတွင် explode ကို ထပ်ထည့်လိုပါက ပိုမိုထင်ရှားစေရန် circle ရဲ့ radius နဲ့ အတူ အောက်ပါ အတိုင်း ပြောင်းနိုင်ပါသည်။

```
plt.pie(Users, labels=Programming, colors=pro_colors,
        startangle=45,explode=explode,radius=3)
plt.pie(man_woman,colors=m_w_colors,radius=2,
        startangle=45,explode=explode_gender)
```

Matplotlib Real Time Data

ယခု သင်ခန်းစာတွင် real time data တွေကို matplotlib နဲ့ ဖော်ပြသွားမှာ ဖြစ်ပြီး real time data တွေကိုတော့ ယခု သင်ခန်းစာမှာ computer cpu ကနေ တစ်ဆင့် ရယူမှာ ဖြစ်ပါတယ်။ cpu နှင့် ပတ်သက်သည့် function call များကိုခေါ်သုံးနိုင်ရန်အတွက် psutil ဆိုသည့် library ကို install လုပ်ပေးရမှာ ဖြစ်ပါတယ်။ ထို့ပြင် real time data များသည် အမြဲတမ်းတိုးပွားလာမှာ ဖြစ်တဲ့ အတွက် ထို တိုးပွားလာတဲ့ data တွေကို လိုက်ပြ ပေးနိုင်ရန် FuncAnimation ဆိုသည့် method ကို matplotlib.animation ထဲမှ import လုပ်ပေးရန်လည်း လိုအပ်ပါသည်။

```
1  import psutil
2  import matplotlib.pyplot as plt
3  from itertools import  count
4  from matplotlib.animation import FuncAnimation
5  plt.style.use('fivethirtyeight')
6
7  x_vals=[]
8  y_vals=[]
9
10 index=count()
11 def draw(i):
12     x_vals.append(next(index))
13     y_vals.append(psutil.cpu_percent())
14     plt.cla()
15     plt.plot(x_vals,y_vals)
16 animat = FuncAnimation(plt.gcf(),draw, interval=1000)
17
18 plt.tight_layout()
19 plt.show()
```

အထက်ပါ program တွင် line 10 ၌ count method ကို ခေါ်ထားပါသည်။ count method သည် ပုံမှန်အားဖြင့် parameter နှစ်ခု ယူပါသည်။ ပထမတစ်ခုသည် စမှတ်ဖြစ်ပြီး ဒုတိယ တစ်ခုသည် step ဘယ်လို ပုံစံ တိုးသွားမလဲ ဆိုတာ ဖြစ်ပါတယ်။ စာရေးသူအနေဖြင့် count method ထဲတွင် မည်သည့် parameter မှ ထည့်ပေးထားခြင်းမရှိသည့်အတွက် default zero ဖြင့် အလုပ် လုပ်သွားမည် ဖြစ်သည်။ စာရေးသူအနေဖြင့် line 12 တွင် next ကို သုံးထားတာဖြစ်သည့်အတွက် သူ့ရဲ့ နောက် values တွေကို ဖော်ပြနေမည် ဖြစ်သည်။ ပထမဆုံး အကြိမ်တွင် zero က စသော်လည်း next ဒုတိယ value သည် 1 ဖြစ်သည့် အတွက် 1 ကနေ စတင် သတ်မှတ်လိမ့်မည် ဖြစ်သည်။ ယခု အဆင့်အား ပိုမို နားလည်ရန် debug လုပ်သင့်ပေသည်။ line 13 တွင် cpu_percent() ကို သုံးထားပါသည်။ cpu_percent() method သည် ယခုလက်ရှိ မိမိတို့အသုံးပြုနေသော cpu ရဲ့ အသုံးပြုနှုန်းကို percentage နဲ့ ပြန်လည်ဖော်ပြ ပေးပါသည်။ အကယ်၍ cpu_percent() method ကို သုံးမည် ဆိုလျှင် parameter အနေဖြင့် time interval(seconds) ကိုထည့်ပြီး သုံးသင့်ပါသည်။ သို့သော် စာရေးသူသည် line 16 တွင် interval ကို 1000 milliseconds ခံထားပါသည်။ 1000 milliseconds သည် 1 second ဖြစ်ပါသည်။ line 14 တွင် ရေးထားသော cla() method သည် clear an axes ကို ဆိုလိုခြင်းဖြစ်ပြီး ယခုလက်ရှိ figure တွင် ဖော်ပြနေသော axes တွေကို clear လုပ်ပေးရန် သုံးပါသည်။

Line 16 FuncAnimation method ကတော့ matplotlib ထဲမှ animation ဆိုတဲ့ class မှ လာခြင်းဖြစ်ပြီး ပြောင်းလဲ နေတဲ့ data တွေ wave form တွေ စတဲ့ အချက်လက်တွေကို animation ပုံစံဖြင့် ဖော်ပြရာတွင် အသုံးပြုပါတယ်။ plt.gcf() method သည် get current figure ကို ဆိုလိုခြင်း ဖြစ်ပြီး ယခုလက်ရှိ ဖော်ပြထားတဲ့ figure ကို ရယူလို တဲ့အချိန်မှာ အသုံးပြုပါတယ်။ အကယ်၍ figure မရှိဘူး ဆိုလျှင်လည်း figure() function ကို သုံးပြီး figure တစ်ခုကို ဖန်တီးပါသည်။ ဒုတိယ တစ်ခုကတော့ draw ဆိုတဲ့ function ကို လှမ်းခေါ်တာ ဖြစ်ပါတယ်။ draw function ကို ခေါ်တဲ့ အချိန်မှာ သူ့နောက်မှာ ဘာ parameter မှ ပါမသွားသော်လည်း draw function ကို ကြေငြာတဲ့ အခါမှာတော့ paramter တစ်ခုကို ထည့်ပေးထားရပါတယ်။ ဘာကြောင့်လဲဆိုတော့ draw function ကို ခေါ်တဲ့ အချိန်မှာ

FuncAnimation method ကနေ တစ်ဆင့် frame , *fargs ဆိုတဲ့ parameter နှစ်ခု ထည့်ပေးလိုက်တဲ့အတွက်ကြောင့် ဖြစ်ပါတယ်။ draw ထဲမှ i ကို တစ်နည်းအားဖြင့် frame number လို့လည်း ခေါ်ပါတယ်။ နောက်ဆုံး တစ်ခု interval ကတော့ frame တစ်ခု နဲ့ တစ်ခုကြား ခေါ်တဲ့ ကြာချိန် ဖြစ်ပါတယ်။ ထို interval ရဲ့ default value သည် 200 milliseconds ဖြစ်ပါတယ်။

Wave Form (FuncAnimation)

နောက်ထပ် program တစ်ပုဒ်အနေဖြင့် FuncAnimation ကို ပိုမိုကျွမ်းကျင်သွားစေရန် wave form program ကို ဆက်လေ့လာကြပါစို့။

```
1   import numpy as np
2   from matplotlib import pyplot as plt
3   from matplotlib import animation
4
5   fig = plt.figure()
6   ax = plt.axes(xlim=(0, 2), ylim=(-2, 2))
7   line, = ax.plot([], [], lw=2)
8   def init():
9       line.set_data([], [])
10      return line,
11  def animate(i):
12      x = np.linspace(0, 2, 1000)
13      y = np.sin(2 * np.pi * (x - 0.01 * i))
14      line.set_data(x, y)
15      return line,
16  anim = animation.FuncAnimation(fig, animate, init_func=init,
17                                 frames=200, interval=20, blit=True)
18
19  plt.show()
```

အထက်ပါ program တွင် line 5 ၌ fig object တစ်ခုကို Figure() method မှ တည်ဆောက်ထားပါသည်။ Figure များ ဆွဲရန်အတွက် ဖြစ်ပါသည်။ Line 6 တွင် axes method ကို သုံးပြီး xlim and ylim value များကို သတ်မှတ်ပေးထားပါသည်။ lw ကတော့ linewidth ကို ဆိုလိုချင်တာပါ။ line 7 , line 10 , line 15 များနောက်မှ line နောက်တွင် , ပါဝင်သည်ကို သတိပြုပါ။ , မပါဝင်ဘူးဆိုလျှင် Line2D object is not iterable ဆိုတဲ့ TypeError တက်ပါလိမ့်မယ် line 12 and 13 သည် numpy ထဲမှ linspace နှင့် sin method ကို ခေါ်သုံးထားတာဖြစ်ပြီး ရှေ့ပိုင်းသင်ခန်းစာများတွင် ရှင်းခဲ့ပြီး ဖြစ်ပါသည်။

FuncAnimation method ထဲတွင် သုံးခုမြောက်၌ init_func သည် နောက်ထပ် ဘာမှ မရှိသေးတဲ့ frame တစ်ခု ထပ်ဆွဲဖို့ အတွက် function call ခေါ်ပေးတာဖြစ်ပါတယ်။ အခေါ်ခံရသော function နှင့် နောက်ဆုံး parameter ဖြစ်သည့် blit သည် ဆက်စပ်နေပါတယ်။ အကယ်၍ blit ရဲ့ value ကို True ဟု ပေးခဲ့မည် ဆိုလျှင် init_func မှ ခေါ်လိုက်သော init function သည် return value ကို ပေးကို ပေးရမည် ဖြစ်ပြီး ထို value များကို blitting algorithm က အသုံးပြုပြီး figure ရဲ့ ဘယ်အပိုင်းတွေဟာ ပြန်ဆွဲဖို့ လိုသလိုဆိုတာကို ဆုံးဖြတ်ပါတယ်။ အကယ်၍ blit value သည် false ဖြစ်နေခဲ့မည် ဆိုလျှင် return value တွေကို ပြန်မသုံးတော့ပါဘူး။ သတိပြုရန် အချက်မှာ line 15 မှ line, သည် modified ဖြစ်တဲ့ data အသစ်တွေ ဖြစ်တဲ့ tuple data တွေကို return ပြန်ပေးခြင်းဖြစ်ပြီး ထို data တွေသည် Animation framework ထဲ၌ ဘယ်အပိုင်းကို animate လုပ်ရမယ် ဆိုတာ ပြောပြ ပေးပါတယ်။

3D Plot

ယခု lesson မှာတော့ chart တွေကို 3d plot တွေဖြင့် ဖော်ပြသွားမှာ ဖြစ်ပါတယ်။ 3d plot ဖြင့် ဖော်ပြပေးနိုင်ဖို့အတွက် axes3d ကို Import လုပ်ပေးရန် လိုအပ်ပြီး ထို object သည် mpl_toolkits.mplot3d class ထဲမှဖြစ်ပါသည်။ ထို့ကြောင့် matplotlib library ကို install လုပ်ထားရုံဖြင့် အသုံးပြုနိုင်ပါသည်။

