BIG DATA INSTITUTE
สถาบันข้อมูลขนาดใหญ่

Python Basics

Big Data Institute (BDI):
4 August 2023

Angkhana Prommarat, Ph.D.
Data Scientist

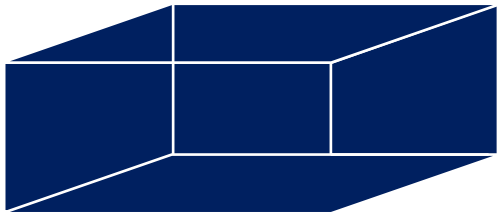
Original materials developed by Chainarong Tumapha & Wisith Wanishyanon

Agenda

BIG DATA INSTITUTE

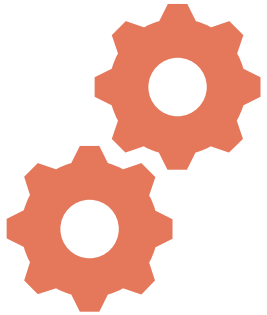
- Introduction to Python Programming
- Colab Setup
- 01_Expression
- 02_Variables
- 03_Operation
- 04_Data Types
- 05_Conditions
- 06_Loops
- 07_Function
- 08_Numpy
- 09_Pandas

Fundamental programming



Variables

เก็บข้อมูล

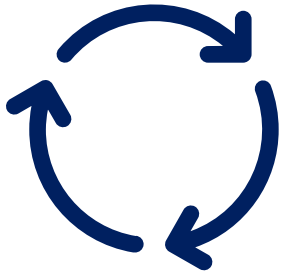


Function

เรียกใช้ชุดคำสั่ง

Flow control

ตัดสินใจ



Loop

ทำซ้ำ

Raking of programming languages in 2023

BIG DATA INSTITUTE

Programming Language	Ratings	Change
Python	13.45%	+0.71%
C	13.35%	+1.76%
Java	12.22%	+1.22%
C++	11.96%	+3.13%
C#	7.43%	+1.04%
Visual Basic	3.84%	-2.02%
JavaScript	2.44%	+0.32%
PHP	1.59%	+0.07%
SQL	1.48%	-0.39%
Assembly language	1.20%	-0.72%

Delphi/Object Pascal	1.01%	-0.41%
Go	0.99%	-0.12%
Scratch	0.95%	+0.29%
Swift	0.91%	-0.31%
MATLAB	0.88%	+0.06%
R	0.82%	-0.39%
Rust	0.82%	+0.42%
Ruby	0.80%	-0.06%
Fortran	0.78%	+0.40%
Classic Visual Basic	0.75%	-0.28%

TIOBE Index - TIOBE

Why Python?

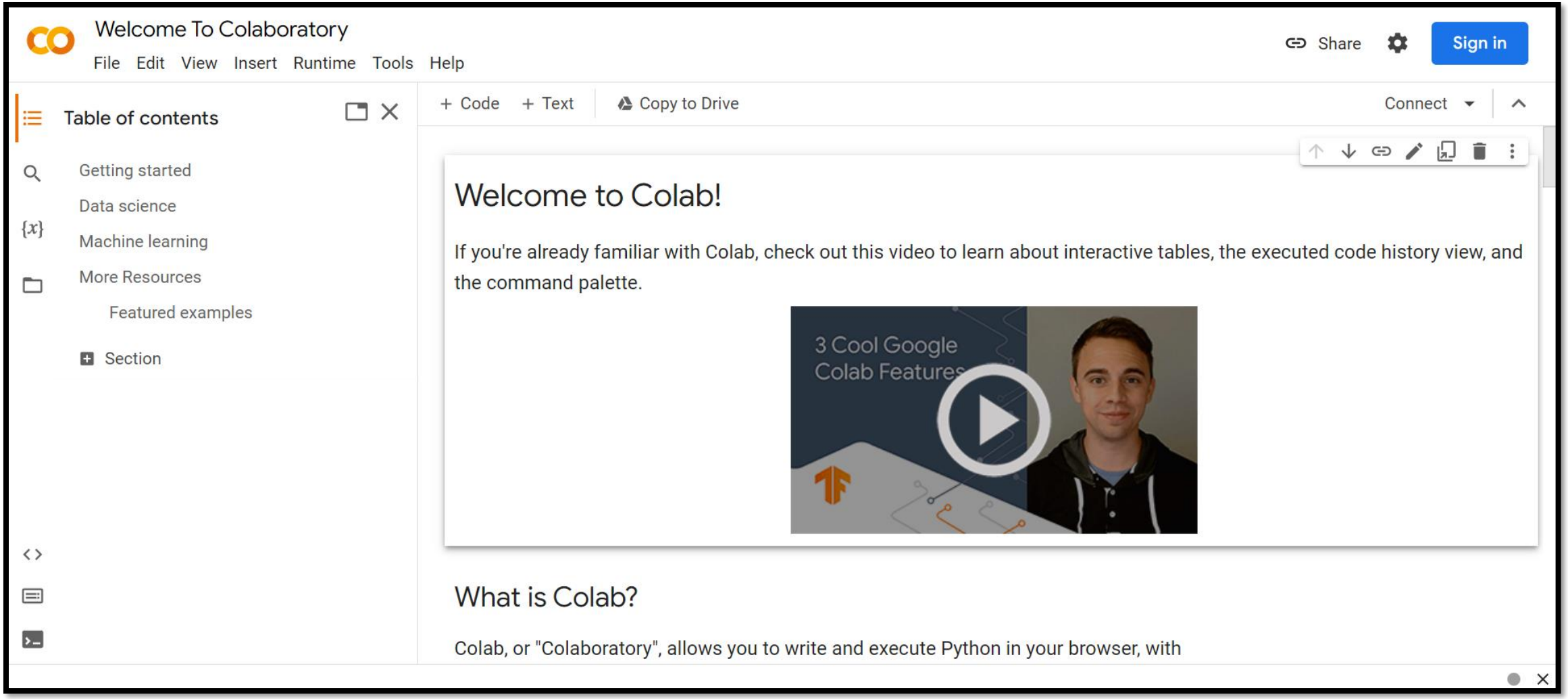
- ✓ ง่ายต่อการเรียนรู้
- ✓ สามารถนำไปใช้งานจริงได้ ตัวอย่างผลิตภัณฑ์ซอฟแวร์ เช่น Google, Facebook, Netflix
- ✓ มีชุมชนนักพัฒนาที่แข็งแกร่ง
- ✓ มี library คลอบคลุมการใช้งาน
- ✓ มีการใช้งานอย่างแพร่หลาย โดยเฉพาะในด้าน Data Science
- ✓ ใช้งานได้ฟรี

