



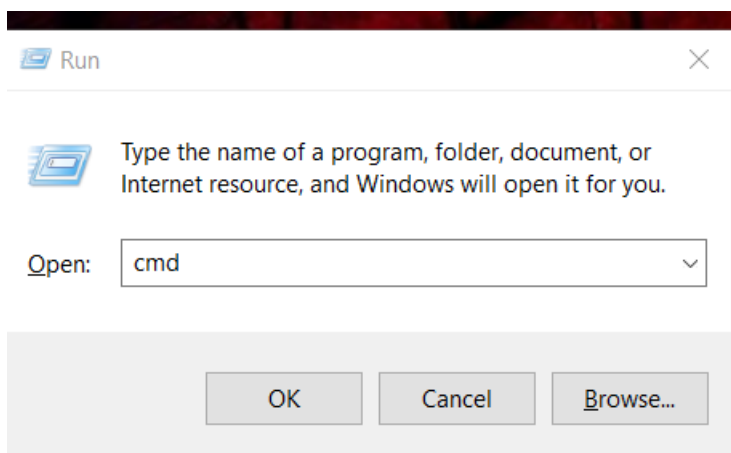
LEARN PYTHON PROGRAMMING STEP BY STEP

Title

1. Installation
2. Starting Hello World

Installation

Python အား install ပြုလုပ်ရန် python အား download ဆွဲပေးဖို့ လိုအပ်ပါသည်။ ။
<https://www.python.org/downloads/> ယခု link ကို သွားပါ မိမိ အသုံးပြုနေသော os ကို auto detect လုပ်ပေးပါလိမ့်မည်။ download button ကို နှိပ်လိုက်သည်နှင့် တစ်ပြိုင်နက် download စတင် ဆွဲပေးပါလိမ့်မည်။ exe file ကို install လုပ်ချိန်တွင် သတိပြုရန်မှာ ပထမဆုံး ပေါ်လာသော window form တွင် add path ကို အမှန်ခြစ် ပေးခဲ့ပါ။ python install လုပ်ပြီးလျှင် စစ်ဆေးရန် window key+r ကို နှိပ်ပါ ထို့နောက် cmd ဟု ရေးပြီး ok ကိုနှိပ်ပါ။



ထိုသို့ ရေးပြီးနောက် ပေါ်လာ သော console တွင် python ဟု ရေးပါ။ ။

```
C:\Windows\system32\cmd.exe - python
Microsoft Windows [Version 10.0.17763.615]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\MAT>python
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 19:29:22) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

အထက်ပါ အတိုင်းပေါ်လာလျှင် python ကို အောင်မြင်စွာ Install လုပ်ပြီးသလို python ကို စတင် ရေးသားနိုင်ပါပြီ။

Starting Hello World

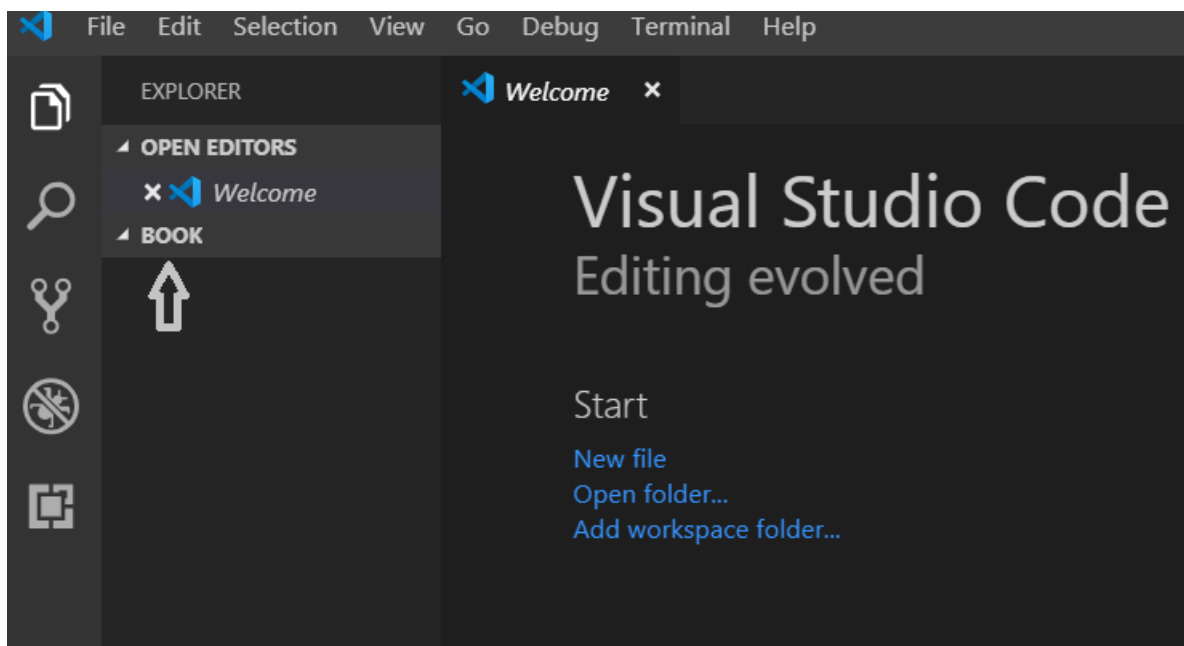
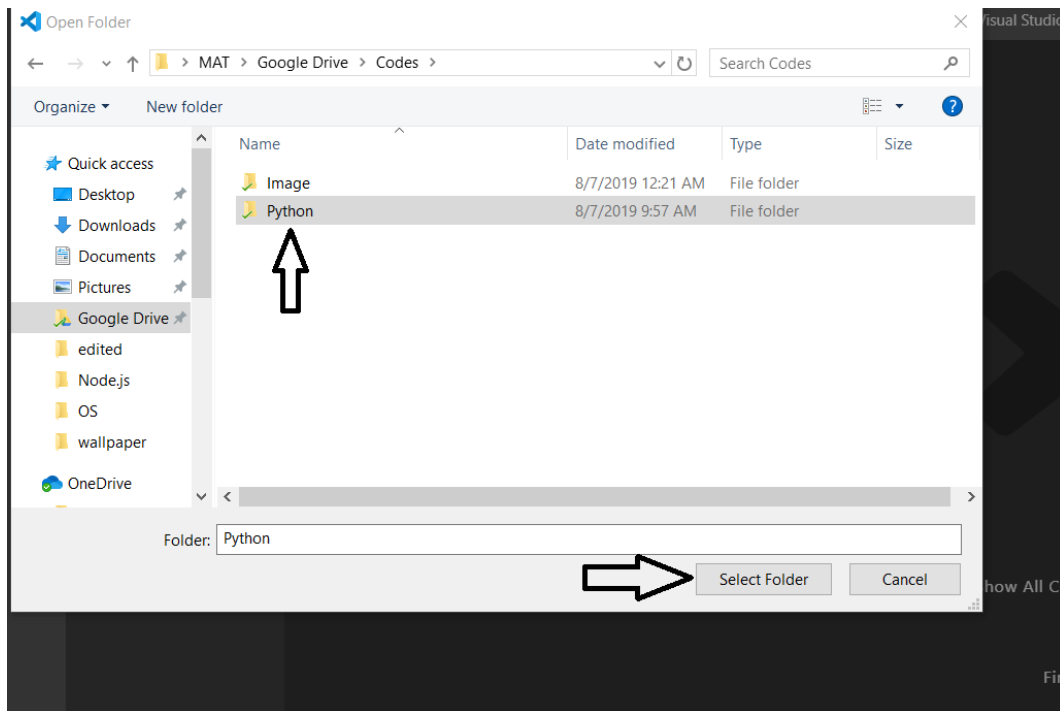
Hello world program တစ်ပုဒ်ကို စတင်ရေးသားပါမည် ။ `print("Hello World")` ဟုရေးပြီး enter နှိပ်ကြည့်ပါ ။ အောက်ပါ ပုံ အတိုင်း Hello World ဆိုသည့် Output ပေါ်လာလျှင် program တစ်ပုဒ်ကို အောင်မြင်စွာ ရေးသားနိုင်ပါပြီ။