X and y အတွက် value တွေကိုတော့ line 4 and 5 တွင်ဖော်ပြထားပြီး numpy array ကို သုံးပြီး float data တွေကို ထုတ်ထားပါသည်။ Line 7 မှာတော့ meshgrid method ကိုသုံးကာ One dimensional array နှစ်ခုကနေ multi dimensional array ရအောင် ဖန်တီးလိုက်တာပါ။ ထိုမှ တစ်ဆင့် x and y positions များကို ရှိပါတယ်။ numpy သင်ခန်းစာရဲ့ meshgrid array မှာ ဖော်ပြ ခဲ့ပြီး ဖြစ်ပါတယ်။ line 8 တွင် x , y ရဲ့ မှ value များရဲ့ အမြင့်များကို ဖော်ပြချင်တဲ့ အတွက် z တန်ဘိုး ကိုပါ ရှာထားပါတယ်။ ထို့ကြောင့် data တစ်ခုမှာ x , y နှင့် အမြင့် ပမာဏ z တန်ဘိုးပါ ရှိနေမှာ ဖြစ်ပါတယ်။

```python
import  matplotlib.pyplot as plt , numpy as np
from mpl_toolkits.mplot3d import  axes3d

x = np.array(range(0,3),float)
y = x.copy()

xpos ,ypos = np.meshgrid(x,y)
z = np.random.rand(3,3)

print("xpos\n",xpos)
print("ypos\n",ypos)
print("z",z)
```

```
xpos
 [[0. 1. 2.]
 [0. 1. 2.]
 [0. 1. 2.]]
ypos
 [[0. 0. 0.]
 [1. 1. 1.]
 [2. 2. 2.]]
z [[0.35921468 0.59425909 0.72024001]
 [0.43059528 0.50480525 0.65090816]
 [0.1035428  0.43071996 0.65811003]]
```

အထက်ပါ program ရဲ့ Output များကို ကြည့်မည်ဆိုလျှင် xpos data 9 ခု ypos data 9 ခုကို ရရှိနေပါသည်။ ထို xpos and ypos ရဲ့ x,y ရဲ့ အမှတ်များကိုထုတ်ကြည့်မည်ဆိုလျှင် x,y စုစုပေါင်း အမှတ် ၉ ခု ရရှိနေပါပြီ။ သို့သော် ယခုအတိုင်း plot လုပ်ကြည့်မည်ဆိုလျှင် scatter plot နှင့်အောက်ပါ ပုံအတိုင်းထုတ်ကြည့်နိုင်သည်။ plt.scatter(xpos,ypos)

ဒုတိယ တစ်ဆင့် အနေဖြင့် xpos and ypos များကို row ပုံစံ ပြန်ပြောင်းဖို့ လိုပါသေးသည်။ အဘယ်ကြောင့်ဆိုသော် meshgrid မှ data များသည် two dimensions ဖြစ်နေပါသည်။ ထို data အားလုံးအား row ပုံစံဖြင့် One dimensional အတိုင်းပြန်ယူလိုပါက flatten() method ကို အသုံးပြု ရပါမည်။ သတိပြုရန် အချက်မှာ 3D ဖြစ်ဖို့ ဆိုလျှင် x,y,z values များ လိုအပ်ပါသည်။ z values ရ ရန်အတွက် np.zero_like(xpos) ကို သုံးလိုက်ပါသည်။ zero_like သည် zero တွေချည်းပဲ ဖန်တီး ပေးမှာ ဖြစ်ပြီး xpos သည် zero အရေအတွက် ဖြစ်ပါသည်။

```python
xpos = xpos.flatten()
ypos = ypos.flatten()
zpos = np.zeros_like(xpos)
```

တတိယ အဆင့် အနေဖြင့် x,y,z တို့ရဲ့ dx, dy and dz dimensions များကို ဖန်တီးပေးရန် လိုအပ်ပါသည်။ dx and dy values များကို np.ones(xpos) ဆိုပြီး နှစ်ခုလုံး အတွက် ဖန်တီးပေး နိုင်ပါသည်။

dx= np.ones(xpos)

dy=np.ones(xpos) or dy.copy() ဟုရေး သားနိုင်ပါသည်။ dz ကတော့ မိမိတို့ ထည့်ပေးချင်သည့် dimensions များကို ထည့်ပေးနိုင်ပါသည်။ ဥပမာ အနေဖြင့် အောက်ပါအတိုင်း ရေးသား နိုင်သည်။

```
dx = np.ones(10)
dy = np.ones(10)
dz = [1,2,3,4,5,6,7,8,9,10]
```

အထက်ပါ အတိုင်း np.ones ကို သုံးမည် ဆိုလျှင် Numpy.float64 ဆိုသည့် error တက်နိုင်ပါသည်။အကယ်၍ error တက်ခဲ့လျှင် np.ones_like(zpos) ကို သုံးနိုင်ပါသည်။ ထိုကဲ့သို့ အသုံးပြုမည်ဆိုလျှင် 1 များထုတ်ပေးပါမည်။ 1 များ ထုတ်ပေးခဲ့လျှင် 1 အပြည့် ဖြစ်နေသည့် အတွက် အောက်ပါ ပုံအတိုင်း တစ်ခုနှင့် တစ်ခု ပူးကပ် နေနိုင်ပါသည်။ ထို့ကြောင့် စာရေးသူ အနေဖြင့် program ထဲတွင် 0.5*np.ones_like(zpos) ဟု ရေးထားပါသည်။

ယခုဆိုလျှင် 3d bar အတွက် data အချက်လက်များ အားလုံး လုံလောက်ပြီ ဖြစ်ပါသည်။ program အပြည့်အစုံကို စာရေးသူ အနေဖြင့်အောက်မှာ ဖော်ပြထားပါသည်။ စာရေးသူ အနေဖြင့် dz အတွက် values များကို np.random.rand မှ တစ်ဆင့် ရယူ ထားပါသည်။ ထိုသို့ ရရှိလာသော values များကို flatten သုံးကာ စာရေးသူတို့ လိုချင်သည့် ပုံစံ အတိုင်း ပြန်ပြောင်း ထားပါသည်။

```python
import  matplotlib.pyplot as plt , numpy as np
from mpl_toolkits.mplot3d import  axes3d

x = np.array(range(0,3),float)
y = x.copy()

#meshgrid ကိုသုံးပြီး data များ ဖန်တီးထားပါသည်။
#မိမိတို့ အစားထိုးလိုသည့် data များကို x and y နေရာတွင် သုံးနိုင်သည်
xpos ,ypos = np.meshgrid(x,y)
z = np.random.rand(3,3) # z dimension အတွက်ဖြစ်သည်

xpos = xpos.flatten()
ypos = ypos.flatten()
zpos = np.zeros_like(xpos)
#x,y,z အတွက် တည်နေရာများ ဖန်တီး ပေးခြင်း ဖြစ်ပါသည်
#3d ဖြစ်သည်အတွက် x y z values သုံးခု ဖန်တီး ပေးရသည်။

# 1 တွေချည်းပဲ ထွက်လာသည့် values
dx = np.ones_like(zpos)
dy = dx.copy()
dz = z.flatten()

colors = ["b","g","crimson"]*3
fig = plt.figure(figsize=(7,5))
ax = fig.add_subplot(111 , projection ="3d")
ax.bar3d(xpos , ypos ,zpos,dx ,dy,dz,color=colors)
plt.show()
```

အထက်တွင် ဖော်ပြထားသော program အား x_lable , y_lable and z_lable များနှင့် အတူ x တွင် ပါဝင်သော အရေ အတွက် y တွင် ပါဝင်သော အရေ အတွက်တို့ကို ထည့်နိုင်ပါသေးသည်။ အထက်ပါ ပုံကို စာရေးသူတို့အနေဖြင့် MPT, Ooredoo , Telenor စသည့် data များ ထည့်သွင်း ကြည့်ပါမည်။ x နေရာတွင် အရေအတွက်ကို ထားပြီး y နေရာတွင် services များကို ထည့်ကာ ဥပမာ voice , message , internet တို့ကို ထားပြီး z နေရာတွင် ထို services များ အသုံးပြုခြင်း အတွက် ကုန်ကျ စရိတ်များကို ထားပါမည်။

```python
import  matplotlib.pyplot as plt , numpy as np
from mpl_toolkits.mplot3d import  axes3d

x = np.array(range(0,3),float)
y = x.copy()

#meshgrid ကိုသုံးပြီး data များ ဖန်တီးထားပါသည်။
#မိမိတို့ အစားထိုးလိုသည့် data များကို x and y နေရာတွင် သုံးနိုင်သည်
xpos ,ypos = np.meshgrid(x,y)
z = np.random.rand(3,3) # z dimension အတွက်ဖြစ်သည်

xpos = xpos.flatten()
ypos = ypos.flatten()
zpos = np.zeros_like(xpos)
#x,y,z အတွက် တည်နေရာများ ဖန်တီး ပေးခြင်း ဖြစ်ပါသည်
#3d ဖြစ်သည်အတွက် x y z values သုံးခု ဖန်တီး ပေးရသည်။

# 1 တွေချည်းပဲ ထွက်လာသည့် values
dx = 0.5*np.ones_like(zpos)
dy = dx.copy()
dz = z.flatten()

colors = ["b","g","crimson"]*3
fig = plt.figure(figsize=(7,5))
ax = fig.add_subplot(111 , projection ="3d")

# figure ရဲ့ title ကိုထည့်မည်
ax.set_title("Myanmar Operators",fontsize=20)

# figure ရဲ့  x အတွက် lable ကို သတ်မှတ်ပေးမည်
ax.set_xticklabels([3,2,1]) #နံပါတ်စဉ် တပ်ပေးခြင်း
ax.set_xticks(range(3)) # အရေအတွက်ကို ဖော်ပြ ပေခြင်း
ax.set_xlabel("GoodServices",labelpad=7)
```

```
34
35  # y သည်လည်း x အတိုင်းပင် ဖြစ်သည် သို့သော် y သည် sevices များကို
36  # ဖော်ပြသည် ။
37  ax.set_yticklabels(["Voice","SMS","Internet"])
38  ax.set_yticks(range(3))
39  ax.set_ylabel("Operators",labelpad=7)
40
41  # z ကတော့ pricing ကို ဖော်ပြဖို့ အတွက်သုံးထားပါသည်။
42  ax.set_zticklabels([0,100,200,300,400])
43  ax.set_zticks(np.arange(0.0,1.0,0.2))
44  ax.set_zlabel("Pricing Kyats")
45
46  x_leg=plt.Rectangle((0,0),1,1,fc="b")
47  y_leg=plt.Rectangle((0,0),1,1,fc="g")
48  z_leg=plt.Rectangle((0,0),1,1,fc="crimson")
49
50  labels =["MPT","Ooredoo","Telenor"]
51  ax.legend([x_leg,y_leg,z_leg],labels,fontsize=12,loc=(1.1,0.4))
52  ax.bar3d(xpos , ypos ,zpos,dx ,dy,dz,color=colors)
53  plt.show()
```

Image Processing Project

ယခု သင်ခန်းစာတွင် image ပုံတစ်ခုကို ခဲတံဖြင့် ရေးဆွဲထားသည့် ပုံစံသို့ ပြောင်းသည့် project တစ်ခုကို လုပ်ဆောင်သွားမှာ ဖြစ်ပါတယ်။ အောက်မှာပြထားတာကတော့ ယခု project မှာ အသုံးပြုမည့် libraries များ ဖြစ်ပါတယ်။

```
1  import cv2
2  import easygui
3  import numpy as np
4  import imageio
5
6  import sys
7  import matplotlib.pyplot as plt
8  import os
9  import tkinter as tk
10 from tkinter import filedialog
11 from tkinter import *
12 from PIL import ImageTk, Image
```

Line 1 cv2 သည် open-source computer vision library တစ်ခု ဖြစ်ပြီး machine learning , image processing တို့ အတွက် python ,c++ , java,etc စတဲ့ programming languages တွေနဲ့ support လုပ်ထားပါတယ်။ Image နဲ့ videos တွေကို processing လုပ်ဖို့ဆိုရင်တော့ opencv က အသုံးများဆုံး library ထဲမှာ ပါ ပါတယ်။ ဥပမာ objects တွေကို detection လုပ်ခြင်း၊ faces recognition လုပ်ခြင်း ၊လူတွေ ရေးထားတဲ့ handwriting စာတွေကို computer က နားလည်အောင် ပြန် ပြောင်းယူခြင်း စတဲ့ နေရာတွေမှာလည်း အသုံးပြုပါတယ်။ opencv ထွက်ထားတဲ့ version တွေကိုတော့ https://github.com/opencv/opencv/releases ယခု Link မှာ သွားရောက် ကြည့်ရှု နိုင်ပါတယ်။

