Programming with Python: Introduction for Beginners

Contents Pages

Python programming language	2
Setting Up for development Environment	3
Number	8
Variable	9
String Operation	10
Simple Input and Output	12
Type Conversion	12
Comment	14
Operators	15
Arithmetic Operators	15
Assignment Operators	19
Increment and Decrement Operator	19
Prefix and Postfix	20
Combing Assignment and Arithmetic Operators	22
Relational Operator	23
Logical Operator	24
If. . . Else	25
Range	26
Looping	38
Array	33
List	34
Dictionary	38
File Handling	40
Exception	51

Python programming language မိတ်ဆက်

Python programming language ဟာ beginner တွေအတွက် programming ကို စတင်သင်ယူဖို့ အကောင်းဆုံး language တစ်ခုဘဲ ဖြစ်ပါတယ်။Computer Science နယ်ပယ်ထဲကမဟုတ်ဘဲ ပြင်ပကနေ programming ကို စတင်သင်ယူမဲ့ သူတွေအတွက်လည်း အရိုးရှင်းဆုံး အလွယ်ကူဆုံး စွမ်းရည်အမြင့်ဆုံး languageတစ်ခုလည်း ဖြစ်ပါတယ်။စွယ်စုံသုံး languageတစ်ခုဟုပင် သတ်မှတ်နိုင်လောက်အောင် စွမ်းရည် မြင့်မားလှပါတယ်။ Python ဟာ အလွန်လွယ်ကူတဲ့ ဘာသာစကားတစ်ခုပါ။စက်တွေကိုခိုင်းတဲ့ဘာသာစကားထဲက တစ်ခုပေါ့။မြန်မာစာ၊ English စာတွေလိုပါပဲ စက်ကိုခိုင်းဖို့ python language , Java language စတဲ့ ရေးပုံရေးနည်းတွေရှိပါတယ်။

လေ့လာတဲ့အခါမှာလည်း ဘာသာစကားလိုပဲ လေ့လာရပါတယ်။ English စာကျွမ်းဖို့ English လို အမြဲလေ့ကျင့်ဖို့လိုအပ်သလိုပဲ python ကိုကျွမ်းချင်ရင် အမြဲရေးနေဖို့လိုအပ်ပါတယ်။

Python Language ကို 1991 ခုနှစ်မှာ စတင်ပြီး တီထွင်ခဲ့သူကတော့ Duch လူမျိုး Netherlands နိုင်ငံသား တစ်ယောက်ဖြစ်တဲ့ Guido Van Rossum ဘဲ ဖြစ်ပါတယ်။သူကတော့ Master of mathematics and computer science ဘွဲ့ကို University of Amsterdam မှ 1982 ခုနှစ်မှာ ရရှိခဲ့ပါတယ်။

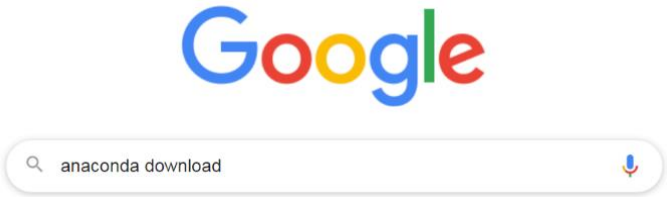
2000ခုနှစ်မှာ Python version 2.0 2008ခုနှစ်မှာ version 3.0 မှ အဆင့်ဆင့် update ပြုလုပ်ခဲ့ရာမှ ယခုအခါ version 3.8 အထိ ရောက်ရှိခဲ့ပြီ ဖြစ်ပါတယ်။

Setting Up for development Environment

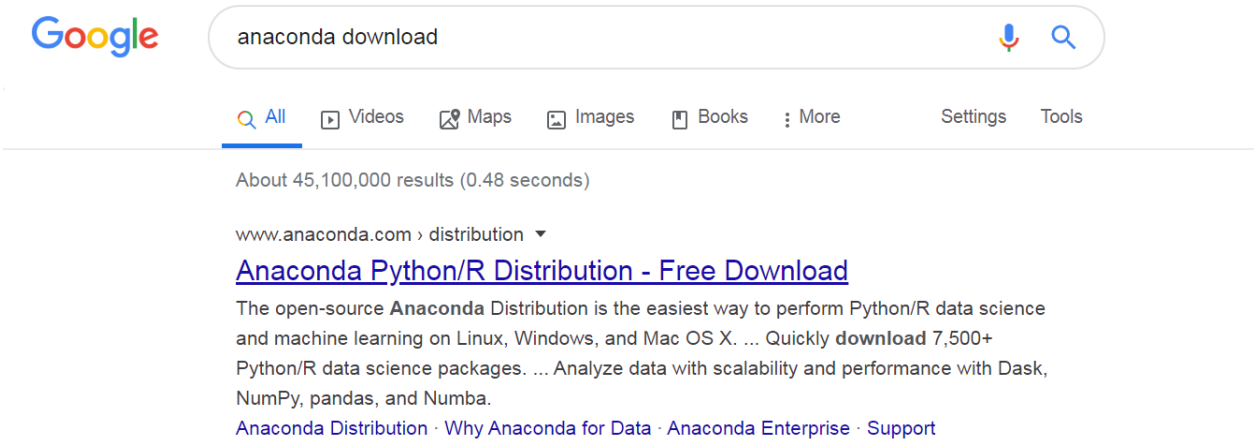
Python ကို run လို့ရတဲ့ environment များစွာရှိပါတယ်။ထိုကဲ့သို့များစွာရှိသည့်အထဲမှ Jupiter Notebook ကို စမ်းပြထားပါတယ်။

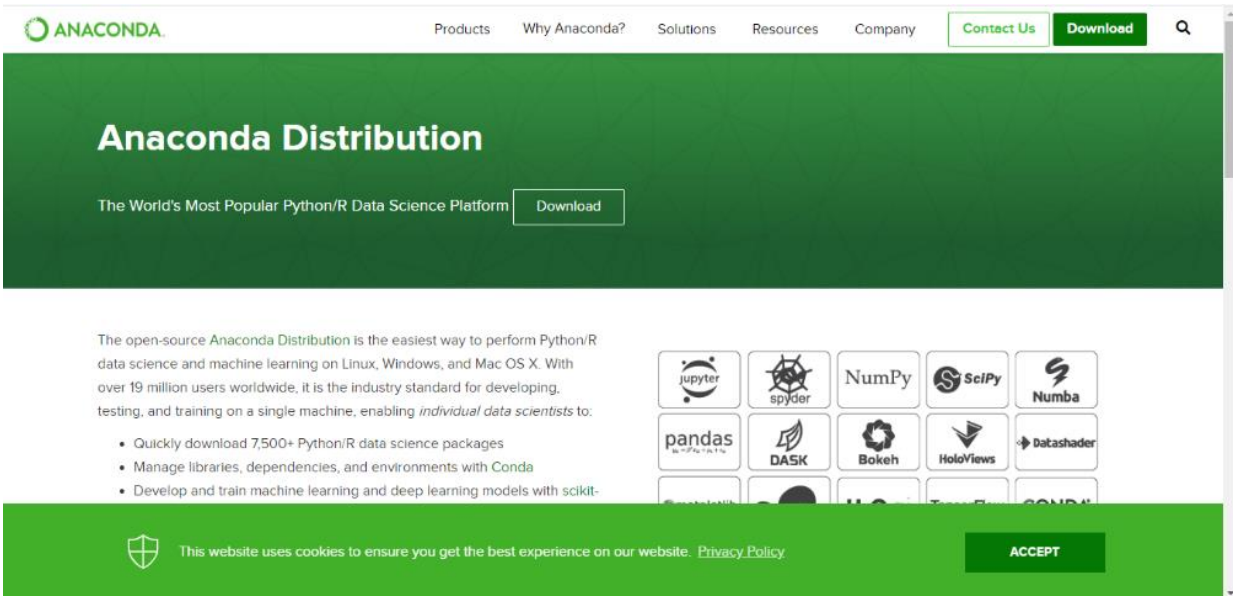
Install ပြုလုပ်ပုံ အဆင့်ဆင့်ကို ဖော်ပြပေးလိုက်ပါတယ်