```
C:\Windows\system32\cmd.exe - python
C:\Users\MAT>python
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 19:29:22) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("hello world")
hello world
>>>
```

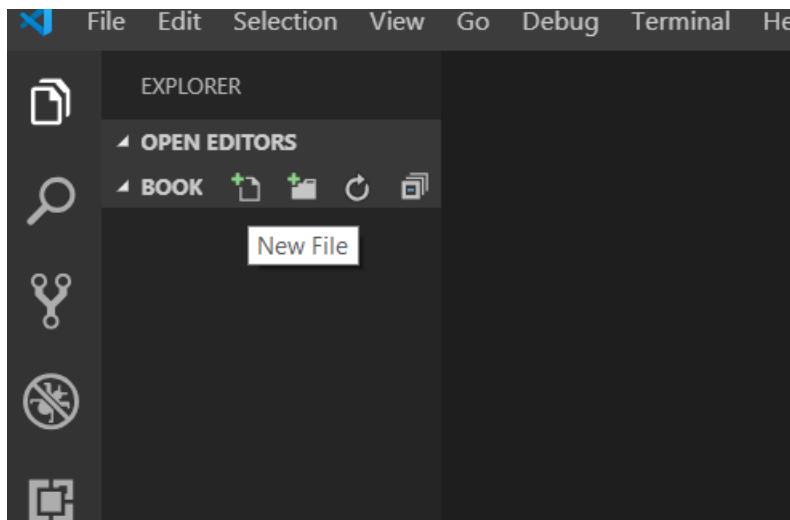
Using Visual Studio Code

Console ကိုလည်းကောင်း Visual Studio Code ကိုလည်းကောင်း အသုံးပြုသွားပါမည်။ visual studio code ကို download ပြုလုပ်နိုင်ရန် အောက်ပါ link ကိုသွားပါ ။ <https://code.visualstudio.com/> install လုပ်ပြီးနောက် visual studio code ကိုဖွင့်ပါ ။ ထို့နောက် file ထဲကိုသွားပါ ပြီးလျှင် open folder ကို နှိပ်ပါ ။ ပေါ်လာသော window form နေရာတွင် မိမိ ကြိုက်နှစ်သက်ရာ နေတာ တစ်ခုကို သွားပြီး folder တစ်ခု တည်ဆောက်ပါ ။ ပြီးလျှင် ထို folder

ကို တစ်ချက် သာ နှိပ်ပြီး select ဟု နှိပ်လိုက်လျှင် vs code တွင် မိမိတို့ ဆောက်လိုက်သော folder ပေါ်လာသည်ကို မြင်ရပါမည်။ ထို folder ထဲတွင် program များကို ရေးသားမည်ဖြစ်ပါသည်။

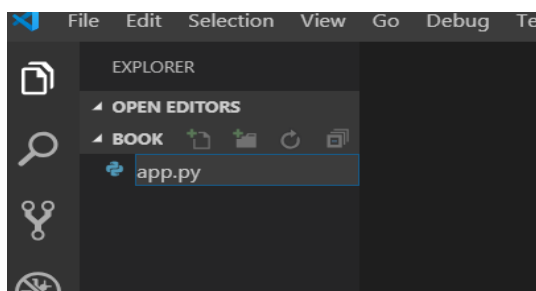


စာရေးသူ အနေဖြင့် book ဆိုသည့် folder တစ်ခု တည်ဆောက် ထားသည့် အတွက် book folder လေးကို မြင်ရပါမည်။ ထို folder ထဲတွင် python program file များကို ထည့်သွင်းရေးသား သွားမည်ဖြစ်ပါသည်။ Welcome page ကို ပိတ်လိုက်ပါ။ ထို့နောက် book folder အောက်တွင် app.py ဆိုသည့် python file တစ်ခု တည်ဆောက်ပါမည်။ book folder ဘေးလေးကို mouse တင်လိုက်သည်နှင့် တစ်ပြိုက်နက် အောက်ပါ ပုံအတိုင်း new file ဆိုသည့် စာသားလေးပေါ်လာပါ မည်။ထို file လေးကို တစ်ချက် နှိပ်လိုက်ပါ။

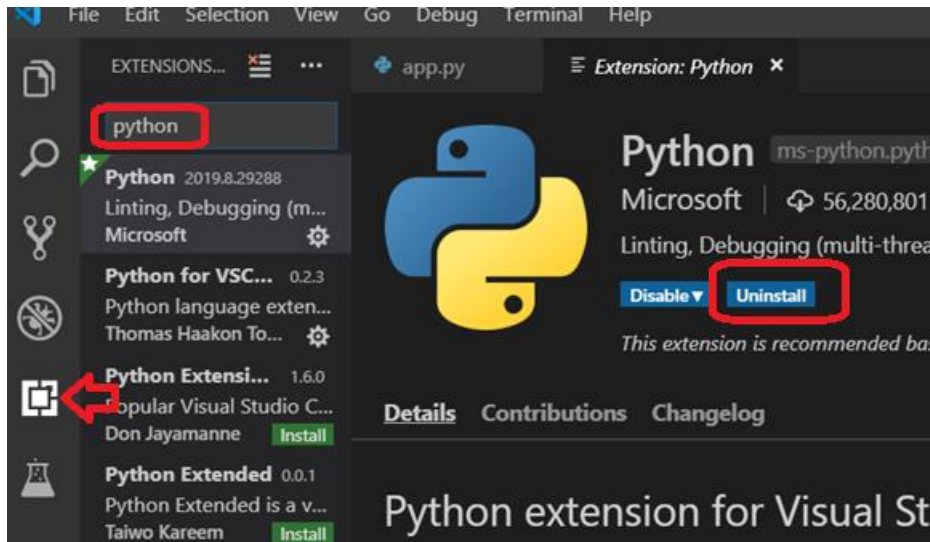


ထိုသို့ နှိပ်လိုသည်နှင့် တစ်ပြိုင်နက် အကွက်လေး တစ်ကွက် ပေါ်လာပါမည်။ ထို အကွက်လေးထဲတွင် app.py ဆိုသည့် စာသားကိုရေးပြီး enter ။ ထို့နောက်တွင် app.py ဆိုသည့် python file တစ်ခုကို အောက်ပါ

အတိုင်း မြင်တွေ့ရပါမည်။



ထိုကဲ့သို့ file create လုပ်လိုက်သည်နှင့် notification များ တက်လာပါမည်။ install များကိုသာ နှိပ်ပေးပါ ။ extension များကို install လုပ်ခြင်းဖြစ်သည်။ ထို့နောက် အောက်ပါ ပုံထဲက အတိုင်း extension ကိုသွားပါ။



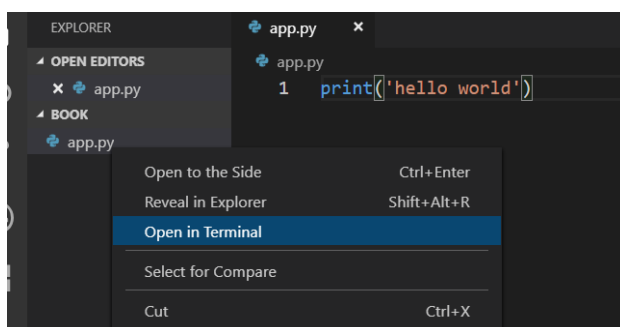
အနီရောင် မြှားဖြင့်ပြထားသော လေးထောင့် အကွက်လေးကို နှိပ်ပါ ။ ထို့နောက် ပေါ်လာသော search အကွက်တွင် python ကိုရေးပါ။ ထို့နောက် ပထမဆုံးနေရာတွင် Python ဆိုသည့် စာတန်း ပေါ်လာပါမည်။ ထိုစာတန်းကို နှိပ်လိုက်သည်နှင့် ညာဘက်တွင် python ပေါ်လာမည် ။ ထိုနေရာတွင် install ကို နှိပ်ပေးပါ။

Start Python in Visual Studio Code

Python code ကို vs code တွင် စရေးကြည့်ပါမယ် ။

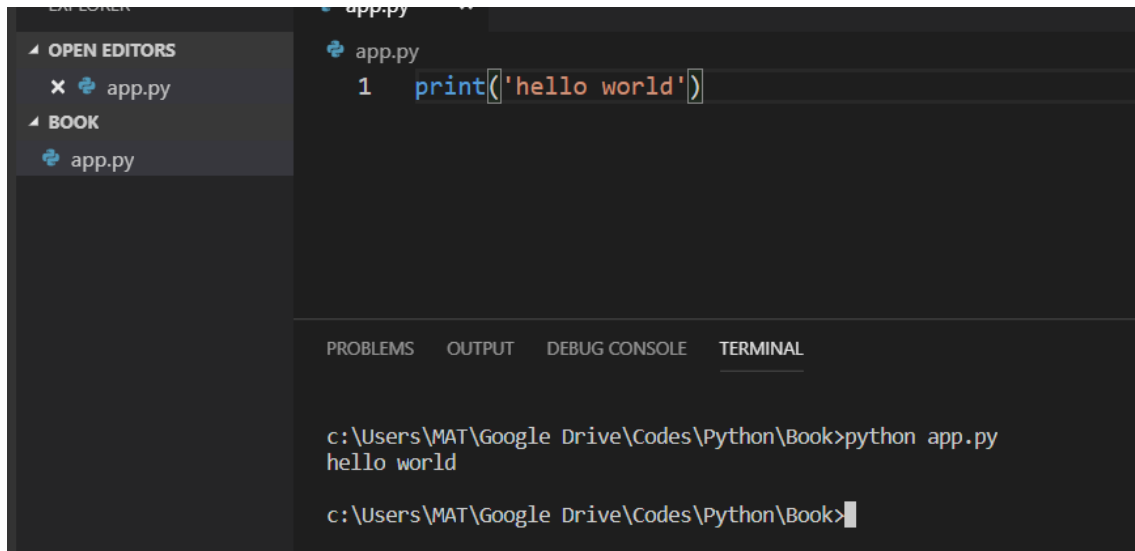
```
print('hello world')
```

အထက်ပါ အတိုင်းရေးပြီး control+s ကို နှိပ်ပါ ပြီးလျှင် program run ရန် အောက်ပါ ပုံထဲ က အတိုင်း app.py ဆိုသည့် file ကို right click ထောက်ပြီး open in terminal ဆိုတာကို နှိပ်ပါ ။



ပုံပါ အတိုင်း နှိပ်ပြီးသည်နှင့် တစ်ပြိုင်နက် terminal တစ်ခု ပွင့်လာပါမည်။ ထို terminal သည် မိမိတို့ python program များကို run ရန် နေရာ ဖြစ်သည်။ python program ကို

run ရန် python app.py ဟုရေးကာ run ပေးရမည်။ထိုသို့ရေးပြီး enter နှိပ်လိုက်သည်နှင့် တစ်ပြိုင်နက် output ထွက်လာသည်ကို မြင်ရပါမည်။



```

app.py
1 print('hello world')

c:\Users\MAT\Google Drive\Codes\Python\Book>python app.py
hello world

c:\Users\MAT\Google Drive\Codes\Python\Book>

```

Python Standard Input and Output

Python programming တွင် built-in functions များစွာ ပါဝင်ပါတယ်။ standard input အနေဖြင့် input() function ကို အသုံးပြုပြီး standard output အနေဖြင့် print() ကို အသုံးပြု ပါသည်။ format function ကိုအသုံးပြုပြီးတော့လည်း format(ပုံစံတကျ) ဖြင့် print ထုတ်နိုင် ပါတယ်။ format function ကိုသုံးရာတွင် curly bracket {} ကိုအသုံးပြုပါတယ်။ format function ကိုသုံးရာတွင် format ရှေ့၌ (dot) . ကိုရေးပေးရမည်ဖြစ်ပြီး format function ထဲမှာလည်း မိမိဖော်ပြချင်သော data များကို အစဉ်လိုက် , ခြားပြီးရေးပေးရပါမည်။

```

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

```

Output အနေဖြင့် 10 and 5 တို့သည် curly bracket {} ထည့်ထားသော နေရာများ တွင် အစဉ်လိုက်ပေါ်လာပါသည်။ C programming တွင်မူ integer များအတွက် ထိုကဲ့သို့ စာသားများ ကြားတွင် ဖော်ပြလိုသော အခါမျိုး၌ %d ကိုသုံးသည်။ format function အခြားသော နည်းလမ်း ဖြင့်လည်း ထပ်မံ အသုံးပြုနိုင်ပါသေးသည်။ format ထဲတွင် ထည့်ထားသော element များသည် index 0 မှ စတင်ပြီး အစဉ်လိုက် နေရာယူပါသည်။ ထိုသို့ အစဉ်လိုက်နေရာယူထားသော elements များကို print လုပ်သော အချိန်တွင် မိမိလိုသလို index

number များကို သုံးပြီး ရှေ့နောက် ပြောင်းလဲ နိုင်ပါသည်။ ရှေ့ဆုံး element ကို ပေါ်စေချင်သော အခါမျိုးတွင် curly bracket အတွင်း၌ {0} ကို အသုံးပြုပါသည်။ ထိုနည်းတူ ဒုတိယ element ကိုပေါ်စေချင်သော အခါမျိုး၌လည်း {1} ကို အသုံးပြုပါသည်။ အောက်တွင် sample program နှင့် output တို့ကို ဖော်ပြထားပါသည်။

```
a=10;b=5
print('hello {2} {3} form {0}
{1}'.format('green','hackers','win','htut'))
#output
hello win htut form green hackers
```

ထို့ပြင် format function ကိုသုံးပြီး string များကို print ထုတ်ရာတွင်လည်း keyword arguments များကိုလည်း သုံးနိုင်ပါသေးသည်။ sample program နှင့် output ကို အောက်တွင် ဖော်ပြထားပါသည်။

```
print('Hello {name} team {gh}'.format(name='WinHtut',gh='GH'))
#output
Hello WinHtut team GH
```

Python တွင် ဒသမ ကိန်းများကို print လုပ်ရာ၌ % operator ကိုသုံးပြီး မိမိတို့ လိုသလို control လုပ်နိုင်ပါသေးသည်။ ဒသမ နှစ်နေရာ ထိ print လုပ်ချင်သော အခါတွင် %.2f လို့ ရေးရပြီး 3 နေရာထိ print လုပ်ချင်သော အခါတွင်မူ %.3f ဟုရေးရပါသည်။ sample program ကို အောက်တွင် output နှင့် တကွ ဖော်ပြထားပါသည်။