Line 2 easygui သည် Graphical User Interface application တွေ ရေးဖို့ အတွက် အရမ်းကို လွယ်ကူတဲ့ module တစ်ခု ဖြစ်ပါတယ်။ GUI နဲ့ ပတ်သက်တဲ့ လုပ်ဆောင်ချက်တွေကို function call ခေါ်လိုက်ရုံနဲ့ အလွယ် တကူကို ရရှိ နိုင်ပါတယ်။ ဥပမာ message box တစ်ခုကို title , button တွေနဲ့ ဖော်ပြချင်တယ် ဆိုရင် msgbox ဆိုတဲ့ function ကို ခေါ်သုံးလိုက်ရုံပါပဲ။ သူ့မှာ parameter သုံးခု ထည့်ပေး ရပါတယ်။ ပထမ တစ်ခုကတော့ ပေါ်လာမယ့် form မှာ ပေါ်စေချင်တဲ့ စာသားတွေဖြစ်ပြီး ဒုတိယ တစ်ခုကတော့ title ပါ ထို title ကတော့ form ရဲ့ ထိပ်ဆုံးမှာ ပေါ်စေချင်တဲ့ စာသားကို ထည့်ဖို့ ဖြစ်ပါတယ်။တတိယ တစ်ခုကတော့ buttonပါ၊ ထို button ကို နှိပ်လိုက်ရင် form လေးက ပြန်ပိတ်သွားမှာ ဖြစ်ပါတယ်။ ထို button အတွက် စာသားကိုလည်း variable တစ်ခု ကြေငြာသလို ထည့်ပေးနိုင်ပါတယ်။ အောက်မှာတော့ sample program တစ်ပုဒ် ရေးပြထားပါတယ်။

```
1  from easygui import *
2  title = "Using EasyGui"
3
4  msg = "WinHtut-Member@GreenHackers , Founder@NationalCyberCity"
5
6  button = "ClickMe"
7  msgbox(msg, title, button )
8
```

Line 3 imageio သည် cross-platform library တစ်ခုဖြစ်ပြီး image , video , volumetric တွေရဲ့ data တွေကို ဖတ်တဲ့ နေရာမှာ အသုံးပြုပါတယ်။ imageio က python ရဲ့ built-in library မဟုတ်တဲ့အတွက် အသုံးပြုမည် ဆိုလျှင် Install လုပ်ပေးရန် လိုအပ်ပါသည်။ pip install imageio imageio ကို သုံးပြီး image တစ်ခုရဲ့ data တွေကို ဖတ်တဲ့ပုံကိုတော့ အောက်မှာဖော်ပြထားပါတယ်။ သူ့ရဲ့ syntax ကတော့ `imageio.imread(uri, format=None, **kwargs)` ဖြစ်ပြီး uri နေရာမှာ image file name သို့မဟုတ် http address သို့မဟုတ် file object တွေကိုပါ ထည့်ပေးနိုင်ပါတယ်။ ဒုတိယ တစ်ခုကတော့ format ပါ၊ file ကို ဖတ်တဲ့ နေရာမှာ အသုံးပြုဖို့ဖြစ်ပါတယ်။ ဒုတိယ နဲ့ တတိယက optional ပါ မထည့်လဲ ရပါတယ်။

```
1  import imageio
2
3  image = imageio.imread('myphoto.jpg')
4  print(image.shape)
5  #output (2048, 1536, 3)
```

2048 pixels သည် image ရဲ့ width ဖြစ်ပြီး 1536 pixels သည် image ရဲ့ height ဖြစ်ပါတယ်။ အကယ်၍ movie တွေကနေ တစ်ဆင့် ဖတ်ချင်ရင်တော့ get_reader နဲ့ ဖတ်ပါတယ်။

Line 6 sys module ကတော့ system တွေရဲ့ Information အချက်လက်တွေကို ဖော်ပြပေးဖို့ သုံးပါတယ်။ ဥပမာ မိမိတို့ system မှာ ဘယ် platform ကို သုံးထားလဲ သိချင်ရင် sys.platform ဟု ရေးလိုက်ရုံဖြင့် သိနိုင်ပါတယ်။ ထိုနည်းတူ မိမိတို့ system ရဲ့ memory store လုပ်ပုံမှာ little endian or big endian လား သိချင်ပါက sys.buteorder ဟု ရေးလိုက်ရုံဖြင့် သိနိုင်ပါတယ်။ ထို့ပြင် python interpreter ရဲ့ constants, functions and methods တွေရဲ့ information တွေကိုလည်း သိနိုင်ပါတယ်။ နောက်ပြီး မိမိတို့ယခု လက်ရှိ ရေးနေတဲ့ project မှာလည်း path ကို သိလိုပါက sys.path ဟုရေးလိုက်ရုံဖြင့် သိနိုင်ပါတယ်..etc။

```
>>> import sys
>>> sys.platform
'win32'
>>> sys.byteorder
'little'
>>> sys.path
['', 'C:\\Users\\winht\\AppData\\Local\\Programs\\Python\\Python38-32\\python38.zip', 'C:\\Us
\\AppData\\Local\\Programs\\Python\\Python38-32\\DLLs', 'C:\\Users\\winht\\AppData\\Local\\Pr
thon\\Python38-32\\lib', 'C:\\Users\\winht\\AppData\\Local\\Programs\\Python\\Python38-32',
```

Line 8 os module ကတော့ operating system နဲ့ သက်ဆိုင်တဲ့ လုပ်ဆောင်ချက်တွေကို လုပ်ဆောင်နိုင်ဖို့ သုံးပါတယ်။ ဥပမာ directory တစ်ခုကို ဖန်တီးလိုသည် ဆိုပါစို့။ os.mkdir() ဆိုတဲ့ function ကို ခေါ်သုံးနိုင်ပြီး ထို function ထဲတွင် မိမိတည်ဆောက်လိုတဲ့ နေရာကို parameter အနေဖြင့် ထည့်ပေးမည်ဆိုလျှင် directory တစ်ခုကို အလိုလျောက် တည်ဆောက်ပေးသွားမှာ ဖြစ်ပါတယ်။

```
>>> import os
>>> os.mkdir("c:\\myFolder")
>>> 
```

ယခု အတိုင်း ရေးလိုက်မည် ဆိုလျှင် c directory အောက်မှာ myFolder ဆိုသည့် folder တစ်ခုကို တည်ဆောက်ပေးမှာ ဖြစ်ပါတယ်။ နောက်ဥပမာ တစ်ခုအနေဖြင့် terminal မှာ ယခု လက်ရှိ မိမိရောက်နေတဲ့ နေရာ သိလိုလျှင် os.getcwd() ဟု ရေးပြီး သိနိုင်ပါသေးသည်။ cwd ဆိုတာ current working directory ကို ဆိုလိုခြင်း ဖြစ်ပါတယ်။ ထို့ပြင် တစ်နေရာမှ တစ်နေရာကို ချိန်းလို ပါကလည်း os.chdir() ဟု ရေးနိုင်ပါသေးတယ်။ chdir ဆိုတာ change directory ကို ဆိုလိုခြင်း ဖြစ်ပါတယ်။ အောက်မှာ getcwd နှင့် chdir function တို့ကို စမ်းပြထားပါတယ်။

```
>>> import os
>>> os.getcwd()
'C:\\Users\\winht\\PycharmProjects\\Cartoonifier\\venv\\testCode'
>>> os.chdir("testfolder")
>>> os.getcwd()
'C:\\Users\\winht\\PycharmProjects\\Cartoonifier\\venv\\testCode\\testfolder'
>>>
```

Line 9 tkinter ကတော့ ယခု လက်ရှိမှာ PYTHON Programming ရဲ့ Graphical User Interface နှင့် ပတ်သက်လာလျှင် အသုံးများဆုံး graphical library တစ်ခု ဖြစ်ပြီး c/c++ ဖြင့် ရေးသားထားပါတယ်။ GUI elements တွေနဲ့ function တွေကို အသုံးပြုဖို့ ဆိုလျှင် Tkinter module ကို import လုပ်ပေးရပါတယ်။ အောက်မှာတော့ sample program တစ်ပုဒ်ကိုရေးပြ ထားပါတယ်။ pycharm မှာ Tkinter ကို Install လုပ်ပေးစရာ မလိုပါဘူး pycharm ကို Installation လုပ်ကတည်းက ပါပြီးသား ဖြစ်ပါတယ်။

```
1  from tkinter import *
2  gui =Tk()
3
4  gui.title("Hello GUI")
5  gui.geometry("300x200")
6  gui.mainloop()
7
```

အထက်ပါ program ရဲ့ line 2 မှာတော့ Tk() function ကို ခေါ်ပြီး gui ဆိုတဲ့ object တစ်ခု တည်ဆောက်ပါတယ်။ ပြီးလျှင် gui object ထဲမှ title ကိုသုံးပြီး ပေါ်လာမယ့် window form ရဲ့ ထပ်ဆုံးမှာ ပေါ်စေချင်တဲ့ စာသားကို ထည့်ပါတယ်။ ထို့နောက် geometry ကိုသုံးပြီး window form ရဲ့ size ကို သတ်မှတ် ပေးပါတယ်။ မိမိတို့ create လုပ်လုံတဲ့ form ကို မပိတ်မချင်း ပေါ်စေချင်တယ် ဆိုရင်တော့ နောက်ဆုံးမှာ mainloop() method ကို အသုံးပြုဖို့ လိုအပ်ပါတယ်။ ထို့ပြင် Tkinter ကိုသုံးပြီး buttons များ image , background, text စသည်တို့ကိုလည် ထည့်နိုင်ပါသေးတယ်။ geometry method သည် window form ရဲ့ size ကိုသာမက program run လိုက်ချိန် တွင် ပေါ်လာမည် window form ရဲ့ Location ကို ပါ ထည့်ပေးနိုင်ပါတယ်။

```
1  from tkinter import *
2  gui =Tk()
3
4  btn=Button(gui,text="ClickMe",fg="green")
5  btn.place(x=80,y=100)
6  gui.title("Hello GUI")
7  gui.geometry("300x200+10+20")
8  gui.mainloop()
9
```

အထက်ပါ program တွင် line 4 ၌ Button function ကိုသုံးပြီး button တစ်ခု ဖန်တီးထားပါသည်။ ထို Button function ထဲတွင် ပထမဆုံး parameter အနေဖြင့် gui object ကိုထ ည့်ပေးရပါမည်။ ထို့ပြင် button ပေါ်၌ ပေါ်စေချင်သောစာသား နှင့် အခြား parameter များကိုလည်း ထည့်နိုင်ပါသေးသည်။ line 5 ကတော့ gui object ကနေတစ်ဆင့် ဖန်တီးထားတဲ့ window form ထဲတွင် button ပေါ်စေချင်သည့် နေရာကို ထည့်ပေးခြင်း ဖြစ်ပါသည်။ line 7 မှာတော့ window form ပေါ်လာစေချင်သည့် နေရာကို paramter များ ထည့်ပေးထားပါတယ်။ 10 သည် x position အတွက်ဖြစ်ပြီး 20 သည် y position အတွက် ဖြစ်ပါတယ်။