*ပထမဦးစွာ google browser မှ download anaconda ကို ရိုက်၍ရှာပါ။

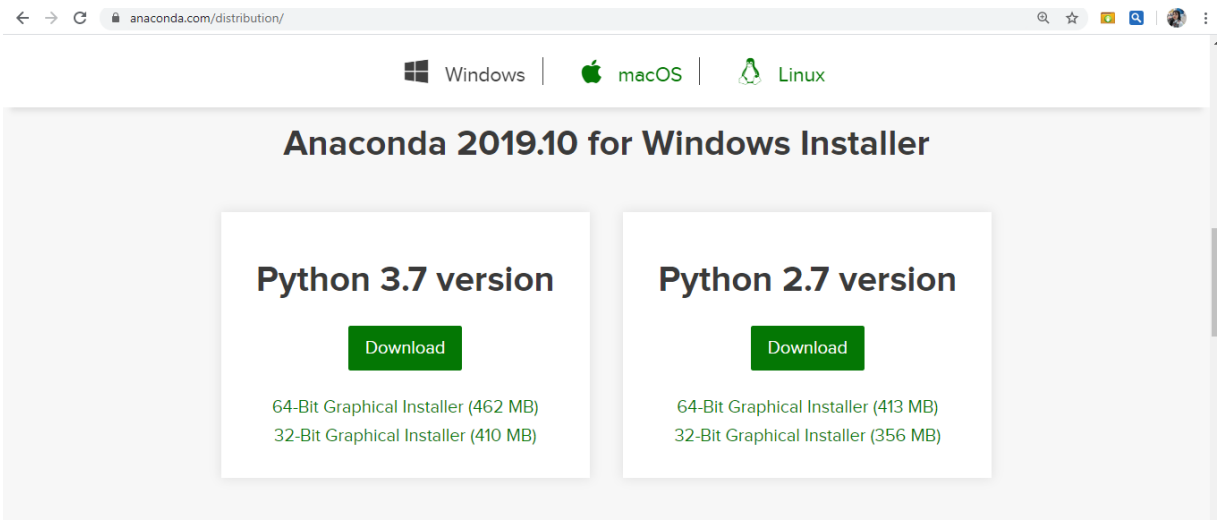


*ပြီးလျင် မြင်ရသည့်အတိုင်း Anaconda Website ထဲသို့ ဝင်ပါ။

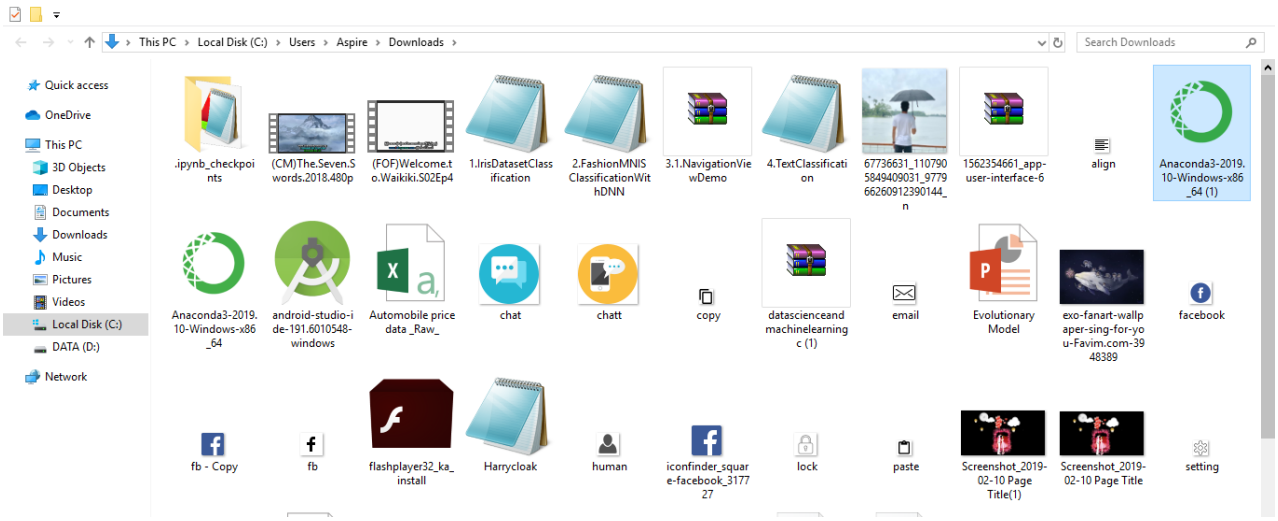




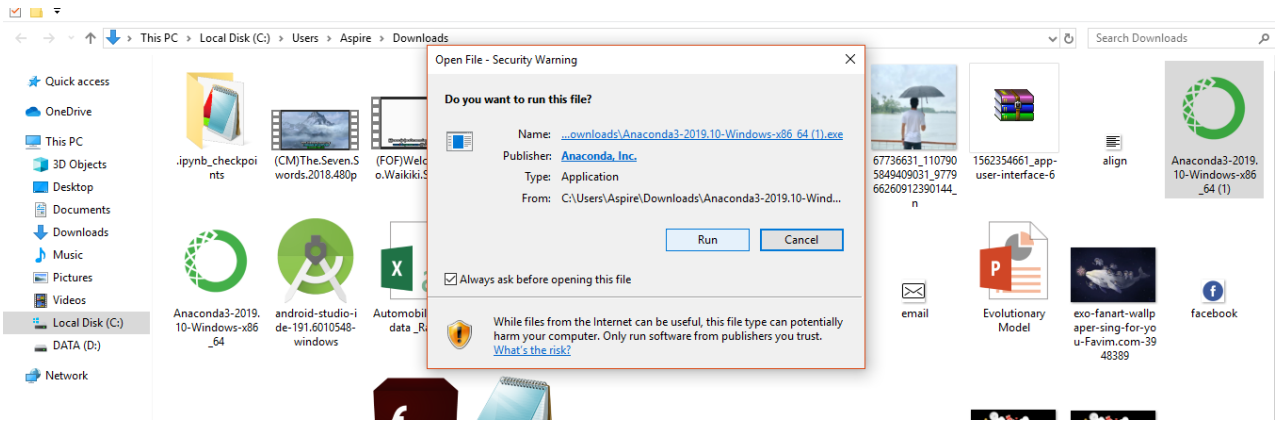
*python 3.7 version ကိုရွေးပါ။ Download ပြုလုပ်နေချိန် ခဏစောင့်ဆိုင်းပေးပါ။



*Download ပြုလုပ်ပြီးပါက မိမိ Computer ထဲရှိ Local disk / C / Users /Aspire / Downloads အောက်သို့သွား၍Anaconda file ကိုရှာပါ။ တွေ့ရှိပါက double click နှိပ်၍ install ပြုလုပ်ပါ။

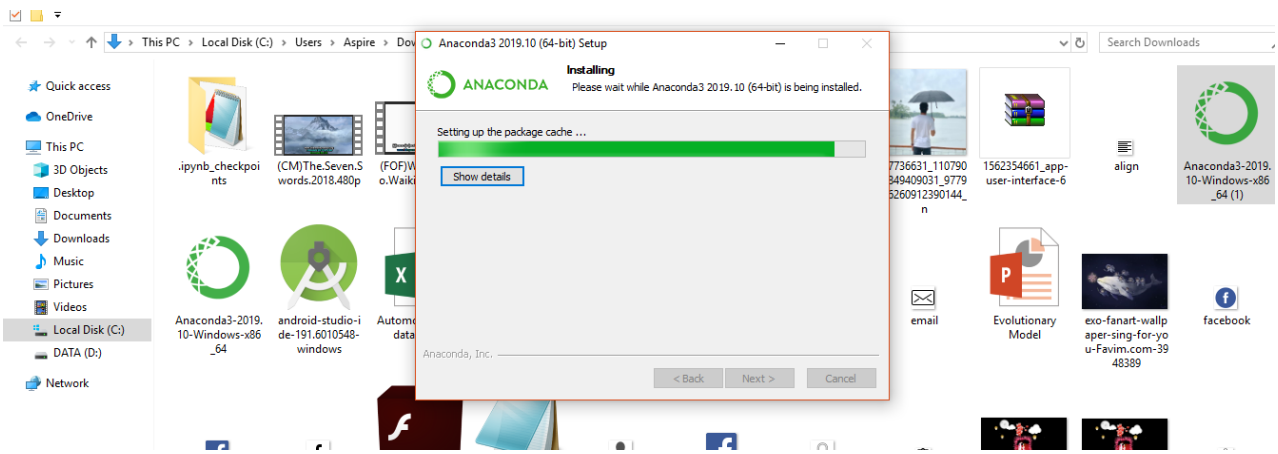


Run ကိုclick ပါ။Next ကို သာလျင် click သွားပါ။



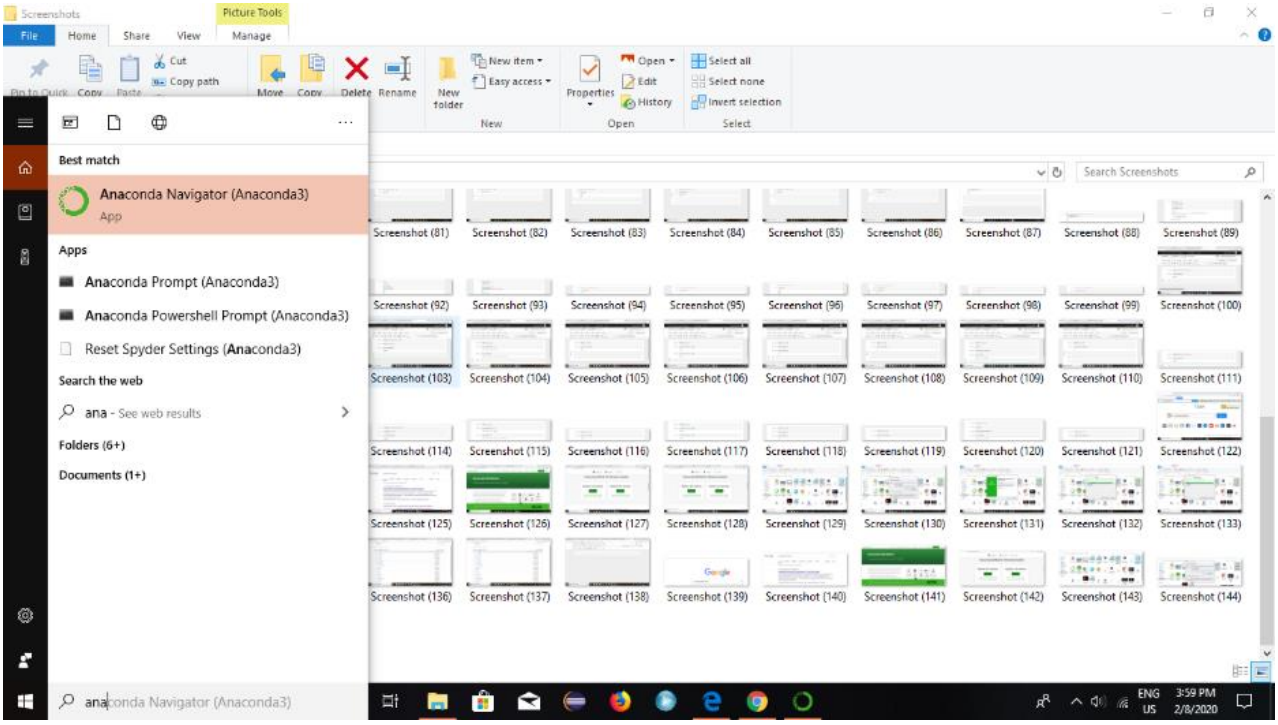
Install ပြုလုပ်နေပါ ဗြီ။

(Install ပြုလုပ်နေစဉ် မည်သည့်ခ လုတ် မျှနှိပ်စရာမလိုပါ)

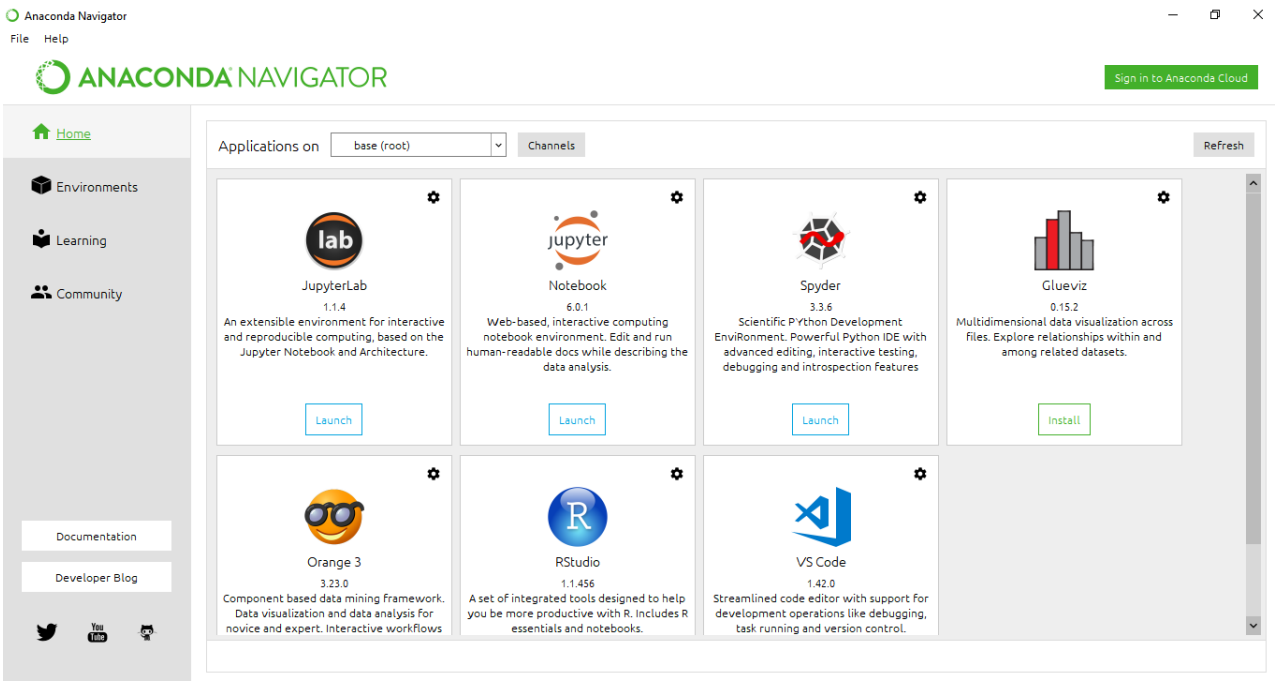


*Install ပြုလုပ်ပြီးပါ က search bar မှ Anaconda ဟုရိုက်၍ရှာပါ။

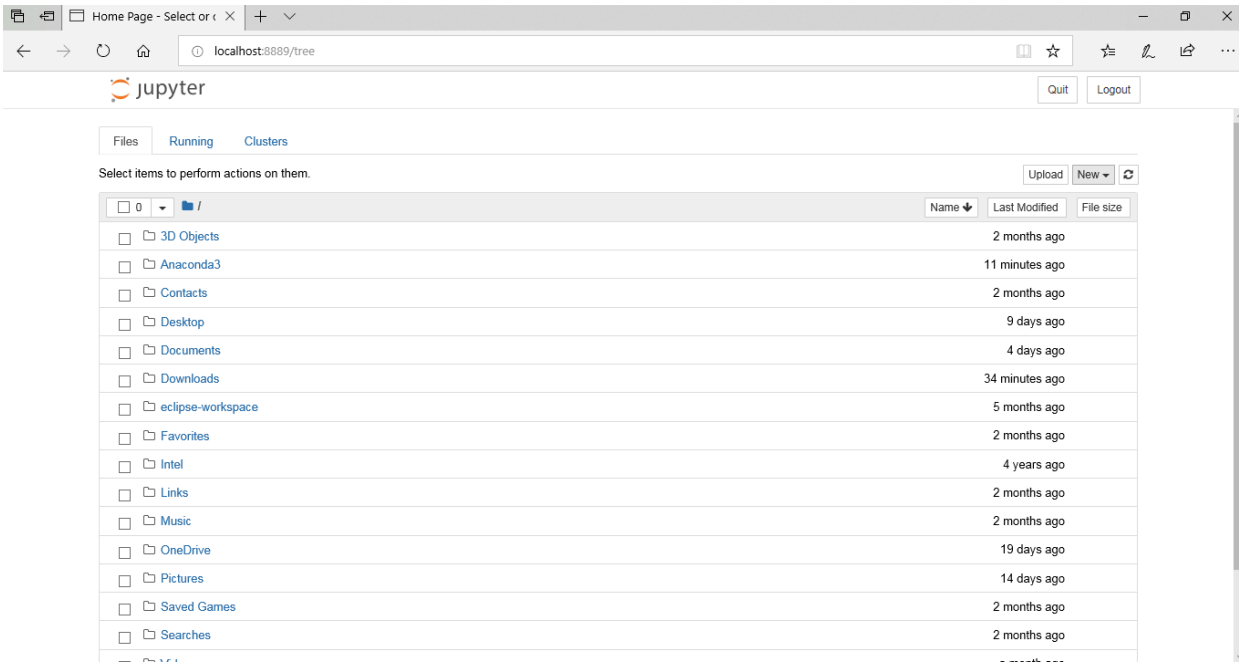
ခဏစောင့်ပါ။ (မည်သည့်ခလုတ်မျှနှိပ်စရာမလို)