```
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
```

Standard Input in Python Programming

Standard input ဆိုတာ user ဆီမှ ထည့်ပေးလိုက်သော သို့မဟုတ် program ထဲသို့ ဝင်လာသော data or elements များကို input လို့ ခေါ်ပါတယ်။ keyboard and mouse တို့သည်

Input များဖြစ်ပြီး monitor , soundbox and etc တို့သည် output များ ဖြစ်ကြပါတယ်။ 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

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

Standard Data Types in Python Programming Language

Python programming မှာဆိုရင် standard data type အနေနဲ့ ၅ မျိုးရှိပါတယ်။

- Numbers
- String
- List
- Tuple
- Dictionary

Numbers in Python

Number data type တွေ ဆိုတာ numeric values တွေကို သိုလှောင်ဖို့ အတွက် သုံးပါတယ်။ value တစ်ခု assign သတ်မှတ်ပေးလိုက်တာနဲ့ number objects ကို ပြုလုပ်နိုင်ပါတယ်။

For example :

```
Var1=1
```

```
Var2=2
```

del ဆိုတဲ့ statement ကို အသုံးပြုပြီး number objects တွေကို ဖြတ်ထုတ်နိုင်ပါတယ်။

For example:

```
del var1
```

del ဆိုတာကို သုံးပြီး single object တစ်ခု တည်းတင်မကပဲ multiple objects တွေ ကိုပါ ဖြတ်ထုတ်နိုင်ပါသေးတယ်။

For example:

```
del var1[var2,var3,var4]
```

Python Programming Language မှာ Number data type ကို numerical types အနေနဲ့ သုံးမျိုး support ပေးထားပါတယ်။

- Int (signed integers)
- Float(floating point real values)
- Complex(complex)
- Booleans

Int မှာဆိုရင် long integers အနေနဲ့ ကိုယ်စားပြုပါတယ်။ဆိုလိုချင်တာက python မှာ integer တွေ အားလုံးဟာ long တွေ ဖြစ်တယ် ။

int	float	complex
10	0.0	3.14j
100	15.20	45.j
-786	-21.9	9.322e-36j
080	32.3+e18	.876j
-0490	-90.	-.6545+0j
-0x260	-32.54e100	3e+26j
0x69	70.2-E12	4.53e-7j

Float ဆိုတာကတော့ ဒဿ ကိန်းတွေကို ဆိုလိုတာပါ။

Complex numbers

Complex numbers တွေဆိုတာ real number နဲ့ imaginary number ပါဝင်တာကို ဆိုလိုတာပါ။

For Example :

Bj

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

3+7j

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

Real Number + Imaginary Number = Complex Number

Booleans

Booleans data type ထဲမှာတော့ **True** and **False** ပါဝင်မှာဖြစ်ပါတယ်။ **True** ကို number အနေနဲ့ ပြောမယ်ဆိုရင် 1 ဖြစ်ပြီး **False** သည် 0 ဖြစ်ပါသည်။

```
Python 3.6.5 (v3.6.5:f59c0932b4, Ma
Type "help", "copyright", "credits"
>>> bool(1)
True
>>> bool(2)
True
>>> bool(0)
False
>>> bool(0.0)
False
>>> bool(2.1)
True
>>>
```

bool(1) ဆိုတဲ့ code ကိုရေးလိုက်ရင် result အနေနဲ့ **True** ဆိုတာကို ရမှာဖြစ်ပါတယ်။ ရှေ့ **bool** ဆိုတာ ကတော့ သူ့ရဲ့ data type ဖြစ်ပါတယ်။ **bool(0)** လို့ရေးရင်တော့ **False** ဆိုတဲ့ result ကိုရမှာဖြစ်ပါတယ်။ ဒါမ ကိန်းတွေ ထည့်ရင်လည်း number တစ်ခုခုဖြစ်နေခဲ့ ရင်

True ဆိုတဲ့ result ကိုပဲရရှိမှာဖြစ်ပါတယ်။ **bool(-2)** ထိုကဲ့သို့ အနုတ်ကိန်းတွေ ထည့်ခဲ့ရင်လည်း **True** ကိုပဲ ရရှိမှာဖြစ်ပါတယ်။

Type Conversion

Python တွင် မည်သည့် value မှာမဆို data type တွေရှိပါတယ်။ Data type တွေဆိုတာ data တွေကို ခွဲခြားထားခြင်းဖြစ်ပြီး compiler or interpreter အား data တွေကို ဘယ်လို ပုံစံနဲ့ အသုံးပြုမည်ဖြစ်ကြောင်း ပြောကြားခြင်းပင် ဖြစ်သည်။ Type Conversion ဆိုသည်မှာ data type တစ်ခုကနေ အခြား data type တစ်ခုကို ပြောင်းလဲခြင်းကို ဆိုလိုပါတယ်။ Data တွေကို ကိုင်တွယ်တဲ့နေရာမှာ type တွေကို လိုသလို ပြောင်းလဲခြင်းဖြင့် ပိုမို အသုံးပြုရ လွယ်ကူစေပါတယ်။ Type conversion မှာဆိုလျှင် implicit and explicit ဟူပြီး နှစ်မျိုးရှိပါသည်။

Implicit Type Conversion or *coercion*

Implicit type conversion သည် run time မှာ Python ကနေပြီး data တွေကို တိုက်ရိုက် conversion ပြုလုပ်ခြင်းဖြစ်သည်။ user ကနေ ပြုလုပ်ပေးစရာ မလိုပါဘူး။ data များအား မည်သည့် data type ဖြစ်သည်ကို သိလိုပါက **type()** function ကို အသုံးပြုနိုင်ပါသည်။ အောက်ပါ Program တွင် **c_sum** ရဲ့ data type ကိုသိနိုင်ရန် **type()** function ကိုသုံးထားပါသည်။

sample program

```
a_int = 1
b_float = 1.0
c_sum = a_int + b_float
print(c_sum)
print(type(c_sum))
```

Output

2.0

<class 'float'>

အထက်ပါ program တွင် int and float တို့ကို ပေါင်းရာ၌ int သို့ ပြောင်းလဲ မသွားပဲ output တွင် float data type သို့ ပြောင်းလဲ သွားပါသည်။ python တွင် int data size ထက် float data size က ပိုများပါသည်။ float မှ int သို့ ပြောင်းလျှင် float တွင်ဝင်နေသော ဒသမ ကိန်းများ ပျောက်ဆုံး သွားနိုင်ပါသည်။

Explicit Type Conversion

Explicit Type Conversion သည် user မှ data များကို လိုသလို ပြောင်းလဲခြင်းဖြစ်သည်။ Python တွင်မူ int() , float() , str() စသည့် function များကို explicit type conversion တွင်အသုံးပြုနိုင်ပါသည်။ Explicit type conversion ကို typecasting လို့လည်းခေါ်ဆိုပါသည်။

Sample program

```
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))
```

အထက်ပါ program အား run ကြည့်လျှင် output အနေဖြင့် data type နှစ်ခု ထွက်လာပါမည်။ပထမ တစ်ခုသည် python မှ handle လုပ်ထားသော implicit type ဖြစ်ပြီး ဒုတိယ တစ်ခု ဖြစ်သည့် int သည် user မှ handle လုပ်ထားသော explicit type ဖြစ်သည်။

သတိပြုရန် အချက်များ

- implicit type တွင် data loss ဖြစ်ခြင်းကို ကာကွယ်နိုင်ရန် python interpreter သည် data size သေးရာမှ ကြီးရာသို့ ပြောင်းပေးသည်။
- explicit type သည် data loss ဖြစ်နိုင်ပါသည် အဘယ်ကြောင့်ဆိုသော် user မှ predefined function များကိုသုံးပြီး data type များအား လိုသလို ပြောင်းလဲခြင်းကြောင့် ဖြစ်သည်။

Data Type Conversion with String

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

Sample Program (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 program အား run ကြည့်လျှင် output 25 ကို ရပါမည် ။ program error တက်မည် မဟုတ်ပါ။

ယခုရေးထားသော integer type အား string type ဖြင့် အောက်ပါ အတိုင်း ပြန်လည် ရေးသားပါမည်။

```
int_one=15
```

```
int_two=10
```

```
total=int_one+int_two
```

```
#printing like integer type
```

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

အထက်ပါ program အား run ကြည့်လျှင် အောက်ပါအတိုင်း Type Error ရပါမည် ဆိုလိုသည်မှ implicit နည်းလမ်းနဲ့ integer မှတစ်ဆင့် string သို့ မပြောင်းလဲနိုင်ကြောင်းဖော်ပြထားခြင်းဖြစ်သည်။

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
```

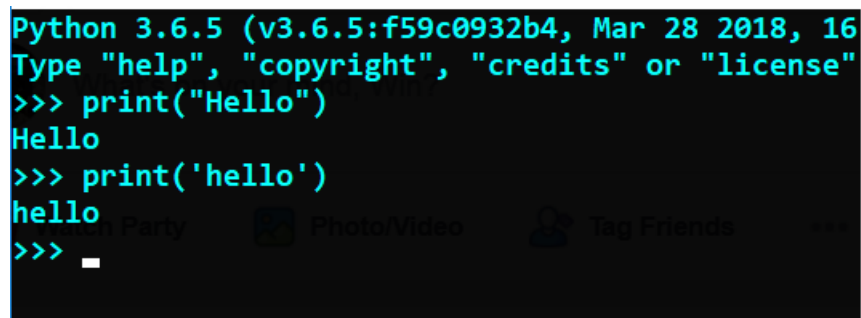
Explicit နည်းလမ်းကို သုံးပြီး int မှ string သို့ type casting ပြုလုပ်ပါမည်။ အောက်ပါ program ကိုကြည့်လျှင် + operator နှင့် , တို့ကွာခြားသွားသည်ကိုမြင်နိုင်ပါသည်။ အောက်တွင်ဖော်ပြထားသော sample program အား run ကြည့်လျှင် output အနေဖြင့် 25 ကိုရပါမည်။

```
int_one=15

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

String in Python

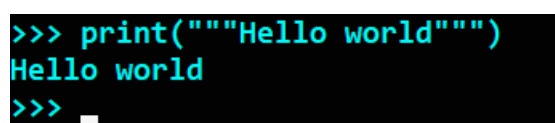
Python programming မှာဆိုရင် string ကို single quotes ('some thing') or double quotes (" some thing ") ထဲမှာထည့်ပြီး တစ်စု တစ်စည်းထဲ သတ်မှတ် ထားခြင်းကို ဆိုလိုပါတယ်။