Reference: https://python-thailand.github.io/intro/why-python.html

Google Colab Setup

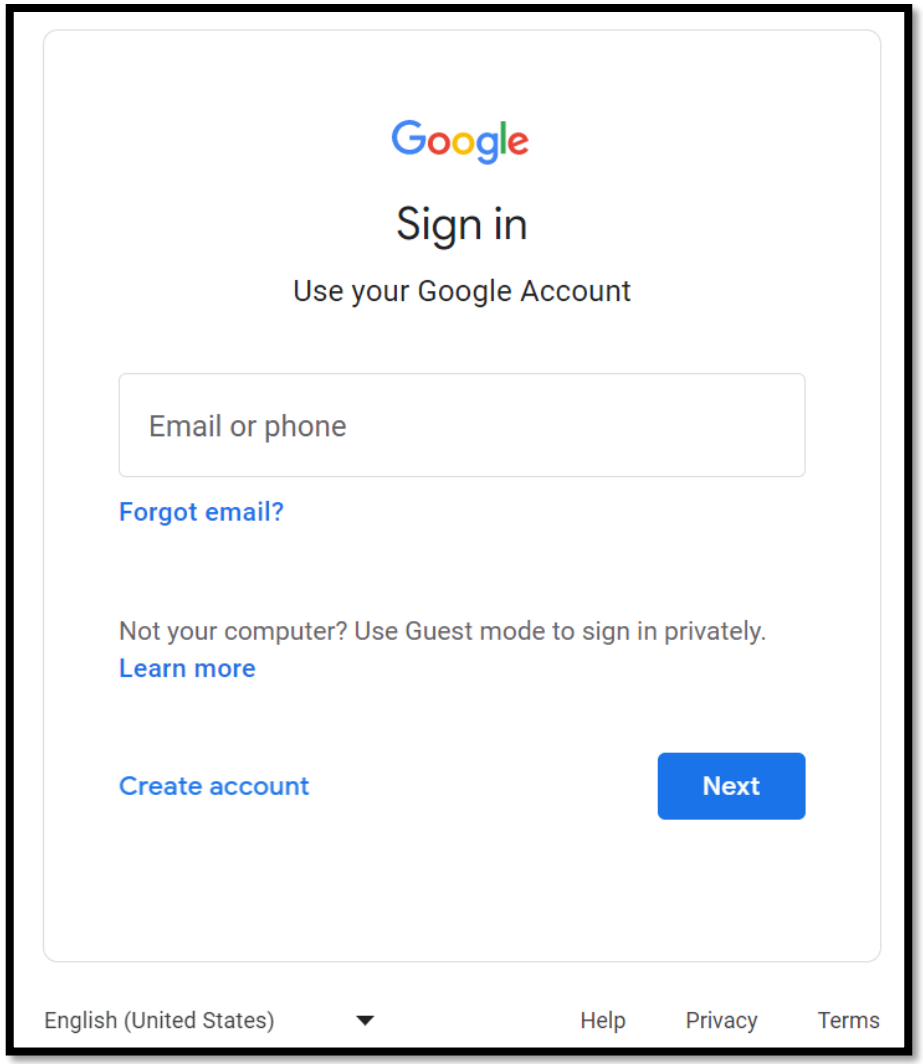
Google Colab Setup

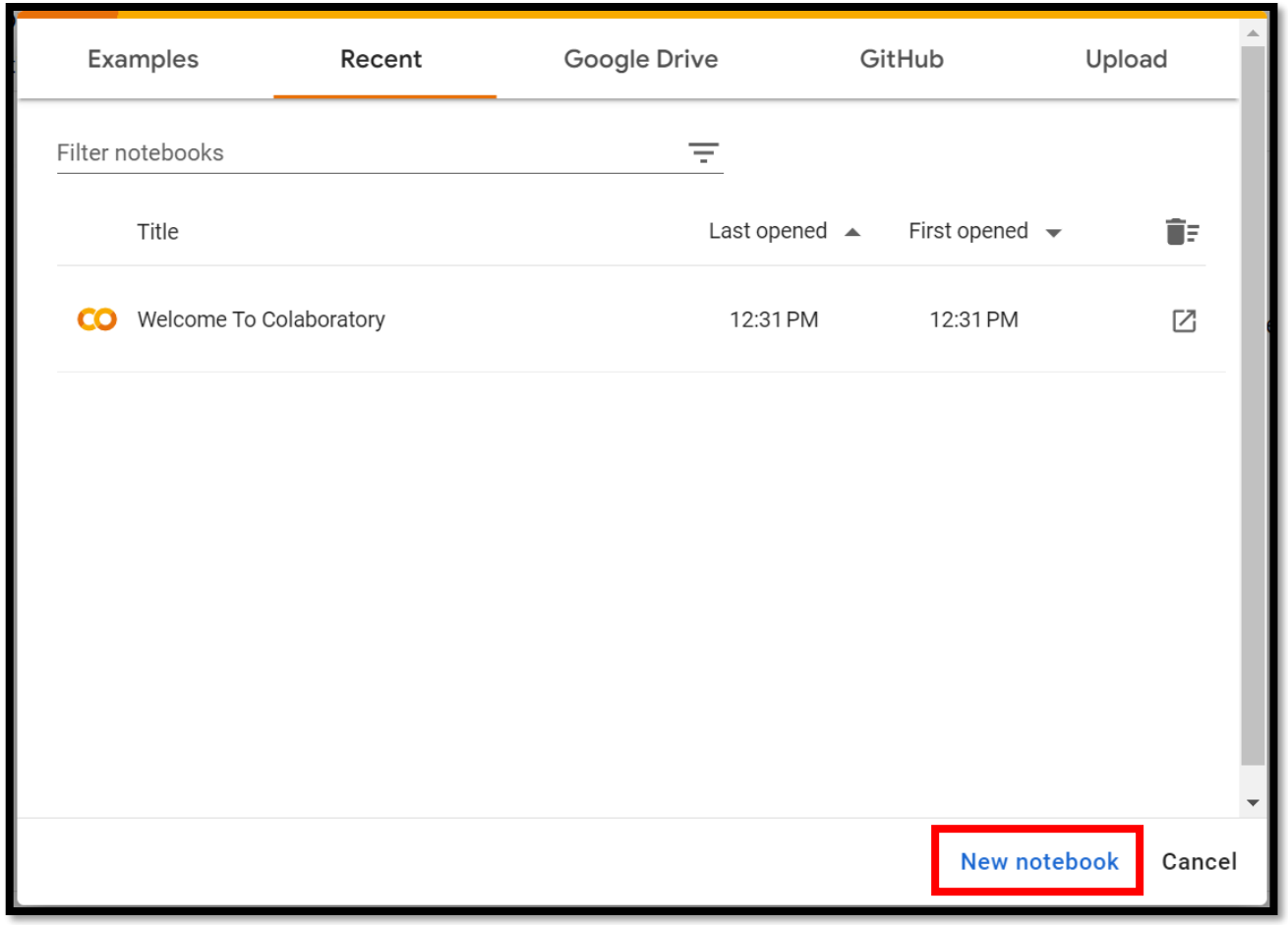
1) เข้าไปที่ google colab: https://colab.research.google.com



2) กรอก Email และ Password เพื่อ Sign in เข้าสู่ระบบ

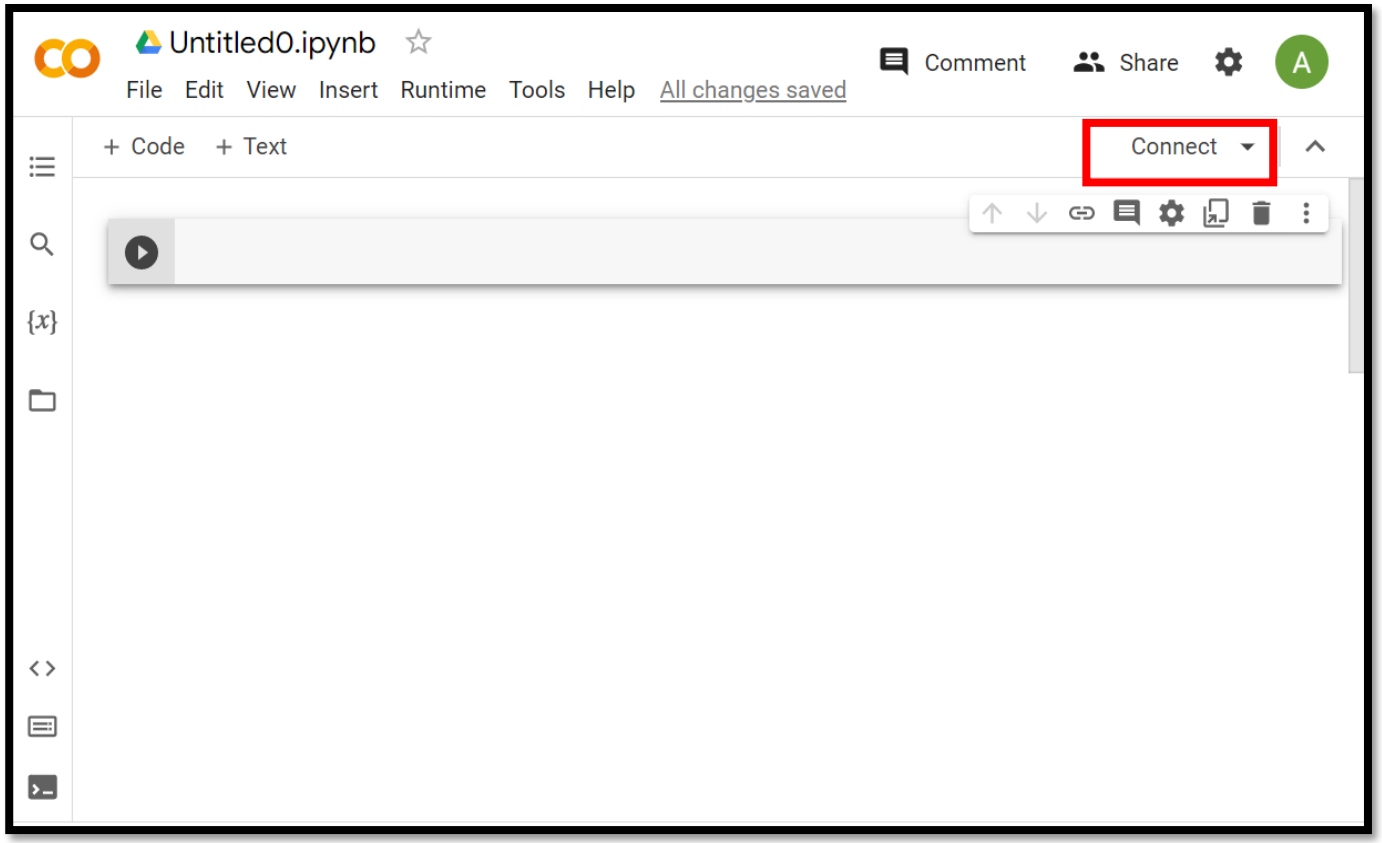
3) คลิกเลือก New notebook

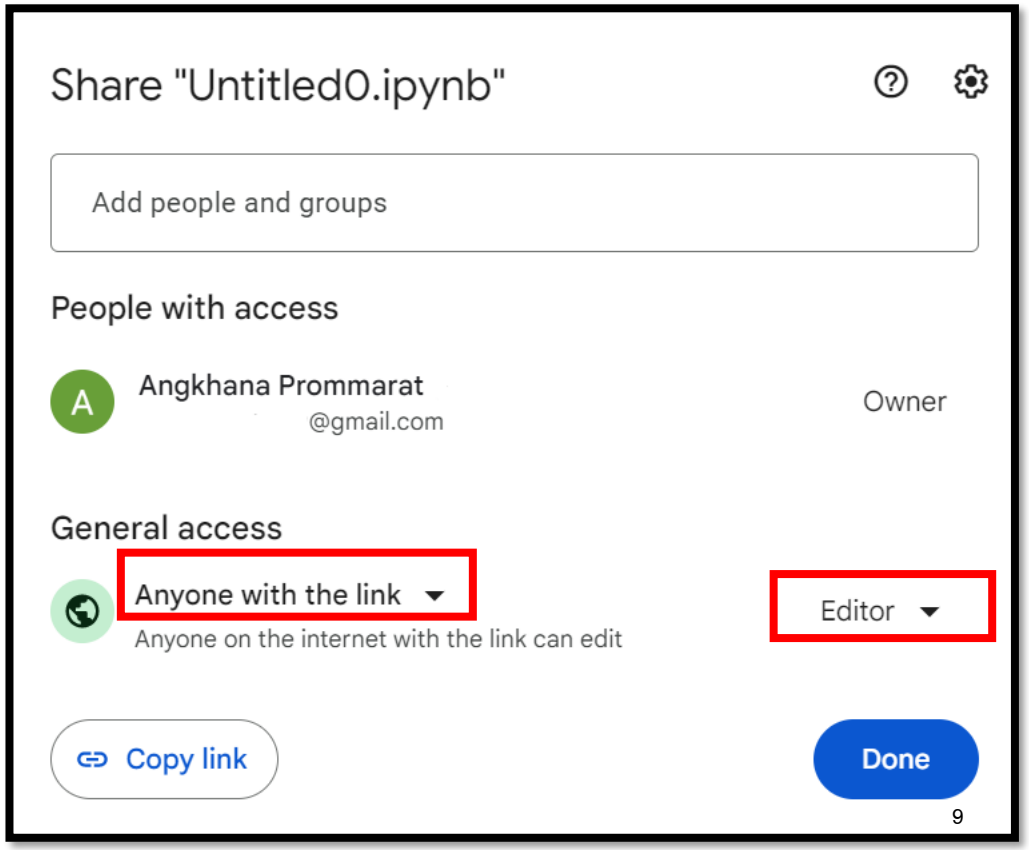




4) ตรวจสอบสถานะ Connect ของ colab

5) คลิก Share ปรับการเข้าถึงให้ทุกคนที่มี link สามารถเข้าถึงได้ โดยคลิกเลือก “Anyone with the link” และปรับการอนุญาตให้สามารถแก้ไขได้ โดยคลิก “Editor”