Main program line 12 PIL ကတော့ Python Image Library ဖြစ်ပါတယ်။ အမျိုးမျိုးသော Image file format တွေကို ဖွင့်တာတွေ အစားထိုးတာတွေ လုပ်ဖို့ သုံးပါတယ်။ ဥပမာ မိမိတို့ ပုံ တစ်ပုံကို blur ဝါးသွား စေချင်တယ်ဆိုပါစို့။ အောက်ပါ အတိုင်း ရေးသားနိုင်ပါတယ်။ မိမိတို့အ သုံးပြုချင်တဲ့ ပုံကိုတော့ program file ရေးတဲ့ နေရာနဲ့ same folder ဖြစ်ရပါမည်။ program အသေးစိတ်ကိုတော့ main program မှာ ပြန်လည် ရှင်းပြပါမည်။

```
1  from PIL import Image,ImageFilter
2
3  o_image = Image.open("myphoto.jpg")
4  blurred_image = o_image.filter(ImageFilter.GaussianBlur(radius=5))
5
6  o_image.show()
7  blurred_image.show()
```

ပထမဆုံးအနေဖြင့် မိမိတို့ main program အတွက် window form တစ်ခု တည်ဆောက်ပါမည်။ program မှာ အောက်ပါ အတိုင်း ဖြစ်သည်။

```
14    gui=tk.Tk()
15    gui.title('Transform Your Image !')
16    gui.configure(background='#218384')
17    gui.geometry('450x500+20+20')
18    label=Label(gui,background='#00FFFF')
19
```

နောက်တစ်ဆင့် အနေဖြင့် window ၌ button တစ်ခု တည်ဆောက်ပါမည်။ ထို button တွင် button ကို နှိပ်လိုက်ချိန်၌ function တစ်ခုကို ပါ တစ်ပြိုင်တည်း သွားခေါ်ပေးပါမည်။

```
upload=Button(gui,text="Transfomr Your Image",command=getImage,padx=10,pady=5)
upload.configure(background='#612184', foreground='#F9F60A',font=('calibri',10,'bold'))
upload.pack(side=TOP,pady=200)

gui.mainloop()
```

အခေါ်ခံရမည့် function ကို command ဆိုသည့် attribute ဖြင့် ရေးထားပါသည်။ command နောက်မှ getImage သည် function name ဖြစ်ပါသည် padx နှင့် pady တို့သည် padding x and y တို့ဖြစ်ပါသည်။ getImage function သည် easygui ကို သုံးပြီး မိမိတို့ ပြောင်းလိုသည့် image file ရဲ့ path ကို ယူပေးမည် ဖြစ်ပါသည်။ ထိုကဲ့သို့ ရယူ လာသော image ရဲ့ page ကို image transform လုပ်မည့် function ဆီသို့ ပြန်ပေးလိုက်ပါမည်။

```
20    def getImage():
21        ImagePath=easygui.fileopenbox()
22        transForm(ImagePath)
23
```

```
25    def transForm(ImagePath):
26        originalmage = cv2.imread(ImagePath)
27        originalmage = cv2.cvtColor(originalmage, cv2.COLOR_BGR2RGB)
28        print(originalmage)
```

Line 26 transForm function မှာတော့ cv2 library ထဲမှ imread function ကို သုံးကာ image file ကို ရယူပါမည်။ image တွေကို computer ထဲမှာ သိမ်းတဲ့ ထို image ရဲ့ number တွေကိုပဲ သိမ်းတာပါ။ imread နဲ့ Image file တစ်ခုကို ဖတ်ပြီး print ထုတ်ကြည့်မည် ဆိုလျှင် Image ကို မြင်ရမည် မဟုတ်ပဲ number တွေကိုပဲ မြင်ရမှာပါ။ imread function ကို သုံးပြီး ဖတ်လို့ရသည့် image သည် BGR (blue,green,red) ဆိုသည့် ပုံစံဖြင့် လာပါတယ်။ သို့သော် python image library ဖြစ်တဲ့ PIL(Pillow) နှင့် OpenCV မှာ သုံးသည့် function များသည် RGB(red,green,blue) ပုံစံ ဖြင့် အလုပ်လုပ်ပါတယ်။ ထို့ကြောင့် OpenCV function များကို အသုံးပြုမည် ဆိုလျှင် BGR မှ တစ်ဆင့် RGB သို့ပြောင်းလဲပေးရန်လိုအပ်ပါသည်။ OpenCV တွင် cvtColor ဆိုသည့် function ကို သုံးပါသည်။

```
None
[[[ 64  81 107]
  [ 65  82 108]
  [ 65  82 108]
  ...
  [163 157 152]
  [169 163 158]
  [181 175 170]]
```

```
[[[107  81  64]
  [108  82  65]
  [108  82  65]
  ...
  [152 157 163]
  [158 163 169]
  [170 175 181]]
```

အထက်ပါ ပုံ နှစ်ခုမှ ပထမ ပုံသည် imread function မှ ဖတ်လို့ ရလာသည့် BGR valuesများ ဖြစ်ပြီး ဒုတိယ ပုံသည် openCV မှ ရရှိလာသော RGB values များ ဖြစ်ပါသည်။ RGB values တွေကို မြင်ရပြီဆိုတော့ ဘယ်ဟာက r ဘယ်ဟာက g ဘယ်ဟာက b လဲဆိုတာကို ခွဲခြားတတ်ဖို့ လိုပါသေးတယ်။ ထိုသို့ မခွဲခြားခင် မိမိတို့ ရရှိလာတဲ့ color values array သည် ဘယ်လောက် dimension array လဲဆိုတာကို ဦးစာ သိဖို့ လိုပါသေးတယ်။ ထိုသို့သိနိုင်ရန် Numpy သင်ခန်းစာမှာ သင်ကြားခဲ့ပြီးဖြစ်သည့် ndim ကို အောက်ပါ အတိုင်း သုံးလိုက်မည် ဆိုလျှင် dimension ကို သိရမည် ဖြစ်သည်။

```
25  def transForm(ImagePath):
26      originalmage = cv2.imread(ImagePath)
27      originalmage = cv2.cvtColor(originalmage, cv2.COLOR_BGR2RGB)
28      print(originalmage)
29      print(originalmage.ndim)
```

ယခုအတိုင်း ရေးလိုက်မည် ဆိုလျှင် output အနေဖြင့် 3 ကို ရရှိပါသည်။ ထို values များအား rgb ခွဲခြားနိုင်စေရန် အောက်ပါပုံအား ကြည့်ခြင်းဖြင့် သိနိုင်ပါသည်။

နောက်တစ်ဆင့် အနေဖြင့် ဖတ်လိုက်သော file အား image file ဟုတ်မဟုတ်ကို စစ်ဆေးပါမည်။ အကယ်၍ image file မဟုတ်ဘူး ဆိုလျှင် image file မဟုတ်ကြောင်းပြောပြီး program ကို ရပ်လိုက်ပါမည်။

```
def transForm(ImagePath):
    originalmage = cv2.imread(ImagePath)
    originalmage = cv2.cvtColor(originalmage, cv2.COLOR_BGR2RGB)
    print(originalmage)
    print(originalmage.ndim)

    if originalmage is None:
        print("It is not a valid image File .")
        sys.exit()
```

အထက်ပါ အတိုင်း Image ကို စစ်ဆေး ပြီးပါက image ဖြစ်လျှင် original image ကို resized လုပ်ပါမည်။ ယခု အဆင့်သည် မူရင်းပုံနှင့် Output ပုံကို တိုက်ကြည့်ရန် အတွက် ဖြစ်ပါသည်။ နောက်ထပ် 2 nd ပုံကတော့ original image ကို gray scale ပြောင်းမှာ ဖြစ်ပါတယ်။ GrayScale ပြောင်းရန် အတွက်လည်း openCV တွင် cvtColor ကိုပဲ ပြန်သုံး နိုင်ပါသည်။

```
    if originalmage is None:
        print("It is not a valid image File .")
        sys.exit()
    #For 1 image
    image1 = cv2.resize(originalmage, (900, 450))

    #For 2 image
    grayScale= cv2.cvtColor(originalmage, cv2.COLOR_BGR2GRAY)
    image2 = cv2.resize(grayScale, (900, 450))
```

Why Gray?

ဘာကြောင့် grayscale ကိုချိန်းရလဲ ဆိုလျှင် RGB သည် red ,green and blue တို့ ပေါင်းထားခြင်း ဖြစ်ပြီး rgb value တစ်ခုနှင့် တစ်ခုမှာ မူ မတူညီကြပါ။ သို့သော် gray တွင် r g b vlaues အားလုံးသည် အတူတူပင်ဖြစ်ပါသည်။ထို့ပြင် rgb သည် 24 bits ရှိပြီး r=8bit , g=8bit and b=8bit တို့ဖြစ်ပြီး gray သည် 8 bit သာ ဖြစ်သည် ထို့ကြောင့် grayscale image တွင် 8bit ဖြစ်သည့် အတွက် values သည် အမဲ မှ အဖြူထိ color value မှာ 0 to 256 ထိသာ ရှိပါသည် ။ image များသည် pixel များဖြင့်သာ ဖော်ပြခြင်း ဖြစ်ပြီး gray scale image အတွက်တော့ single byte တစ်ခုတည်းနှင့်သာ ဖော်ပြနိုင်ပါသည်။ ထို့ပြင် image processing လုပ်ဆောင်သည့် processing တော်တော်များများတွင် grayscale ဖြင့် အဆင်ပြေသည့်အတွက် ပိုမိုရှုပ်ထွေးသည့် rgb color images သည် processing လုပ်ဆောင်ရန် ပိုခက်ခဲပါသည်။

MedianBlur (Median Filtering)

MedianBlur ကို denoising လုပ်တယ်လို့လည်း ခေါ်ပါတယ်။ ဆိုလိုချင်တာက image ထဲမှာ ထင်ရှားနေတဲ့ နေရာတွေကို ဖျောက်လိုက်ခြင်း ဖြစ်ပါတယ်။ ဥပမာ မိမိတို့ ဖုန်း camera ရဲ့ ရှိုက်လိုက်တဲ့ ပုံတွေမှာ ကိုယ်မလိုချင်တဲ့ နေရာတွေကို ဝါးလိုက်ကြပါတယ်။ အဲဒါကို blur လုပ်တယ်လို့ ခေါ်ပါတယ်။ Blur လုပ်ခြင်းဟာ image pre-processing step ထဲမှာ ပါဝင်သလို machine learning and deep learning models တွေမှာလည်း အသုံးများပါတယ်။ ပထမဆုံး အနေဖြင့် Median အကြောင်းကို အရင်ဆုံးရှင်းပါမည်။ median ဆိုတာ ကိန်းဂဏန်းတွေကို ငယ် စဉ် ကြီးလိုက် စီပြီး အလယ်က နံပါတ်ကို ယူတာပါ။ အကယ်၍ စုံ ဂဏန်း ဖြစ်နေပြီး အလယ်မှာ number နှစ်ခု ဖြစ်နေပါက ထို number နှစ်ခုကို ပေါင်းပြီး 2 နဲ့ စားပေးပါတယ်။ ရလာတဲ့ value ကို median အဖြစ် ယူတာပါ။ ဥပမာ 9,8,10,5,6,3,2,4,1,7 ယခု ဖော်ပြထားသည့် number ၁၀ ခုရဲ့ median ကို လိုချင်ရင် အရင်ဆုံး ငယ်စဉ် ကြီးလိုက် စီလိုက်ပါက number အရေအတွက်သည် ဆုံဂဏန်း ဖြစ်နေသည့် အတွက် အလယ်တွင် 5,6 နှစ်ခု ရှိနေပါမည်။ ထို နှစ်ခုအား ပေါင်းပြီး ရလာတဲ့ တန်ဘိုးကို ၂ နဲ့ စားခြင်း ဖြင့် median value ကို ရပါပြီ။ အကယ်၍ number အရေ အတွက်သည် မ ဂဏန်း ဖြစ်နေမည် ဆိုလျှင်တော့ ငယ်စဉ် ကြီးလိုက် စီကာ အလယ်ဆုံး number သည် median ဖြစ်ပါသည်။ Median value ကို Python ဖြင့် ရေးနိုင်ရန် statistics module ကို Import လုပ်ပြီး ထို statistics ထဲကမှ median function ကို သုံးနိုင်ပါသည်။ sample program ကို အောက်မှာ ဖော်ပြ ထားပါသည်။ ပထမဆုံးဦးစွာ statistics module ကို install လုပ်ပေးရန် လိုအပ်ပါသည်။

```
Type "help", "copyright", "credits" or "license" for more information.
>>> import statistics
>>> imagePixels=[10,20,30,70,90,45,100,38]
>>> print(statistics.median(imagePixels))
41.5
```

