PYTHON PROGRAMMING

Mr. Tissana Tanaklang

1

PRESENTATION TITLE

AGENDA

- Introduction to Python
- Basic Python Programming
- Functions
- Modules and Packages
- Exceptions
- File Management
- Object Oriented Programming
- Database
- GUI
- Git Version control

2

INTRODUCTION TO PYTHON

3

Python

- Python is a high-level, interpreted, interactive and object-oriented scripting language.
- Python is designed to be highly readable.
- It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

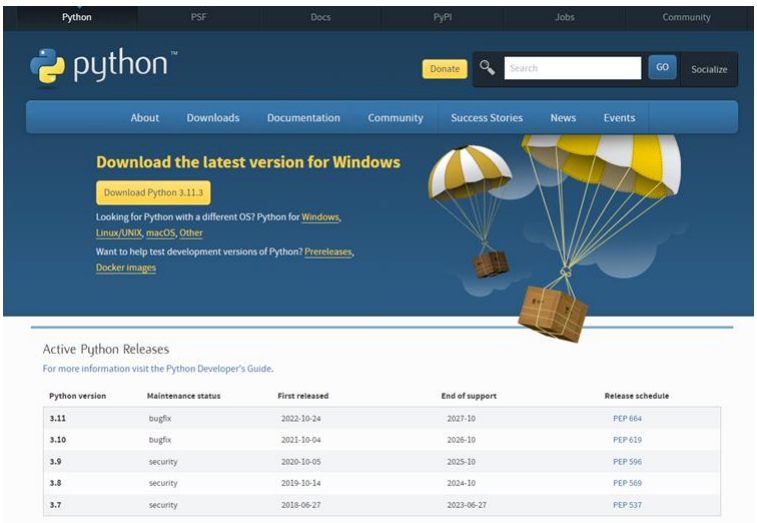
4

Python

- Python is Interpreted
- Python is Interactive
- Python is Object-Oriented
- Python is a Beginner's Language