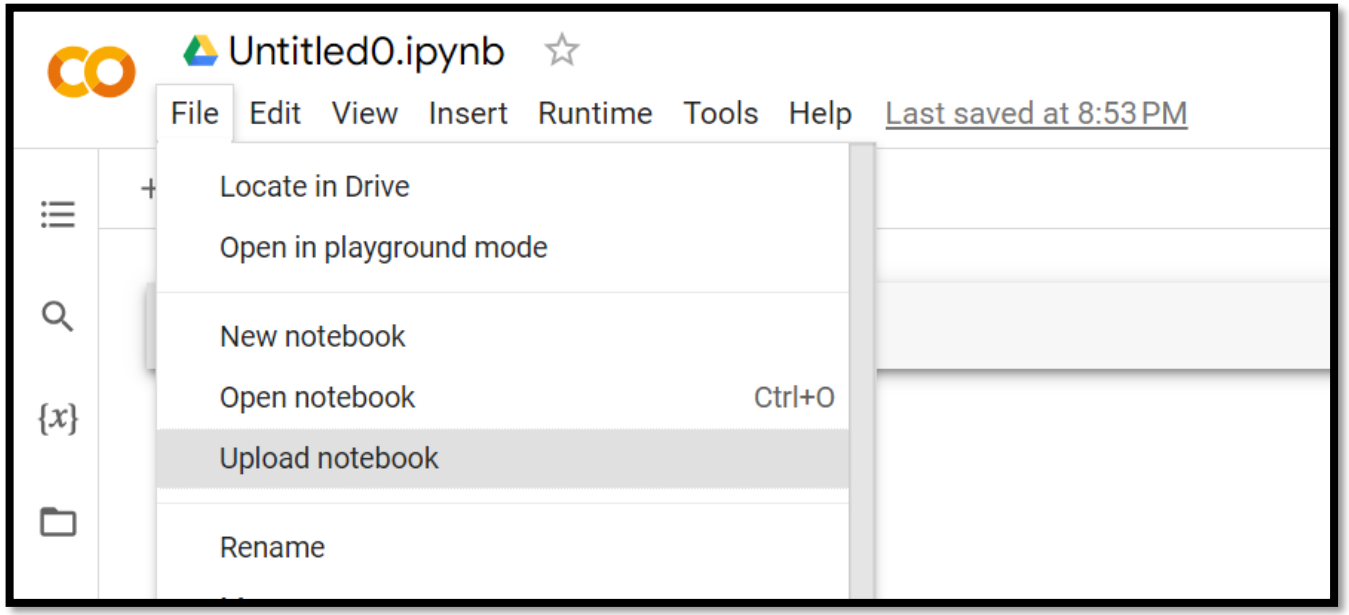


BIG DATA INSTITUTE

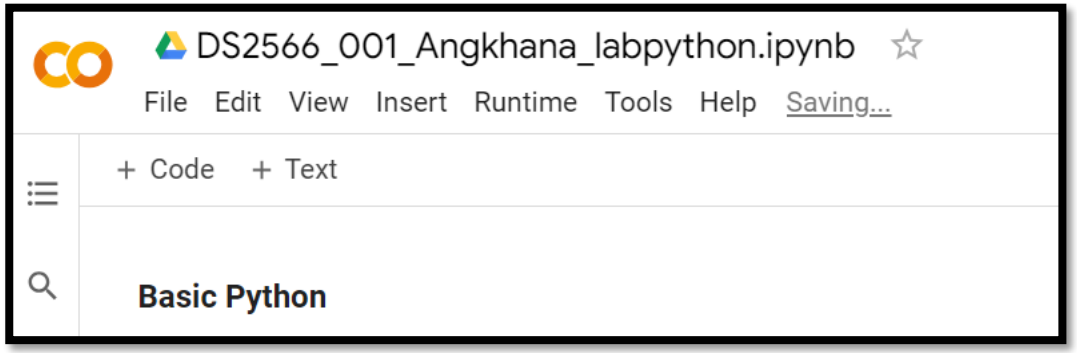
Download เอกสาร

https://shorturl.ac/7arre

6) เข้าไปที่ Google Colab >> File >> Upload notebook >> Upload >> Choose File **DS2566_ID_Name_labpython.ipynb**.



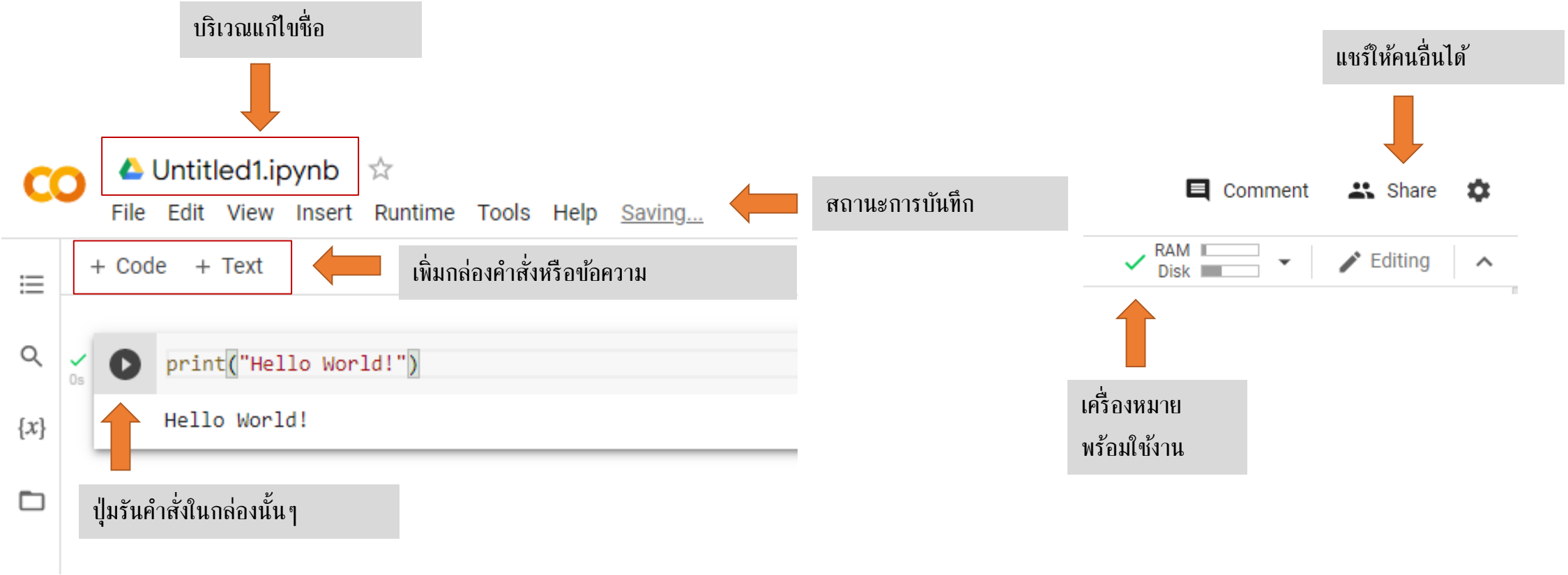
7) เปลี่ยนชื่อไฟล์ โดยใส่ DS2566_ID_ และตามด้วยชื่อภาษาอังกฤษ เช่น DS2566_001_Angkhana_labpython และย้อนกลับไปตั้งค่าการแชร์ใหม่ตามข้อ 5)



01_EXPRESSION

Example 1 : Hello World!

พิมพ์คำสั่ง print(“Hello World!”) ลงใน Google Colab



Comments

Python will not run comments

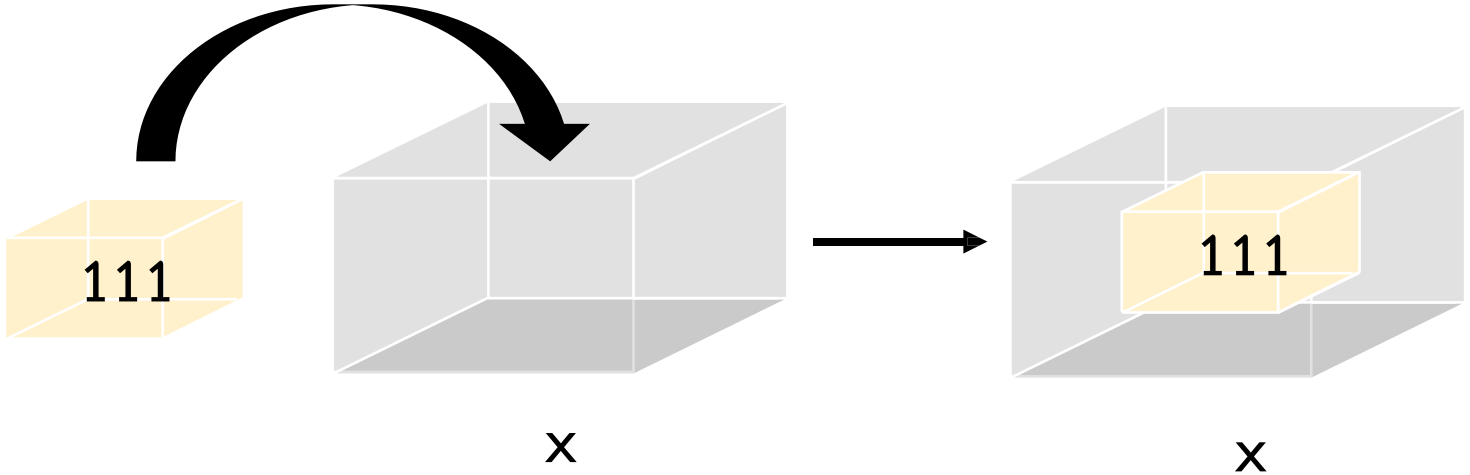
```
#A comment. What you write here will not be executed.
print("Hello World!")
```

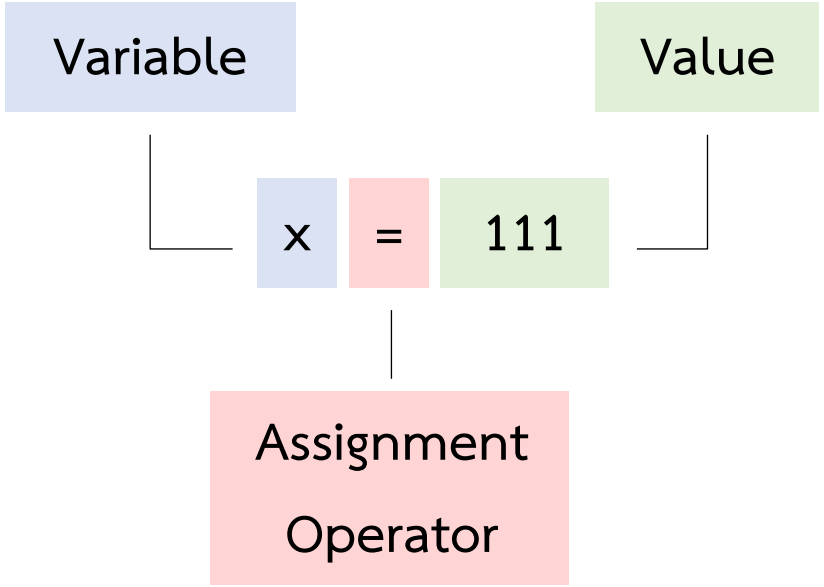
```
print("This is Python Class") #This is a comment
```

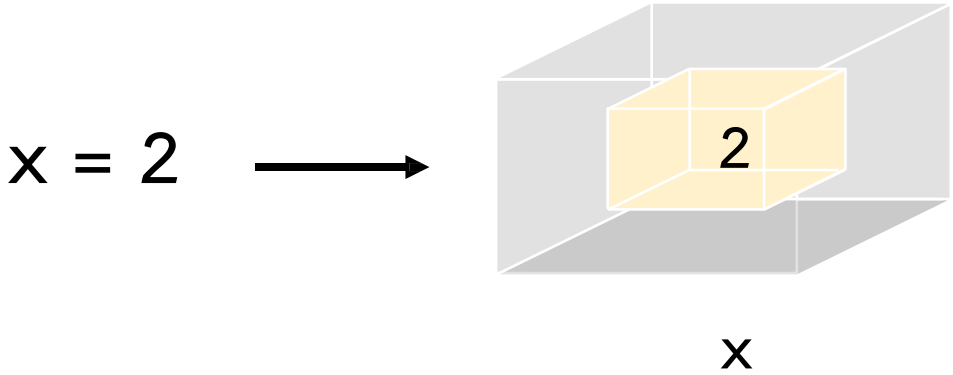
```
"""

What you write here will not be executed
"""

print("This is Python Class")
```

02_VARIABLES

Variables







Please visit this link for more information.