```
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16
Type "help", "copyright", "credits" or "license"
>>> print("Hello")
Hello
>>> print('hello')
hello
>>> _
```

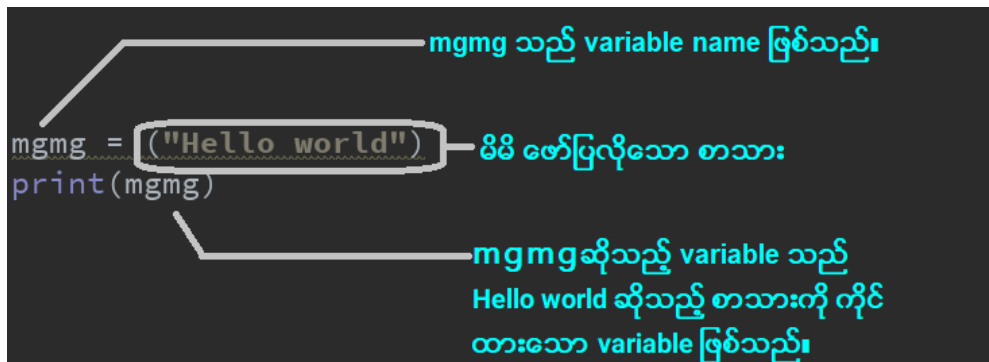
စာသားတွေကို single quote ထဲမှာရေးရင် လည်းရသလို double quote ထဲမှာရေးရင် လည်း ရပါတယ်

သို့သော် မိမိရေးဖော်ပြလိုသော စာသားထဲမှာ single quote ပါနေခဲ့မယ်ဆိုရင် double quote ကိုသုံးသင့်ပါတယ်။ `printf("hello myanmar's people")` ထိုကဲ့သို့ဖြစ်ပါတယ်။ သို့မဟုတ် Triple quote ကိုလည်း အသုံးပြုလို့ရပါသေးသည်။



```
>>> 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 ကို အသုံးပြုပြီး ထည့်လိုက်သည်။ ထို့ကြောင့် mgmg ဆိုသည့် variable ကို program ထဲမှာ မြင် တိုင်းHello world ဆိုသည့် စာကြောင်းကို ပဲ ထုတ်ပေးမည်ဖြစ်သည်။ ဒုတိယ စာ ကြောင်းတွင် mgmg ဆိုသည့် variable ကို ခေါ်သောကြောင့် result အနေနှင့် Hello world ဆိုသည့် စာသားကို ရရှိခြင်းဖြစ်သည်။

Len ကိုအသုံးပြုပြီး စာသားတွေရဲ့ အရေအတွက်ကို ထုတ်ပါမယ်။

```
mgmg = ("Hello world")
print(len(mgmg))
```

`len(length)` သည် မိမိထုတ်လိုက်တဲ့ variable ထဲမှာရှိသော စာသားရဲ့ အလုံးအရေအတွက်ကို ပြပေးပါတယ်

space ကိုပါ character တစ်ခု အနေနဲ့ ထည့်သွင်းရေတွက်ပါတယ်။အထက်ပါ program ကို run လိုက်ရင် output အနေနဲ့ 11 ကိုရရှိမှာဖြစ်ပါတယ်။

Example:

Str="Hello world" or

Str='Hello world'

Python မှာဆိုရင် slice operator တွေကို သုံးပြီး စာသားတွေကို တစ်လုံးချင်းစီ ဖော်ပြခိုင်းလို့ရပါတယ်။သူ့ရဲ့ index တွေကို zero ကနေပဲ စပြီး ရေတွက်ပါတယ်။

Accessing The Character using Slice operator

[] = တစ်လုံးခြင်းစီကို သီးခြား ဖော်ပြလိုတဲ့ အခါမှာ သုံးပါတယ်။

Print (str[0])

```
mgmg = ("Hello world")
print(mgmg[0])
```

ယခု program ကိုရေးကြည့်
လိုက်မယ်ဆိုရင် result အနေနဲ့ H
ကိုရရှိမှာဖြစ်ပါတယ်။

```
str = ("Hello world")
print(str[10])
```

ယခု program ကို run ကြည့်မယ်ဆိုရင်
လည်း result အနေနဲ့ character d ကို ရရှိမှာ
ဖြစ်ပါတယ်။

```
str = ("Hello world")
print(str[11])
```

wh x

```
C:\Users\WinHtut_GreenHackers\PycharmProj
Traceback (most recent call last):
  File "C:/Users/WinHtut_GreenHackers/PycharmProj
    print(str[11])
IndexError: string index out of range
```

ယခု
program ကို run
ကြည့်မယ်ဆိုရင်
out of range
ဆိုတဲ့ error ကို
မြင်တွေ့ရမှာပါ။
ဘာကြောင့်လဲ
ဆိုတော့ စုစုပေါင်း
character 11 လုံး

သာပါဝင်ပြီး index 10 သည် အများဆုံးဖြစ်နေတဲ့အတွက်ပါ။

Str = စာသားတွေ အားလုံးကို ဖော်ပြလိုတဲ့ အခါမှာ သုံးပါတယ်။

Print(str)

```
str = ("Hello world")
print(str)
```

ယခု program ကို run လိုက်ရင် result အနေနဲ့ Hello world ဆိုတဲ့ စာသား အပြည့်အစုံကိုရရှိမှာပါ။

`Print(str[0:5])` ရှေ့က 0 သည် စမဲ့ index number ဖြစ်ပြီး 5 သည် ဆုံးမဲ့ index number ဖြစ်ပါတယ်။ ဥပမာ ကိုအောက်တွင်ကြည့်ပါ။

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

ယခု program ကို run ကြည့်မယ်ဆိုရင် result အနေနဲ့ Hello ဆိုတာကို ရရှိမှာဖြစ်ပါတယ်။ index

အစကနေ အဆုံးထိ character တွေကို ထုတ်ပေးပါတယ်။

ပိုပြီးနားလည်သွား အောင် အောက်က program လေးတွေကို ဆက်ရေးကြည့်နိုင်ပါတယ်။

`[2:5]` = စာသားတွေကိုမိမိ လိုချင်သလောက်ပဲ ဖော်ပြလိုသော အခါမျိုးမှာသုံးပါတယ်။

```
str = ("Hello world")
print(str[2:5])
```

ယခု program ကို 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 ကျန်တဲ့ အဆုံးထိ ဖော်ပြလိုသော အခါမျိုးမှာ သုံးပါတယ်။

```
str = ("Hello world")
print(str[2:])
```

ယခု program ကို run လိုက်ရင် result အနေနဲ့ llo world ဆိုတာကို ရရှိမှာဖြစ်ပါတယ်။ ထည့်ပေး

လိုက်တဲ့ index number ကနေစပြီး နောက်ပိုင်းမှာရှိတဲ့ စာလုံးတွေအားလုံးကို ဖော်ပြပေးမှာ ဖြစ်ပါတယ်။

`[:5]` = ထည့်ပေးလိုက်တဲ့ index number ရဲ့ ရှေ့ထိ စာသားအားလုံးကို ဖော်ပြ ပေးမှာဖြစ်ပါတယ်။ နောက်ပိုင်းစာလုံးတွေကိုတော့ ဖော်ပြပေးမှာ မဟုတ်ပါဘူး။

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

ယခု program ကို run လိုက်မယ်ဆိုရင် result အနေနဲ့ Hello ကိုရရှိမှာဖြစ်ပါတယ်။

```
mgmg = ("Hello world")
print(mgmg[0])
```

ယခုပုံလေးကို ပြန်လည် ဖော်ပြ ပေးထားပါတယ်။ ပိုပြီး နားလည်သွားမှာပါ။

`Print(str * 2)` = စာသားတွေကိုနှစ်ကြိမ်ဖော်ပြလိုသော အခါမျိုးမှာသုံးပါတယ်။

```
str = ("Hello Green Hackers")
print(str*2)
```

C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python3

Hello Green HackersHello Green Hackers

ယခု program ကို run ပြုလုပ်ရင် result အနေနဲ့ စာသားကို နှစ်ကြိမ် ဖော်ပြထား တာကို

မြင်ရမှာဖြစ်ပါတယ်။

`Print (str + "WinHtut")` = စာသားတွေကို တစ်ခုနဲ့တစ်ခု ပေါင်းပြီး ဖော်ပြလိုတဲ့ အခါမှာသုံးပါတယ်။

```
str = ("Hello Green Hackers")
print(str+"WinHtut")
```

C:\Program Files (x86)\Microsoft Visual Studio\Sh

Hello Green HackersWinHtut

result အနေနဲ့

စာသားနှစ်ခုလုံးကို ဖော်ပြ ပေးထားတာကို 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 နဲ့ ရေးထားတဲ့ characters တွေအားလုံးကို uppercase (အကြီး) အဖြစ်ပြောင်းလဲပေးလိုက်ပါတယ်။ တစ်ကယ်လို့ စာသားတွေထဲမှာ uppercase (အကြီး) တွေ ပါနေခဲ့မယ်ဆိုရင် မူလအတိုင်း အကြီးတွေကိုပဲ ပြန်လည်ဖော်ပြပေးမှာပါ။

Syntax:

String.upper()

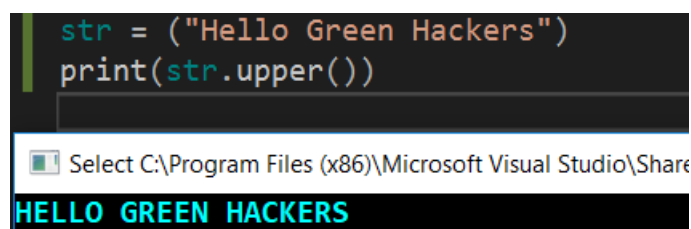
Parameter :

upper(မည်သည့် parameter မှ မပါဝင်ပါ)

Return :

စာလုံးအသေးတွေကို အကြီးအဖြစ် ပြောင်းလဲပေးပါတယ်။

Example



```
str = ("Hello Green Hackers")
print(str.upper())
```

Select C:\Program Files (x86)\Microsoft Visual Studio\Share

HELLO GREEN HACKERS

Result အနေနဲ့ စာလုံးအကြီးတွေကို ပြန်ဖော်ပြပေးတာ မြင်ရမှာပါ။

lower()

Python programming မှာဆိုရင် lower() သည် built-in method ဖြစ်ပြီး string တွေကို လိုအပ်သလို handle လုပ်ဖို့ အသုံးပြုပါတယ်။ uppercase နဲ့ ရေးထားတဲ့ characters တွေအားလုံးကို lowercase (အသေး) အဖြစ်ပြောင်းလဲပေးလိုက်

ပါတယ်။တစ်ကယ်လို့ စာသားတွေထဲမှာ lowercase (အကြီး) တွေ ပါနေခဲ့မယ်ဆို ရင် မူလအတိုင်း အသေးတွေကိုပဲ ပြန်လည်ဖော်ပြပေးမှာပါ။

Syntax:

`String.lower()`

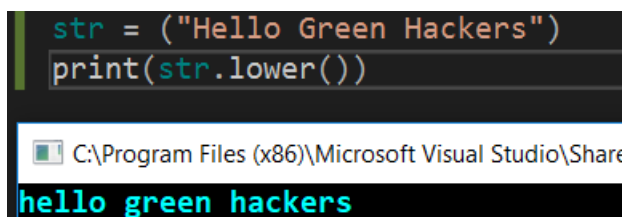
Parameter :

`lower` (မည်သည့် parameter မှ မပါဝင်ပါ)

Return :

စာလုံးအကြီးတွေကို အသေးတွေအဖြစ် ပြောင်းပေးပါတယ်။

Example:

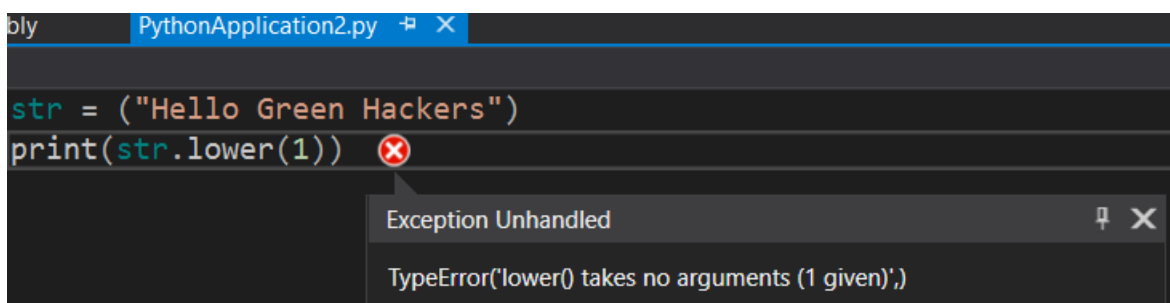


```
str = ("Hello Green Hackers")
print(str.lower())
```

C:\Program Files (x86)\Microsoft Visual Studio\Share
hello green hackers

result အနေနဲ့ characters တွေအား လုံးကို lowercase(စာလုံးအသေး) တွေနဲ့ ဖော် ပြ ထားတာကိုမြင်ရမှာပါ။

Error Example



```
str = ("Hello Green Hackers")
print(str.lower(1))
```

Exception Unhandled
TypeError('lower() takes no arguments (1 given)')

Upper or lower ထဲမှာ အထက်ပါအတိုင်း argument တစ်ခုခု ထည့်လိုက်မယ်ဆိုရင် ပုံပါအတိုင်း error message ကို မြင်တွေ့ရမှာဖြစ်ပါတယ်။

isupper()

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

Syntax:

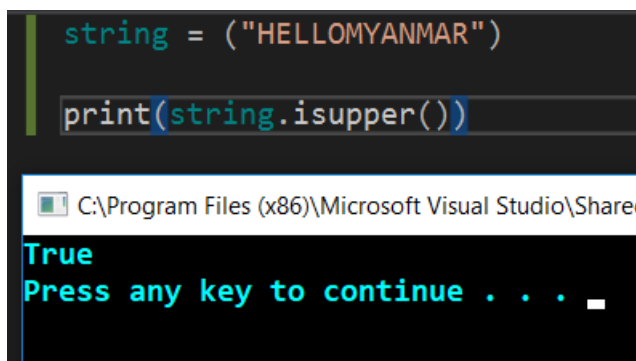
`string.isupper()`

Parameter:

Isupper သည် မညှိသည့် parameter မှ မပါဝင်ပါ။

Returns :

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



```
string = ("HELLOMYANMAR")
print(string.isupper())
True
Press any key to continue . . .
```

ယခု program ကို run ကြည့်ရင် result အနေနဲ့ true ကိုရရှိမှာဖြစ်ပါတယ်။ characters တွေအားလုံး အကြီးတွေ ဖြစ်နေလို့ပါ။

Islower()

islower() သည် python ရဲ့ built-in method ဖြစ်ပြီး string တွေကို handle လုပ်ဖို့အတွက် အသုံးဝင်ပါတယ်။တစ်ကယ်လို့ characters တွေအားလုံးဟာ အသေးတွေချည်းပဲ

ဖြစ်နေခဲ့မယ်ဆိုရင် True ဆိုတဲ့ return value ကိုပြန်ရမှာဖြစ်ပြီး characters တွေထဲမှာ အကြီးတွေရော အသေးတွေရော 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။

```
string = ("hellomyanmar")
print(string.islower())
```

C:\Program Files (x86)\Microsoft Visual Studio
True

ယခု program ကို run ကြည့်ရင် result အနေနဲ့ True ကိုရရှိမှာဖြစ်ပါတယ်။ ဘာကြောင့်လဲ ဆိုတော့ စာလုံးတွေ အားလုံး အသေးတွေဖြစ် နေလို့ပါ။

False Example

```
string = ("helloMyanmar")
print(string.islower())
```

C:\Program Files (x86)\Microsoft Visual Studio\Shared
False
Press any key to continue . . .

M ဆိုတဲ့ character ကို အကြီးပြောင်းလိုက်တဲ့ အတွက် result မှာ False ဆိုပြီး ရရှိခြင်းဖြစ်ပါတယ်။

count()

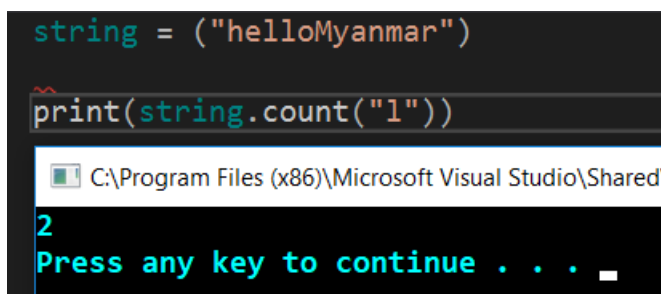
count() သည် python programming ၏ built-in function ဖြစ်ပြီး parameter အနေနဲ့ ထည့်ပေးလိုက်တဲ့ character ရဲ့ ဘယ်နှစ်လုံးရှိသလဲ ဆိုတဲ့ အရေအတွက် ကို return အနေနဲ့ ပြန်ပေးပါတယ်။

syntax

`string.count("မိမိထည့်လိုသော character ")`

parameter

မိမိ ရှာလိုသော character ကိုထည့်ပေးရပါတယ်။



```
string = ("helloMyanmar")
print(string.count("l"))
```

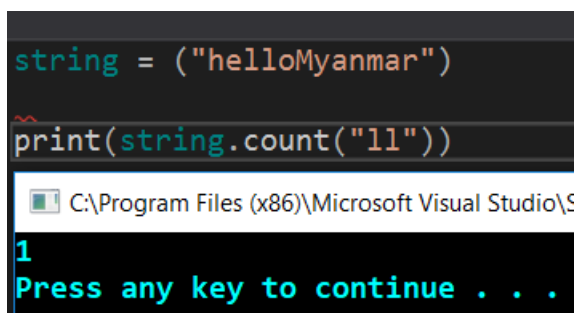
C:\Program Files (x86)\Microsoft Visual Studio\Shared

2

Press any key to continue . . .

ယခု program မှာဆိုရင် character l ရဲ့ ဘယ်နှစ်ကြိမ်ပါလဲ ဆိုတဲ့ အကြိမ် အရေအတွက်ကိုရှာထားပါတယ်။ result အနေနဲ့ 2 ကိုပြန်ပေးပါတယ် ဘာကြောင့်လဲဆိုတော့ string ထဲမှာ ll

ဆိုပြီး character နှစ်လုံးပါနေတဲ့ အတွက်ကြောင့်ဖြစ်ပါတယ်။



```
string = ("helloMyanmar")
print(string.count("ll"))
```

C:\Program Files (x86)\Microsoft Visual Studio\

1

Press any key to continue . . .

ယခု program ကတော့ character ll ဆိုတာကို ဘယ်နှစ်ခါ ပါလဲ စစ်ထားခြင်း ဖြစ်ပါတယ်။ ll သည် တစ်ခါတည်း ပါသည့် အတွက်ကြောင့် 1 ဆိုပြီး return value ပြန်ရခြင်းဖြစ်ပါတယ်။


```
string = ("helloMyanmar")
print(string.count("m"))
```

C:\Program Files (x86)\Microsoft Visual Studio\Shared'

1
Press any key to continue . . . _

character m ဘယ်နှစ်ကြိမ်ပါသလဲဆိုတာကိုရှာထားခြင်းဖြစ်ပါတယ်။

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။

```
string = ("helloMyanmar")
print(string.find("y"))
```

C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python

6
Press any key to continue . . . _

ယခု program 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,count)
```

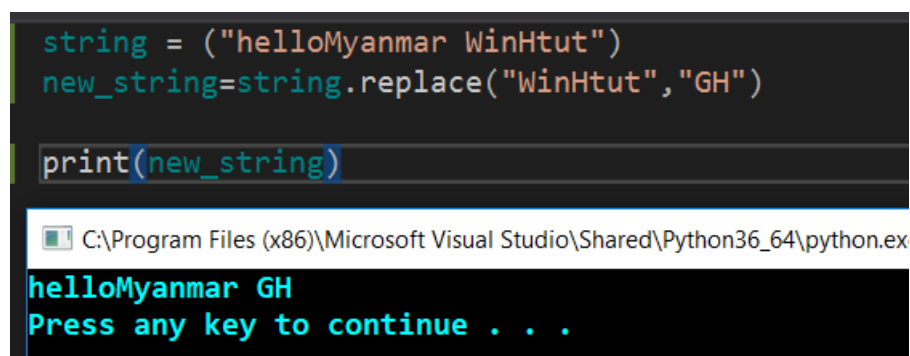
paramters:

old- အစားထိုးချင်တဲ့ စာသားကိုထည့်ပေးရပါတယ်။

new- အသစ်ပေါ်လာစေချင်တဲ့ စာသားကို ထည့်ပေးရပါတယ်။

return value:

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



```
string = ("helloMyanmar WinHtut")
new_string=string.replace("WinHtut","GH")
print(new_string)
```

C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python36_64\python.exe

```
helloMyanmar GH
Press any key to continue . . .
```

ရှေ့က lesson ကဲ့
သို့ string ကို
တိုက်ရိုက် print
ထုတ်ရင်လည်း
ရသလို ယခု ကဲ့သို့

variable အသစ်ထဲကို လည်း assign လုပ်ပြီး print ထုတ်လိုရပါတယ်။

List in Python

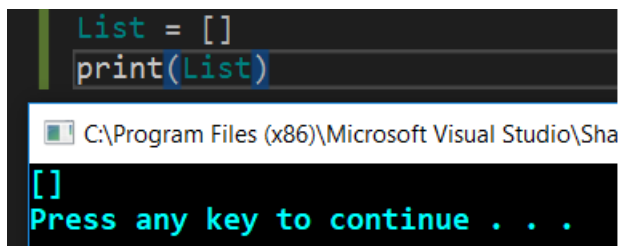
Python မှာဆိုရင် list က နေရာတော်တော်များများမှာ သုံးတဲ့ compound data type ဖြစ်ပါသည်။ Python ရဲ့ list ဟာ တစ်ခြား programming မှာဆိုရင် array နဲ့ ဆင်တူသော်လည်းပဲ python ရဲ့ list ကတော့ powerful tool တစ်ခုဖြစ်ပါတယ်။ List တစ်ခု တည်းမှာကိုပဲ data

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 ကိုကြေ ငြာလို့ရပါတယ်။



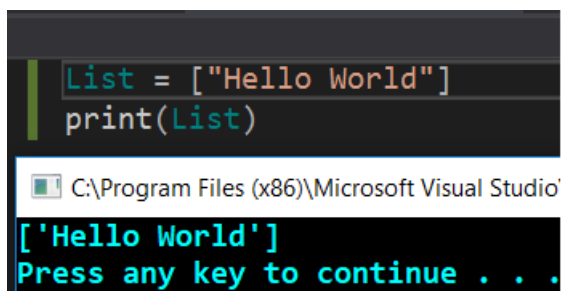
```
List = [ ]
print(List)
```