5

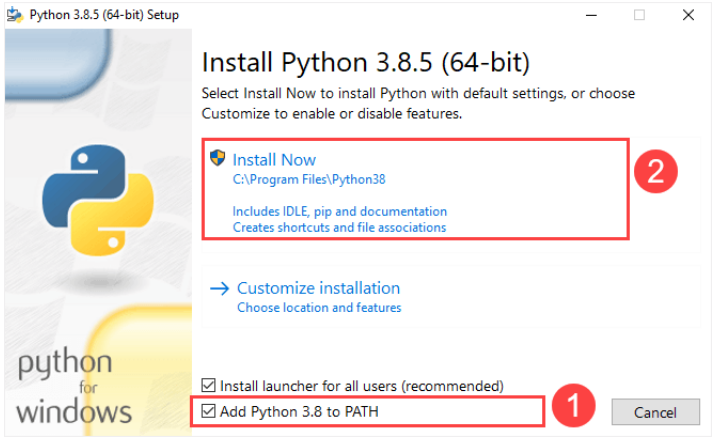
Python Download



https://www.python.org/downloads/

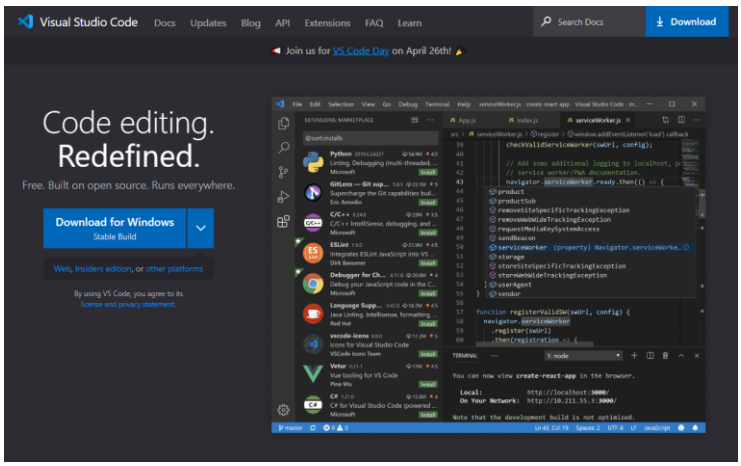
6

Python Installation



7

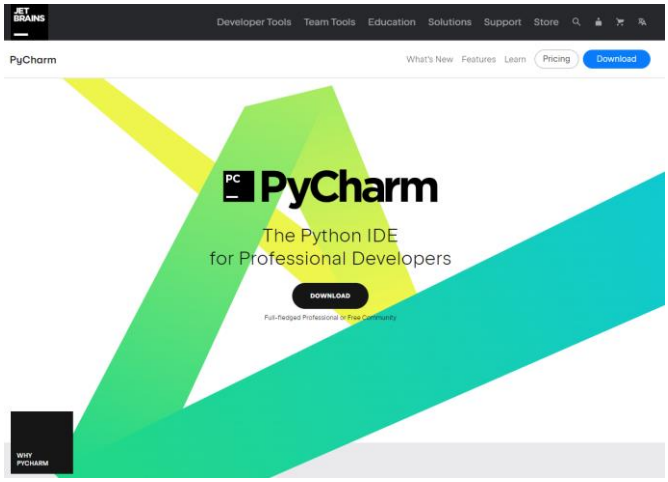
Python Tools



https://code.visualstudio.com/

8

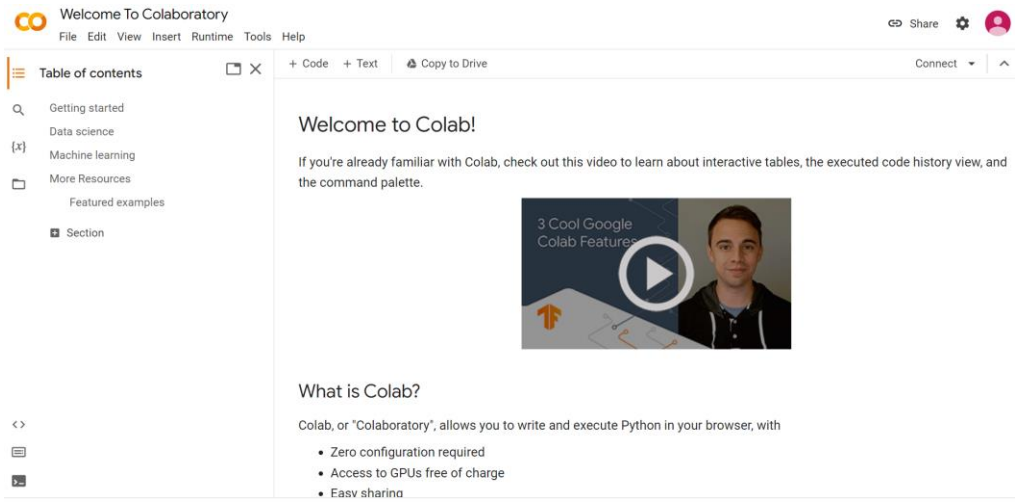
Python Tools



https://www.jetbrains.com/pycharm/

9

Python Tools



10

BASIC PYTHON PROGRAMMING

11

Identifiers

- A Python identifier is a name used to identify a variable, function, class, module or other object.
- An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).
- Python does not allow punctuation characters such as @, $, and % within identifiers.
- Python is a case sensitive programming language.

12

Reserved Words

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

13

Reserved Words

```
# Demo 02-01
import keyword

print(keyword.kwlist)
```

```
"C:\Python Programming\venv\Scripts\python.exe" "C:\Python Programming\02_Basic Python Programming.py"
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue', 'def',
```

14

Reserved Words

```
# Demo 02-02
import keyword

mykey1 = keyword.iskeyword("python")
print(mykey1)

mykey2 = keyword.iskeyword("and")
print(mykey2)
```

```
False
True
```

15

Lines and Indentation

- Python provides no braces to indicate blocks of code for class and function definitions or flow control.
- Blocks of code are denoted by line indentation.