Variable Naming Rules

- Must start with a-z, A-Z, or the _ sign (underscore)
- Within the variable name contains a-z, A-Z, 0-9, or the _ sign
- Do not be spaces or other markings
- Case sensitive
- Do not use reserved words in naming

True	False	and	or	not
None	if	else	elif	is
in	def	del	except	finally
for	from	global	import	lambda
pass	break	class	continue	return
yield	raise	try	while	with
nonlocal	as	assert		

Example 2: Variables

```
Myvar      = "BDI"
my_var     = "BDI"
MyVar01    = "BDI"
_my_var    = "BDI"
```

```
5my_var = "BDI"
My-Var  = "BDI"
My var  = "BDI"
return  = "BDI"
```

03_OPERATION

Operators

BIG DATA INSTITUTE

ARITHMETIC OPERATORS

Operator	Operation
+	Addition
-	Subtraction
*	Multiplication
/	Division
//	Floor Division
%	Modulo
**	Power

Example 3: Arithmetic operator

Python	
Input	a=56 b=5 print(a, '+' ,b ,'=' ,a+b) print(a, '-' ,b , '=' ,a-b) print(a, '*' ,b , '=' ,a*b) print(a, '/' ,b , '=' ,a/b) print(a, '//' ,b , '=' ,a//b) print(a, '%',b , '=' ,a%b) print(a, '^' ,b , '=' ,a**b)
Output	

ASSIGNMENT OPERATORS

Operator	Operation	Example	Equivalent form
+=	Addition	x += 2	x = x + 2
-=	Subtraction	x -= 2	x = x - 2
*=	Multiplication	x *= 2	x = x * 2
/=	Division	x /= 2	x = x / 2
//=	Integer Division	x //= 2	x = x // 2
%=	Congruent Modulo	x %= 2	x = x % 2
**=	Exponentiation	x **= 2	x = x ** 2

Example 4: Assignment operator

Python	
Input	x=5 x+=2 print(x)
Output	

COMPARISON OPERATORS

Operator	Description	Example
==	Is equal to	1==1 give us **True**
!=	Not equal to	3!=4 give us **True**
>	Greater than	2>8 give us **False**
<	Less than	5<7 give us **True**
>=	Greater than or equal to	9>=7 give us **True**
<=	Less than or equal to	5<=3 give us **False**

Example 5: Comparison operator

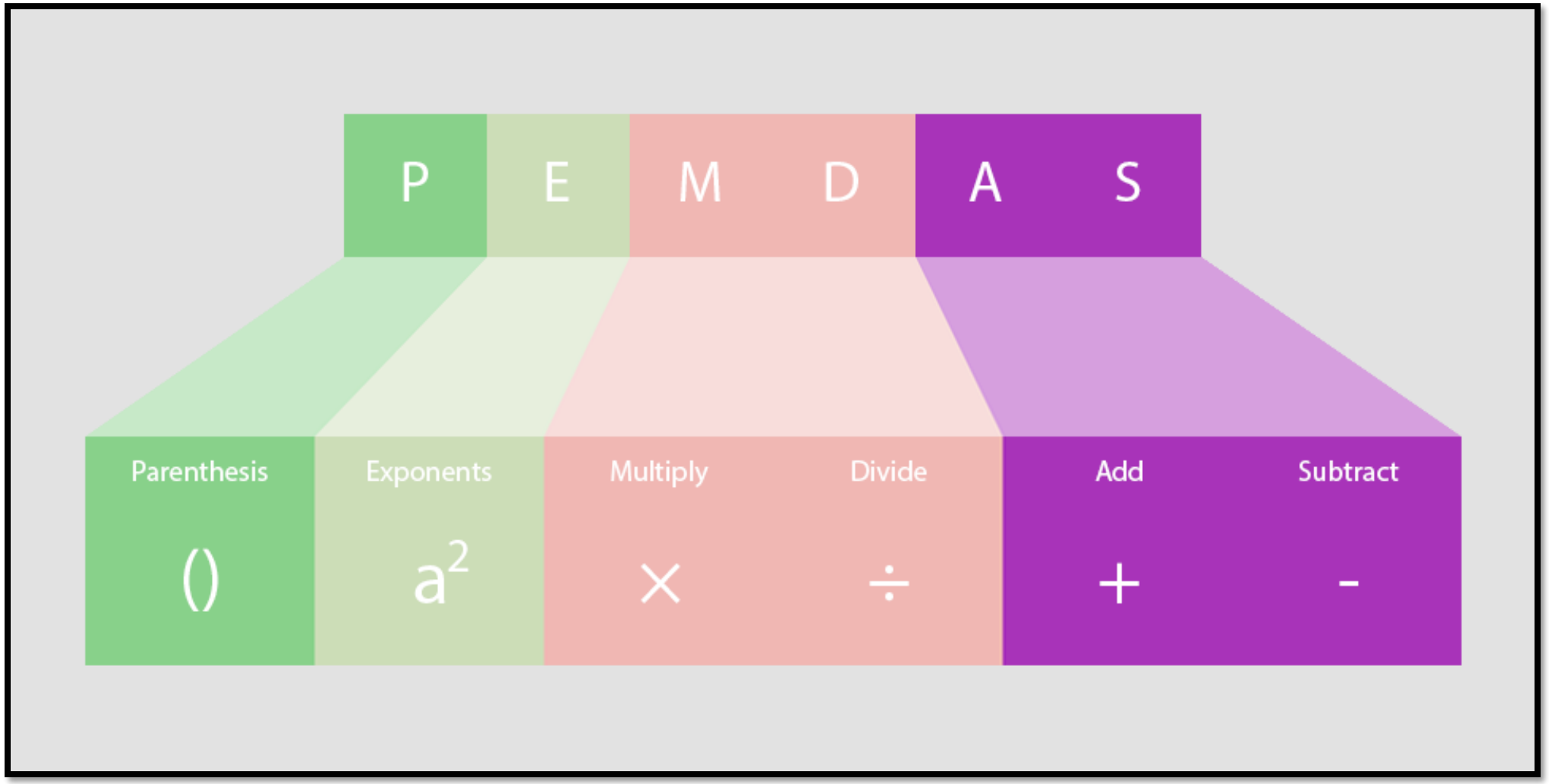
Python	
Input	a=56 b=4 print(a,'==',b, a==b) print(a,'!=',b, a!=b) print(a,'>',b, a>b) print(a,'<',b, a<b) print(a,'>=',b, a>=b) print(a,'<=',b, a<=b)
Output	

What is the answer to this question?

BIG DATA INSTITUTE

7+4*3**2/1-5

PRECEDENCE OF OPERATOR



https://nuuneoi.com/blog/blog.php?read_id=896

What is the answer to this question?

BIG DATA INSTITUTE

7+4*32/1-5**

Example 6: Precedence of operator

Python	
Input	7+4*3**2/1-5
Output	38.0

7 + 4*(3**2)/1 - 5
7 + 4* 9 / 1 - 5
7 + (4* 9)/1 - 5
7 + 36 / 1 - 5
7 + (36 / 1) -5
7 + 36 - 5
(7 + 36) -5
43-5
38

LOGICAL OPERATORS

BIG DATA INSTITUTE

Truth Table

Expression	Result
True **and** True	True
True **and** False	False
False **and** True	False
False **and** False	False

Expression	Result
True **or** True	True
True **or** False	True
False **or** True	True
False **or** False	False

Expression	Result
not True	False
not False	True

Operator	Description	Example
and	Returns True if both statements are true	x **and** y
or	Returns True if one of the statements is true	x **or** y
not	Reverse the result	**not** x

LOGICAL OPERATORS

Example 7: Logical operator

กำหนดให้ x=82 แล้ว x<=100 แสดงผลอย่างไร ???

Python	
Input	**X=82** **X<=100**
Output	

Python	
Input	**not** 54 % 7 > 4 **or** **not** 3 != 3
Output	

Exercise 1: Operation

1. ค่า x มีค่าเท่าใดจากการคำนวณต่อไปนี้

```
x   = -73
x += 15
```

2. ค่า my_variable มีค่าเท่าใดจากการคำนวณต่อไปนี้

```
my_variable    = 157
my_variable  %= 10
```

3. ค่า y จากการคำนวณต่อไปนี้มีค่าเท่าใด

```
y = 28 / 13
```

4. กำหนดให้ y = x + 9 (ใช้ค่า x จากข้อ 1)

จงหาค่าของ y หลังจากการคำนวณต่อไปนี้

```
y /= 10
```

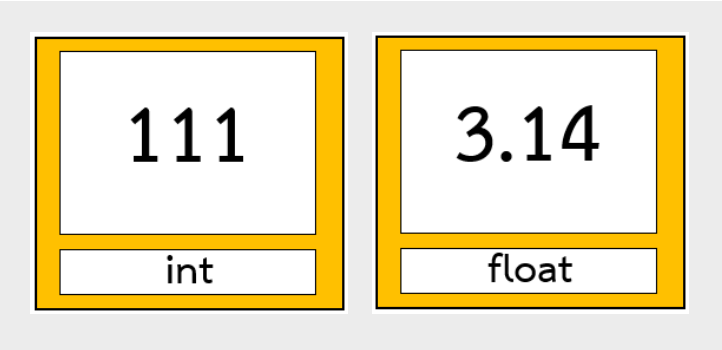
5. จงหาค่าของ 5 – 2 ** 2 % 2 / 4 * 3 // 6

6. จงหาคำตอบของ not 36 % 9 > 2 and 4 != 1

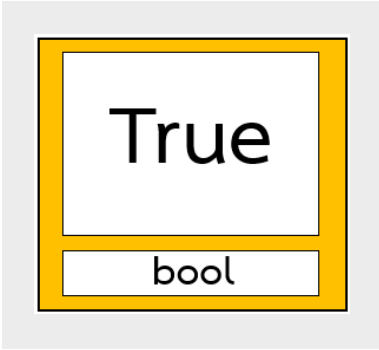
Hint: You may use the commands print() to show a value.