Median ကို သိပြီဆိုတော့ medianblur ကို ဆက်သွားကြပါစို့။ *medianblur ဆိုတာဟာ kernel size အပေါ်မှာ မူတည်ပြီး ရရှိလာတဲ့ pixels တွေထဲကမှ central pixel value ကို medain value ဖြင့် အစားထိုးလိုက်ခြင်း ဖြစ်ပါတယ်။* ဒီနည်းလမ်းလို သုံးခြင်းအားဖြင့် image ပေါ်မှ ထင်ရှားနေတဲ့ အမှုန်တွေ အစက်ပျောက် တွေကို မှုန်ဝါးသွားစေပါတယ်။

ယခု ပုံမှာဆိုလျှင် ညာဘက်ကပုံသည် အနည်းငယ် ဝါးနေပြီး မျက်နှာပေါ်မှာ အစက်ပျောက်တွေ အနဲငယ် ပျောက်သွားသည်ကို မြင်ရမှာပါ။ စာရေးသူသည် ယခု program တွင် kernel size ကို 5 သုံးထားပါသည်။ kernel size သည် positive odd(မ) integer ဖြစ်ရပါမည်။ main program တွင် medianBlur အတွက်ရေးသည့် sample ကို အောက်တွင်ကြည့်ပါ။

```
40
41      #For blur image 3
42      smoothGray = cv2.medianBlur(grayScale, 3)
43      image3 = cv2.resize(smoothGray, (900, 450))
44
```

အထက်ပါ Program တွင် kernel size ကို 3 ဟု သုံးထားပါသည်။ medianBlur function သည် median filter ဆိုသည့် method ကို သုံးပါသည် median filter သည် kernalSize x kernalSize ဖြင့် pixels များကို ချိန်းရာတွင် အသုံးပြုပါသည်။ အောက်တွင် အဖြူရောင်နှင့် ပုံကို image ရဲ့ pixels များဟု မှတ်ယူပြီး အပြာရောင်အား Median mask or kernel ဟု မှတ်ယူပါ။

5	6	10	11
3	4	7	9
7	6	5	2
10	5	12	8

Image Pixel

Median mask

ပထမဆုံး အနေဖြင့် pixels များအား replication နှင့် Zero Padding ပြုလုပ်ပါမည်။

Image Pixels

5	6	10	11
3	4	7	9
7	6	5	2
10	5	12	8

After Pixel Replication of Image

5	5	6	10	11	9
5	5	6	10	11	
3	3	4	7	9	9
7	7	6	5	2	2
10	10	5	12	8	8
10	10	5	12	8	8

After Zero Padding of Image

0	0	0	0	0	0
0	5	6	10	11	0
0	3	4	7	9	0
0	7	6	5	2	0
0	10	5	12	8	0
0	0	0	0	0	0

Pixel Replication ပြုလုပ်တယ်ဆိုတာ မူရင်း ပုံရဲ့ pixel တွေကို left and right အရင်ဆုံး ပွားလိုက်တာပါ ပြီးမှ up and down ကို ပွားတာပါ ထို့ကြောင့် မူရင်း image pixel နှင့် မတူတော့ကြောင်းကို After Pixel Replication ဆိုတဲ့ ပုံမှာ ကြည့်ပါ။ နောက်တစ်ပုံကတော့ padding zero လုပ်တာပါ ပတ်ပတ်လည် အားလုံးကို zero တွေ ဖြည့်လိုက်တာပါ။ နောက်တစ်ဆင့် အနေဖြင့် Median Filtering Mask ကို သုံးပြီး pixel ကို စတင်ပြောင်းလဲကြည့်ပါမည်။ အပြာရောင်နှင့် ပြထားသောပုံသည့် median filter mask အုပ်ထားသော ပုံဖြစ်သည်။ ညာဘက်အစွန်ဆုံး column မှ ဒုတိယ value နေရာတွင် လွတ်နေပါသည် ထိုနေရာအား 11 ဟု ရှိသည်ဟု မှတ်ထားပေးပါ။

အထက်ပါအတိုင်း Median Filter ကို အုပ်ကြည့်ပါက အောက်ပါ ပုံတွင်ပြထားသည့် pixels များကို ရရှိပါမည်။ ထို pixels များထဲမှ median ကို ရှာပြီး ရလာတဲ့ median value ကို အလယ် တည့်တည့်မှာ ထည့်လိုက်ပါမှာ ဖြစ်ပါတယ်။ အောက်ပါ ပုံအရ pixels များထဲမှ median value သည် 5 ဖြစ်ပါမည်။

5	5	6
5	5	6
3	3	4

အလယ်မှ median value သည်လည်း 5 ဖြစ်နေသည့် အတွက် ဘာမှ ပြောင်းလဲ သွားမည် မဟုတ်ပါ။နောက်ထပ် median filter mask ထပ်အုပ်ကြည့်ပါမည်။ အောက်ပါ ပုံအတိုင်း မြင်ရပါမည်။

5	5	6	10	11	9
5	5	6	10	11	
3	3	4	7	9	9
7	7	6	5	2	2
10	10	5	12	8	8
10	10	5	12	8	8

အထက်ပါ ပုံတွင် အပြာရောင် pixel value များထဲမှ median ကို ရှာကြည့်လျှင် 6 ကို ရရှိပါမည်။ pixel များထဲမှ အလယ် value သည်လည်း 6 ဖြစ်နေသည့်အတွက် ပြောင်းလဲ စရာမလိုပါ။

5	5	6	10	11	9
5	5	6	10	11	
3	3	4	7	9	9
7	7	6	5	2	2
10	10	5	12	8	8
10	10	5	12	8	8

အထက်ပါ pixel များထဲမှ median value ကို ရှာကြည့်လျှင် 9 ကို ရရှိပါမည်။ အလယ် value သည် 10 ဖြစ်နေသည့်အတွက် ထို 10 နေရာတွင် 9 ကို အစားထိုးပါမည်။ Ouput အနေဖြင့် အောက်ပါ အတိုင်း ရရှိပါမည်။

5	5	6	10	11	9
5	5	6	9	11	
3	3	4	7	9	9
7	7	6	5	2	2
10	10	5	12	8	8
10	10	5	12	8	8

နောက်ထပ် အစွန်းဆုံးအပိုင်းအား median mask ထပ်အုပ်ကြည့်ကြပါစို့။

5	5	6	10	11	9
5	5	6	9	11	11
3	3	4	7	9	9
7	7	6	5	2	2
10	10	5	12	8	8
10	10	5	12	8	8

အထက်ပါ ပုံထဲမှ pixel value များရဲ့ median value မှာ 9 ဖြစ်သည်။ ထို့ကြောင့် 11 နေရာအား 9 ဖြင့် ချိန်းရမည်ဖြစ်ပြီး output မှာ အောက်ပါ အတိုင်း ဖြစ်သည်။

5	5	6	10	11	9
5	5	6	9	9	11
3	3	4	7	9	9
7	7	6	5	2	2
10	10	5	12	8	8
10	10	5	12	8	8

Columns တွေ အားလုံးပြီးလျှင် rows များကိုလည်း ထပ်အုပ်ဖို့ လိုပါသေးသည်။

5	5	6	10	11	9
5	5	6	9	9	11
3	3	4	7	9	9
7	7	6	5	2	2
10	10	5	12	8	8
10	10	5	12	8	8

အထက်ပါပုံတွင် median value သည် 5 ဖြစ်နေသည့်အတွက် အလယ်မှ 3 အား 5 ဖြစ်ချိန်းရမည် ဖြစ်သည်။ ထို့ကြောင့် Output အနေဖြင့် အောက်ပါအတိုင်း ရရှိပါမည်။

5	5	6	10	11	9
5	5	6	9	9	11
3	5	4	7	9	9
7	7	6	5	2	2
10	10	5	12	8	8
10	10	5	12	8	8

MedianFiltering လုပ်ခြင်း မှာ အထက်ပါပုံများ အတိုင်း အဆင့်ဆင့် နောက်ဆုံး row and column ထိ လုပ်ဆောင်သွားရခြင်းဖြစ်ပြီး အဆင့်များ အားလုံးပြီးသွားပါက Image တွင် အစက်အပျောက်များ လျော့နဲ့သွားမည် ဖြစ်ပါသည်။ အကယ်၍ zero padding ကို အသုံးပြုမည် ဆိုလျှင်လည်း ယခု နည်းလမ်း အတိုင်းများပင် လုပ်ဆောင်သွားရမည် ဖြစ်ပါသည်။

0	0	0	0	0	0
0	0	6	10	11	0
0	3	4	7	9	0
0	7	6	5	2	0
0	10	5	12	8	0
0	0	0	0	0	0

Simple Thresholding

Adaptive Thresholding အကြောင်းကို သိဖို့ဆိုလျှင် Thresholding အကြောင်းကို ဦးစွာသိရပါမည်။ Threasholding ကို မဖော်ပြခင် s=(l-1) -r ဆိုသည့် equation နှင့် အတူ image negative အကြောင်းကို ဖော်ပြလိုပါသည်။ Image Negative သည် image တစ်ခုရဲ့ pixel ကနေ တစ်ဆင့် ထို pixel မှ bit အပေါ်မူတည်ပြီး s = (l - 1) - r equation ကို သုံးကာ တွက်ထုတ်ရခြင်း ဖြစ်သည်။ စာရေးသူတို့အနေဖြင့် ယခု သင်ခန်းစာတွင် pixel ကို 3 bit အနေဖြင့်သာ ထားပြီး တွက်ပြပါမည်။ အကယ်၍ 8 bit ဆိုလျှင်လည်း 3 bit နေရာတွင် 8 bit ကို အစားထိုးပြီး တွက်နိုင်ပါသည်။

1	4	3	7
2	1	6	3
7	3	1	2
5	6	7	1

အထက်ပါ ပုံသည် စာရေးသူတို့ရဲ့ image 3bit pixel ပုံ တစ်ပုံ ဖြစ်သည်။

s= (l -1) - r

s= output image

l=number of bit

r= original image pixel

အထက်ပါ equation တွင် s သည် processing လုပ်ပြီး ထွက်လာမည့် image ပုံ ဖြစ်သည်။ l သည် number of bit ဖြစ်ပြီး စာရေးသူတို့ အနေဖြင့် ယခု ပုံတွင် 3 bit ဖြစ်သည့် အတွက် l =2^3 = 8 ဖြစ်သည်။ r သည် မူရင်း Input image ဖြစ်သည်။