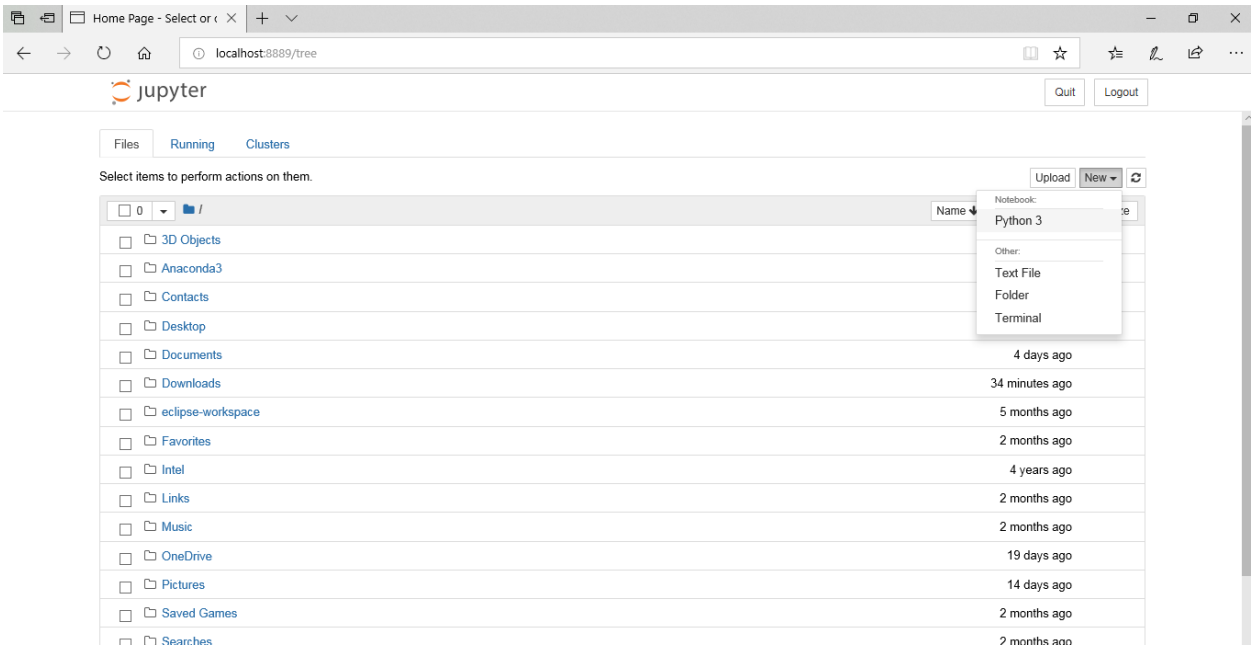
*ပုံတွင်ပြထားသည့်အတိုင်းပေါ်လာပါက Jupyter Notebook ကို Launch လုပ်ပါ။



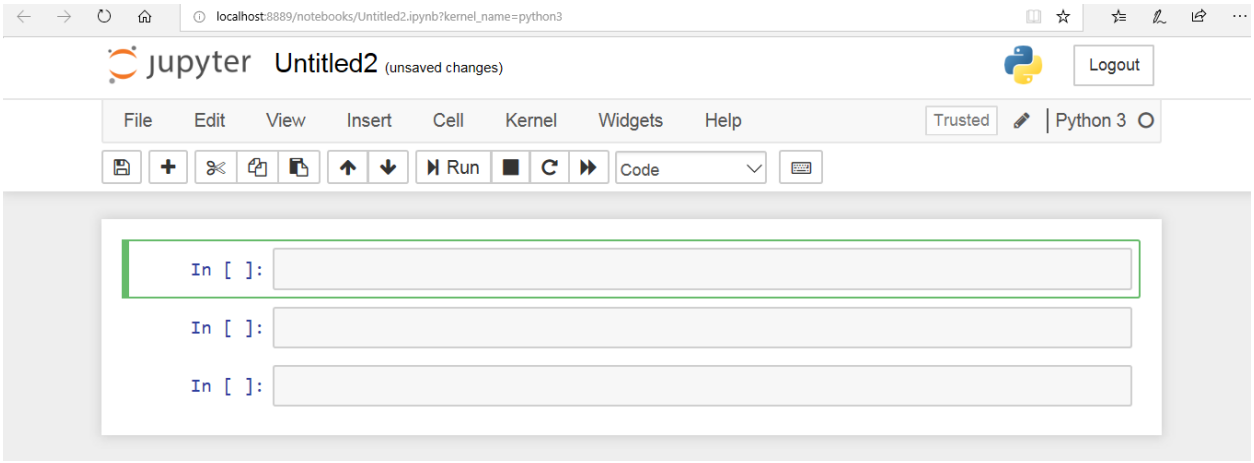
*ပုံပါအတိုင်းပေါ်လာပါလိမ့်မည်။ပြီးနောက် ညာဘက် ထောင့်အပေါ်ဘက်ရှိ New button ကို click ပါ။

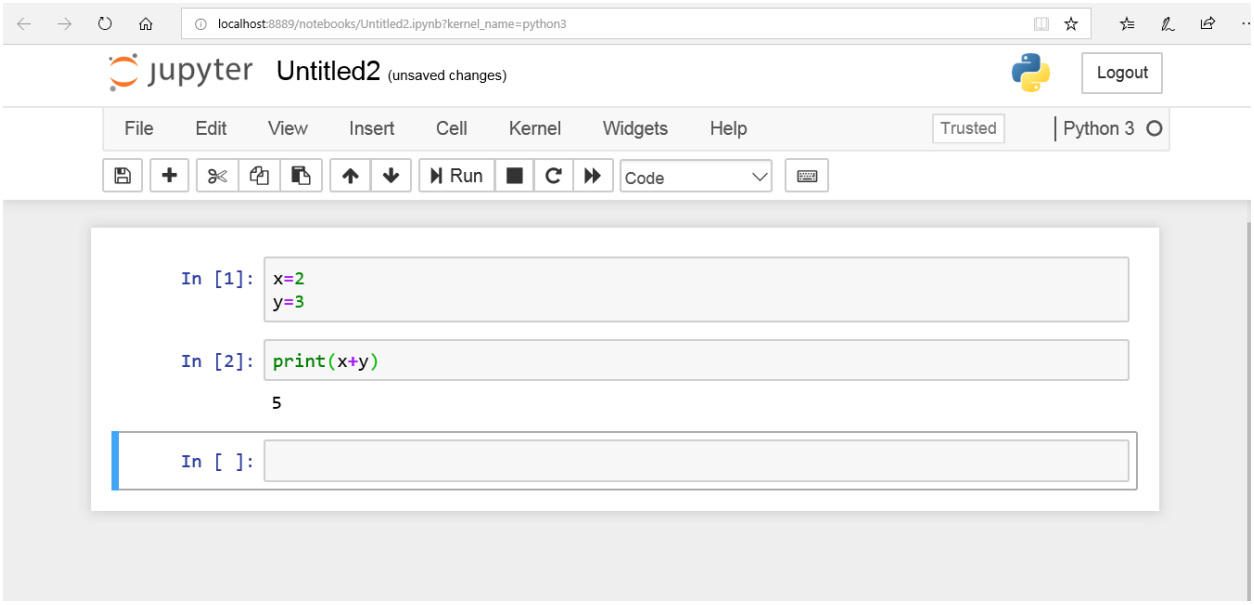


Python 3 ကိုရွေးပါ။



*code များရေး၍ runနိုင်ပါပြီ။





Number

Number မှာဆိုရင် အမျိုးအစား သုံးမျိုး ရှိပါသည်။

1. integer(int)

2. float

3. complex တို့ ဖြစ်ပါသည်။

Integer ဆိုသည်မှာ ကိန်းပြည့်များကို ဆိုလိုခြင်းဖြစ်ပါသည်။

Example: 1, 2, 3, 4 . . .

Float ဆိုသည်မှာ ဒဿမကိန်းများကို ဆိုလိုခြင်းဖြစ်ပါသည်။

Example: 1.2, 2.00, 3.25893 . . .

Complex ဆိုသည်မှာ မသိကိန်းများနှင့် ရောနှောနေသော ကိန်းများကို ဆိုလိုခြင်းဖြစ်ပါသည်။

Example: Z=3j,1+ 4k,h-7b . . .

Variables

Variables ဆိုသည်မှာ programming language အများစုတွင် အရေးအကြီးဆုံး နေရာမှ ပါဝင်နေသော အရာတစ်ခုပင်ဖြစ်ပါသည်။variableများသည်ၚင်းတို့ထံတွင်တန်ဖိုးများအားသိမ်းဆည်းထားခွင့်ပြုသည်။ထို variable များကို program ထဲတွင် နောင်တစ်ချိန် ပြန်လည်အသုံး ပြုနိုင်သည်။

Variables များကို သင့်စိတ်ကြိုက်အကြိမ်ပေါင်း များစွာ သတ်မှတ်၍ရသည်။

Example: fruit, student, name, town

Rules (Variables များကိုသတ်မှတ်ပုံ)

1. Variables များအား စာများ ဂဏန်းများ under score " _ " များနှင့်ရေးနိုင်သည်။

2. Variables များအား ဂဏန်းများနှင့် စတင်၍မရေးသားရ။

3. မိမိကြေညာထားခြင်း မရှိသော variables များအား programထဲတွင်ပြန်လည်ခေါ်ယူအသုံးပြုပါက Errorတက်မည်ဖြစ်ပါသည်။

4. ကြေညာထားပြီးသော variables များအား ဖျက်လို့ပါက del statement အား အသုံးပြု၍ရသည်။

ဖျက်ထားပြီးပြီဖြစ်သော variables များအားလည်း ပြန်၍ reassigned ပြုလုပ်၍ ရပါသည်။

String

Python မှာ စာကြောင်းတွေကို ထုတ်ပြချင်တယ် ဆိုရင်တော string တွေကို အသုံးပြုရပါမယ်။ String တွေကို သတ်မှတ် ချင်တယ် ဆိုလို့ရှိရင် Two single or double quotation marks တွေနဲ့သတ်မှတ် ဖန်တီးလို့ ရပါတယ်။

Example:

string s =”Mg Mg”

string s = ‘Mg Mg’

တစ်ချို့character တွေ (eg. ‘ , “ , / , . . .) ကို ကျတော့ string ထဲမှာ တစ်ခါတည်း ရေးလို့ မရပါဘူး ။ သူတို့ကို ရေးချင်တယ် ဆိုရင်တော့သူတို့လေးတွေရဲ့ရှေ့မှာ backslash\ လေးတွေ ခံပြီးတော့ ရေးပေးရပါမယ်။

Example:

“He\’s a boy.”

>>He’s a boy.

Newlines

နောက်တစ်ကြောင်းဆင်းပြီး ကိုယ်ရဲ့formatအ တိုင်းဖြစ်ချင်ရင် three set of quotes ထဲမှာ ရေးရပါတယ်။

Example:

“”” I am Mg Mg

and I am a Student.”””

>>>I am Mg Mg

and I am a student.