04_DATA TYPES

Data Types

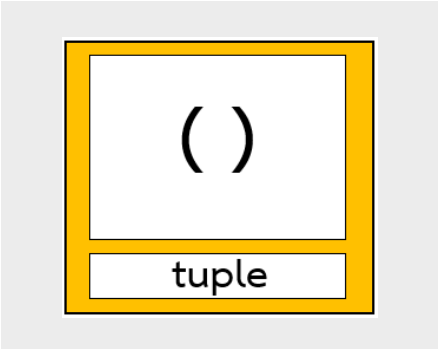


Numbers



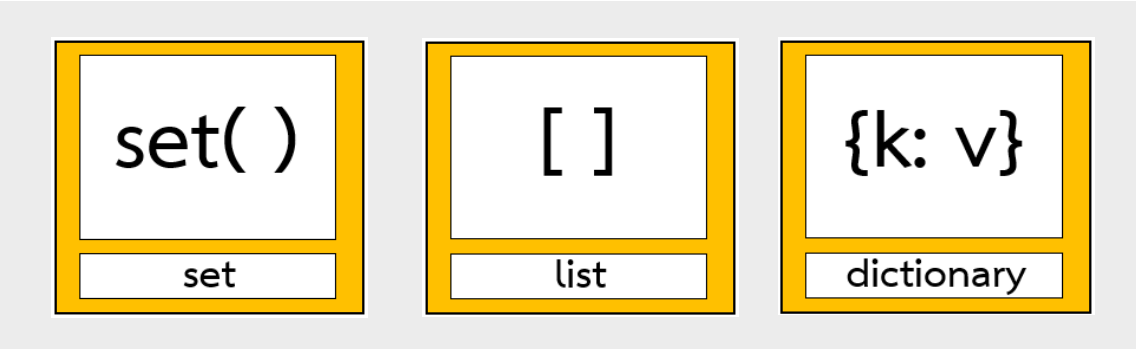
Logical values

Texts



Immutable Containers

(เปลี่ยนค่าข้างในไม่ได้)



Mutable Containers

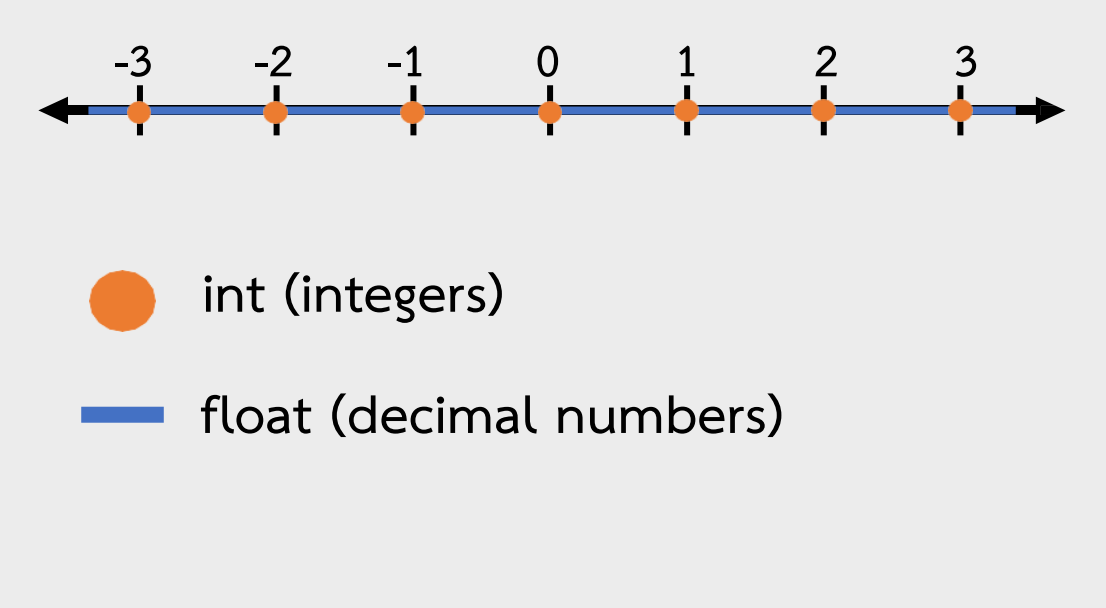
(เปลี่ยนค่าข้างในได้)

Texts may be treated as a container as well.

Example 8: Data Types

```
a='Hello Carbu'
b=45.5
c=90
d=['Python', 'SQL', 'R']
e=('Python', 'SQL', 'R')
f={'Python', 'SQL', 'R'}
g={'Name': 'Carbu', 'Age':  '20'}
h=False
```

Number: int, float, complex



Example 9: Numbers

Python	
Input	my_interger = int(3.414)
Output	

Python	
Input	my_float = float(4)
Output	

Python	
Input	my_round = round(2.453345)
Output	

Python	
Input	my_complex= complex(2,3)
Output	

Text: String

String Methods:

For more methods: [Python String Methods (w3schools.com)]

Method	Description
capitalize()	Converts the first character to upper case
upper()	Converts a string into upper case
lower()	Converts a string into lower case
split()	Splits the string at the specified separator, and returns a list
title()	Converts the first character of each word to upper case
replace()	Returns a string where a specified value is replaced with a specified value
count()	Returns the number of times a specified value occurs in a string
format()	Formats specified values in a string
find()	Searches the string for a specified value and returns the position of where it was found
index()	Searches the string for a specified value and returns the position of where it was found
isnumeric()	Returns True if all characters in the string are numeric
strip()	Returns a trimmed version of the string

Text: String

Example 10: String Methods

กำหนดให้ organize_name = 'bDi'

Python	
Input	organize_name.capitalize()

Python	
Input	organize_name.upper()

Python	
Input	organize_name.lower()

กำหนดให้ my_str = '13-July-2023'

Python #split with dash	
Input	my_str.split(?)

กำหนดให้ my_str1 = 'This is Python class.'

Python #find P	
Input	my_str.find(?)

กำหนดให้ my_str2 = 'I am studying SQL.'

Python #replace SQL with Python	
Input	my_str2.replace(?)

กำหนดให้ my_str3 = 'Data Scientist'

Python #count a	
Input	my_str3.count(?)

Text: String

Example 10: String Methods (cont.)

กำหนดให้ my_str4 = 'Data Scientist'

Python #index i	
Input	my_str4.index(?)

กำหนดให้ my_str5 = "0568"
my_str6 = "2B34"
my_str7 = " "

Python	
Input	x = my_str5.isnumeric() y = my_str6.isnumeric() z = my_str7.isnumeric() print(x, y, z)

กำหนดให้ my_str8 = " Mr. John "

Python	
Input	my_str8.strip()

Text: String

Example 10: String Methods(cont.)

Python	
Input	txt = "My height is {Height:.2f} cm." print(txt.format(Height = 180))
Output	

Python	
Input	txt1 = "There are {num_member} people in {fname} organization".format(num_member=70, fname = "BDI") txt2 = "There are {0} people in {1} organization".format(70, "BDI") txt3 = "There are {} people in {} organization".format(70, "BDI") print(txt1) print(txt2) print(txt3)
Output	

Text: String

Example 11: Basic String Operations:

Concatenation

'สวัสดี' + 'ค่ะ' → 'สวัสดีค่ะ'

'Thank you' * 3 → 'Thank youThank youThank you'

Length

len('Hello World') → 11

Membership

'H' in 'Hello World' → True

'H' not in 'Hello World' → False

These are basic operations for a container.

Text: String

Example 12: String Slicing

BIG DATA INSTITUTE

x = 'Hello World!' or x = "Hello World!"

value	H	e	l	l	o		W	o	r	l	d	!
index	0	1	2	3	4	5	6	7	8	9	10	11
	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

1. x[0]
2. x[5]
3. x[9]
4. x[-12]
5. x[-7]
6. x[-3]

Text: String

Example 13: String Slicing2

x = 'Hello World!'

H	e	l	l	o		W	o	r	l	d	!
0	1	2	3	4	5	6	7	8	9	10	11

x [start : stop : step]

1. x[1: 9]
2. x[0: 11: 2]
3. x[2: 12: 1]
4. x[3: -2: 2]

Immutable Containers: Tuple

➢ *Tuple*

A tuple is a collection that is **ordered** and **unchangeable**. It is written with round brackets and consists of elements that are separated by commas.

Example 14: Tuples

Input	my_tuple1= (1, 2, 3, 4, 4, 5, 5, 5) my_tuple1.count(4)
Output	

Input	my_tuple2= tuple([1, 2, 3, 4, 4, 5, 5, 5]) my_tuple2.count(4)
Output	

Accessing tuple element

Input	#access 3rd index
	my_tuple3= (1, 2, 3, 4, 4, 5, 5, 5)
	my_tuple3[?]

Output	

Input	#remove 2 from the tuple
	my_tuple5 = (1, 2, 3)
	my_tuple5.remove(?)
	print(my_tuple5)

Output	

Input	my_tuple4= ("C", "Python", "HTML") print(my_tuple4[0:2]) print(my_tuple4[1:]) print(my_tuple4[:-1])
Output	

Mutable Containers: List, Set, Dictionary

BIG DATA INSTITUTE

- ***List***

  A List is a mutable sequence that is **ordered** and **changeable**. It is written with square brackets and consists of elements that are separated by commas.

List Methods:

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Adds the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

Python - List Methods (w3schools.com)

Example 15: Lists

Input	my_list1 = [1, 2, 3] print(my_list1)
Output	

Input	my_list2 = list([1, 2, 3]) print(my_list2)
Output	

Input	my_list3 = list('123') print(my_list3)
Output	

Input	#count p
	my_list4 = list('python programming')
	my_list4.count()

Output	

Input	#show reverse
	my_list5 = list('python')
	my_list5.reverse()

Output	

Adding to a list: append(), insert(), extend()

Input	my_list6 = list('python') my_list6.append('!!!!!')
Output	

Input	my_list6 = list('python') my_list6.extend('!!!!!')
Output	

Input	my_list7 = ['python'] my_list7.insert(0, 'I love')
Output	

Input	my_list_1 = ['I', 'love'] my_list_2 = ['python'] my_list_1.extend(my_list_2)
Output	

Accessing list elements

Input	#Access 3rd index
	my_list8 = ['rain','cloud','sun','moon']
	my_list8[3]

Output	

Input	#Access -2nd index my_list8[-2]
Output	

Input	#Access n in the word moon my_list8[3][3]
Output	

Delete from a list: clear(), pop(), remove()

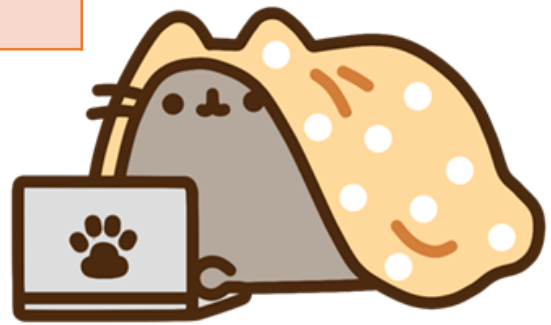
Input	my_list9 = ['apple','orange','banana'] my_list9.clear()
Output	

Input	#Remove apple from the list
	my_list10 = ['apple','orange','banana']
	my_list10.remove(?)