S = (8 -1) - 1 (ပထဆုံး ဖြစ်တဲ့ 1 ဆိုသည့် pixel အတွက် တွက်ခြင်း ဖြစ်သည်)

S = 6

အထက်တွင် ရလာသော s = 6 သည် ပထမဆုံး Pixel အတွက် တွက်ကြည့်ခြင်း ဖြစ်ပြီး ထိုနည်းအတိုင်း pixel 16 ခုလုံးကို တွက်ကြည့်ရမည် ဖြစ်သည်။ ထိုသို့ တွက်ကြည့်ပြီးနောက်တွင် Pixel တစ်ခုစီ တိုင်း အတွက် pixel အသစ်များရရှိမည်ဖြစ်သည်။ထို့ကြောင့် Image Negative အနေဖြင့် အောက်ပါ ပုံကို ရရှိပါမည်။ ယခု ဆက်လက်ပြီး Thresholding ကို လေ့လာ ကြပါစို့ ။ Thresholding သည် gray skill image or color image ကနေ binary image အဖြစ် ဖန်တီးခြင်း ဖြစ်ပါတယ်။ binary image ဆိုတာ1 or 0 image ဖြစ်ပါတယ်။ Thresholding ကို image processing မှာ foreground and background ခွဲခြားဖို့ သုံးပါတယ်။ Thresholding လုပ်ဖို့ ဆိုလျှင် pixel များရဲ့ bit အပေါ် မူတည်ပြီး ယူရပါမယ်။ ဥပမာ အထက်ပါ Image Negative သင်ခန်းစာတွင် 3bit ဖြစ်တဲ့ အတွက် 8 ဖြစ်ပါတယ် (l -1) ဖြစ်တဲ့ အတွက် 7 ပေါ့ ထိုကဲ့သို့ ရလာသော 7 အား တစ်ဝက် ဝက်လျှင် 0 to 3 က တစ်ခုနှင့် 4 to 7 က တစ်ခု ဖြစ်မည်။ 0 to 3 အား zero အဖြစ်သတ်မှတ်ပြီး 4 to 7 အား 1 အဖြစ် သတ်မှတ်ခြင်း ဖြစ်ပါတယ်။ ထို့ကြောင့် 0 to 3 အပိုင်း သည် zero အဖြစ် မှတ်ယူသည့် အတွက် black color ဖြစ်ပြီး white color ဖြစ်ပါသည် ထို့ကြောင့် thresholding ဖြင့် black and white ခွဲခြား နိုင်ပါသည်။ အထက်ပါ တွက်ချင်ခြင်းများသည် 3 bit အတွက် ဖြစ်ပြီး ယခု ဆက်လက်ပြီး 8 bit အတွက် ဆိုလျှင်

8 bit = 2^8 = 256 = 122 <= 123 ထို့ကြောင့် 122 အပိုင်းအား 0 အဖြစ် သတ်မှတ်ပြီး black color အဖြစ်ယူမည် 123 အပိုင်းအား white color အဖြစ် ယူပါမည်။ ထိုနည်းအားဖြင့် black and white ခွဲခြား နိုင်ပါသည် သို့သော် opencv ရဲ့ offcial documentation တွင်မူ 8 bit အတွက်ကို 127 ဟု Threshold value ကို သတ်မှတ်ထားသော်လည်း မိမိ တို့စိတ်ကြိုက် ပြုပြင်နိုင်သည်။ အထက်တွင် ဖော်ပြခဲ့သော နည်းလမ်းသည် thresholding နည်လမ်းများစွာထဲကမှ Binary Thresholding ဖြစ်သည်။ ထို့ပြင် Binary Inverted , Truncate , ToZero , ToZero Inverted စသည့် Thresholding များလည်း ရှိနေပါသေးသည်။ Opencv documentation မှာတော့ အောက်ပါအတိုင်း ဖော်ပြထားပါတယ်။

Enumerator	
THRESH_BINARY Python: cv.THRESH_BINARY	$\texttt{dst}(x,y) = \begin{cases} \texttt{maxval} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$
THRESH_BINARY_INV Python: cv.THRESH_BINARY_INV	$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{maxval} & \text{otherwise} \end{cases}$
THRESH_TRUNC Python: cv.THRESH_TRUNC	$\texttt{dst}(x,y) = \begin{cases} \texttt{threshold} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$
THRESH_TOZERO Python: cv.THRESH_TOZERO	$\texttt{dst}(x,y) = \begin{cases} \texttt{src}(x,y) & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$
THRESH_TOZERO_INV Python: cv.THRESH_TOZERO_INV	$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$

Binary Inverted Thresholding

Binary Inverted ကတော့ Binary thresholdingကို ပြောင်းပြန် လုပ်ထားတာပါ။ binary threshold မှာ threshold value ထက်ကြီးတာကို 1 အဖြစ်ယူပြီး ငယ်တာကို 0 အဖြစ် ယူပါတယ်။ Binary inverted မှာတော့ threshold value ထက် input value က ကြီးနေမည် ဆိုလျှင် 0 ဟု ယူပါသည်။ဥပမာ အားဖြင့် input value သည် 5 ဖြစ်နေလျှင် threshold value သည် 3bit အားဖြင့် 4 ဖြစ်ပါက output value အနေဖြင့် 0 ကိုသာ ရရှိပါမည်။

If (inputValue(x,y) > Threshold value) :

output=0

Else:

output= maxvalue (ယခု သင်ခန်းစာမှာတော့ 7)

Truncate Thresholding

Truncate Thresholding တွင် Input value src(x,y) သည် threshold value ထက် ကြီးနေမည် ဆိုလျှင် output အနေဖြင့် threshold value ကို ရရှိမည် ဖြစ်ပြီး အကယ်၍ input value သည် threshold value ထက် ငယ်နေမည် ဆိုလျှင် output အနေဖြင့် input value အတိုင်းသာ ပြန်ရမည်။

If (inputValue(x,y) > threshold value) :

Output = threshold value

Else:

Output = input value(x,y)

ToZero Thresholding

ToZero Thresholding တွင် input value သည် Thresholding value ထက် ကြီးနေမည် ဆိုလျှင် output အနေဖြင့် input value အတိုင်းသာ ပြန်ရမည်။ အကယ်၍ input value သည် threshold value ထက် ငယ်နေခဲ့မည် ဆိုလျှင်တော့ output အနေဖြင့် zero ကိုသာ ရရှိပါမည်။

If (inputValue(x,y) > threshold value) :

Output = input value(x,y)

Else:

Output = zero

ToZero Invert Thresholding

ToZero Invert Thresholding တွင် ToZero နှင့်ပြောင်းပြန် ဖြစ်ပါသည်။ အကယ်၍ input value သည် threshold value ထက် ကြီးနေမည် ဆိုလျှင် output ကို zero အဖြစ် သတ်မှတ်ပြီး input value သည် threshold value ထက် ငယ်နေခဲ့မည် ဆိုလျှင် input value ကိုသာ ပြန်ရပါသည်။

If (inputValue(x,y) > threshold value) :

Output = input value(x,y)

Else:

Output = zero

Thresholding အားလုံးအတွက် program ကို အောက်တွင် ရေးပြထားပါသည်။ Theory အားလုံးကို ရှင်းပြပြီး ဖြစ်တာမို့ program ကို မိမိတို့ကိုယ်တိုင် နားလည်နိုင်ရန် ကြိုးစားသင့်ပါသည်။ အောက်ပါ program အတိုင်း ပုံ တစ်ပုံကို ထည့်ပြီး run ကြည့်လျှင် မတူညီသည့် thresholding များဖြင့် ပုံ ၅ ပုံကို ပြန်လည် ရရှိလာမှာ ဖြစ်ပါတယ်။

```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
img = cv.imread('myphoto.jpg',0)
ret,thresh1 = cv.threshold(img,127,255,cv.THRESH_BINARY)
ret,thresh2 = cv.threshold(img,127,255,cv.THRESH_BINARY_INV)
ret,thresh3 = cv.threshold(img,127,255,cv.THRESH_TRUNC)
ret,thresh4 = cv.threshold(img,127,255,cv.THRESH_TOZERO)
ret,thresh5 = cv.threshold(img,127,255,cv.THRESH_TOZERO_INV)
cv.imshow("BinaryThresholding",thresh1)
cv.imshow("BinaryInvertedThresholding",thresh2)
cv.imshow("TruncateThresholding",thresh3)
cv.imshow("ToZeroThresholding",thresh4)
cv.imshow("ZeroInvertedThresholding",thresh5)
cv.waitKey(0)
cv.destroyAllWindows()
```

Adaptive Thresholding

အထက်ပါ Thresholding သင်ခန်းစာမှာ threshold value ကို global value အနေနဲ့အသုံးပြုခဲ့ပါတယ်။ global value ဆိုတာက threshold value တစ်ခုတည်းကိုပဲ ပုံမှာ ရှိတဲ့ Pixel အားလုံးအတွက် အသုံးပြုခဲ့တာပါ။ ထိုကဲ့သို့ အသုံးပြုခြင်းက အမြဲတမ်းတော့ အဆင်မပြေနိုင်ပါဘူး ဘာကြောင့်လဲဆိုတော့ ပုံတစ်ပုံမှာ မတူညီတဲ့ lighting area တွေ ရှိပါတယ်။ ထို့ကြောင့် ပိုပြီး တိတိကျကျ သိချင်တဲ့ အခါမျိုးတွေမှာ adaptive thresholding ကို အသုံးပြု နိုင်ပါတယ်။ Adaptive algorithm ဟာ global အနေဖြင့် threshold value ကို ထုတ်မယူပဲ small region တစ်ခုအတွက် threshold value တစ်ခု ယူပါတယ်။ ထို့ကြောင့် ပုံတစ်ပုံမှာ မတူညီတဲ့ region ပေါ်မူတည်ပြီး မတူညီတဲ့ threshold value များစွာ ရရှိနိုင်ပါတယ်။ သို့ဖြစ်ပါ၍ ပိုမိုကောင်းမွန်သော result ကိုလည်း ရရှိနိုင်ပါတယ်။

ယခု ဆိုလျှင် Thresholding နှင့် adaptive thresholding များကို အတန်အသင့် နားလည်သွားပြီ ဖြစ်တဲ့အတွက် opencv library မှ adaptiveThresholding function အကြောင်းကို

ဆက်လက်ဖော်ပြလိုပါသည်။ အောက်ပါ program သည် စာရေးသူတို့ရဲ့ main program မှ adaptiveThresholding အတွက် code ဖြစ်သည်။

```
    getEdge = cv2.adaptiveThreshold(smoothGray, 255,
        cv2.ADAPTIVE_THRESH_MEAN_C,
        cv2.THRESH_BINARY, 9, 9)