Correct	Incorrect

if True:
  print "True"
else:
  print "False"
	if True:
	  print "Answer"
	  print "True"
	else:
	  print "Answer"
	print "False"

16

Data Type

Data Type	Example
Text	str
Numeric	int, float, complex
Sequence	list, tuple, range
Mapping	dict
Set	set, frozenset
Boolean	bool
Binary	bytes, bytearray, memoryview
None	NoneType

17

Print Function

```
# Demo 02-03

print('Welcome to Python Programming')
print("Basic Python Programming")
print(100)
print(500.75)
print()

print('-----------------------------------------------')

language = 'Python'
version = 3.11
print(language)
print(version)
```

```
Welcome to Python Programming
Basic Python Programming
100
500.75

-----------------------------------------------
Python
3.11
```

18

Print Function

```python
# Demo 02-04

print('Python', 'Programming', 'for', 'Beginner')

mystring1 = 'หนึ่ง,'
mystring2 = 'สอง,'
mystring3 = 'สาม,'
print(mystring1, mystring2, mystring3, 'สี่')

name = 'Guido van Rossum'
year = 1994
print(name, 'พัฒนาภาษาไพธอน ในปี ค.ศ.', year)
```

```
Python Programming for Beginner
หนึ่ง, สอง, สาม, สี่
Guido van Rossum พัฒนาภาษาไพธอน ในปี ค.ศ. 1994
```

19

Print Function

```python
# Demo 02-05

mystrnum1 = 'One'
mystrnum2 = 'Two'
mystrnum3 = 'Three'
mystrnum4 = 'Four'
print(mystrnum1, mystrnum2, mystrnum3, mystrnum4, sep=', ')

mystr1 = 'www'
mystr2 = 'tizcloud'
mystr3 = 'com'
print(mystr1, mystr2, mystr3, sep='.')

myd = 30
mym = 'April'
myy = 2023
print('Start Date: ', end='')
print(myd, mym, myy, sep=' ')

myh = 15
mym = 30
print('Start Time: ', end='')
print(myh, mym, sep=':')
```

```
One, Two, Three, Four
www.tizcloud.com
Start Date: 30 April 2023
Start Time: 15:30
```

20

String

- Strings in python are surrounded by either single quotation marks, or double quotation marks.

'Python' is the same as "Python".

21

String Concatenation

```
# Demo 02-06

mystr1 = 'Welcome' + ' to ' + 'Python ' + 'Programming'
print(mystr1)

myname = 'Guido van Rossum'
mystr2 = 'Python Language create by ' + myname + ' at 1994'
print(mystr2)

fullname = "Sunisa Wattanapaisan"
country = "Thailand"
print("My name is " + fullname + " from " + country)
```

```
Welcome to Python Programming
Python Language create by Guido van Rossum at 1994
My name is Sunisa Wattanapaisan from Thailand
```

22

String Concatenation

```
# Demo 02-07

version = '3.11'
mystring = 'Python' + ' ' + version
print(mystring)

mymsg = "Start Date : " + str(30) + " April " + str(2023)
print(mymsg)

print('Distance : ' + str(1) + ' Mile = '+ str(1.6) + ' km.')
```

```
Python 3.11
Start Date : 30 April 2023
Distance : 1 Mile = 1.6 km.
```

23

Escaping Sequences

Escape Sequence	Meaning
\'	Single quote
\"	Double quote
\\	Backslash
\n	Newline
\r	Carriage Return
\t	Horizontal Tab
\b	Backspace
\f	Formfeed
\v	Vertical Tab
\0	Null Character

24

Escaping Sequences

```
# Demo 02-08

print('Please Select:')
print('1 < Transfer within bank\t\tWithdraw > 3')
print('2 < Transfer without bank\t\tCheck Balance > 4')

print("-------------------------------------")

print('Error Exception:\n' + \
    '- Please Enter Name\n' + \
    '- Please Enter Email\n' + \
    '- Please Enter Correct Password')
```

```
Please Select:
1 < Transfer within bank        Withdraw > 3
2 < Transfer without bank       Check Balance > 4
-------------------------------------
Error Exception:
- Please Enter Name
- Please Enter Email
- Please Enter Correct Password
```