Output	

Input	#Remove banana from the list
	my_list11 = ['apple','orange','banana']
	my_list11.pop(?)

Output	



Sorting a list

Input	my_list12 = [13, 2, 7, 5, 11, 3] my_list12.sort() print(my_list12)
Output	

Input	my_list13 = [13, 2, 7, 5, 11, 3] sorted(my_list13)
Output	

Let's Try:

BIG DATA INSTITUTE

Let x = [ ]

Operation	Syntax	Output	Description
append	x.append(9)	[9]	Add an element at the end of the list.
insert	x.insert(0, False)	[False, 9]	Add an element at a specified index.
remove	x.remove(9)	[False]	Remove an element.
copy	x.copy()	[False]	Copy a list.

Please visit this link for more on list methods.

➢ *Set*

A list is a data type that is **unordered**, **unchangeable*** and **does not allow duplicate values**. It is written with curly brackets and consists of elements that are separated by commas.

(* Set *items* are unchangeable, but you can remove items and add new items.)

Example 16: Sets

Creating a set

Input	my_set = {"python", "C#", "MATLAB", "SQL", "SQL"} my_set
Output	

Input	mylist = ["python", "C#", "MATLAB", "SQL", "SQL"] my_set2 = set(mylist) my_set2
Output	

Accessing set members

Input	my_set3 = {"python", "C#", "MATLAB", "SQL", "SQL"} 'python' in my_set3
Output	

Adding set members : add(), update()

Input	#add HTML in this set
	my_set3 = {"python", "C#", "MATLAB", "SQL", "SQL"}
	my_set3.add(?)
	print(my_set3)

Output	

Input	my_set4_1 = my_set3.copy() my_set4_1.update('C++')
Output	

Input	my_set4_2 = my_set3.copy() my_set4_2.update(['C++'])
Output	

Removing set members : remove(), discard()

Input	#Remove 9 from this set
	my_set5 = {2, 3, 5, 7, 9, 11, 13}
	my_set5.remove(?)

Output	

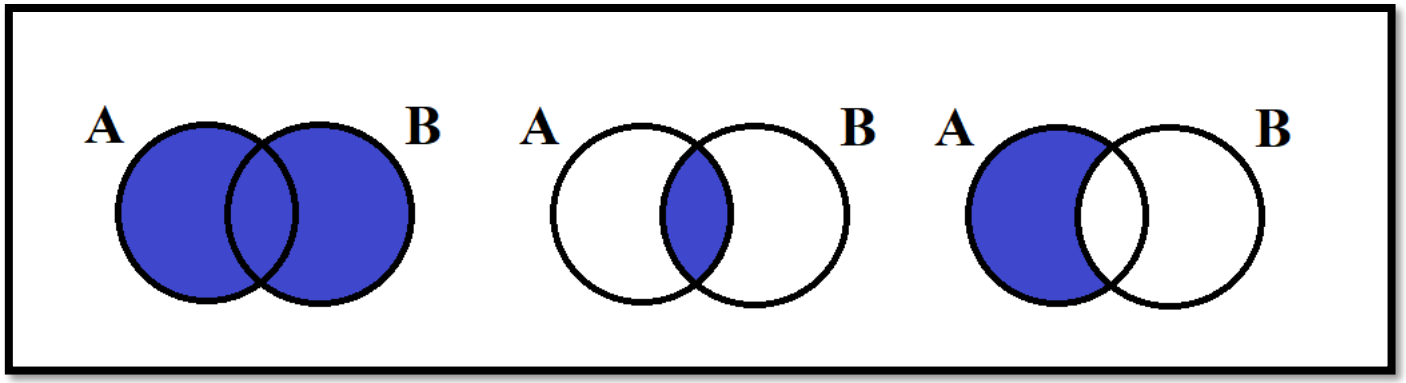
Input	#Remove 9 from this set
	my_set6 = {2, 3, 5, 7, 9, 11, 13}
	my_set6.discard(?)

Output	

Clearing or deleting a set

Input	my_set7 = {"sea", "beach", "sand", "sun"} my_set7.clear() my_set7
Output	

Set operation



Input	my_set8 = {1,2,3,4,6,8,10} my_set9 = {3,4,5,7,9} union_set = my_set8.union(my_set9) intersection_set = my_set8.intersection(my_set9) diff_set = my_set8.difference(my_set9)
Output	

➢ *Dictionary*

A dictionary is a collection that is **unordered**, **changeable** and does not allow duplicates.

Example 17: Dictionary

Creating dictionaries

Input	my_dict1 = {'one':1,'two':2,'three':3} my_dict1
Output	

Input	my_dict2 = dict({'one':1,'two':2,'three':3}) my_dict2
Output	

Accessing dictionaries

Input	my_dict3= {'first_name': 'Angkhana', 'last_name': 'Prommarat', 'organization': 'BDI'} my_dict3['organization']
Output	

Input	'last_name' in my_dict3
Output	

Input	'nick_name' in my_dict3
Output	

Key methods

Input	my_dict4 = {'Cat':'แมว','Pig':'หมู','Rabbit':'กระต่าย','Elephant':'ช้าง','Bird':'นก'} x=my_dict4.keys() y=my_dict4.values() z=my_dict4.items() print(x) print(y) print(z)
Output	

Adding Item to Dictionary

Input	personal_info = {'first_name':'AngAng','last_name':'Pro','organization':'BDI'} x1 = personal_info.keys() x2 = personal_info.values() print(x1,x2)
Output	

Input	personal_info['email']='angkhana.pr@depa.or.th' print(x1,x2)
Output	

Input	personal_info.update({'tel.':'020262333'}) personal_info
Output	

Deleting Item from Dictionary

Input	personal_info2 = {'first_name': 'AngAng', 'last_name': 'Pro', 'organization': 'BDI'} del personal_info2['last_name'] personal_info2
Output	

Input	personal_info3 = {'first_name': 'AngAng', 'last_name': 'Pro', 'organization': 'BDI'} personal_info3.pop('last_name') personal_info3
Output	

Input	personal_info4 = {'first_name': 'AngAng', 'last_name': 'Pro', 'organization': 'BDI'} personal_info4.clear() personal_info4
Output	

Modifying Item in Dictionary

Input	personal_info5 = {'first_name': 'AngAng', 'last_name': 'Pro', 'organization': 'BDI'} personal_info5['first_name']='Adisak' personal_info5
Output	

Exercise 2: Data Types

1. กำหนดให้ A='Big Data Institute'
จงหาคำตอบของ A[5:-5:3]

2. กำหนดให้ B=['Yellow', 'Mango', 'อีโต้']
จงลบคำว่า 'อีโต้' ออกจาก list

3. กำหนดให้ C=[3, 5, 2, 4, 8, 100, 1, 6, 5, 7]
จงลบเลข 100 ออกจาก list แล้วจัดเรียง list ใหม่โดยให้ผลลัพธ์เรียงลำดับจากมากไปหาน้อย

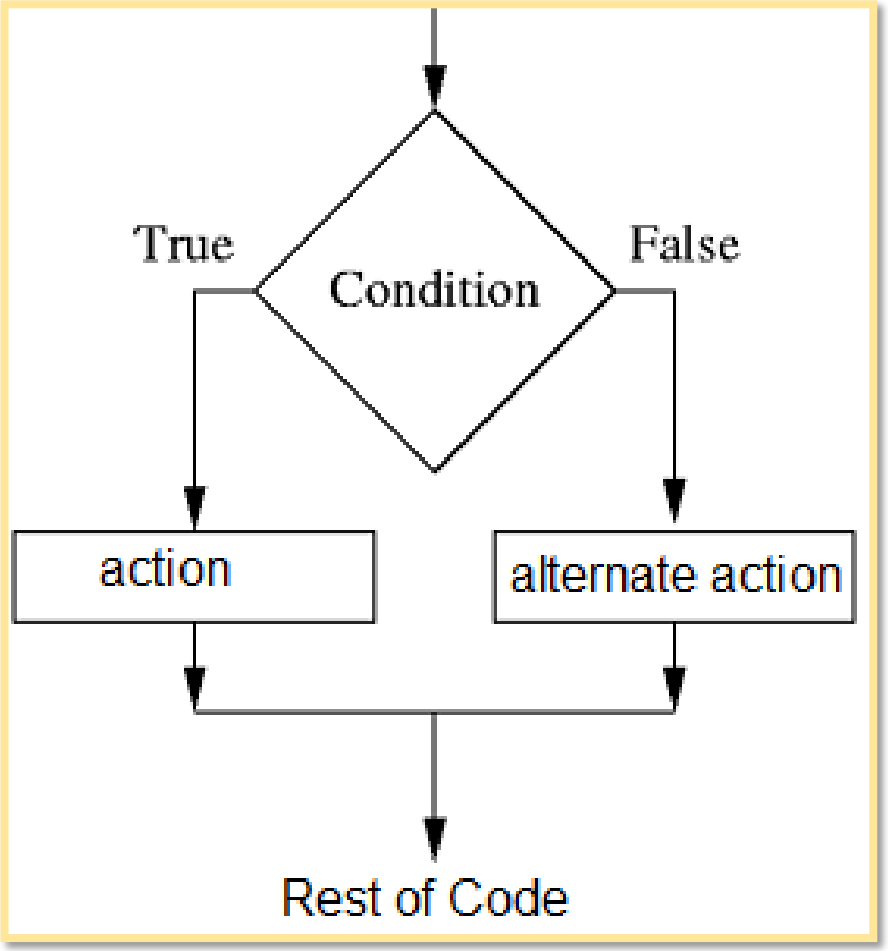
4. กำหนดให้ D={'Banana' : 'กล้วย', 'Mango' : 'เงาะ' , 'Durian' : 'ทุเรียน'}
จงจัดการความหมายของผลไม้ ใน dictionary ให้ถูกต้อง

5. กำหนดให้ E={9, 2, 3, 4, 5, 6, 8}, F={2, 3}, G={3, 8}
จงหาผลลัพธ์ (E ∩ F)-G

BIG DATA INSTITUTE

05_CONDITIONS

Conditions



[If Statements Explained - Python Tutorial (pythonbasics.org)]

If statement

Logical expression

```
if condition:

    statement1
    statement2
    ...
```

If the condition holds, the following codes will be executed.

Conditions

If-else statements

```
if condition:

    statement11
    statement12
    ...

else:

    statement21
    statement22
    ...
```

If the condition holds, the following codes will be executed.

Otherwise, this group of codes will be executed.

If-elif-else statements

```
if condition1:

    statement11
    statement12
    ...

elif condition2:

    statement21
    statement22
    ...

else:

    statementN1
    statementN2
    ...
```

If the condition1 does not hold but the condition2 is satisfied, execute these codes.