Concatenation

String တစ်ခုနှင့်တစ်ခု စာကြောင်းတွေ တစ်ခုနှင့်တစ်ခု ပေါင်းချင်တယ် ဆိုရင် concatination ကို သုံး ရ ပါမယ်။ ပေါင်းတဲ့အခါမှာstring တွေက single quote နဲ့ပဲ ဖြစ်ဖြစ် double quotes နဲ့ပဲဖြစ်ဖြစ် ရေးလို့ရပါတယ်။

Example:

1.“spam”+’eggs’

>>>spameggs

2.”Mya Mya”+”and”+”Aye Aye”

>>>Mya Mya and Aye Aye

Note: Numbers နှင့် strings တွေ concatination လုပ်လို့မရပါဘူး။

Example: 1+’2’

>>>TypeError

Multiplication

String တွေကို int တွေနဲ့လည်း မြှောက်လို့ရပါတယ်။ output အနေနဲ့ကတော့ original string ရဲ့ repeated version အနေနဲ့ ရလာမှာပါ။ string အချင်းချင်း မြှောက်လို့ မရပါဘူး။

Example:

‘2’*4

>>>2222

‘3’*’N’

>>>TypeError

Simple input and output

Program ဆိုတာ input ကို လက်ခံပြီး output ကို ပြန်ထုတ်ပေးတာပါ။

ဥပမာ။ ။ ဖုန်းခေါ်တဲ့ အခါ ဖုန်းနံပါတ်ကို လက်ခံပြီး ထည့်လိုက်တဲ့ဖုန်းနံပါတ်နဲ့ ချိတ်ဆက်ပေးသလိုပါ။

simple input အဖြစ် input ( ) ကို သုံးပါတယ်။

simple output အဖြစ် print ( ) ကို သုံးပါတယ်။

Example

```
x = input ( ' Enter phone numbers : ')
print ( 'Connecting ' , x )
```

သူ့ကို run ရင်

Enter phone numbers :

ဆိုပြီး စာနဲ့ ကွက်လပ်ပေါ်ပါလိမ့်မယ်။

ကွက်လပ်မှာ 09688755437 လို့ ရိုက်လိုက်ရင် အောက်ကလို ပြပါလိမ့်မယ်။

Enter phone numbers : 09688755437

Connecting 09688755437

ကွက်လပ်မှာရေးတဲ့စာဟာ input( ) နေရာ ဝင်ပြီး x ရဲ့တန်ဖိုးဖြစ်သွားတာပါ။

Type conversion (အမျိုးအစားပြောင်းလဲခြင်း)

Type conversion ဆိုသည်မှာ number များအား အမျိုးအစား တစ်ခုမှ တစ်ခုသို့ ပြောင်းလဲခြင်းပင် ဖြစ်ပါသည်။

"2" + "2"

စာနှစ်ခုပေါင်းရင် 22 ရပါတယ်။

2 + 2

ကိန်းပြည့်နှစ်ခုပေါင်းတော့ 4 ပါ။

int("2") ဟာ စာနှစ်ကို ကိန်းပြည့်နှစ်အဖြစ်ပြောင်းတာပါ။

int("2") + int("2")

စာတွေကို ကိန်းပြည့်ပြောင်းပြီးပေါင်းလို့ 4 ပါ။

မိမိကြေညာလိုက်သောnumber၏typeအားသိလိုပါကtype()functionအားအသုံးပြု၍ ကြည့်ရှုနိုင်ပါသည်။

```
In [1]: print(type(2))
        print(type(3.54))
        print(int(3.23))
        print(float(5))
        print(str(2))
```

```
<class 'int'>
<class 'float'>
3
5.0
2
```

Type ပြောင်းချင်ပါက ပြောင်းချင်သော variable အမျိုးအစားပေါ်မူတည်ပြီး ပြောင်းနိုင်ပါတယ် ။ Integer ပြောင်းချင်ရင် int()၊ float ပြောင်းချင်ရင် float() ၊ String ပြောင်းချင်ရင် str() အစရှိသည်ဖြင့် ပြောင်းနိုင်ပါတယ်။

Comment

Comment ဆိုသည်မှာ မှတ်ချက်ပေးခြင်းသူ၍လည်း အဓိပ္ပာယ်ရပါသည်။code များကို ရေးသားသော အခါတွင်မူ မိမိရေးသားသော code၏ ရည်ရွယ်ချက် အား သိသာစေရန် ရေးသားလေ့ရှိပါသည်။ထို comment စာကြောင်းအား code များကို run ရာတွင် ထည့်သွင်း၍ run လေ့မရှိပါ။

hash sign ဖြင့် စ၍ ရေးနိုင်ပါသည်။

Example:

this is printing a statement

Print (“Hello everyone!!! My name is Ko ko”)

Output:

Hello everyone!!! My name is Ko ko

Sign ၏နောက်တွင်ရေးထားသော စာကြောင်းမှာ output ထွက်သည့်အခါ ထည့်သွင်း၍ runခြင်းအား မပြုလုပ်ပါ။စာကြောင်းတစ်ကြောင်းအား ထုတ်ပေးသည်ဟူသော မှတ်ချက်တစ်ခုအား ပေးရုံသာပေးသည့် သဘောပင် ဖြစ်ပါသည်။

Operators

Operators ဆိုတာကတော့ programming language တိုင်းမှာ အမြဲလိုလို ပါဝင်ပြီး formula တွေ၊ logical ဆိုင်ရာ condition (နောက်အခန်းတွေမှာ ပါဝင်မည်) တွေ ရေးရာမှာ ကူညီပေးတဲ့ အကူ Object လေးတွေလို့ ပြောရင်လည်း မမှားဘူးပေါ့။ Operator တွေကို အုပ်စုခွဲပြရင်တော့ အုပ်စု ၄ စုရှိပြီး အဲဒါတွေကတော့ အောက်ပါ အမျိုးအစားတွေပါဘဲ ။

(၁) Arithmetic operator

(၂)Assignment operator

(၃)Relational operator

(၄)Logical operator ဆိုပြီးတော့ တွေ့ရမှာပါ။ Operator တစ်ခုစီအလိုက် သီးသန့် ရှင်းပြပေးပါမယ်။

Arithmetic Operator

Arithmetic operator တွေကို အောက်က ဇယားကွက်မှာ ရှင်းပြထားပါတယ်။

Operator	Evaluation
+	ပေါင်းခြင်း
-	နှုတ်ခြင်း
*	မြှောက်ခြင်း
/	စားခြင်း
%	အကြွင်း

ဒီ ဇယားမှာ ပါဝင်တဲ့ operator တွေရဲ့ စွမ်းရည် တွေကတော့ Math Operator ပါပဲ။ ပေါင်း၊နုတ်၊မြှောက်၊စား တွေကို လုပ်ပေးနိုင်ပါတယ်။ အစား operator မှာ နှစ်မျိုး ကွဲတဲ့ အတွက် ကြောင့် (/) ကို စားလဒ် ပြလို့ မှတ်ပြီး (%) ကို အကြွင်း ပြလို့ ခေါ်ပါသေးတယ်။ Arithmetic Operator တွေအကြောင်း ကို ရှင်းလင်းအောင် အောက်မှာ ဥပမာ နဲ့ တကွ ရှင်းပြ ထားပါတယ်။လေ့လာ ကြည့် ပါ။

Example:

```
1.  print ( 3+3 )
2.  print (6-2 )
3.  print ( 5*3)
4.  print (4/2)
5.  print (15%2)
```

Output (ရလဒ်):

1: အဖြေက 6

2: အဖြေက 4

3: အဖြေက 15

4: အဖြေက 2

5: အဖြေက 1 (15 ကို 2 နဲ့ စားပြီး ကြွင်းသော အကြွင်း)

ထပ်ကိန်း

print (3**3)

Output = 27

အစား

print (10 / 4)

Output = 2.5

print(10 // 4)

Output = 2

အကြွင်း

print (10 % 3)

Output = 1

print(2**3*4/8+4-6)

Python မှာ ထပ်ကိန်းကို အရင်ရှင်းပါတယ်။

2 သုံးထပ်က 8 ပေါ့။ print(8*4/8+4-6)

အမြောက်ကို ဒုတိယရှင်းပါတယ်။

8 နဲ့ 4 မြှောက်ရင် 32 ပေါ့။

print(32/8+4-6)

အစားကို တတိယရှင်းပါတယ်။

32 ကို 8 နဲ့စားရင် 4 ပေါ့။

print(4+4-6)

အပေါင်းက လေးခုမြောက်ပေါ့။

4 နဲ့ 4 ပေါင်းရင် 8 ပေါ့။

print(8-6)

နောက်ဆုံးက အနှုတ်ပေါ့။

8 ထဲက 6 နှုတ်ရင် 2 ပေါ့။

print(2)

Output = 2

==========

ဒဿမကိန်းများ

decimal point ထည့်ရင် ဒဿမကိန်းဖြစ်ပါတယ်။

example = 5.0

စားရင်လည်း ဒဿမကိန်းဖြစ်ပါတယ်။

print(10 / 4)

Output = 2.5

ဒဿမကိန်းကို မြှောက်ရင်လည်း ဒဿမကိန်းဖြစ်ပါတယ်။

print (2.5 * 4)

Output = 10.0

ဒဿမကိန်းကို နှုတ်ရင်လည်း ဒဿမကိန်းဖြစ်ပါတယ်။

print(10.0 - 4)

Output = 6.0

ဒဿမကိန်းကို ပေါင်းရင်လည်း ဒဿမကိန်းဖြစ်ပါတယ်။

print(10.0 + 4)

Output = 14.0

Assignment Operator (=)

Assignment Operatorရဲ့အဓိပ္ပါယ် ကတော့ ရှေ့မှာ ရှင်းပြထားတဲ့ variable တစ်ခုခုထဲ ဂဏန်း တန်ဖိုး အစားသွင်း တယ် ထည့်လိုက်တယ် လို့ ယူဆလို့ရပါတယ်။ Assign လုပ်ခြင်း (ထည့်ခြင်း)လို့ လဲ ခေါ်ပါတယ်။

Example:

1. a = 2
2. b = 5
3. c = a + b
4. print (c)

ဆိုရင် နံပါတ် 1 စာကြောင်းရဲ့အဓိပ္ပါယ် က variable a ထဲကို ဂဏန်း တန်ဖိုး 2 ကို ထည့် တယ် (assign) လုပ်တယ်လို့ အဓိပ္ပါယ် ရတယ်။ နံပါတ် 2 စာကြောင်း မှာ ကျတော့ variable b ထဲကို ဂဏန်းတန်ဖိုး 5 ထည့်လိုက်တယ်လို့ အဓိပ္ပါယ် ရပါတယ် ။ အဲ့ စာနှစ်ကြောင်း အပြီးမှာ a ရဲ့တန်ဖိုးဟာ 2 နဲ့ ညီပြီး b ရဲ့တန်ဖိုးဟာ 5

နဲ့ ညီမျှတယ် လို့ အဓိပ္ပါယ် ရတယ်။ နံပါတ်သုံး စာကြောင်းမှာ variable c ထဲကို a နဲ့ b ရဲ့ ပေါင်းလဒ် အဖြေကို ထည့် မယ်ဆိုတော့ a ရဲ့တန်ဖိုး က 2 ၊b ရဲ့တန်ဖိုးက 5 သူတို့နှစ်ခုရဲ့ပေါင်းလဒ် 7 ကို c ထဲထည့် တော့ variable c ရဲ့တန်ဖိုးဟာ 7 ဖြစ်သွားပါတယ် ။

အဲ့တာကြောင့် နံပါတ် 4 စာကြောင်းမှာ c ရဲ့တန်ဖိုး ထုတ် ကြည့်တဲ့အခါ နောက်ဆုံးအဖြေ 7 ဆိုပြီး ထွက်လာမှာ ဖြစ်ပါတယ်။ အဲ့ဒီလောက်ဆိုရင် Assignment Operator အကြောင်းသိသွားလောက်ပြီ ထင်ပါတယ်။ ကဲ အဲဒါဆို ဆက်လေ့လာရအောင်။ တချို့ Assignment Operator စာကြောင်းတွေမှာ အခြား operator တွေဖြစ်တဲ့ Increment Operator ၊ Decrement Operator တွေနဲ့ ရေးထားတာတွေလည်း ပါ နိုင် ပါတယ်။ အဲ့ ဒီ Increment operator ၊ Decrement Operator တွေအကြောင်းကို ဆက်ပြီး လေ့လာကြရအောင်။