```

အထက်ပါ program တွင် ADAPTVIE_THRES_MEAN_C သည် ပုံတစ်ပုံမှ သတ်မှတ်ထားသော Pixel area တစ်ခုရဲ့ mean တန်ဘိုး ဖြစ်ပါတယ်။ နောက်ဆုံး C ကတော့ constant value ကို ပြောတာပါ။ ပုံတိကျ အောင် ပြောရမည် ဆိုလျှင် MxM မှ mean value ရဲ့ minus constant ဖြစ်ပါတယ်။ သတ်မှတ်ထားသော Pixel area ဆိုတာက M x M နေရာကို ပြောတာပါ။ အထက်ပါ ပုံတွင် ဖော်ပြထားပြီး ဖြစ်ပါတယ်။ အထက်ပါ program မှာဆိုလျှင်တော့ 9 x 9 သည် Mx M ဖြစ်တယ်။ THRES_BINARY ကတော့ အသုံးပြုမည့် thresholding type ဖြစ်ပါတယ်။ Program ရဲ့ Output image နှင့် ယခု အဆင့်ထိ program အားလုံးကို အောက်မှာ ဖော်ပြထားပါတယ်။

```python
def getImage():
    ImagePath=easygui.fileopenbox()
    transForm(ImagePath)


def transForm(ImagePath):
    originalmage = cv2.imread(ImagePath)
    originalmage = cv2.cvtColor(originalmage, cv2.COLOR_BGR2RGB)
    print("original",originalmage)
    print(originalmage.ndim)

    if originalmage is None:
        print("It is not a valid image File .")
        sys.exit()
    #For 1 image
    image1 = cv2.resize(originalmage, (900, 450))

    #For 2 image
    grayScale= cv2.cvtColor(originalmage, cv2.COLOR_BGR2GRAY)
    image2 = cv2.resize(grayScale, (900, 450))

    #For blur image 3
    smoothGray = cv2.medianBlur(grayScale, 3)
    image3 = cv2.resize(smoothGray, (900, 450))

    getEdge = cv2.adaptiveThreshold(smoothGray, 255,
        cv2.ADAPTIVE_THRESH_MEAN_C,
        cv2.THRESH_BINARY, 9, 9)
    image4 = cv2.resize(getEdge, (960, 540))
    cv2.imshow("Adaptive Threshold",image4)

```

Gaussian Kernel and Filtering

ယခု သင်ခန်းစာမှာတော့ Gaussian Filter အကြောင်းကိုဖော်ပြသွားမှာ ဖြစ်ပြီး Gaussian Filter သည် image processing နယ်ပယ်တွင် image တွေကို blur လုပ်ဖို့နှင့် image မှာရှိတဲ့ noise and detail တွေကို လျော့ချဖို့ သုံးပါတယ်။ Gaussian Filter ကို အသုံးပြုမည်ဆိုလျှင် ဦးစွာ Gaussian kernel အကြောင်းကို သိရပါမည်။ Gaussian kernel အတွက် တွက်ချက်ရမည့် formula သည် အောက်ပါ အတိုင်း ဖြစ်ပါတယ်။

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Standard Deviation Sigma= 0.6 (weight)ဟု ထားပြီး တွက်ကြည့်ပါမည်။ ထို့ပြင် Kernel Size ကိုတော့ 3x3 ထားပါမည်။ ထို့ပြင် X , Y value များကို အောက်ပါ အတိုင်း ပေးထားပါမည်။ အထက်ပါ formula မှ e သည် euler number ဖြစ်သည်။ euler number

အားရှာသည့် formula သည် အောက်ပါ အတိုင်း ဖြစ်သည်။ e ရဲ့ constant သည် e = 2.71828 **ဖြစ်သည်။**

$$e = \sum_{n=0}^{\infty} \frac{1}{n!} = 1 + \frac{1}{1} + \frac{1}{1 \cdot 2} + \frac{1}{1 \cdot 2 \cdot 3} + \cdots$$

$$\frac{1}{2\pi\sigma^2} = \frac{1}{2 \times 3.14 \times 0.6 \times 0.6} = \frac{1}{2.2619}$$

$$\frac{-(x^2+y^2)}{2\sigma^2}$$

ယခု equation **ထဲတွင်** x , y and sigma **တန်ဘိုးများကို အစားသွင်းပါကအောက်ပါအတိုင်း တန်ဘိုးများ ပြန်လည် ရရှိပါမည်။** $x^2 + y^2$ **ကို** numpy **အားသုံးပြီး တွက်ချက်ပုံနှင့် ထိုနှစ်ခုအား ပေါင်းခြင်းကိုလည်း အောက်တွင် ဖော်ပြထားပါသည်။**

```
mx = [[-1,1,1],[-1,0,1],[-1,2,1]]
sx = np.square(mx)
my = [[-1,-2,-1],[0,0,0],[2,1,1]]
sy = np.square(my)
mz = np.add(sx,sy)
print(mz)
```

```
[[2 5 2]
 [1 0 1]
 [5 5 2]]
```

အထက်တွင် ရလာသော အဖြေအား 2sigma^2 **ဖြင့် စားရန် ကျန်ရှိနေပါသေးသည်။ စားလိုက်သော အခါ အဖြေသည် အောက်ပါ အတိုင်းရရှိပါသည်။** 2sigma^2 **ရဲ့** value **သည်** 2.2619 **ဖြစ်သည့် အတွက် အောက်ပါတိုင်း** Numpy **ဖြင့် စားလိုက်ပါသည်။**

```
mx = [[-1,1,1],[-1,0,1],[-1,2,1]]
sx = np.square(mx)
my = [[-1,-2,-1],[0,0,0],[2,1,1]]
sy = np.square(my)
mz = np.add(sx,sy)
dz = np.divide(-mz,2.2619)
print(dz)
```

```
[[-0.88421239 -2.21053097 -0.88421239]
 [-0.44210619  0.          -0.44210619]
 [-2.21053097 -2.21053097 -0.88421239]]
```

ထိုကဲ့သို့ ရလာသော matrix အား exponent ရှာပါမည်။ exponent ရှာပြီး ရရှိလာသော matrix အား 0.442106 ဖြင့် စားရပါမည်။ 0.442106 သည် 1/2PiSigma^2 အားရှင်းပြီး ရလာသော value ဖြစ်သည်။

$$\frac{1}{2\pi\sigma^2}$$

```
mx = [[-1,1,1],[-1,0,1],[-1,2,1]]
sx = np.square(mx)
my = [[-1,-2,-1],[0,0,0],[2,1,1]]
sy = np.square(my)
mz = np.add(sx,sy)
dz = np.divide(-mz,2.2619)
mExp = np.exp(dz)*0.442106
print(mExp)
```

```
[[0.18260718 0.04847357 0.18260718]
 [0.2841333  0.442106   0.2841333 ]
 [0.04847357 0.04847357 0.18260718]]
```

အထက်ပါ output များကို ကြည့်ပါက အလယ်မှ 0.442106 သည် အများဆုံး ဖြစ်နေပြီး အခြားသော pixel များရဲ့ intensity value သည် အလယ်မှ 0.442106 နှင့် ဝေးလာသည်နှင့် အမျှ လျော့ကျသွားသည်ကို တွေ့မြင်ရပါမည်။ နောက်တစ်ဆင့်အနေဖြင့် ပုံတစ်ပုံရဲ့ Pixel တန်ဘိုးများကို အောက်ပါ အတိုင်း ဥပမာထားပြီး တွက်ကြည့်ပါမည်။

```
Original Image Pixel
[[ 30  40 139]
 [ 70  79 130]
 [ 60  37 170]]
```

*

```
MyKernel
[[0.18260718 0.04847357 0.18260718]
 [0.2841333  0.442106   0.2841333 ]
 [0.04847357 0.04847357 0.18260718]]
```

=

```
New Pixel Multiply by myKernel
[[ 5.47821538  1.9389428  25.38239792]
 [19.88933069 34.926374   36.93732842]
 [ 2.90841421  1.79352209 31.04322048]]
```

⇩

160.297745

အထက်တွင် ရရှိလာသော 160.2977အား ယခုလက်ရှိ kernel အုပ်ထားသော pixel ရဲ့ အလယ် သို့ အစားထိုးလိုက်မည် ဖြစ်ပါသည်။ ထို့နောက် နောက်ထပ် ဘယ်ဘက်သို့ pixel တစ်ခု ရွှေ့ပြီး ဆက်လက် တွက်ချက်သွားမှာ ဖြစ်ပါတယ်။

```
[[ 30  40 139]
 [ 70  79 130]
 [ 60  37 170]]
```

[30 40 139]

[70 160 130]

[60 37 170]

Program အပြည့်အစုံနှင့် output တို့ကိုတော့ အောက်မှာ ဖော်ပြထားပါတယ်။

```
originalPixel =[[30,40,139],[70,79,130],[60,37,170]]

mx = [[-1,1,1],[-1,0,1],[-1,2,1]]
sx = np.square(mx)
my = [[-1,-2,-1],[0,0,0],[2,1,1]]
sy = np.square(my)
mz = np.add(sx,sy)
dz = np.divide(-mz,2.2619)
myKernel = np.exp(dz)*0.442106
print("MyKernel")
print(myKernel)

print("Original Image Pixel")
print(np.array(originalPixel))

FinlaPixel = np.multiply(originalPixel,myKernel)
print("New Pixel Multiply by myKernel")
print(FinlaPixel)

Sum_NewPixel = np.sum(FinlaPixel)
print("Sum of NewPixel")
print(Sum_NewPixel)
```

```
MyKernel
[[0.18260718 0.04847357 0.18260718]
 [0.2841333  0.442106   0.2841333 ]
 [0.04847357 0.04847357 0.18260718]]
Original Image Pixel
[[ 30  40 139]
 [ 70  79 130]
 [ 60  37 170]]
New Pixel Multiply by myKernel
[[ 5.47821538  1.9389428  25.38239792]
 [19.88933069 34.926374   36.93732842]
 [ 2.90841421  1.79352209 31.04322048]]
Sum of NewPixel
160.2977459885186
```

အထက်တွင် Gaussian Kernel နှင့် Gaussian Filtering အကြောင်းများကို တိတိကျကျ အသေးစိတ် နားလည်သွားပြီဖြစ်တဲ့အတွက်စာရေးသူတို့ရဲ့ main program သို့ ဆက်သွားကြရအောင်။

```
GauImage = cv2.GaussianBlur(originalImage,(7,7),0)
image5 = cv2.resize(GauImage, (960, 540))
cv2.imshow("GaussianBlur",image5)
```

CV2.bitwise_and()

Bitwise and logical operation သည် image တွေကို ပေါင်းရာမှာ အသုံးဝင်ပါတယ်။ images နှစ်ခုရဲ့ သက်ဆိုင်ရာ Pixels တွေကိုပေါင်းပေးပါတယ်။ များသောအားဖြင့် Input images တွေရဲ့ မတူညီတဲ့အချက်တွေကို သိနိုင်ဖို့အသုံးပြုကြပါတယ်။ bitwise_and() ထဲတွင် ထည့်ပေးလိုက်သည့် mask အပေါ်မူတည်ပြီး အချို့သော နေရာတွေကို highlighting လုပ်ပြီး ပြန်ပြ ပေးပါတယ်။ ဥပမာ စာရေးသူတွင် ပုံနှစ်ပုံ ရှိသည် ဆိုပါစို့။ ထို ပုံ ပုံနှစ်ပုံအား ပေါင်းပြီး ပြန်ရလာတဲ့ result ကို အောက်တွင် ကြည့်နိုင်ပါသည်။ အရေးကြီးသည့်အချက်မှာ image များကို operation လုပ်ရာတွင် ထို image များရဲ့ size ကို တူညီအောင် အရင်ဆုံး လုပ်ပေးရပါတယ်။ ဥပမာ image1 သည် 800 x 740 pixel ဆိုလျှင် ဒုတိယ image2 သည်လည်း 800 x 740 ဖြစ်ရမည်။ ထိုကဲ့သို့ ညှိပေးရန် အတွက်လည်း OpenCV တွင် resize ဆိုတဲ့ function ကို သုံးနိုင်ပါတယ်။ resize function သည် parameter နှစ်ခု ယူပြီး ပထမ တစ်ခုသည် resize လုပ်ချင်သော image ဖြစ်ပြီး ဒုတိယ တစ်ခုသည် မိမိ ပြောင်းလိုသည့် pixel ဖြစ်ပါသည်။

```
import cv2 as cv

img1 = cv.imread("img1.png")
img2 = cv.imread("img2.png")
r_img1 = cv.resize(img1, (800, 740))
r_img2 = cv.resize(img2, (800, 740))
bitwise = cv.bitwise_and(r_img2,r_img1)
cv.imshow("Bitwise_and",bitwise)
cv.waitKey(0)
cv.destroyAllWindows()
```