25

Insert information to String

```
# Demo 02-09

mydate = 30
print(f'Date : {mydate}')

language = 'Python'
title = f'This is {language} Programming'
print(title)

print(f"{language} for Data Analytics")

home = 0
away = 6
result = F"Manchester United FC. {home} - {away} Liverpool FC."
print(result)
```

```
Date : 30
This is Python Programming
Python for Data Analytics
Manchester United FC. 0 - 6 Liverpool FC.
```

26

Insert information to String

```
# Demo 02-10

print(f'50 + 70 = {50 + 70}')

mynum1 = 150
mynum2 = 50
print(f"{mynum1} - {mynum2} = {mynum1 - mynum2}")
```

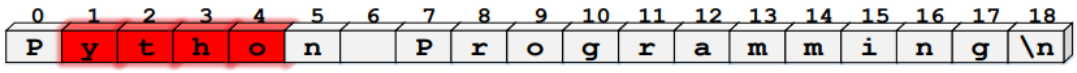
```
50 + 70 = 120
150 - 50 = 100
```

27

String

```
# Demo 02-11

mystring = "Python Programming"

print ("mystring[7]: ", mystring[7])
print ("mystring[1:5]: ", mystring[1:5])
```

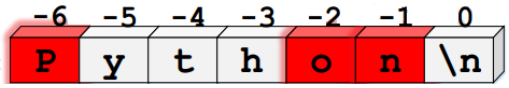
```
mystring[7]:  P
mystring[1:5]:  ytho
```



28

String

```
# Demo 02-12

mystring = 'Python'

print ("mystring[-1]: ", mystring[-1])
print ("mystring[-2]: ", mystring[-2])
print ("mystring[-6]: ", mystring[-6])
```




```
mystring[-1]:  n
mystring[-2]:  o
mystring[-6]:  P
```

29

String

```
# Demo 02-13

mystring = "Welcome to Python Programming"

print('mystring[5]: ', mystring[5])
print('mystring[2:10]: ', mystring[2:10])
print('mystring[2:18:2]: ', mystring[2:18:2])
print('mystring[:8]: ', mystring[:8])
print('mystring[8:]: ', mystring[8:])
print('mystring[::3]: ', mystring[::3])
print('mystring[::-2]: ', mystring[::-2])
```

```
mystring[5]:  m
mystring[2:10]:  lcome to
mystring[2:18:2]:  loet yhn
mystring[:8]:  Welcome
mystring[8:]:  to Python Programming
mystring[::3]:  WceoyoPgmn
mystring[::-2]:  gimroPnhy teolW
```

30

List

- Lists are used to store multiple items in a single variable.
- Lists are one of 4 built-in data types in Python used to store collections of data, the other 3 are Tuple, Set, and Dictionary, all with different qualities and usage.

31

List

```
# Demo 02-14

mylst = [10, 20, 30, 40, 'Python', 50, 'Java']
print(mylst)
print(type(mylst))

print('------------------------------------------------')

mylist1 = [] # Empty list
mylist2 = [10, 20, 30, 40, 50 ]
mylist3 = ["Thailand", "Singapore", "Malaysia", "Vietnam"]
mylist4 = ['Python', 'Java', 1994, 1996]
mylist5 = ['Name', "Surname", 300, 0.5,mylist2]

print(mylist1)
print(mylist2)
print(mylist3)
print(mylist4)
print(mylist5)
```

```
[10, 20, 30, 40, 'Python', 50, 'Java']
<class 'list'>
------------------------------------------------
[]
[10, 20, 30, 40, 50]
['Thailand', 'Singapore', 'Malaysia', 'Vietnam']
['Python', 'Java', 1994, 1996]
['Name', 'Surname', 300, 0.5, [10, 20, 30, 40, 50]]
```