```
[ ]
Press any key to continue . . .
```

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။



```
List = ["Hello World"]
print(List)
```

```
['Hello World']
Press any key to continue . . .
```

result အနေနဲ့ ['Hello World'] ဆိုတာကို ရရှိမှာ ဖြစ်ပါတယ်။

Creating a list with Multiple Data

list တစ်ခုထဲသို့ data အများကြီး ထည့်လိုတဲ့အခါ , ခြားပြီး ထည့်ရုံပဲ။

```
List = ["Hello World", "Green", "Hackers"]
print(List)
```

C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python36_6

```
['Hello World', 'Green', 'Hackers']
Press any key to continue . . .
```

result အနေနဲ့ မိမိတို့

ထည့်လိုက်သော data သုံးခု ကို ပြန်ရမှာဖြစ်ပါတယ်။

Accessing data

List တစ်ခုထဲမှာ ရှိတဲ့ data တွေကို access(ရယူ) လိုတဲ့ အခါမျိုးမှာဆိုရင် ၎င်းတို့ရဲ့ index number တွေကို သုံးပြီးတော့ ရယူနိုင်ပါတယ်။

```
List = ["Hello World", "Green", "Hackers"]
print(List[0])
print(List[1])
print(List[2])
```

C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python36_

```
Hello World
Green
Hackers
Press any key to continue . . .
```

index တွေကိုရေတွက်တဲ့ အခါ

မှာ zero ကနေ စတင်ပြီးရေ

တွက်ရပါတယ်။ ယခု program

မှာဆိုရင် Hello World သည်

index zero ဖြစ်ပြီး ကျန်တဲ့ data

များသည်လည်း အစဉ်

လိုက်ဖြစ်ပါသည်။ ထို့ကြောင့် result အနေနဲ့ Hello World / Green / Hackers တို့ကို ရရှိခြင်းဖြစ်ပါသည်။

```
List=['green','hackers','winhtut']
```

```
print(List[-1])
```

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

Accessing Data From Multi-Dimensional list

Array မှာကဲ့သို့ List ကို multi-dimensional array ပုံစံဖြင့်လည်း အသုံးပြုလို့ ရပါသေး သည်။

multi-dimensional ကို နားလည်ဖို့ ဆိုလျှင် ပထမဦးစွာ row and column ကို နားလည်ရန်

လိုအပ်ပါသည်။ row and column ကို သေချာ ခွဲခြားနိုင်ရန် သင်ထောက်ကူပုံ တစ်ခု 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

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

အထက်ပါ Program ကို run ကြည့်လျှင် output အနေဖြင့် hackers and gh ဆိုတာကို ရပါမည်။ multi-dimensional list ကို ကြောငြာသော အခါတွင် [] square bracket များခွဲပြီး ကြောငြာပေးရပါမည် ။ ပထမ square bracket ကို 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])` print လုပ်လိုက်သော အချိန်တွင် 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 လုပ်သော အချိန်တွင်လည်း 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 ထုတ်မည်ဖြစ်ကြောင်းထည့်ပေးရပါမည်။ `remove()` method ကိုသုံးရာတွင် မိမိထုတ်ပြန်လိုသော name ကို အတိအကျ ရေးပေးပြီးထုတ်ပြန်နိုင်ပါသည်။ ဥပမာ list တစ်ခု ထဲတွင် element သုံးခု ရှိခဲ့လျှင် [1 ,2,3]ထို element သုံးခုထဲမှ ပထမဆုံး တစ်ခု ဖြစ်သည့် 1 ကို ထုတ်ပြန်လိုသော် `List.remove(1)` ဟု ရေးရပါမည်။ နောက် ထပ် ဥပမာ တစ်ခု အနေဖြင့် list တစ်ခု ရှိမည် ထို List ထဲတွင် ['win','htut','gh'] စသည် တို့ ရှိမည်။ ထို list ထဲမှ gh ဆိုသည့် စာသားကို ထုတ်ပြန်လိုလျှင် `list.remove('gh')` ဟုရေးပေးရမည်ဖြစ်ပါသည်။ sample program ကို အောက်တွင်ရေးပြထားပါသည်။

```
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)
```

အထက်ပါ program အား run ဂြာည့်လျှင် အောက်ပါ အတိုင်း မြင်တွေ့ရပါမည်။

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 ဒုတိယ အကြိမ်တွင် ထို list ထဲမှ 3 ကို ထုတ်လိုက်ပါသည်။ ထို့ကြောင့် ဒုတိယ အကြိမ် print လုပ်သော အခါတွင် list ထဲမှ 3 ပျောက်သွားခြင်းဖြစ်သည်။ remove() method သည် index number နှင့် အလုပ်လုပ်ခြင်း မဟုတ်ပဲ parameter အဖြစ် ထည့်ပေးလိုက်သည့် အတိုင်း အတိအကျ အလုပ်လုပ်ခြင်းဖြစ်သည်။ အောက်ဆုံးတွင် win ဆိုသည့် စာသားကို list ထဲမှ ထုတ်ထားပါသည်။ ထို့ကြောင့် နောက်ဆုံး output တွင် win မပါ လာခြင်းဖြစ်သည်။

pop() Method Popping elements from list

Pop method() ကိုသုံးရာတွင် parameter အနေဖြင့် index number ထည့်ပေးရမည် ဖြစ်ပြီး ထို parameter အတိုင်း list ထဲမှ ဖြတ်ထုတ်ပါသည်။ 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 ဖြတ်ထုတ်ပါသည်။

Syntax :

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

Example Program

```
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)
```

အထက်ပါ program အား run ကြည့်လျှင် output အနေဖြင့် အောက်ပါ အတိုင်းမြင်တွေ့ ရပါမည်။

အထူးသတိပြုရန်မှာ pop သည် index number ဖြင့် အလုပ်လုပ်သောကြောင့် pop တစ်ခါ လုပ်ပြီးတိုင်း index number များပြောင်းလဲသွားမည်ဖြစ်သည်။

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

```
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)
```

အထက်ပါ program အား run ကြည့်လျှင် output အနေဖြင့် 6 ကိုရရှိပါမည်။

Error

`list` ထဲတွင်မပါရှိသော `elements` များအား `index` method ထဲတွင် `argument` အဖြစ် ထည့်သုံးသော အခါတွင်မူ `value error` ကို ရရှိပါမည်။

Sample Error Program

```
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'))
```

အထက်ပါ program အား run ကြည့်လျှင် output အနေဖြင့် အောက်ပါ အတိုင်း `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 များ ထပ်ထည့်ရန် အသုံးပြုပါသည်။ append နောက်တွင် ထည့်ပေးလိုက်သော element ကို list elements များရဲ့ နောက်ဆုံးနေရာတွင် သွားရောက်ထည့်ပေးပါသည်။ list ထဲကို elements များသာ မက list များပါ ထပ်ထည့်နိုင်ပါသေးသည်။ append method သည် original list ကို သာ modifies လုပ်ပေးပြီး မည့်သည့် return value မျှ ပြန်ပေးမည် မဟုတ်ပါ ။

sample program

#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 နံမည်အား my_list ဟု နံမည် ပေးပါမည်။ထို my_list list ထဲတွင် elements အချို့ ထည့်ထားပါမည်။ ထို့နောက် my_list ထဲအား ထပ်ထည့်ရန် list တစ်ခု ထပ်ဆောက်ပါ မည် ထို list နံမည်ကို new_list ဟု နံမည် ပေးလိုက်ပါမည်။ ထို new_list ထဲတွင်လည်း elements အချို့ထည့်ထား ပါမည်။ ပြီးလျှင် append method ကိုသုံးပြီး list တစ်ခုမှ အခြား list တစ်ခုသို့ ပေါင်းထည့်ရန် my_list.append(new_list) ဟုရေးပါမည်။

sample program

#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)
```

အထက်ပါ program အား run ပြီးလျှင် output အနေဖြင့် အောက်ပါအတိုင်း list နှစ်ခု ပေါင်းသွားသည်ကို တွေ့ရပါမည်။

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 ထပ်ထည့်ခြင်းဖြစ်သည်။ 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 မည့်သည့် return value မှ ပြန်မည်မဟုတ်ပါ။ list များကို extend လုပ်ရာတွင် operator + or += များကိုလည်း အသုံးပြုနိုင်ပါသေးသည်။

Sample program

#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)
```

အထက်ပါ program အား 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 ကိုသာ သုံးပြီးလည်း အောက်ပါ အတိုင်း list များကို extending ပြုလုပ်နိုင်ပါသေးသည်။

```
#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)
```

ယခု program အား run ကြည့်လျှင်လည်း extend() method မှာကဲ့သို့ output တူနေမည် ဖြစ်ပါသည်။

insert() method

list တစ်ခုထဲမှာ ရှိတဲ့ elements တွေထဲကို အခြားသော elements များ ထပ်ပေါင်းထည့်လို သော အခါမျိုးမှာ 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 လုပ်ဖြစ်ခြင်း ဖြစ်ပြီး မည်သည့် return value မျှ ပြန်ပေးမည်မဟုတ်ပါ။ insert သည် index number ဖြင့် အလုပ်လုပ် သောကြောင့် မိမိတို့

အနေဖြင့် 5 ခု မြောက်နေရာမှာ element insert လုပ်လိုလျှင် argument အနေဖြင့် 4 ကို ထည့်ပေးရပါမည်။ အဘယ်ကြောင့်ဆိုသော် python index number သည် zero ကနေ စပါသည်(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 နောက်ဆုံး နေရာများတွင်သာ အစဉ်လိုက် နေရာယူသည် မိမိ ထည့်လိုသော နေရာသို့ ရောက်မည် မဟုတ်ပါ။

Sample program inserting an element to list

#declaring and initializing a list

```
my_list=[1,2,3,4,5,6]
```

#printing original list

```
print('original list',my_list)
```

#inseting an element to list

```
my_list.insert(4,'win')
```

#printing elements after modifies

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

အထက်ပါ program အား run ငြာည့်လျှင် 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။

#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)
```

ယခု program အား run ငြာည့်လျှင် 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။ပထမဆုံး အနေဖြင့် list တစ်ခု တည်ဆောက်ပါမည်။ထို list ထဲတွင် 1,2,3,4,5,6,1 စသည့် elements များထည့်ထားပါမည်။

sample program

#declaring and initializing a list

```
my_list=[1,2,3,4,5,6,1]
times=my_list.count(1)
print(times)
```

အထက်ပါ program တွင် my_list.count(1) သည် my_list ဆိုသည့် list ထဲမှာ 1 ဆိုသည့် element အကြိမ်မည်မျှ ပါဝင်သည်ကို ရေတွက်ရန်ဖြစ်သည်။ အထက်ပါ program အား run ဖြည့်လျှင် 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

#declaring and initializing a list

```
my_list=[1,2,3,4,5,6,1]
my_list.reverse()
print(my_list)
```

အထက်ပါ program အား run ဖြည့်လျှင် output အနေဖြင့် နဂိုမူရင်းရှိသော original list ထဲမှ elements များအား ပြောင်းပြန် လုပ်ထားသည်ကို မြင်ရပါမည်။