Output Image

နောက်ဆုံး အဆင့်အနေဖြင့်စာရေးသူတို့ရဲ့ main program တွင် processing လုပ်ထားသော image အားလုံးကို တစ်စု တစ်စီးတည်း ပြန်လည် ဖော်ပြပါမည်။ ထို့ပြင် main program ရဲ့ User Interface တွင်လည်း Save TransFormed Image တစ်ခုကို ထပ်ထည့်ပါမည်။ အကယ်၍ ထို save ဆိုသည့် button ကို နှိပ်လိုက်လျှင် ImageSave ဆိုသည့် function ကို သွားခေါ်ပါမည်။ program မှာ အောက်ပါ အတိုင်း ဖြစ်သည်။

```
images=[image1,image2,image3,image4,image5,image6]
fig, axes = plt.subplots(3,2, figsize=(8,8), subplot_kw={'xticks':[], 'yticks':[]},
                         gridspec_kw=dict(hspace=0.1, wspace=0.1))
for i, ax in enumerate(axes.flat):
    ax.imshow(images[i], cmap='gray')

save1=Button(gui,text="Save TransFormed Image",command=lambda: ImageSave(image6, ImagePath),
             padx=10,pady=5)
save1.configure(background='#612184', foreground='#F9F60A')
save1.pack(side=TOP,pady=20)

plt.show()
```

ImageSave function ကို ခေါ်ရာတွင် image ကိုရော image path ကိုပါ တစ်ခါတည်း ထည့်ပေးလိုက်ပါမည်။ Image save function သည် os.path ထဲမှ dirname ဆိုသည့် function ကို သုံးကာ ImagePath ကို ရယူ လိုက်ပါမည်။ ထို့ပြင် os.path.splitext ကို သုံးကာ image ရဲ့ extension ကိုလည်း ရယူမှာ ဖြစ်ပါတယ်။ ယခုအပိုင်းအား နားမလည်ပါကရှေ့ပိုင်းတွင် os module နှင့် ပတ်သက်ပြီး ဖော်ပြထားသည့် သင်ခန်းစာအား ပြန်လည်ကြည့်ရှုရန် လိုအပ်ပါသည်။ နောက်ဆုံး အဆင့်အနေဖြင့် openCV မှ imwrite function ကို သုံးကာ နောက်ဆုံးရရှိလာသော Image အား သိမ်းဆည်း လိုက်မှာ ဖြစ်ပါတယ်။

```
def ImageSave(image6, ImagePath):
    newName="TransFormed Image"
    path1 = os.path.dirname(ImagePath)
    extension=os.path.splitext(ImagePath)[1]
    path = os.path.join(path1, newName+extension)
    cv2.imwrite(path, cv2.cvtColor(ReSized6, cv2.COLOR_RGB2BGR))
    info= "Image saved by name " + newName +" at "+ path
    tk.messagebox.showinfo(title=None, message=info)
```

cv2.COLOR_RGB2BGR သည် image processing project အစပိုင်းတွင် ဖော်ပြခဲ့ပြီး ဖြစ်သည့်အတွက် ယခုအခါတွင် ထပ်မံမဖော်ပြလိုတော့ပါ။ အစပိုင်းတွင် RGB သို့ ပြောင်းခဲ့ခြင်း ဖြစ်သည့်အတွက် ယခုကဲ့သို့ ပြန်သိမ်းသောအခါတွင်လည်း BGR သို့ ပြန်ပြောင်းကာ သိမ်းဆည်းခြင်း ဖြစ်ပါသည်။ စာရေးသူတို့ရဲ့ project တစ်ခုလုံး program အပြည့်အစုံကိုတော့ အောက်မှာ ပြန်လည် ဖော်ပြထားပါတယ်။

```
import cv2
import easygui
import numpy as np
import imageio

import sys
import matplotlib.pyplot as plt
import os
import tkinter as tk
from tkinter import filedialog
from tkinter import *
from PIL import ImageTk, Image

gui=tk.Tk()
gui.title('Transform Your Image !')
gui.configure(background='#218384')
gui.geometry('450x500+20+20')
label=Label(gui,background='#00FFFF')

def getImage():
  ImagePath=easygui.fileopenbox()
  transForm(ImagePath)


def transForm(ImagePath):
  originalImage = cv2.imread(ImagePath)
  originalImage = cv2.cvtColor(originalImage, cv2.COLOR_BGR2RGB)
  print("original",originalImage)
  print(originalImage.ndim)
```

```
  if originalImage is None:
     print("It is not a valid image File .")
     sys.exit()
  #For 1 image
  image1 = cv2.resize(originalImage, (900, 450))

  #For 2 image
  grayScale= cv2.cvtColor(originalImage, cv2.COLOR_BGR2GRAY)
  image2 = cv2.resize(grayScale, (900, 450))

  #For blur image 3
  smoothGray = cv2.medianBlur(grayScale, 3)
  image3 = cv2.resize(smoothGray, (900, 450))

  getEdge = cv2.adaptiveThreshold(smoothGray, 255,
     cv2.ADAPTIVE_THRESH_MEAN_C,
     cv2.THRESH_BINARY, 9, 9)
  image4 = cv2.resize(getEdge, (960, 540))
  # cv2.imshow("Adaptive Threshold",image4)

  GauImage = cv2.GaussianBlur(originalImage,(7,7),0)
  image5 = cv2.resize(GauImage, (960, 540))
  cv2.imshow("GaussianBlur",image5)

  FinalImage = cv2.bitwise_and(GauImage, GauImage, mask=getEdge)
  image6 = cv2.resize(FinalImage, (960, 540))

  images=[image1,image2,image3,image4,image5,image6]
  fig, axes = plt.subplots(3,2, figsize=(8,8), subplot_kw={'xticks':[], 'yticks':[]},
                    gridspec_kw=dict(hspace=0.1, wspace=0.1))
  for i, ax in enumerate(axes.flat):
     ax.imshow(images[i], cmap='gray')
          save1=Button(gui,text="Save   TransFormed   Image",command=lambda:   ImageSave(image6,
ImagePath),
           padx=10,pady=5)
  save1.configure(background='#612184', foreground='#F9F60A')
  save1.pack(side=TOP,pady=20)
  plt.show()
def ImageSave(image6, ImagePath):
  newName="TransFormed Image"
  path1 = os.path.dirname(ImagePath)
  extension=os.path.splitext(ImagePath)[1]
  path = os.path.join(path1, newName+extension)
  cv2.imwrite(path, cv2.cvtColor(ReSized6, cv2.COLOR_RGB2BGR))
  info= "Image saved by name " + newName +" at "+ path
  tk.messagebox.showinfo(title=None, message=info)
```

```
upload=Button(gui,text="Transform Your Image",command=getImage,padx=10,pady=5)
upload.configure(background='#612184', foreground='#F9F60A',font=('calibri',10,'bold'))
upload.pack(side=TOP,pady=150)
gui.mainloop()
```

```
import cv2
import easygui
import numpy as np
import imageio

import sys
import matplotlib.pyplot as plt
import os
import tkinter as tk
from tkinter import filedialog
from tkinter import *
from PIL import ImageTk, Image

gui=tk.Tk()
gui.title('Transform Your Image !')
gui.configure(background='#218384')
gui.geometry('450x500+20+20')
label=Label(gui,background='#00FFFF')

def getImage():...
def transForm(ImagePath):
    originalmage = cv2.imread(ImagePath)
    originalImage = cv2.cvtColor(originalmage, cv2.COLOR_BGR2RGB)
    print("original",originalImage)
    print(originalImage.ndim)

    if originalImage is None:
        print("It is not a valid image File .")
        sys.exit()
    #For 1 image
    image1 = cv2.resize(originalImage, (900, 450))

    #For 2 image
    grayScale= cv2.cvtColor(originalImage, cv2.COLOR_BGR2GRAY)
    image2 = cv2.resize(grayScale, (900, 450))
```

```
34
35      #For 2 image
36      grayScale= cv2.cvtColor(originalImage, cv2.COLOR_BGR2GRAY)
37      image2 = cv2.resize(grayScale, (900, 450))
38
39      #For blur image 3
40      smoothGray = cv2.medianBlur(grayScale, 3)
41      image3 = cv2.resize(smoothGray, (900, 450))
42
43      getEdge = cv2.adaptiveThreshold(smoothGray, 255,
44          cv2.ADAPTIVE_THRESH_MEAN_C,
45          cv2.THRESH_BINARY, 9, 9)
46      image4 = cv2.resize(getEdge, (960, 540))
47      # cv2.imshow("Adaptive Threshold",image4)
48
49      GauImage = cv2.GaussianBlur(originalImage,(7,7),0)
50      image5 = cv2.resize(GauImage, (960, 540))
51      cv2.imshow("GaussianBlur",image5)
52
53      FinalImage = cv2.bitwise_and(GauImage, GauImage, mask=getEdge)
54      image6 = cv2.resize(FinalImage, (960, 540))
55
56      images=[image1,image2,image3,image4,image5,image6]
57      fig, axes = plt.subplots(3,2, figsize=(8,8), subplot_kw={'xticks':[], 'yticks':[]},
58                               gridspec_kw=dict(hspace=0.1, wspace=0.1))
59      for i, ax in enumerate(axes.flat):
60          ax.imshow(images[i], cmap='gray')
61
62      save1=Button(gui,text="Save TransFormed Image",command=lambda: ImageSave(image6, ImagePath),
63                   padx=10,pady=5)
64      save1.configure(background='#612184', foreground='#F9F60A')
65      save1.pack(side=TOP,pady=20)
66      plt.show()
67
```

```
66      plt.show()
67
68  def ImageSave(image6, ImagePath):
69      newName="TransFormed Image"
70      path1 = os.path.dirname(ImagePath)
71      extension=os.path.splitext(ImagePath)[1]
72      path = os.path.join(path1, newName+extension)
73      cv2.imwrite(path, cv2.cvtColor(ReSized6, cv2.COLOR_RGB2BGR))
74      info= "Image saved by name " + newName +" at "+ path
75      tk.messagebox.showinfo(title=None, message=info)
76
77  upload=Button(gui,text="Transform Your Image",command=getImage,padx=10,pady=5)
78  upload.configure(background='#612184', foreground='#F9F60A',font=('calibri',10,'bold'))
79  upload.pack(side=TOP,pady=150)
80
81  gui.mainloop()
```

Program အားလုံး run ပြီးချိန်၌ output သည် အထက်ပါ အတိုင်းထွက်လာမည် ဖြစ်ပြီး အကယ်၍ output image ကို သိမ်းထား လိုပါကလည်း Save Transformed Image ကိုနှိပ်ပြီး သိမ်းဆည်း နိုင်ပါသည်။

ဆက်လက် ရေးသား နေဆဲ ဖြစ်သည်.....