Increment and Decrement Operator

Increment and Decrement operator ဆိုတာ variable တစ်ခုခု ရဲ့တန်ဖိုးကို increment (တိုးလိုက်ခြင်း) ၊ decrement (လျော့လိုက်ခြင်း) တွေ လုပ်လိုက်ခြင်းဖြစ်သည်။ i + + ဆိုတဲ့ ပုံစံလေးကို ရှင်းစပြရင် အလွယ်ဆုံးနည်းပါ။ ဒီ ပုံစံလေးကို မှတ်ထားပါ။ ဒါဆိုရင် ဥပမာ ပြထားပါတယ် ။ လေ့လာကြည့်ပါ။

1. i =6
2. i + +

ဆိုရင် ပထမ စာကြောင်းမှာ variable i ထဲကို ဂဏန်းတန်ဖိုး 6 ကို ထည့်လိုက်ပါတယ်။ ဒုတိယ စာကြောင်းက i ++ ရဲ့အဓိပ္ပါယ် က i ရဲ့တန်ဖိုးကို 1 တိုးလိုက်ခြင်း တစ်နည်းအားဖြင့် 1 ပေါင်းလိုက် ခြင်းလို့ ဆိုလိုပါတယ်။ i= i+1 (i ရဲ့တန်ဖိုးထဲ အရင် အဟောင်း i ရဲ့တန်ဖိုး နဲ့ 1 နဲ့ ပေါင်းပြီး ရလာတဲ့ တန်ဖိုးအသစ်ကို i ထဲဘဲ ပြန်ထည့်ခြင်း) နဲ့ အဓိပ္ပါယ်တူပါတယ်။

ကဲ နောက်ထပ် i + + နဲ့ ပုံစံ တူရေးလို့ရတဲ့ formulas ကို ဥပမာ ရေးပြပါမယ်။

1. i=5
2. i + =1
3. print(i)

ဆိုပြီး program ရေးလိုက်ရင် နောက်ဆုံး အဖြေမှာ 6 ဆိုပြီး ထွက်လာပါလိမ့်မယ်။ဘာကြောင့်လဲဆိုတော့ =(equal sign) ရဲ့ညာဘက်မှာရှိတဲ့ တန်ဖိုးဟာ =(Equal Sign) ရဲ့ဘယ်ဘက်ရှိ (+) operator ကြောင့် i ရဲ့တန်ဖိုးထဲကို လာပြီးတော့ ထည့်ပေါင်းပေးရမှာ ဖြစ်ပါတယ်။ i= i+1 နဲ့ ပုံစံ တူပါတယ် ။ i=i+3 ဆိုရင် i+=3 / i=i-3 ဆိုရင် i-=3 /

i=i*3 ဆိုရင် i*=3 / i=i/2 ဆိုရင် i/=2 ဆိုပြီးရေးနိုင်ပါတယ်။ကဲ အဲဒါဆို Increment operator အကြောင်း နားလည်လောက်ပြီ ထင်ပါတယ်။ ဆက်လက်ပြီး decrement operator အကြောင်း ဆက်လေ့လာပါဦး။

Increment operator (++) က 1 တိုးတာဆိုရင် Decrement Operator (- -) က1 လျော့သွားတာကို ဆိုလိုပါတယ် ။ ကဲ ဥပမာ လေးနဲ့ ကြည့်လိုက်ရအောင်။

Example:

```
1. i=6
2. i - -
3. print (i)
```

ဆိုရင် program output(အဖြေ) ဟာ 5 ရလာမှာဖြစ်ပါတယ်။ စာကြောင်း 2 ရဲ့အဓိပ္ပါယ် က i ရဲ့တန်ဖိုးကို 1 လျော့လိုက်ခြင်း ဖြစ်တယ် တစ်နည်းအားဖြင့် i=i-1 နဲ့ လည်းညီမျှပါတယ်။ အဲ့ဒီလောက်ဆို increment and decrement operator တွေအကြောင်း သိသွားလောက်ပြီ ထင်ပါတယ်။

Prefix and Postfix

Prefix (ချက်ချင်းတိုး)

Prefix ဆိုတာက တန်ဖိုးမှာ increment operator ကိုဘယ်ဘက်မှာ ထည့်သွင်း ရေးသားအသုံးပြု တာပါ ။ ဒီတော့ တန်ဖိုးဟာ လက်ရှိ ရောက်ဆဲ စာကြောင်းမှာတင် print ကိုတန်းထုတ်လို့ ရသလို 1တိုးပြီး သားဖြစ်နေပါပြီ ။ အောက်မှာ ရေးထားတာတွေ လေ့လာကြည့်ပါ။

Example:

```
i = 2
print ( ++i)
```

ဒီလို prefix ပုံစံ နဲ့ ရေးလိုက်တာကြောင့် ရောက်ဆဲ စာကြောင်းမှာတင် i ရဲ့တန်ဖိုးရဲ့amount ဟာ 3 ဖြစ် သွားပါပြီ ။ increment operator (++) ကို ကိုယ်စားလှယ်တန်ဖိုးဖြစ်တဲ့ i တို့ x တို့ရဲ့ဘယ်ဘက်မှာ ထည့်သွင်း

ရေးသားရင် i တို့ x တို့ ရဲ့တန်ဖိုးဟာ မူလတန်ဖိုးထက် 1 တိုးသွားတယ် ဆိုတာဟာ increment ရဲ့ prefix သဘော တရားပါဘဲ ။ ဒါ့ကြောင့် prefix ဆိုတာ 1 ချက်ချင်းတိုး/ အရင် တိုးလို့ဘဲ မှတ်ထားလိုက်ပါ ။

Postfix (နောက်တစ်ကြောင်းမှတိုး)

Postfix ဆိုတာကတော့ increment operator (++) ကို i တို့ x တို့ လို ကိုယ်စားလှယ် ကိန်းတန်ဖိုးရဲ့ဘယ် ဘက်မှာ ထည့်သွင်း ရေးသားရတာပါပဲ ။ Prefix က 1 ကို ရောက်ဆဲ စာကြောင်းမှာတင် ချက်ချင်း တိုးပေးနိုင် တယ်ဆိုတော့ postfix ဆိုတာ နောက်တစ်ကြောင်းမှာမှ 1 တိုးပေးနိုင်တယ် လို့ သဘောပေါက်ထားပါ ။ sample ရေးပြပါမယ် ။

Example:

```
x = 2
x + +          # x = 2
print ( x)        # x = 3
```

ဒီလို ရေးခြင်းဟာ တိုးသွားတယ် ဆိုပေမယ့် postfix ပုံစံ ရေးခြင်းကြောင့် နောက်တစ်ကြောင်း မှသာ 1 ကို တိုးပေးနိုင်တာပါ။ တကယ်လို့ အောက်မှာ ရေးပြထားတဲ့ ပုံစံအတိုင်း ရေးလိုက်ရင် တော့ increment လုပ်ပေမယ့် တိုးမလာတာကို တွေ့ရမှာပါ ။

```
1.  x = 2
2.  print ( x + + )       # x = 2
```

မူလ တန်ဖိုးပဲ လာပြပါမယ် ။ ဒါဆိုရင် အဖြေဟာ 2 ပဲ ထွက်လာတော့ မူလတန်ဖိုး ထက် 1 တိုး မလာ ဘူး ပေါ့ ။ ဒါပေမယ့် postfix ဟာ နောက်တစ်ကြောင်းမှာမှ 1 တိုးနိုင်တယ်ဆိုတာကြောင့် အဲဒီ အောက်မှာ ဒီ အတိုင်း ဆက်ရေးကြည့်ပါဦး ။

```
3.  print ( x )              # x = 3
```

ဒီ line 3 အဖြစ် ဆက်ရေးလိုက်တာဟာ increment မလုပ်တော့ဘူးနော်။ line 2 မှာ တိုးထားတာကို လှမ်း ခေါ်ပြီး print ထုတ်လိုက်တာပဲ ။ ဒီတော့ အဖြေ ဟာ 3 လို့ output ထွက်လာတာပေါ့ ။ ဒါကြောင့် postfix ပုံစံ ရေး ရင် နောက်တစ်ကြောင်းမှာမှ 1 တိုးမယ်ဆိုတာကို သေသေချာချာ နားလည်သဘောပေါက်ပြီး မှတ်ထားပါ။

ဒါဆိုရင်တော့ increment နဲ့ decrement operator တွေ ရေးသားအသုံးပြု ရာမှာ prefix နဲ့ postfix ပုံစံ ရေးသားနည်းကို လုံးဝ နားလည် သဘောပေါက်သွား ပြီ ဆိုတာကို ယုံကြည်ပါတယ် ။

Combining the Assignment and Arithmetic operators

Assignment နဲ့ Arithmetic operators တွေကို ရှင်းပြပြီးတဲ့ အခါမှာ ဒီ operator တွေကို ပေါင်းစပ် အသုံးပြုလို့ရတဲ့ method တွေကိုရေးပြပါ ။ Operator ကိုနိုင်နင်းအောင် အသုံးပြု နိုင်မှသာလျှင် programmer ကောင်းတစ်ယောက် Developer ကောင်းတစ်ယောက် ဖြစ်လာမှာပါ။ ကဲ အောက်မှာ ရေးထားတဲ့ ဥပမာ လေးကစပြီး လေ့လာကြတာပေါ့ ။

Example:

```
myAge = 5

temp = myAge + 2

myAge = temp

print ( myAge)
```

ဒီ sample လေးကို ကြည့်ရင် myAge ရဲ့တန်ဖိုးက 5 ပါ။ temp = myAge +2 လို့ ရေးတဲ့အတွက် temp = 5+ 2 မှ temp=7 ရလာပြီပေါ့ ။ ဒီနေရာမှာ သိထားရမှာက temp ဆိုတာ ကိုယ်စားလှယ်ကိန်းပါ။ ကိုယ်စားလှယ် လို့ ဘာလို့ ပြောသလဲ ဆိုတော့ temp ဟာ တွက်ချက်လို့ ရထားတဲ့ အဖြေကို သူ့အနေနဲ့ ယာယီ သိုလှောင်ပြီး သိမ်းမှတ်ပေးထားတာပါ။ နောက်တစ်ကြောင်းမှာတော့ myAge = temp လို့ရေးပြီး temp ရဲ့တန်ဖိုးကို myAge ထဲကို ပြန်လွဲ ပေးလိုက်ပါပြီ ။ အဖြေထုတ် တဲ့ အခါမှာတော့ myAge နဲ့ပဲ ထုတ်ပေးမှာပါ။ ကဲ အပေါ်မှာ ရေးခဲ့သလို ရေးတဲ့အခါမှာ အနည်းငယ် complex ဖြစ်သလို သိပ်မကျွမ်း ရင် အမှားတွေ ရေးမိပြီး error တွေ တက်လာနိုင် ပါတယ်။ ဒီတော့ assignment operator ကိုအသုံးပြုပြီးတော့ ဒီလိုရေးရင် ပိုလွယ်မှာပေါ့။

Example:

```
 myAge = 5

myAge = myAge +2          # 5 + 2
```

ကဲ ဒီလိုရေးလိုက်တော့ ပိုမလွယ်ဘူးလား။ ကဲ ဒါဆိုရင် ဒီထက် လွယ်တဲ့ နည်းလမ်းကို ရေးပြ ပါဦးမယ် ။

Sample:

```
myAge = 5
```

myAge + = 2

ဒီလို ရေးပြီး အဖြေထုတ်လိုက်ခြင်း အားဖြင့် 7 ဆိုတဲ့ အဖြေ ကိုရ မှာပါပဲ။ ဒီလို ရေးတာကို self- assign လုပ်တယ်လို့ ခေါ်ပါတယ် ။ Self- assign လုပ်ရာမှာ (+=) addition အပြင် Subtraction (-=) ၊ division (/ =) ၊ multiplication (* =) နဲ့ modulus (% =) တွေကိုလည်း ရေးလို့ ရပါတယ် ။

Relational Operator

Relational Operators တွေကို အောက် Table မှာ ရေးပြထားပါတယ်။ လေ့လာကြည့်ပါ။

Table. The Relational Operator

Name	Operator	Sample	Evaluates
Equals	==	100==50 50==50	False True
Not Equals	!=	100!=50 50!=50	True False
Greater Than	>	100>50 50>50	True False
Greater Than or Equals	>=	100> =50 50> = 50	True True
Less Than	<	100< 50 50< 50	False False
Less Than or Equals	< =	100< = 50 50< = 50	False True