32

List

```python
# Demo 02-15

mylist1 = ['Thailand', 'Vietnam', 2017, 2021]
mylist2 = [100, 200, 300, 400, 500, 600, 700, 800]

print ("mylist1[0]:", mylist1[0])
print ("mylist1[-1]:", mylist1[-1])
print ("mylist2[3]:", mylist2[3])
print ("mylist2[-5]:", mylist2[-5])
print ("mylist2[1:5]:", mylist2[1:5])
print ("mylist2[::2]:", mylist2[::2])
print ("mylist2[2::2]:", mylist2[2::2])
print ("mylist2[2:8:2]:", mylist2[2:8:2])
print ("mylist2[:8]:", mylist2[:8])
print ("mylist2[5:]:", mylist2[5:])
```

```
mylist1[0]: Thailand
mylist1[-1]: 2021
mylist2[3]: 400
mylist2[-5]: 400
mylist2[1:5]: [200, 300, 400, 500]
mylist2[::2]: [100, 300, 500, 700]
mylist2[2::2]: [300, 500, 700]
mylist2[2:8:2]: [300, 500, 700]
mylist2[:8]: [100, 200, 300, 400, 500, 600, 700, 800]
mylist2[5:]: [600, 700, 800]
```

33

List

```python
# Demo 02-16

mylist1 = ['Thailand', 'Vietnam', 2017, 2021]
print ("Old Value : ",mylist1[2])

mylist1[2] = 2023
print ("New Value : ",mylist1[2])

print('----------------------------------------------')

mylist2 = ['python', 'c#', 1994, 2000]
print ("Before deleting : ",mylist2)

del mylist2[2]
print ("After deleting : ",mylist2)
```

```
Old Value :  2017
New Value :  2023
----------------------------------------------
Before deleting :  ['python', 'c#', 1994, 2000]
After deleting :  ['python', 'c#', 2000]
```

34

List

```
# Demo 02-17

mylst1 = [10, 20, 30]
mylst2 = [40, 50, 60]
mylst3 = [70, 80, 90, 100]

print("Length of mylst1 :", len(mylst1))
print("Length of mylst2 :", len(mylst2))
print("Length of mylst2 :", len(mylst3))
print('-------------------------------------')
print("mylst1 + mylst2 :",mylst1 + mylst2)
print("mylst1 + mylst2 + mylst3  :",mylst1 + mylst3 + mylst2)
print('-------------------------------------')
print("mylst1 * 3 :",mylst1*3)

print("Elements in mylst1 :")
for value in mylst1:
 print (value)
```

```
Length of mylst1 : 3
Length of mylst2 : 3
Length of mylst2 : 4
-------------------------------------
mylst1 + mylst2 : [10, 20, 30, 40, 50, 60]
mylst1 + mylst2 + mylst3  : [10, 20, 30, 70, 80, 90, 100, 40, 50, 60]
-------------------------------------
mylst1 * 3 : [10, 20, 30, 10, 20, 30, 10, 20, 30]
Elements in mylst1 :
10
20
30
```

35

List

```
# Demo 02-18

mylst1 = [10,20,30,40,50]
mylst2 = [90,80,70,60,50]
mylst3 = [90,80,70,60,50]
mylst4 = [80,50]

print("mylst1 < mylst2 :",mylst1 < mylst2)
print("mylst1 > mylst2 :",mylst1 > mylst2)
print("mylst2 >= mylst1 :",mylst2 >= mylst1)
print("mylst2 == mylst3 :",mylst2 == mylst3)

print("50 in mylst3 :",50 in mylst3)
print("30 not in mylst4 :",30 not in mylst4)
```

```
mylst1 < mylst2 : True
mylst1 > mylst2 : False
mylst2 >= mylst1 : True
mylst2 == mylst3 : True
50 in mylst3 : True
30 not in mylst4 : True
```