[1, 6, 5, 4, 3, 2, 1]

sort() method

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

Sample Program

#declaring and initializing a list

```
my_list=[1,2,3,4,5,6,1,0]
my_list.sort()
print(my_list)
```

ယခု program အား 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။

#declaring and initializing a list

```
my_list=['u','i','e','a','o']
my_list.sort()
print(my_list)
```

copy() method

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

Sample Program

#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)
```

အထက်ပါ program အား run ဂြာည့်လျှင် အောက်ပါအတိုင်း output များကို မြင်တွေ့ရပါမည်။ new_list ထဲသို့ append လုပ်ထားသည့် elements ကိုလည်း မြင်တွေ့ရပါမည်။

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

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

clear() method

clear() method သည် list ထဲမှ ရှိသော elements များအားလုံးကို ဖျက်ပြစ်ရန် အတွက် အသုံးပြုပါသည်။ clear() method သည် မည်သည့် return value မှပြန်မည်မဟုတ်ပါ။ clear() method ကို အသုံး မပြုနိုင်သည့် အချိန်မျိုးတွင်မူ del operator ကိုလည်း အသုံးပြုနိုင်ပါသည်။

Sample Program

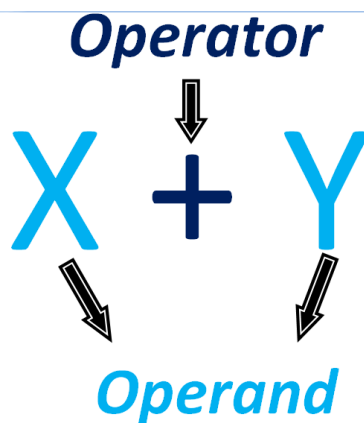
#declaring and initializing a list

```
my_list=['u','i','e','a','o']
my_list.clear()
print(my_list)
```

အထက်ပါ program အား run ဂြာည့်လျှင် output အနေဖြင့် square brackets[] ကိုသာရရှိမည်။ အဘယ်ကြောင့်ဆိုသော် list ထဲရှိ elements များအားလုံးကို clear လုပ်ပြစ်သောကြောင့် ဖြစ်သည်။

Operators

အခြေခံ အားဖြင့် ပေါင်းခြင်း နှုတ်ခြင်း 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 generally အားဖြင့် 8 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 တွေလုပ်ဆောင်ဖို့ အတွက် အသုံးပြုပါတယ် ဥပမာ ပေါင်းခြင်း နှုတ်ခြင်း စားခြင်း မြှောက်ခြင်း တို့ဖြစ်ပါတယ်။ Python programming မှာ arithmetic operators တွေကို အခြေခံ အားဖြင့် 7 မျိုးခွဲခြားနိုင်ပါတယ်။

1. Add (ပေါင်းပေးရာတွင် အသုံးပြုပါတယ်။ operands နှစ်ခု ဖြစ်နိုင်သလို unary plus တစ်ခုတည်းလည်း ဖြစ်နိုင်ပါတယ်)
2. Subtract (နှုတ်ပေးရာတွင် အသုံးပြုပါတယ်။ operands နှစ်ခုဖြစ်နိုင်သလို unary minus တစ်ခုတည်းလည်း ဖြစ်နိုင်ပါတယ်)
3. Multiply (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 အချိန်မျှ အသုံးပြုပါတယ် left operand ကို right 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 အနေဖြင့် ထားပြီး မြှောက်တာပါ 2**3 ဆိုလျှင် 2 power 3 ဆိုတာမျိုးပါ)

Sample program ကို output နှင့်တကွ ဖော်ပြထားပါသည်။

```
a = 13
b = 4
print('a + b =',a+b)
#add output a + b = 17
print('a - b =',a-b)
#subtract output 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 6 မျိုးခွဲခြား ထားနိုင်ပါသည်။ relational operator များကို operand များအား နှိုင်းယှဉ်လို သည့် အခါမျိုးတွင် အသုံးပြုပါသည်။

1. Greater that { $x > y$ } (left operand ဖြစ်သည့် x သည် right operand ဖြစ်သည့် y ထက် ကြီးလျှင် မှန်သည်)
2. Less that { $x < y$ } (left operand ဖြစ်သည့် x သည် right operand ဖြစ်သည့် y ထက် ငယ်လျှင် မှန်သည်)
3. Equal to { $x == y$ } (left operand and right operand နှစ်ခုလုံးဟာ တူညီနေလျှင်မှန်သည်)
4. Not equal to { $x != y$ } (left operand and right operand 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။

```
a = 13
b = 4
print('a > b =',a > b)
```

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

Logical Operators

Python programming မှာ logical operator သုံးမျိုးရှိပါတယ် ။ သူတို့ သုံးမျိုးစလုံးသည် အခြားသော programming တော်တော်များများတွင် symbols များကို အသုံးပြုကြပြီး python တွင်မူ စားသားများဖြင့် 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။ဘေး တစ်ဘက်စီတွင် ရှိသော conditions နှစ်ခုလုံးမှန်မှ သာလျှင် output ကို ထုတ်ပေးပါသည်။

```
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။

```
a=10
b=9
c=13
if a > b or a > c:
    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 မ မှန်ဘူးဟု ဆိုလိုခြင်းဖြစ်ပြီး output အနေဖြင့် မ မှန်ဘူးဆိုသည့် အတိုင်း false ထွက်လာမှာဖြစ်ပါတယ်။ condition သည် မှားနေလျှင် not false မ မှားဘူးဟု ဆိုလိုခြင်း ဖြစ်ပြီး output အနေဖြင့် true ထွက်လာမှာ ဖြစ်ပါတယ်။

Sample program

```
x=True
print('not x is',not x)
# not logical operator Output not 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 ဖော်ပြထားသော ပုံကို ကြည့်ခြင်းအားဖြင့် အလွယ်တကူ သဘော ပေါက်နိုင်ပါတယ်။

AND

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

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

sample program 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 သူရဲ့ binary value မှာ 8 bits ဖြင့်ကြည့်လျှင် 0011 1100 ဖြစ်သည်။ ထိုနည်းတူ b value ဖြစ်သည့် 13 ကို binary value 8 bits ဖြင့်ကြည့်လျှင် 0000 1101 ဖြစ်သည်။

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 ပြန်ရ ရှိခြင်း 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

bitwise or (|) sample program

```
a = 60          # 60 = 0011 1100
b = 13          # 13 = 0000 1101
c = 0
c = a | b;      # 61 = 0011 1101
print (" Value of c is ", c)
```

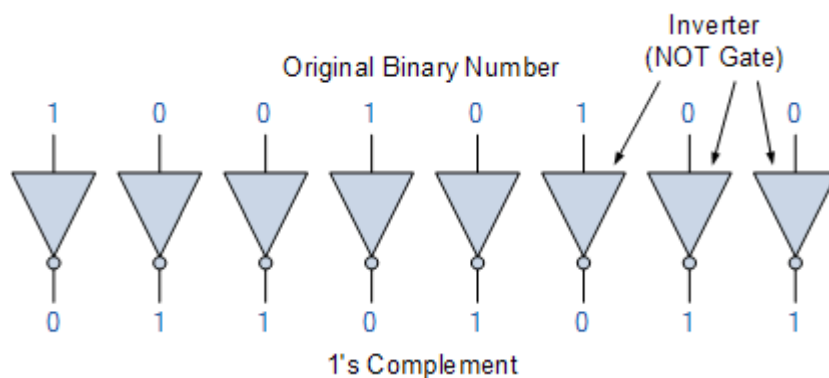
အထက်ပါ program အား run ကြည့်လျှင် output အနေဖြင့် 61 ကို ပြန်ပေးပါမည်။

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 NOT (~)

bitwise not operator သည် unary operator အမျိုးအစား ဖြစ်သည်။ ($\sim x$) unary operator ဆိုသည်မှာ operand တစ်ခုတည်း ပါရှိသော operator ကို ဆိုလိုခြင်းဖြစ်သည်။ Bitwise NOT ကို binary 1's complement လုပ်တယ်လို့လည်း ခေါ်ပါတယ်။ 1's complement လုပ်ခြင်းဆိုသည်မှာ binary value များကို ပြောင်းပြန်လှန် invert လုပ်ခြင်းနှင့် တူညီသည်။ $x = 10$ ဟု ထားမည်ဆိုလျှင် 8 bit အနေဖြင့် 0000 1010 ရှိမည် decimal အနေဖြင့် -11 ပြန်ရပါမည်။ ထို 0000 1010 အား bitwise not (ones complement) ပြုလုပ်လျှင် 1111 0101 ပြန်လည်ရရှိပါမည်။ 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