ညီတာစစ်ချင်ရင် ==

မညီတာစစ်ချင်ရင် !=

ငယ်တာစစ်ချင်ရင် <

ကြီးတာစစ်ချင်ရင် >

. . .

```
print( 1 == 1 ) # True

print( 1 != 1 ) # False


print( 1 <  2 ) # True

print( 1 >  ) # False
```

Logical Operator

တစ်ခုထက် ပိုတဲ့အခြေအနေတွေကို မှန်မှားနှိုင်းယှဉ်ဖို့ သုံးပါတယ်။and ,or ,not ဆိုပြီး သုံးခုရှိပါတယ်။

and တန်းဆက်

အခြေအနေအားလုံးမှန်လျှင် True ဖြစ်မည်။

Example: print (2 == 2 and 2 < 5)

အခြေအနေနှစ်ခုလုံးမှန်လို့ True ဖြစ်ပါတယ်။

or ပြိုင်ဆက်

အခြေအနေအားလုံးထဲမှ တစ်ခုမှန်ရုံမျှဖြင့် True ဖြစ်မည်။

Example: print (2 == 2 or 2 > 5)

အခြေအနေနှစ်ခုမှာ တစ်ခုမှန်လို့ True ဖြစ်ပါတယ်။

not အခြင်း

အခြေအနေအားလုံးမှားလျှင် True ဖြစ်မည်။

Example: print (not (2 == 2))

အခြေအနေမှန်လို့ False ဖြစ်ပါတယ်။

---

if Statement

Python Programming Language မှာ If ကို အခြေအနေ အမျိုးမျိုး ရှိတဲ့ အနေအထားတွေမှာ အသုံးပြုပါတယ် ။ တကယ်တော့ if statement ဆိုတာ conditional statement တစ်ခုပါပဲ။ if ကို သုံးတဲ့အခါ if ရဲ့နောက်က expression ကို မှန် မမှန် စီစစ်ပါတယ်။ မမှန်ခဲ့ရင် သူ့ရဲ့နောက်က statement ကို အလုပ်မလုပ် တော့ဘဲ ကျော်ပြီး နောက်ထပ် if expression တစ်ကြောင်းကို ဆင်းသွားမှာပါ။ မှန်ခဲ့ရင် if expression ရဲ့နောက်မှာ ပါတဲ့ statement ကို အဖြေထုတ်ပြီး တွက်ချက်ပေးသွားမှာပါ။ if expression အခြေအနေက မှန်နေမှ အတွင်းက စာကြောင်းတွေကို အလုပ်လုပ်ပေးနိုင်မှာကို ဆိုလိုတာပါ။အောက်မှာ အသေးစိတ် ထပ်မံ ရှင်းပြထားပါတယ်။ အခြေအနေတစ်ခုကိုင်စွဲထားပြီး မှန်လျှင် code အချို့ကို အလုပ်လုပ်စေမည်။

ပုံသေနည်း

if အခြေအနေ :

အခြေအနေမှန်လျှင် အလုပ်လုပ်မည့် code များ

ဥပမာ။ ။

```
name = "Mg Mg"
if name == "Mg Mg" :
   print(" He is Mg Mg.")
```

ရှင်းလင်းချက်။ ။ name ဟာ မောင်မောင်ကို ကိုယ်စားပြုတာဖြစ်လို့ name = "Mg Mg" ဟာ True ဖြစ်ပါတယ်။if နောက်က အခြေအနေမှန်လို့ block ထဲက code ကို အလုပ်လုပ်ပါတယ်။

==========

else

မည်သည့် အခြေအနေမှ စွဲကိုင်မထားဘဲ if ထဲက အခြေအနေမှားလျှင် အလုပ်လုပ်မည်။

ဥပမာ။ ။

```
password = input("Enter your password : ")
if  password == "34691" :
   print("Facebook Opened.")
else :
   print("Wrong Password & Try Again.")
```

အထက်ပါ program ကို run ရင်

Enter your password : လို့ ပေါ်လာပါ့မယ်။

မိမိရိုက်ထည့်တဲ့ နံပါတ်က 34691 ဖြစ်ရင် if ရဲ့ အခြေအနေ မှန်ပြီး အလုပ်လုပ်ပါလိမ့်မယ်။

အခြားနံပါတ်ဖြစ်ခဲ့ရင် if ရဲ့ အခြေအနေမှားပြီး else ထဲက code က အလုပ်လုပ်ပါလိမ့်မယ်။

Range

ကိန်းများပါဝင်သော စာရင်းတစ်ခု ဖန်တီးရန် အသုံးပြုသည်။

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

x သည် တစ်မှ ကိုးထိရှိသော စာရင်းတစ်ခုဖြစ်သည်။

သူ့ကို range သုံး၍လည်းရေးနိုင်ပါတယ်။

```
x = range ( 1 , 10 )
```

ဒါဆိုရင် တစ်ကနေ ကိုးထိ ရှိတဲ့ စာရင်းတစ်ခုကို range နဲ့ ဖန်တီးလိုက်တာပါ။

```
y = range(10)
```

ဒါဆိုရင်သုညကနေ ကိုးထိ ရှိတဲ့ စာရင်းတစ်ခုကို range နဲ့ ဖန်တီးလိုက်တာပါ။ကွင်း အတွင်းမှာ ကိန်းဂဏန်းတစ်ခုဘဲ ရေးထားရင် Default အားဖြင့် 0 ကနေ စယူပါသည်။

အသုံးပြုပုံ

```
range ( စမှတ် , ဆုံးမှတ် , အစီအစဉ် )
```

နမူနာ

2 ကနေ 2000 ထိ စုံကိန်းတွေပါတဲ့ စာရင်း ဖန်တီးပြပါမယ်။

စမှတ် = 2

ဆုံးမှတ် = 2000

အစီအစဉ် = 2 ခုစီခြား

ကိုယ့်ဘာသာစမ်းကြည့်ပါ။

```
x = range ( 2 , 2000 , 2)
print (x)
```

နမူနာ

```
print(range(20)==range(0,20))
```

```
>>>true
```

Looping

စာကြောင်းတွေ (သို့မဟုတ်) codeတွေကို user အလိုရှိသလောက် အကြိမ်အရေအတွက်နဲ့ အညီ ထပ်ခါထပ်ခါ ထုတ်ပေးတာကို ခေါ်ပါတယ်။

While loop

While loop ကတော့ တစ်ကြိမ်ထက်ပိုပြီး run လိုရပါတယ်။whileရဲ့ နောက်မှာ true ဖစ်တဲ့ condition ရယ် false ဖစ်တဲ့ condition ရယ် ဆိုပြီး condition နှစ်ခု လိုက်လို့ ရပါတယ်။condition မှန်နေသရွေ့ while loopထဲက statement သာ အလုပ်လုပ် နေမှာဘဲ ဖစ်ပါတယ်။

Syntax

while အခြေအနေ :

လုပ်ဆောင်ချက်

Example:

```
num = 1
while num < 10 :
   print ("num")
   #incrementing the value of num
   num = num + 3
```

Output:

1

4

7

Infinite While Loop

Infinite while loop ဆိုတာ ဘယ်တော့မှအဆုံးသတ်သွားခြင်းမရှိသည့် looping များဖြစ်ပါသည်။

Exampl1:

```
while True:
    print("hello")
```

ရှင်းလင်းချက်

hello ဆိုသော စာကြောင်းကိုဘဲ အဆုံးမရှိ ဆက်တိုက် ထုတ်နေမှာဖြစ်ပါသည်။

Example2:

```
num = 1
while num<5:
    print(num)
```

ရှင်းလင်းချက်

Num ၏တန်ဖိုးကို ရှေ့က ဥပမာ အပုဒ် မှာလို တန်ဖိုးတိုးထားခြင်းမရှိသည့်အတွက် num၏ Assignment တန်ဖိုး 1 ကိုသာ အဆုံးမရှိ ထုတ်ပေးနေမည်သာဖြစ်သည်။

==========

For loop

For loop ကတော့ loop counter သို့ မဟုတ် loop variable ပေါ်မှာ မူတည်ပြီးတော့ statement တွေ codeတွေကို ထပ်တလဲလဲ ထုတ်ပေးတာဘဲ ဖြစ်ပါတယ်။ while loop နဲ့ မတူညီတဲ့ အချက်ကတော့ တိကျတဲ့ looping အရေအတွက်ကို စိတ်တိုင်းကျ သတ်မှတ်နိုင်ခြင်းဘဲ ဖြစ်ပါတယ်။

syntax:

For [inerating variable] in [sequence]:

[Do something]

Example:

For i in range (0,5):

    print(i)

Output:

0

1

2

3

4

ရှင်းလင်းပြသချက်

Range 0 to 5 အတွင်းရှိ ကိန်းများအား i အဖစ် ထုတ်ပြလိုက်ခြင်း ဖြစ်ပါသည် ။

==========

Nested for loop

Nested ဆိုသည့် အတိုင်း for loop တစ်ခုအတွင်း တွင် နောက်ထပ် for loop တစ်ခု ရှိနေခြင်းပင် ဖြစ်ပါသည်။

Syntax

For iterating variable in sequence:

    For iterating variable in sequence:

        Statement (s)

    Statement (s)

Example:

```
In [4]: for i in range(1,6):
            for j in range(i):
                print("*", end=' ')
            print(  )
```

```
*
* *
* * *
* * * *
* * * * *
```

==========

Nested While loop

Nested while loop ဆိုသည့်အတိုင်း while loop တစ်ခု အတွင်းတွင် နောက်ထပ် while loop တစ်ခု ထပ်ရှိနေခြင်း ပင် ဖြစ်ပါသည်။

syntax

```
while expression:
    while expression:
        statement(s)
    statement(s)
```

Example1:

```
In [5]:
i = 6
while(i>0):
    j=6
    while(j>i):
        print("*",end=' ')
        j-=1
    i-=1
    print()
```

```
*
* *
* * *
* * * *
* * * * *
```

Example2:

```
In [1]:
i = 1
j = 5
while i<4:
    while j<8:
        print(i,",",j)
        j = j + 1
        i = i + 1
```

```
1 , 5
2 , 6
3 , 7
```

Break and Continue

```
In [1]:
#break
for number in range(1,11):
    if number ==5:
        break
    else:
        print(number)
```

```
1
2
3
4
```

**Break က သူ့ကိုတွေ့သော နေရာမှစပြီး ဆက်လက် အလုပ်လုပ်ဆောင်နိုင်ခြင်းမရှိဘဲ Program မှ ထွက်သွားမည် ဖြစ်ပါသည်။

```
In [2]: #continue
for number in range(1,11):
    if number==5:
        continue
    else:
        print(number)
```

```
1
2
3
4
6
7
8
9
10
```

**Continue သည် သူ့ကို တွေ့သော နေရာတစ်ခုတည်းတွင်သာ အလုပ်မလုပ်ဆောင်နိုင်ခြင်းသာဖြစ်ပြီး အခြား statement များက ဆက်လက် အလုပ်လုပ်နေမည်ဖြစ်သည်။

Array

Array ဆိုသည်မှာ values များကို variable တစ်ခုထဲတွင် သိန်းဆည်းထားခြင်းဖြစ်သည်။

Example:

```
In [1]: cars = ["Ford", "Volvo", "BMW"]

In [2]: print(cars[0])
Ford
```

Array ထဲမှ element များကို index number ကို refer လုပ်ပြီးထုတ်နိုင်သည်။

Array ၏ length ကိုသိလိုပါ က lens method ကို သုံး၍ ကြည့်နိုင်သည်။

Example:

```
In [2]: cars = ["Ford", "Volvo", "BMW"]

In [3]: print(len(cars))
        3
```

Array ထဲမှ values များကို for in loop ကိုအသုံးပြု၍ထုတ်ကြည့်နိုင်သည်။

Example:

```
In [1]: cars = ["Ford", "Volvo", "BMW"]

In [2]: for x in cars:
          print(x)
        Ford
        Volvo
        BMW
```