Example 18: Condition

Input	p1=5
	
	if p1<10:
	  print('p1 below ten')
	if p1>3:
	  print('p1 is greater than three')
	if p1>7:
	  print('p1 is greater than seven')

Output	

Input	weight = int(input('Fill your weight:'))
	if weight>80:
	  print('You have belly fat.')
	else:
	  print('You don\'t have belly fat.')

Output	

Example 18: Condition

จงสร้างโปรแกรมเช็คค่าฝุ่น โดยสามารถ input ค่าความเข้มข้นของ PM2.5 µg/m^3 เข้าไปได้ จากนั้นให้แสดงผลลัพธ์เป็นค่าคุณภาพอากาศตามช่วง (เช่น คุณภาพอากาศปานกลาง) ดังตาราง

ตารางที่ 2 ค่าความเข้มข้นของสารมลพิษทางอากาศที่เทียบเท่ากับค่าดัชนีคุณภาพอากาศ

AQI	$PM_{2.5}$ (มคก./ลบ.ม.) เฉลี่ย 24 ชั่วโมงต่อเนื่อง	PM_{10} (มคก./ลบ.ม.) เฉลี่ย 24 ชั่วโมงต่อเนื่อง	
0 - 25	0 - 25	0 - 50	<-- คุณภาพอากาศดีมาก
26 - 50	26 - 37	51 - 80	<-- คุณภาพอากาศดี
51 - 100	38 - 50	81 - 120	<-- คุณภาพอากาศปานกลาง
101 - 200	51 - 90	121 - 180	<-- คุณภาพอากาศเริ่มมีผลกระทบต่อสุขภาพ
201 ขึ้นไป	91 ขึ้นไป	181 ขึ้นไป	<-- คุณภาพอากาศมีผลกระทบต่อสุขภาพ

```
Input concentration of PM 2.5:
```

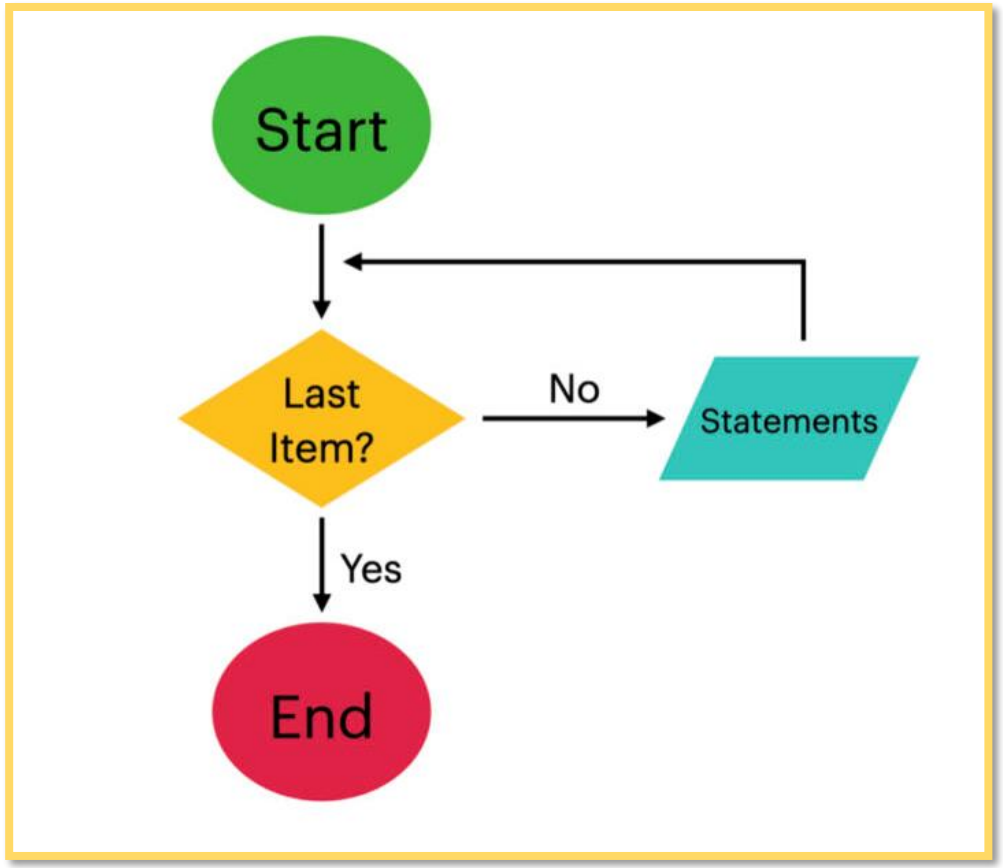
```
Input concentration of PM 2.5:56
คุณภาพอากาศเริ่มมีผลกระทบต่อสุขภาพ
```

Air4Thai (pcd.go.th)

06_LOOPS

For loop

A for loop is used for repetitive actions. The iteration is continued until the stop condition is met. If the stop condition is not met it will loop infinitely.



Flowchart of a For Loop - codingem.com

Count-controlled loop

```
for var in container:

    statement1
    statement2
    ...
```

Example 19: For Loops

BIG DATA INSTITUTE

Input	for char in 'Python!': print(char)
Output	

Input	for elm in {'A','B'}: print(elm)
Output	

Input	for item in ('This',True,3.14,-2): print(item)
Output	

Input	for num in [1,2,3,4,5]: print(num)
Output	

Input	for num in range(1,6): print(num)
Output	

Input	x = {'1':-25, 11:'What', 'How':1.1, -3.14:'!', 4:True} for key in x.keys(): print(key)
Output	

A sum of numbers in list 1-5

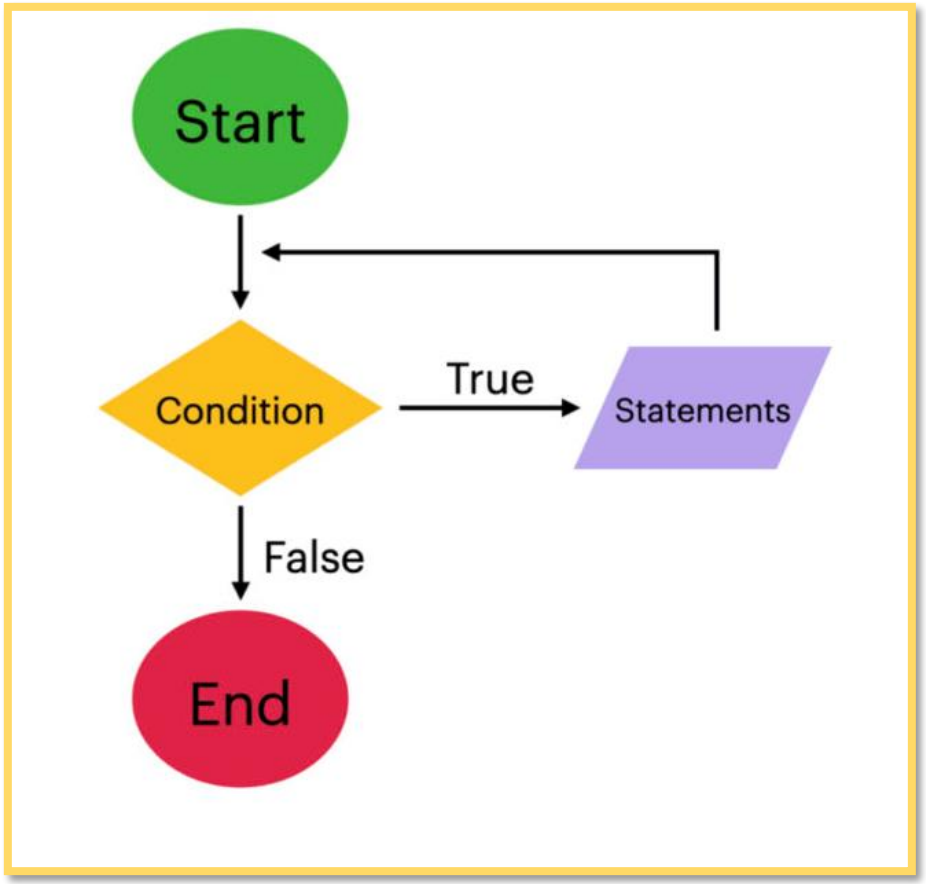
Input	sum=0 for num in [1,2,3,4,5]: sum=sum+num print(sum)
Output	

A sum of even numbers in range 1-100

Input	
Output	

While loop

A while loop is used for repetitive actions. Unlike for loops, the number of iterations is unknown. While loop executes a set of statements as long as a condition is true. When the condition becomes false, the statements are no longer executed.




```
while condition:

    statement1
    statement2
    ...
```

Flowchart of a For Loop - codingem.com

Example 20: While Loops

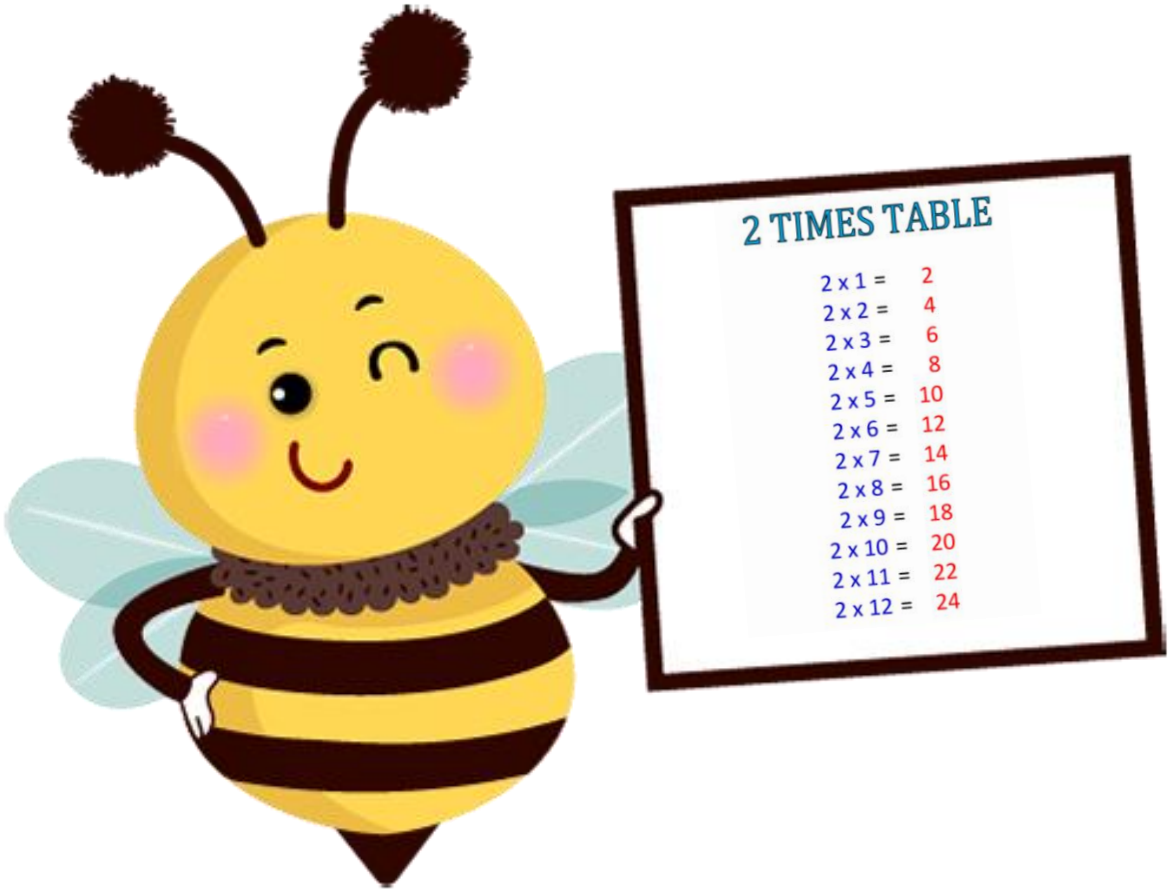
Input	i = 1
	while(i <= 5):
	  print("Good Morning")
	  i = i + 1