36

List

```
# Demo 02-19

mylst = [i*1 for i in range(10)]

print("List Member: ",mylst)
print('---------------------------------------')

del mylst[0], mylst[2], mylst[4]
print("List After Remove Member: ",mylst)
print('---------------------------------------')

del mylst
print("List After Remove Member: ",mylst)
```

```
List Member:  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
---------------------------------------
List After Remove Member:  [1, 2, 4, 5, 7, 8, 9]
---------------------------------------
Traceback (most recent call last):
  File "C:\Python Programming\02-19-Lis.py", line 13, in <module>
    print("List After Remove Member: ",mylst)
                                       ^^^^^
NameError: name 'mylst' is not defined
```

37

Tuples

- Tuples are used to store multiple items in a single variable.
- Tuple is one of 4 built-in data types in Python used to store collections of data, the other 3 are List, Set, and Dictionary, all with different qualities and usage.
- A tuple is a collection which is ordered and unchangeable.
- Tuples are written with round brackets.

38

Tuples

```
# Demo 02-20

mytup1 = (100, 50.75)
mytup2 = ('Python', 'Java')
mytup3 = (30, 15.5, 'Thailand',"Malaysia")
mytup4 = (mytup1, mytup2, mytup3)
mytup5 = mytup1 + mytup2
mytup6 = ()


print('Result of mytup1 : ', mytup1)
print('Result of mytup2 : ', mytup2)
print('Result of mytup3 : ', mytup3)
print('Result of mytup4 : ', mytup4)
print('Result of mytup5 : ', mytup5)
print('Result of mytup6 : ', mytup6)
```

```
Result of mytup1 :  (100, 50.75)
Result of mytup2 :  ('Python', 'Java')
Result of mytup3 :  (30, 15.5, 'Thailand', 'Malaysia')
Result of mytup4 :  ((100, 50.75), ('Python', 'Java'), (30, 15.5, 'Thailand', 'Malaysia'))
Result of mytup5 :  (100, 50.75, 'Python', 'Java')
Result of mytup6 :  ()
```

39

Tuples

```
# Demo 02-21

mytup1 = (("Name", "Sunisa"),("Age","38"),("Gender","Female"))

print(mytup1[0])
print(mytup1[1])
print(mytup1[2])

print('------------------------------')

mytup2 = (100, 35.75)
mytup2[0] = 155
mytup2[1] = 75.55
print(mytup2)
```

```
Traceback (most recent call last):
  File "C:\Python Programming\02-21-Tuples.py", line 12, in <module>
    mytup2[0] = 155
    ~~~~~~^^^
TypeError: 'tuple' object does not support item assignment
('Name', 'Sunisa')
('Age', '38')
('Gender', 'Female')
```

40

Tuples

```
# Demo 02-22

mytup = ('Arsenal', 'Manchester City', 2002, 2022)
print (mytup)
print (mytup[0])
print (mytup[1])
print (mytup[2])
print (mytup[3])

del mytup[2]
print ("After mytup[2] Deleted : ",mytup)
```

```
('Arsenal', 'Manchester City', 2002, 2022)
Arsenal
Manchester City
2002
2022
Traceback (most recent call last):
  File "C:\Python Programming\02-22-Tuples.py", line 10, in <module>
    del mytup[2]
        ~~~~~^^^
TypeError: 'tuple' object doesn't support item deletion
```

41

Tuples

```
# Demo 02-23

mytup = ('Arsenal', 'Manchester City', 2002, 2022)

del mytup
print ("After mytup Deleted : ",mytup)
```

```
Traceback (most recent call last):
  File "C:\Python Programming\02-23-Tuples.py", line 6, in <module>
    print ("After mytup Deleted : ",mytup)
                                    ^^^^^
NameError: name 'mytup' is not defined
```

42

Tuples

```
# Demo 02-24

mytup1 = (10, 20, 30)
mytup2 = (40, 50, 60, 70)

print("Length of mytup1 : ",len(mytup1))
print("Length of mytup2 : ",len(mytup2))
print("mytup1 + mytup2 : ",mytup1 + mytup2)
print("mytup1 * 3 : ",mytup1*3)

print("Elements in mytup1 :")
for x in mytup1:
 print (x)
```

```
Length of mytup1 :  3
Length of mytup2 :  4
mytup1 + mytup2 :  (10, 20, 30, 40, 50, 60, 70)
mytup1 * 3 :  (10, 20, 30, 10, 20, 30, 10, 20, 30)
Elements in mytup1 :
10
20
30
```