Array ထဲသို့ value များထပ်ထည့်လိုပါက append () method ကိုသုံး၍ထည့်နိုင်သည်။

Example:

```
In [3]: cars = ["Ford", "Volvo", "BMW"]

In [4]: cars.append("Honda")

In [5]: print(cars[2])
        BMW
```

Array ထဲမှ value များကို ဖျက်ချင်ပါက pop( ) နှင့် remove( ) method ကို အသုံးပြုနိုင်သည်။

Example:

```
In [2]: cars = ["Ford", "Volvo", "BMW"]

In [3]: cars.pop(1)
Out[3]: 'Volvo'
```

```
In [2]: cars = ["Ford", "Volvo", "BMW"]

In [3]: cars.remove("Volvo")

In [ ]:
```

List

List ဆိုတာကတော့ python ရဲ့ object တစ်ခုပဲဖြစ်ပါတယ်။List တေကို array ရဲ့ item index တွေကို သိုလှောင်သိမ်းဆည်းထားဖို့အသုံးပြုပါတယ်။

*List ကို လေးထောင့်ကွင်း [ ] ဖြင့် comma (,)ပြီး ဖန်တီးနိုင်ပါတယ်။

Example: Words=[“hello”,”world”,”!”]

```
In [1]: words=["Hello","World","!"]

In [2]: print(words[1])
        World
```

လေးထောင့်ကွင်းထဲမှာ ဘာတစ်ခုမှမပါခဲ့ရင်တော့ empty list လို့ခေါ်ပါတယ်။

Example: Words=[ ]

```
In [1]: words=[]

In [2]: print(words)
        []
```

***ယေဘုယျ အားဖြင့် List ထဲမှာ single item များသာပါရှိတတ်သော်လဲ list ထဲတွင် list ထပ်၍ပါနေသော အခြေအနေမျိုးလဲရှိနိုင်သည်။

Example: Thing=[1,2,[4,5,6,7],”she”]

```
In [2]: thing=[1,2,[4,5,6,7],"she"]

In [3]: print(thing[2])
        [4, 5, 6, 7]
```

***list ထဲရှိ မူလ ထည့်ထားသော item value များကို ပြောင်းလဲ၍ရသည်။

List များကို string များကဲ့သို့ မြှောက်၍ရသလို ပေါင်း၍လဲရသည်။

Example:

```
In [1]: thing=[1,2,[4,5,6,7],"she"]

In [2]: thing[2]="Python"

In [3]: print(thing[2])
Python
```

```
In [1]: nums=[1,2,3,4]

In [2]: print(nums+[5,6,7])
[1, 2, 3, 4, 5, 6, 7]

In [4]: print(nums*3)
[1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4]
```

***item တစ်ခုဟာ list ထဲတွင်ရှိမရှိ ကို in operator သုံး၍ စစ်ဆေးနိုင်သည်။

Example:

```
In [3]: nums=[1,2,3,4]

In [5]: print(1 in nums)
True
```

***item တစ်ခု list ထဲတွင်ရှိမနေကြောင်းကိုလဲ not operator ကို ထပ်လောင်းအသုံးပြုကာစစ်ဆေးနိုင်သည်။

Example:

```
In [3]: nums=[1,2,3,4]

In [6]: print(not 10 in nums)
        True

In [7]: print(10 not in nums)
        True
```

***list ထဲသို့ value ထပ်ထည့်ရန် append method ကိုသုံး၍ ထည့်သွင်းနိုင်သည်။

Example:

```
In [1]: nums=[1,2,3,4]

In [2]: nums.append(5)

In [5]: print(nums[4])
        5
```

***List ထဲတွင်ရှိသော items အရေအတွက်ကိုသိလိုလျှင် lens function ကိုသုံး၍စစ်ဆေးနိုင်သည်။

Example:

```
In [2]: nums=[1,2,3,4]

In [4]: print(len(nums))
        4
```

***list ထဲသို့ value ထပ်ထည့်၍ရသော function တစ်ခုမှာ insert function ဖြစ်သည်။

Insert နှင့် append ကွားခြားချက်မှာ append သည် list ၏နောက်ဆုံးတွင်သာ ထပ်ထည့်၍ရသော်လည်း insert function သည် list ၏နောက်ဆုံးမှမဟုတ်ဘဲ မိမိထည့်လိုသော index နေရာ၌ ထည့်သွင်းနိုင်သည်။

Example:

```
In [1]: nums=[1,2,3,4]

In [2]: index=1

In [3]: nums.insert(index,"9")

In [4]: print(nums)
[1, '9', 2, 3, 4]
```

***index method သည် မိမိရှာလိုသော itemကို list ထဲရှိ ပထမဆုံးတွေ့ရသော index ကို return ပြန်ပေး၏။

Example:

```
In [2]: nums=[1,2,3,4]

In [4]: print(nums.index(2))
1
```

Dictionary

Dictionary ဆိုတာက data တည်ဆောက်ပုံစံ တစ်မျိုးဖြစ်ပါတယ်။Arbitrary keys တေကို values တေအဖြစ်ပြောင်းလဲတဲ့နေရာမှာသုံးပါတယ်။List များသည်လည်း သေချာသောrangeအတွင်းမှာရှိတဲ့ integer key များပါဝင်သော dictionary များပင်ဖြစ်ပါသည်။Dictionary များသည်လဲ list များကဲ့သို့ လေးထောင့်ကွင်း [ ] ကိုအသုံးပြု၍ indexing လုပ်နိုင်သည်။

Example:

```
In [1]: ages = {"Dave": 24, "Mary": 42, "John": 58}
        print(ages["Dave"])
        print(ages["Mary"])

24
42
```

Dictionary ထဲတွင်မပါဝင်သော key ကို index ပြုလုပ်ပါက keyError တက်နိုင်သည်။

Example:

```
In [1]: primary = {
          "red": [255, 0, 0],
          "green": [0, 255, 0],
          "blue": [0, 0, 255],
        }

        print(primary["red"])
        print(primary["yellow"])
```

```
[255, 0, 0]

---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-1-a9dad115fdda> in <module>
      6
      7 print(primary["red"])
----> 8 print(primary["yellow"])

KeyError: 'yellow'
```

ပြောင်းလဲ၍ မရသော objects များကိုသာ dictionary ထဲတွင် key အဖြစ်အသုံးပြုနိုင်သည်။

ပြောင်းလဲနိုင်သော objectများကိုအသုံးပြုမိပါက TypeError တက်နိုင်သည်။

Example:

```
In [1]: bad_dict = {
          [1, 2, 3]: "one two three",
        }
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-1-4999e6c58e6f> in <module>
      1 bad_dict = {
----> 2   [1, 2, 3]: "one two three",
      3 }

TypeError: unhashable type: 'list'
```

List များကဲ့သို့ dictionary. Key များကိုလဲ မတူညီသော valueများထည့်သွင်းနိုင်သည်။

List နှင့် dictionary မတူသောအချက်မှာ

```
In [1]: squares = {1: 1, 2: 4, 3: "error", 4: 16,}
        squares[8] = 64
        squares[3] = 9
        print(squares)
```

```
{1: 1, 2: 4, 3: 9, 4: 16, 8: 64}
```

Key တစ်ခု dictionaryထဲတွင်ရှိမရှိကို in,not in ဖြင့်စစ်ဆေးနိုင်သည်

Example:

```
In [1]: nums = {
          1: "one",
          2: "two",
          3: "three",
        }
        print(1 in nums)
        print("three" in nums)
        print(4 not in nums)

        True
        False
        True
```

Dictionaryတွင်အခြားအသုံးဝင်သော method တစ်ခုမှာ get methodဖြစ်သည်။

သူ၏အဓိက တာဝန်မှာ indexing ပင်ဖြစ်သော်လည်း အကယ်၍ dictionaryထဲတွင်key မရှိပါက default noneအစား သတ်မှတ်ပေးလိုက်သော value ကိုreturn ပြန်နိုင်သည်။

Example:

```
In [1]: pairs = {1: "apple",
           "orange": [2, 3, 4],
           True: False,
           None: "True",
         }

        print(pairs.get("orange"))
        print(pairs.get(7))
        print(pairs.get(12345, "not in dictionary"))

        [2, 3, 4]
        None
        not in dictionary
```

File Handling

python တွင် ဖိုင်များကို ဖန်တီးခြင်း ဖတ်ခြင်း အသစ်ထပ်ထည့်ခြင်း ဖျက်ပစ်ခြင်း စသည်တို့ လုပ်ဆောင်နိုင်သော function များစွာရှိသည်။ Files တွေရဲ့contentsတွေကို read ,write နဲ့ append လုပ်ဖို့ အတွက်open()ဆိုတဲ့ function ကို အသုံးပြုနိုင်ပါတယ်။

Syntax:

Variable= open(“filename.txt”)

Variable.close()

open() function တွင် filename နှင့် mode ဟူသော တန်ဖိုးနှစ်ခုကို ပါဝင်ပစ္စည်းအဖြစ် ထည့်သွင်းရသည်။

အသုံးပြုပုံ

open(ဖိုင်အမည် , ဖွင့်မည့်ပုံစံ)

filename နေရာတွင် ဖွင့်မည့် ဖိုင်အမည်ကို ထည့်ရမည်။

mode သည် ဖိုင်ဖွင့်မည့် ပုံစံကို ဖြစ်ပြီး လေးခုရှိသည်။

==========

ဖိုင်ဖွင့်နည်း လေးမျိုး

"r" mode

အရှည်ကောက်မှာ read mode ဖြစ်ပြီး ရှိပြီးသားဖိုင်များကို ဖတ်ရန်အတွက် ဖွင့်ခြင်း ဖြစ်သည်။ mode တန်ဖိုးမထည့်ပေးလျှင် read mode အဖြစ် ဖွင့်ပေးမည်။

ဖိုင်မရှိလျှင် error တက်မည်။

"a" mode

အရှည်ကောက်မှာ append mode ဖြစ်ပြီး ဖိုင်များကို တန်ဖိုးအသစ်ထပ်ထည့်ရန်အတွက် ဖွင့်ခြင်း ဖြစ်သည်။

ဖိုင်အဟောင်းရှိလျှင် မူလရေးထားတဲ့ တန်ဖိုးထဲကို အသစ်ထပ်တိုးမည်။

ဖိုင်မရှိလျှင် အသစ်ဆောက်ပေးမည်။

"w" mode

အရှည်ကောက်မှာ write mode ဖြစ်ပြီး ဖိုင်များကို အသစ်ရေးသားရန်အတွက် ဖွင့်ခြင်း ဖြစ်သည်။

ဖိုင်အဟောင်းရှိလျှင် မူလရေးထားတာတွေကို ဖျက်ပြီး အသစ်ထည့်မည်။

ဖိုင်မရှိလျှင် အသစ်ဆောက်ပေးမည်။

"x" mode

အရှည်ကောက်မှာ create mode ဖြစ်ပြီး ဖိုင်များကို တည်ဆောက်ရန်အတွက် ဖွင့်ခြင်း ဖြစ်သည်။

ဖိုင်အသစ်ဆောက်မည်။

ဖိုင်အဟောင်းရှိလျှင် error ပြမည်။

==========

အထက်ပါ ဖိုင်ဖွင့်နည်း လေးခုတွင် t နှင့် b ဟူသော mode နှစ်ခုစီ ထပ်မံရွေးချယ်နိုင်သည်။

"t" mode

အရှည်ကောက်မှာ text mode ဖြစ်ပြီး ဖိုင်များကို စာအတွက် ဖွင့်ခြင်း ဖြစ်သည်။

(default value ဖြစ်သည်။mode တန်ဖိုးမထည့်ပေးလျှင် text mode အဖြစ် ဖွင့်ပေးမည်။)

"b" mode

အရှည်ကောက်မှာ binary mode ဖြစ်ပြီး ဖိုင်များကို 01 ပုံစံအတွက် ဖွင့်ခြင်း ဖြစ်သည်။

ရုပ်ပုံများ vedio များကို ဖွင့်ရာတွင်သုံးသည်။

==========

program ရေးသားနည်း

ဖိုင်ကို ဖတ်ခြင်း

ရှိပြီးသားဖိုင်တစ်ခုကို စာအဖြစ်ဖတ်လိုပါက ဖိုင်အမည် ရေးရုံဖြင့် လုံလောက်ပါတယ်။

```
f = open("myfile.txt")
```