```
a=10
c = ~a;
print (" Value of c is ", c)
# Value of c is  -11
```

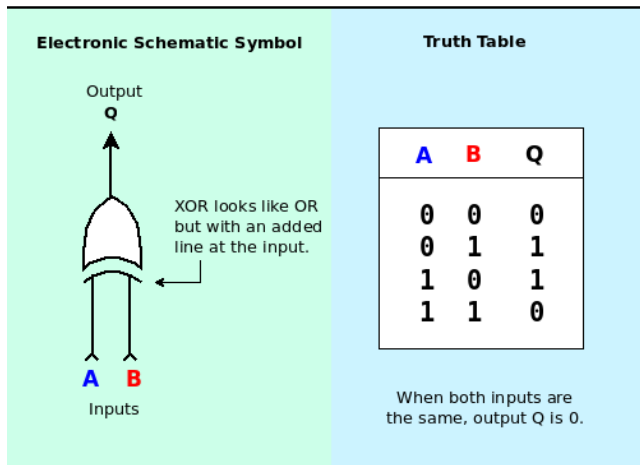
Bitwise XOR (^)

Bitwise XOR (^) သည် တူရင် 1 and 1 ဆိုလျှင် 0 output ပေးပြီး မတူရင် 0 and 1 ဆိုလျှင် 1 output ပေးပါသည်။ သို့သော် bitwise xor သည် 0 and 0 ဆိုလျှင်တော့ 0 သာ

output ပေးပါသည်။ ဆိုလိုသည်မှာ input value တွေမှာ data မရှိသော အခါတွင် 0 ကို output ပြန်ပေးခြင်းဖြစ်သည်။ ဥပမာ အနေဖြင့် $a=5$ (0000 0101) နှင့် $b=3$ (0000 0011) တို့အား XOR လုပ်ကြည့်လျှင် output အနေဖြင့် 0000 0110 ပြန် ရပါမည် decimal value အနေဖြင့် 6 ကို ပြန်ရပါမည်။

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

XOR



Sample program

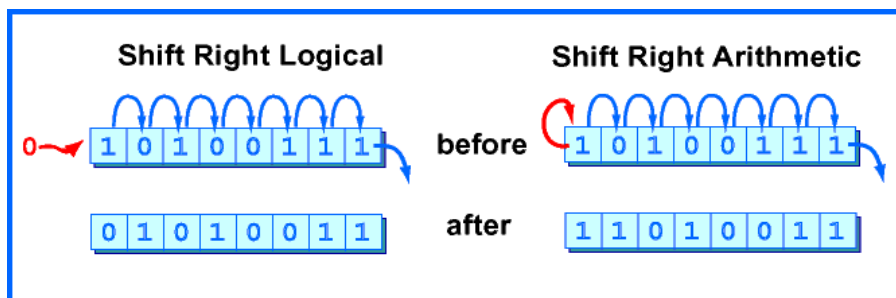
```
a=5
b=3
c = a ^ b;
print ("Value of c is ", c)
#Value of c is 6
```

Bitwise right shift (>>)

Bitwise right shift operator သည် binary value များကို bit အလိုက် ကိုင်တွယ်ရာတွင် အရမ်းကို အသုံးဝင်ပါသည်။ variable တစ်ခုကို $x=10$ (0000 1010) အဖြစ်ထားပါမည်။ $y=x>>2$ ဟုရေးမည် 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) ညဘက်မှ bit 2 လုံး ဖြုတ်မှာ ဖြစ်သည့်အတွက် 0000 10 သာ ကျန်မည် 8 bit အနေဖြင့် ကြည့်လျှင် 0000 0010 ဟု သိနိုင်သည်။ ထို့ကြောင့် decimal value ဖြင့် ကြည့်လျှင် 2 ကို ရ ရှိခြင်းဖြစ်သည်။ အထက်ပါ ဖော်ပြ ထားချက်များသည် right shift 2 လုပ်ခင်းဖြစ်သည် ။ right shift တစ်လုံး ထုတ်ပုံကို အောက်တွင် ဖော်ပြထားသည်။



Python Sample Program

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

အထက်ပါ program 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 ဖြုတ်ပြစ်ရမည် ဖြစ်သည်။ ထို့ကြောင့် binary value အနေဖြင့် 0011 11 decimal အနေဖြင့် 15 ရ ရှိခြင်းဖြစ်သည်။

3 bit right shift

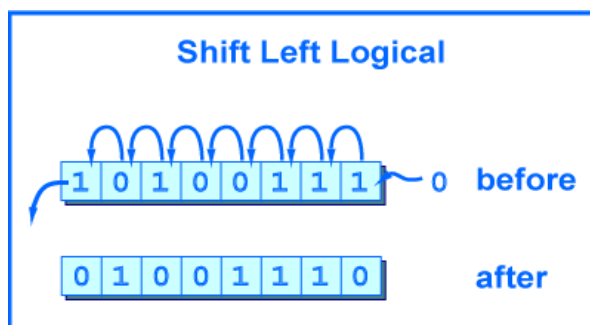
```
a=60
c = a >> 3;
print (" Value of c is ", c)
```

အထက်ပါ program အား run ကြည့်လျှင်လည်း decimal အနေဖြင့် 7 ကို ရရှိပါမည်။ အဘယ်ကြောင့်ဆိုသော် 60 ၏ binary value သည် 0011 1100 ဖြစ်ပြီး ထို value အား 3 bit right shift လုပ်လျှင် 0011 1100 ညဘက်မှ bit သုံးခုကို ဖြုတ်ပြစ်ရမည် ဖြစ်သည်။ binary value အနေဖြင့် 0011 1 ကျန်ရှိမည်ဖြစ်ပြီး decimal အနေဖြင့် 7 ရရှိခြင်းဖြစ်ပါသည်။

Bitwise Left (<<)

Bitwise left သည် ဘယ်ဘက်မှ bit များအား ဖြုတ်ထုတ်လိုက်ခြင်း ဖြစ်ပြီး တစ်နည်းအားဖြင့် 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 shift 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 သည် ညာဘက်မှ 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

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

အထက်ပါ program အား 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 အားဖြင့် 240 ဖြစ်သည်။

Assignment operators

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

Assignment operator ကို အခြားသော operator တွေနဲ့ ပေါင်းပြီး သုံးလိုလည်း ရပါသေးတယ် ၎င်းတို့ကို compound operator လို့ခေါ်ပါတယ်။

$a += 5$ သည် $a = a + 5$ ကိုသာပင် ဆိုလိုခြင်းဖြစ်သည်။ ထိုနည်းတူ $a -= 5$ သည် လည်း $a = a - 5$ သာပင်ဖြစ်သည်။ Python programming တွင် compound operator များစွာ ကို 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 နီးပါမတွေ့ရတဲ့ english letter ရေးသလို operators တွေ ပါ နေပါသေးတယ် identity operator and membership operator လို့ ခေါ်ပါသေးတယ်။

Identity operators

Python programming မှာ is and 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 တူသည့် အတွက် address အတူတူ ပင်ဖြစ်သည် အဘယ်ကြောင့်ဆိုသော် python programming သည် အခြား သော 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။

```
a=10
```

```
b=10
```

```
c=10
print(id(a),id(b),id(c))
```

အထက်ပါ အတိုင်း program ရေးပြီး စမ်းကြည့်လျှင် output အနေဖြင့် memory address များ အတူတူ ထွက်လာသည်ကို မြင်ရပါမည်။ Identity operator သည် မှန်လျှင် true ပြန်ပေးပြီး များလျှင်တော့ 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။

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

Sample program for is not

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 အား run ကြည့်လျှင် false ရပါမည်။

Example 1

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

Example 2

```
a=10
```

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

ယခု program အား run ကြည့်လျှင် true ရပါမည် အဘယ်ကြောင့်ဆိုသော် a နှင့် b သည် မတူတော့သောကြောင့် memory address များလည် မတူတော့သည့်အတွက် a နှင့် b သည် မတူဘူးဟု ဆိုလျှင် true ဆိုသည့် output ကို ပြန်ရခြင်းဖြစ်သည်။ ၎င်းတို့ address များအား အောက်ပါ အတိုင်း id() method ကိုသုံးပြီး စစ်ဆေးနိုင်ပါသည်။

```
a=10
b=20
c= a is not b
print(c)
print('a =',id(a), 'b=',id(b))
#output
#True
#a = 1657627968 b= 1657628128
```

အထက်ပါ program အား run ကြည့်လျှင် output အနေဖြင့် memory address များ မတူသည်ကို မြင်တွေ့နိုင်ပါသည်။

Dive to String

String များကိုလည်း အောက်ပါ အတိုင်း 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။

```
string1='hello'
string2='hello'
print(id(string1),id(string2))

string3=string1 is string2
print(string3)
```

```
if string1 == string2:
    print('They are same')
```

```
#output
#45938560 45938560
#True
#They are same
```

Dive To List

List များသည် string များနှင့် လုံးဝ မတူပါ။ 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 အဘယ်ကြောင့် ဆိုသော သူတို့ သိမ်းဆည်းသော memory address များ မတူညီခြင်းကြောင့်ဖြစ်သည်။ sample program ကို အောက်တွင် identity operator နှင့် relational operator နှစ်ခုလုံးနှင့်ပါ ကွဲပြားစွာ ရေးပြထားပါသည်။ output များကို စစ်ဆေးလျှင် memory address များ မတူညီသည်ကို မြင်ရပါမည်။

```
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
```


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 မိမိရှာလိုသော sequence ထဲမှာ မရှိလျှင် True ကို ပြန်ထုတ်ပေးပြီး ရှိနေလျှင် false ကို ပြန်ထုတ်ပေးပါသည်။လေ့ကျင့်ရန် example program အား အောက်တွင် ဖော်ပြထားပြီး output နှင့်ပါ ယှဉ်တွဲပြထားပါသည်။

Sample Program

```
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
```

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။

1 Example program

```
a = 33
b = 200
if b > a:
    print("b is greater than a")
```

b သည် a ထက်ကြီးခဲ့ရင် b is greater than a ဆိုသည့် စာသားကို ဖော်ပြ ပေးပါ ဟု ရေးသားထားခြင်းဖြစ်ပါသည်။

Indentation Error

```
a = 33
b = 200
if b > a:
print("b is greater than a")
```

အထက်ပါ code 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 အောက် တည့်တည့်တွင် 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 လို့ သုံးကြပါတယ်။ programming language တော်တော်များများနှင့် ယခု python programming မှာလည်းအသုံးပြုပုံ မှာ အတူတူပင်ဖြစ်သည်။ elif သည် သူ့ရှေ့မှာရှိသော condition တစ်ခုကို အရင် စစ်ဆေးပြီးမှ ထို condition ပြီးမှသာ elif ကို လာစစ်ပါသည်။ထို့ကြောင့် elif ကို second option အနေဖြင့် အသုံးပြုပါ သည်။

1 Example program

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

ယခု ကဲ့သို့ condition နှစ်ခုကို တစ်ခုပြီး တစ်ခု စစ်ဆေးလိုသော အခါ မျိုးမှာ အသုံးပြုသလို နှစ်ခုထက် များသော condition များကို စစ်ဆေးလိုသော အခါ မျိုးများတွင်လည်း အသုံးပြုပါသည်။

2 Example program

အောက်ပါ အတိုင်းလည်း elif ကို နှစ်ခုသုံးပြီး ရေးသားနိုင်ပါသည်။

```
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')
```

else statement

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

```
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")
```

အထက်ပါ program 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။

```
i = 1
while i < 6:
    print(i)
    i += 1
```

while loop with the break statement

while loop သည် condition မ မှန်တော့သည့် အချိန်မှာလည်း အဆုံး သတ်နိုင်သလို break statement ဖြင့်လည်း အဆုံး သတ်နိုင်ပါသည်။

```
i = 1
while i < 6:
```

```
print(i)
if i == 3:
    break
i += 1
```

For loops

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

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    print(x)
```

အထက်ပါ program 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

```
for x in "banana":
    print(x)
```

banana ဆိုတဲ့ string ထဲမှာ ရှိတဲ့ စာလုံးများကို တစ်လုံးခြင်းစီ ထုတ်ခြင်းဖြစ်ပါတယ်။

For loops with break statement

ယခု program တွင် list တစ်ခု တည်ဆောက်မည်ဖြစ်ပြီး ထို list ထဲတွင် မိမိ ထည့်ချင်သော 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။

```
fruits = ["aung", "maung", "winhtut", "greenhackers"]
for x in fruits:
```

```
print(x)
if x == "winhtut":
    break
```

For loops with continue statement

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

Introduction to Function

Function များတွင် standard library function and programmer defined function ဟု နှစ်မျိုးရှိပါသည်။ Standard library function သည် မူရင်း ပါဝင်သော 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

Range() function သည် return value အနေဖြင့် zero မှစပြီး number များကို အစဉ်လိုက် တစ်ပေါင်းပြီး ထုတ်ပေးပါသည်။ ထို function ထဲတွင် arguments အနေဖြင့် ထည့်ပေးလိုက်သော number တစ်ခုကို မရောက်ခင် အထိ number များကို sequence အလိုက် ထုတ်ပေးပါသည်။

```
for x in range(10):
    print(x)
```

ယခု program ကို run ကြည့်လျှင် 0 မှစပြီး 9 ထိ number များကို အစဉ်လိုက် မြင်တွေ့ ရပါမည်။

```
for x in range(2, 6):
```

```
print(x)
```

range() function ထဲတွင် အထက်ပါ အတိုင်း arguments နှစ်ခုလည်း ထည့်နိုင်ပါသည်။ ပထမ တစ်ခု သည် starting point ဖြစ်ပြီး နောက် တစ်ခုသည် ending point ဖြစ်သည်။ program အား run ပြုလုပ်လျှင် 2 မှာ စပြီး 5 ထိ အစဉ်လိုက် ပေါ်နေသည်ကို မြင်တွေ့ နိုင်ပါသည်။