43

Tuples

```
# Demo 02-25

mytup1 = ('Napoli', 'Lazio', 750, 600)
mytup2 = (10, 20, 30, 40, 50, 60, 70)

print("mytup[0]:", mytup1[0])
print("mytup1[-1]:", mytup1[-1])
print("mytup2[5]:", mytup2[5])
print("mytup2[-3]:", mytup2[-3])
print("mytup2[1:5]:", mytup2[1:5])
print("mytup2[::2]:", mytup2[::2])
```

```
mytup[0]: Napoli
mytup1[-1]: 600
mytup2[5]: 60
mytup2[-3]: 50
mytup2[1:5]: (20, 30, 40, 50)
mytup2[::2]: (10, 30, 50, 70)
```

44

Tuples

```
# Demo 02-26

mytup1 = (10,20,30,40,50)
mytup2 = (90,80,70,60,50)
mytup3 = (90,80,70,60,50)
mytup4 = (80,70)

print("mytup1 < mytup2 :",mytup1 < mytup2)
print("mytup1 > mytup2 :",mytup1 > mytup2)
print("mytup2 >= mytup1 :",mytup2 >= mytup1)
print("mytup2 == mytup3 :",mytup2 == mytup3)
print("50 in mytup3 :",50 in mytup3)
print("mytup4 not in mytup2 :",mytup4 not in mytup2)
```

```
mytup1 < mytup2 : True
mytup1 > mytup2 : False
mytup2 >= mytup1 : True
mytup2 == mytup3 : True
50 in mytup3 : True
mytup4 not in mytup2 : True
```

45

Dictionary

- Dictionaries are used to store data values in key:value pairs.
- A dictionary is a collection which is ordered*, changeable and do not allow duplicates.
- As of Python version 3.7, dictionaries are ordered. In Python 3.6 and earlier, dictionaries are unordered.

46

Dictionary

```
# Demo 02-27

mydict1 = {}
mydict2 = {'Arsenal': '75', 'Machester City': '70', 'Machester United': '59'}
mydict3 = { 'Newcastle United': 56 }
mydict4 = { 'Liverpool': 47, 2023: 20 }

print(mydict1)
print(mydict2)
print(mydict3)
print(mydict4)
```

```
{}
{'Arsenal': '75', 'Machester City': '70', 'Machester United': '59'}
{'Newcastle United': 56}
{'Liverpool': 47, 2023: 20}
```

47

Dictionary

```
# Demo 02-28

mydict = {'Name': 'Sunisa', 'Age': 30, 'Class': 'First'}

print ("mydict['Name']: ", mydict['Name'])
print ("mydict['Age']: ", mydict['Age'])
```

```
mydict['Name']:  Sunisa
mydict['Age']:  30
```

48

Dictionary

```
# Demo 02-29

mydict1 = {}
print("Original Dictionary : ",mydict1)

mydict1['Key1'] = 150
print("Update data1 to dictionary : ",mydict1)

mydict1['Key2'] = 450
print("Update data2 to dictionary : ",mydict1)
```

```
Original Dictionary :  {}
Update data1 to dictionary :  {'Key1': 150}
Update data2 to dictionary :  {'Key1': 150, 'Key2': 450}
```

49

Dictionary

```
# Demo 02-30

mydict = {'Name': 'James Wilson', 'Age': 35, 'Class': 'First'}
print("Original Dictionary :",mydict)

del mydict['Class'] # delete member dictionary
print("After removed :",mydict)

mydict.clear() # remove all entries in dictionary
print("After cleared all member :",mydict)

del mydict     # delete entire dictionary
print("After deleted form memory :",mydict)
```

```
Traceback (most recent call last):
  File "C:\Python Programming\02-30-Dictionary.py", line 13, in <module>
    print("After deleted form memory :",mydict)
                                        ^^^^^^
NameError: name 'mydict' is not defined. Did you mean: 'dict'?
Original Dictionary : {'Name': 'James Wilson', 'Age': 35, 'Class': 'First'}
After removed : {'Name': 'James Wilson', 'Age': 35}
After cleared all member : {}
```