Output	

Two times table

Input	i = 1
	while i <= 12:
	  print('2 x', i,'=',2 * i)
	  i += 1

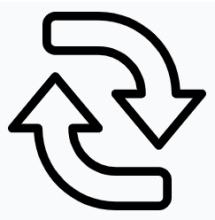
Output	



Extra!!!: จากบทเรียนเรื่อง Loops จงสร้างสูตรคูณทั้ง 12 แม่

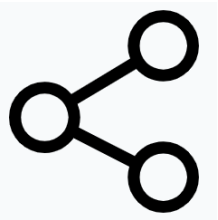
07_FUNCTION

Functions



Reuse

Shorten



Share

Function declaration

Parameters

```
def function_name(input1, input2, ...):
    '''
    docstring: description of the function
    '''

    statement1
    statement2
    ...

    return output1, output2, ...
```

Function called

Arguments

```
x = function_name(input1, input2, ...)
```

Please visit this link for more information.

Example 21: Function

Input	`def my_function():`
	`  print("Hello from a function")`
	
	`my_function()`

Output	

Input	`def show_student(name, organization='No organization'):`
	`  print('Name:', name)`
	`  print('Organization:', organization)`
	
	`show_student('Mr.A')`
	`show_student('Mr.A', 'BDI')`
	`show_student(name= 'Mr.A')`
	`show_student(organization= 'BDI', name='Mr.A')`

Output	

Input	`def triangle_area(base,height):`
	`  area = 1/2 * base*height`
	`  return area`
	
	`print('Area of triangle is: ', triangle_area(10,4), 'unit square')`

Output	

Exercise 3:

จงสร้างฟังก์ชัน convert_month ในการแปลงเดือนจากตัวเลขเป็นชื่อเดือน

โดยกำหนดให้ input จะเป็นรูปแบบของ dd-mm-yyyy

(Hint: สร้าง dict , ใช้ Accessing the string slicing)

```
convert_month('19-05-2564')
```

```
'19 พฤษภาคม 2564'
```

```
convert_month('09-09-2549')
```

```
'9 กันยายน 2549'
```

```
convert_month('31-12-2500')
```

```
'31 ธันวาคม 2500'
```

08_Numpy

Numpy

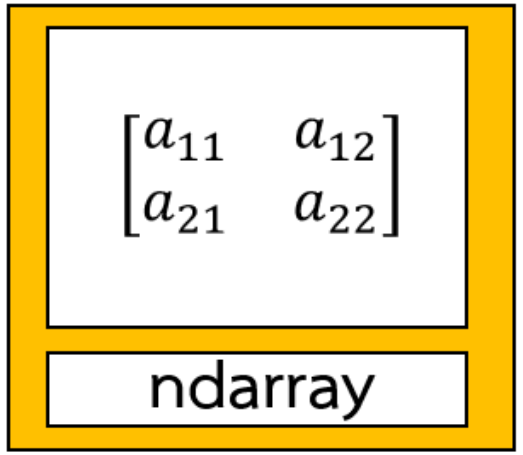
- NumPy is a python package / library that stands for 'Numerical Python'
- It is used for scientific computing by using array objects
- It uses array/vector and matrix operations which save coding time and execution time; no need for while loops in lists

Why Numpy?

- ✓ NumPy arrays are faster and more compact than Python lists.
- ✓ An array consumes less memory and is convenient to use.
- ✓ NumPy uses much less memory to store data and it provides a mechanism of specifying the data types.

Importing library

Python	
Input	**import numpy as np**



NumPy

NumPy: the absolute basics for beginners — NumPy v1.24 Manual

Array

An array is a type of data that can store the same type of data sequentially. The data is contained in the same variable called an array variable. It uses index to access the data.

Example 22: Array

Creating Array

Input	a = np.array([1,2])
Output	

Input	b = np.array([1,2], [3,4])
Output	

Input	c = np.array([[1,2],[3,4],[5,6]])
Output	

Input	d = np.zeros((5,2))
Output	

Input	e = np.ones((5,5))
Output	

Input	f = np.eye(4)
Output	

Input	g = np.random.random((3,3))
Output	

Accessing Array

Input	arr1= np.array([1,2,3,4]) arr1[2]
Output	

Input	arr1[:3]
Output	

Input	arr2= np.array([[1,2,3,4],[2,3,2,1],[4,4,2,1]]) arr2[2][2]
Output	

Basic Math

Input	ar1 = np.array([1,2,3,4,5,6]) ar2 = np.array([7,8,9,10,11,12])
Output	

Input	r1=np.add(ar1,ar2) #ar1+ar2
	r2=np.subtract(ar1,ar2) #ar1-ar2
	r3=np.multiply(ar1,ar2) #ar1*ar2
	r4=np.divide(ar1,ar2) #ar1/ar2
	r5=np.sum(ar1)
	r6=np.average(ar1)
	r7=np.sqrt(ar1)
	print([r1])
	print([r2])
	print([r3])
	print([r4.round(3])
	print([r5])
	print([r6])
	print([r7.round(2)])

Output	

Reshaping

Input	ar3 = np.array([[1,2], [3,4], [5,6]]) ar3.shape
Output	

Input	ar3.reshape(2,3)
Output	

Transpose

Input	#generate new array and reshape
	ar4 = np.arange(9).reshape((3,3))
	ar4

Output	

Input	ar4.T
Output	

09_Pandas

Pandas

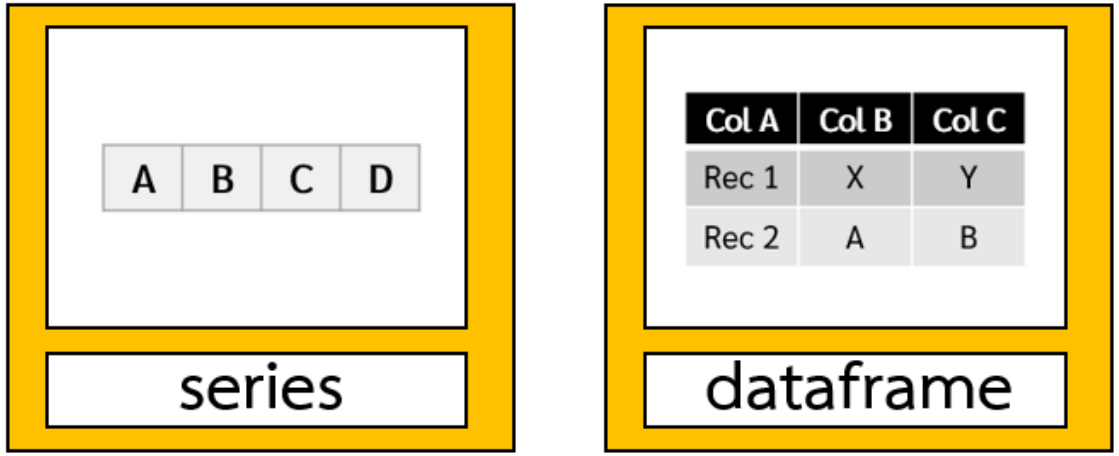
- ❑ Pandas is a Python library used for working with data sets.
- ❑ It has functions for analyzing, cleaning, exploring, and manipulating data.
- ❑ The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis"

Why Pandas?

- ✓ Pandas allows us to analyze big data and make conclusions based on statistical theories.
- ✓ Pandas can clean messy data sets and make them readable and relevant.
- ✓ Relevant data is very important in data science.

Importing library

Python	
Input	**import pandas as pd**



Pandas

Pandas Introduction (w3schools.com)

Example 23: Creating the DataFrame

BIG DATA INSTITUTE

Basic Creating DataFrame

Input	df=pd.DataFrame(columns=['Name','Gender','Age']) df.loc[0]=['Annie','Female',23] df.loc[1]=['Tom', 'Male', 57] df.loc[2]=['Eddy', 'Male', 35]

Creating DataFrame from a dictionary

Input	d1 = {'Name': ['Annie', 'Tom','Eddy'], 'Gender': ['Female', 'Male', 'Male'], 'Age':[23, 57, 35]} df = pd.DataFrame(d1)

Creating DataFrame from a list

Input	a_list = [['Annie', 'Female', 23], ['Tom', 'Male', 57], ['Eddy', 'Male', 35]] df = pd.DataFrame(a_list,columns=['Name','Gender','Age'])

Creating DataFrame from an array

Input	arr3=np.arange(9).reshape(3,3) arr3 df=pd.DataFrame(arr3,columns=['A','B','C']) df

Example 24: Dealing with an Excel file into a DataFrame

BIG DATA INSTITUTE

Importing an excel file

Input	df=pd.read_excel('Superstore.xlsx')

Input	df2=pd.read_excel('Superstore.xlsx', sheet_name='returns')

Reading an excel file from google drive

Input	from google.colab import drive drive.mount('/content/drive')

Input	root_path = '/content/drive/MyDrive/Accident62.xlsx'

Input	df3 = pd.read_excel(root_path)

Exporting an excel file

Input	writer_data = pd.ExcelWriter('Export_Data_Test.xlsx') df.to_excel(writer_data, sheet_name = 'Export1', index = False) writer_data.save()

DataFrame Methods

Method	Description
.shape	Show an ordered pair that is the number of rows and columns
.columns	Show all column names
.index	Show index information of the DataFrame
.head()	Show the first few rows
.tail()	Show the last few rows
.dtypes()	Show data type of all columns
.info()	Show information about the DataFrame
.loc[]	Show detail in a row
.iloc[]	Show detail in a row-based index
.describe()	Description of the data in the DataFrame that contains numerical data

Post-test and send colab link

BIG DATA INSTITUTE

Post-Test: Basic Python Programming_BDI_G2

https://rb.gy/w2mke

BIG DATA INSTITUTE

Facebook

Twitter

Blockdit

YouTube

Line
Official