==========

ဒါမှမဟုတ် ပြည့်ပြည့် စုံစုံ ရေးချင်ရင် အောက်ကလို ရေးပါ။

```
f = open("myfile.txt", "rt")
```

"r" နဲ့ "t" mode က default value ဖြစ်လို့ မထည့်လည်းရပါတယ်။

==========

server တစ်ခုပေါ်တွင် ရှိပြီးသားဖိုင်ကို ဖွင့်ဖတ်ခြင်း

ဖွင့်ခြင်းအပိုင်း

open() function မှာ read mode နဲ့ဖွင့်ပါတယ်။

ဖတ်ခြင်းအပိုင်း

read () function နဲ့ ဖတ်ပါတယ်။

mobile C မှာ folder တစ်ခုဆောက်ပါ။

folder ထဲမှာ xyz.txt ဆိုတဲ့ ဖိုင်တစ်ခုဆောက်ပြီး

abcdefg

hijklmnop

လို့ ရေးပါ။

file lesson.1.py ဆိုတဲ့ ဖိုင်ဆောက်ပြီး အောက်ကအတိုင်းရေးပါ။

```
f = open("xyz.txt", "r")
print(f.read())
```

runရင်

abcdefg

hijklmnop လိုပြပါလိမ့်မယ်။

==========

မိမိဖတ်လိုသော အစိတ်အပိုင်းကိုရွေးချယ်ဖတ်ခြင်း

ပထမဆုံးစာငါးလုံးကို ဖတ်ပါ့မယ်။

file lesson.2.py ဆိုတဲ့ ဖိုင်တစ်ခုဆောက်ပြီး အောက်က program ကို ရေးပါ။

```
f = open("xyz.txt", "r")
print(f.read(5))
```

runရင်

abcd

လို့ပြပါလိမ့်မယ်။

ငါးလုံးထိဖတ်တာဆိုတော့ ငါးလုံးထိတာနဲ့ရပ်တော့ လေးခုပဲပြပါတယ်။

range အကြောင်းရရင် စိတ်ကြိုက် ရွေးနိုင်ပါတယ်။

နောက်ဆုံးငါးလုံးဆိုရင် -5 ပေါ့။

==========

readline ()

တစ်ကြောင်းချင်းဖတ်ရင် readline() function ကို သုံးပါတယ်။

ဘာမှမရေးရင် တစ်ကြောင်းလုံးဖတ်ပါမယ်။

file lesson.3.py ဆိုတဲ့ ဖိုင်တစ်ခုဆောက်ပြီး အောက်က program ကို ရေးပါ။

```
f = open("xyz.txt", "r")
print(f.readline())
```

runရင်

abcdefg လို့ပြပါလိမ့်မယ်။

==========

ပထမအကြောင်းက ရှေ့သုံးလုံးကို ဖတ်ပါ့မယ်။

ပြီးရင် ကျန်တဲ့စာကို ဆက်ဖတ်ပါ့မယ်။

file lesson.4.py ဆိုတဲ့ ဖိုင်တစ်ခုဆောက်ပြီး အောက်က program ကို ရေးပါ။

```
f = open("xyz.txt", "r")
print(f.readline(4))
print(f.readline())
```

run ရင်

```
abc
defg
```

==========

နှစ်ခါရေးရင် ပထမဆုံးနှစ်ကြောင်းဖတ်ပါတယ်။

file lesson.5.py ဆိုတဲ့ ဖိုင်တစ်ခုဆောက်ပြီး အောက်က program ကို ရေးပါ။

```
f = open("xyz.txt", "r")
print(f.readline())
print(f.readline())
```

runရင်

abcdefg

hijklmnop

လို့ပြပါလိမ့်မယ်။

==========

အားလုံးကို တစ်ကြောင်းချင်းဖတ်ချင်ရင် loop ပတ်လို့ရပါတယ်။

file lesson.6.py ဆိုတဲ့ ဖိုင်တစ်ခုဆောက်ပြီး အောက်က program ကို ရေးပါ။

```
f = open("xyz.txt", "r")
for x in f:
  print(x)
```

runရင်

abcdefg

hijklmnop

လို့ပြပါလိမ့်မယ်။

==========

Close Files

ဖိုင်ပိတ်ခြင်း

ဖိုင်များကို ဖွင့်ပြီးတိုင်းပြန်ပိတ်ဖို့မမေ့ပါနဲ့။

မပိတ်လည်းရကောင်းရပေမယ့် ပိတ်တာက ကောင်းတဲ့အလေ့အကျင့်တစ်ခုပါ။

file lesson.7. py ဆိုတဲ့ ဖိုင်တစ်ခုဆောက်ပြီး အောက်က program ကို ရေးပါ။

```
f = open("xyz.txt", "r")
print(f.readline())
f.close()
```

run ရင်

abcdefg

တစ်ချို့နေရာတွေမှာ ဖိုင်ကို ပိတ်မှ ပြောင်းလိုက်တဲ့အရာတွေ သက်ရောက်တာပါ။

==========

Python File Write

ဖိုင်တွင် ရေးသားခြင်း

mode နှစ်မျိုးရှိပါတယ်။

"a" mode

ဖိုင်ရဲ့မူလတန်ဖိုးအဆုံးကနေ အသစ်ထပ်ထည့်မယ်။

"w" - mode

ဖိုင်ကို အကုန်ဖျက်ပြီး အစကနေ ပြန်ရေးမယ်။

==========

append mode

file lesson.8. py ဆိုတဲ့ ဖိုင်တစ်ခုဆောက်ပြီး အောက်က program ကို ရေးပါ။

```
f = open("xyz.txt", "a")
f.write("Hello Files")
```

```
f.close()
```

run ရင် ဘာမှမပြပါဘူး။

ဒါပေမယ့် xyz.txt ကို ဖွင့်ကြည့်ရင် အောက်ကလို အသစ်တိုးလာပါလိမ့်မယ်။

```
abcdefg

hijklmnopHello Files
```

==========

write mode

file lesson.9. py ဆိုတဲ့ ဖိုင်တစ်ခုဆောက်ပြီး အောက်က program ကို ရေးပါ။

```
f = open("xyz.txt", "w")

f.write("Hello Files")

f.close()
```

run ရင် ဘာမှမပြပါဘူး။

ဒါပေမယ့် xyz.txt ကို ဖွင့်ကြည့်ရင် အောက်ကလို အသစ်ပြောင်းပါလိမ့်မယ်။

```
Hello Files
```

==========

Create a New File

ဖိုင်အသစ်ဖန်တီးခြင်း

ဖိုင်အသစ်ဖန်တီးခြင်းသုံးမျိုးရှိပါတယ်။

"x" - Create mode

"a" - Append mode

"w" - Write mode

==========

create mode

my self.txt ဆိုတဲ့ ဖိုင်တစ်ခုတည်ဆောက်ပါ့မယ်။

file lesson.10. py ဆိုတဲ့ ဖိုင်တစ်ခုဆောက်ပြီး အောက်က program ကို ရေးပါ။

```
f = open("my self.txt", "x")
```

run ရင် ဘာမှမပြပါဘူး။

ဒါပေမယ့် အပြင်မှာ my self.txt ဆိုတဲ့ ဖိုင်တစ်ခု ဖြစ်လာပါလိမ့်မယ်။

မရှိသေးတဲ့ဖိုင်အမည် အသစ်ကို ဖွင့်ချင်ရင်

x mode

a mode

w mode

ကြိုက်ရာသုံးနိုင်ပါတယ်။

အပေါ်နမူနာရဲ့ x နေရာမှာ a သို့မဟုတ် w ပြောင်းစမ်းရုံပါပဲ။

==========

ဖိုင်ကို ဖျက်ခြင်း

os module ထဲက remove () function နဲ့ဖျက်ပါတယ်။

file lesson.11. py ဆိုတဲ့ ဖိုင်တစ်ခုဆောက်ပြီး အောက်က program ကို ရေးပါ။

```
import os
os.remove("my self.txt")
```

run ရင် ဘာမှမပြပါဘူး။

my self.txt ဆိုတဲ့ဖိုင်တော့ ဖျက်ခံရပါတယ်။

==========

ဖိုင်ရှိမရှိစစ်ခြင်း

exists () function ကို သုံးပါတယ်။

ရှိရင် True ဖြစ်ပါတယ်။

file lesson.12. py ဆိုတဲ့ ဖိုင်တစ်ခုဆောက်ပြီး အောက်က program ကို ရေးပါ။

```
import os
if  os.path.exists("my self.txt"):
  os.remove("my self.txt")
else:
  print("The file does not exist.")
```

run ရင်

The file does not exist.

လို့ပြပါတယ်။

ဘာကြောင့်လဲဆိုတော့ ဖိုင်မရှိလို့ပါ။

==========

Delete Folder

ဖိုင်တွဲကို ဖျက်ခြင်း

os.rmdir() method နဲ့ ဖျက်ပါတယ်။

ဘာတန်ဖိုးမှမပါတဲ့ ဖိုင်တွဲပဲဖျက်ပေးပါတယ်။

file lesson.13. py ဆိုတဲ့ ဖိုင်တစ်ခုဆောက်ပြီး အောက်က program ကို ရေးပါ။

စမ်းမပြတော့ပါဘူး။

```
import os

os.rmdir("ဖိုင်တွဲအမည်")
```

Exception

What is an Exception?

Exception ဆိုတာ program တစ်ပုဒ်ကို execution လုပ်လိုက်တဲ့ အခါမှာ ဖြစ်ပေါ်လာတဲ့ error တစ်မျိုးပါပဲ။

အဲ့လိုမျိုးerror တွေ ဖြစ်တဲ့အခါ program က exception တစ်ခုကို ထုတ်ပေးပါတယ်။ Exception တွေကို program မှာ ဖြစ်တတ်တဲ့errors တွေနှင့် special conditions တွေကိုကိုင်တွယ်နိုင်ဖို့ နည်းလမ်း အများ ကြီးမှာ အသုံးပြုပါတယ်။

Exception Errors

Types of Errors	**Descriptions**
IOError	If the file cannot be opened.
ImportError	If the module cannot find
ValueError	ပေးလိုက်တဲ့ argument က type တူ ၊တန်ဖိုး မတူ
KeyboardInterrupt	User က မှားနှိပ်မိတာမျိုး
EOFError	End-of-file condition

အခြား Exception များလည်းရှိပါသည်။ အခု စမ်းပြချင်တဲ့ Exception ကတော့ ZeroDivisionException ဖြစ်ပါတယ်။ သူက အပိုင်းကိန်းတွေမှာ ပိုင်းခြေက zero ဖြစ်ရင် စားလို့မရတာကြောင့် တက်တဲ့ exception ဖြစ်ပါတယ်။လေ့လာကြည့်ပါ။

```
In [2]: length=10
        width=0
        length/width
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-2-1c41d4f3fe73> in <module>
      1 length=10
      2 width=0
----> 3 length/width
      4

ZeroDivisionError: division by zero
```

ပိုင်းခြေ widthက 0 ဖြစ်နေတဲ့အတွက် ZeroDivisionException တက်လာပါတယ်။

```
In [3]: #ZeroDivisionError: division by zero
        try:
            length=10
            width=0
            length/width
        except Exception:
            print("Division by zero is invalid!Kindly change your input")
```

```
Division by zero is invalid!Kindly change your input
```

Exception တက်ခြင်းကို ဖြေရှင်းချင်ရင် Exception ကိုဖြစ်ပေါ်စေတဲ့ စာကြောင်းတွေကို try ထဲတွင် ရေးသားပြီး Exception တက်ရင် ဖြစ်ပေါ်စေချင်သော စာကြောင်းများကို except Exception သို့မဟုတ် except Exceptionရဲ့ Type (ဥပမာ ။ except ZeroDivisionError)ဟူ၍အတွင်းရေးသားနိုင်ပါတယ်။

==========End==========

Written By

Mg Thwin Htoo Aung(UTYCC)

Mg Pau Deih Tung(UTYCC)

Ma Myint Cho Cho Kyaw(UTYCC)

Ma Khine Thet Thet Zaw(UTYCC)

Supporting Resources:

www.tutorailgateway.org

www.programiz.com/python-programming

www.facebook.com/groups/225596184681998

www.anaconda.com

Photo Credits

All images © Shutterstock, Inc
Screenshot of Jupiter Notebook