50

Dictionary

```
# Demo 02-31

mydict1 = {1:"Python", 2:'Java', 3:"SQL", 4:"R"}
for i in mydict1:
    print(mydict1[i])

print("-------------------------------------------")

mydict2 = {1:"Python", 2:(100, 200), '3':[30, 40, 50], 4:{"Name":"Sunisa"}}
for i in mydict2:
 print(mydict2[i])
```

```
Python
Java
SQL
R
---------------------------
Python
(100, 200)
[30, 40, 50]
{'Name': 'Sunisa'}
```

51

Sets

- Sets are used to store multiple items in a single variable.
- Set is one of 4 built-in data types in Python used to store collections of data, the other 3 are List, Tuple, and Dictionary, all with different qualities and usage.
- A set is a collection which is unordered, unchangeable*, and unindexed.
- Set items are unchangeable, but you can remove items and add new items.

52

Sets

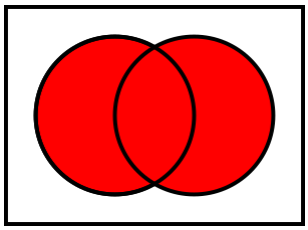
```
# Demo 02-32

mybasket = {'apple', 'orange', 'apple', 'pear', 'orange', 'banana'}
print(mybasket)

print('--------------------------------')

setA = set('Python Programming')
setB = set('Web Application Development')
setC = set('11222333444455555')
print(setA)
print(setB)
print(setC)
```

```
{'apple', 'banana', 'pear', 'orange'}
--------------------------------
{'r', 'a', ' ', 'm', 'i', 't', 'n', 'h', 'y', 'o', 'g', 'P'}
{'a', 'b', ' ', 'p', 'e', 'W', 'l', 'D', 'i', 't', 'n', 'v', 'o', 'm', 'A', 'c'}
{'4', '1', '2', '5', '3'}
```

53

Sets - Union

```
# Demo 02-33

myset1 = set((10,10,20,30,50,80,120))
myset2 = set((20,30,50,70,100,120))

print("Set myset1 :",myset1)
print("Set myset2 :",myset2)
print("Set myset1 | myset2 :",myset1|myset2)
```

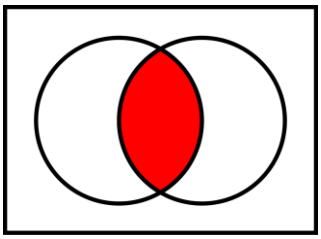


```
Set myset1 : {10, 80, 50, 20, 120, 30}
Set myset2 : {100, 70, 50, 20, 120, 30}
Set myset1 | myset2 : {100, 70, 10, 80, 50, 20, 120, 30}
```

54

Sets - Intersection

```
# Demo 02-34

myset1 = set((10,10,20,30,50,80,120))
myset2 = set((20,30,50,70,100,120))

print("Set myset1 :",myset1)
print("Set myset2 :",myset2)
print("Set myset1 & myset2 :",myset1&myset2)
```

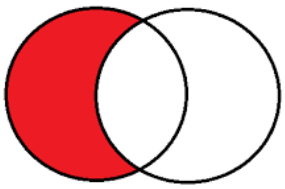


```
Set myset1 : {10, 80, 50, 20, 120, 30}
Set myset2 : {100, 70, 50, 20, 120, 30}
Set myset1 & myset2 : {120, 50, 20, 30}
```

55

Sets - Difference

```
# Demo 02-35

myset1 = set((10,10,20,30,50,80,120))
myset2 = set((20,30,50,70,100,120))

print("Set myset1 :",myset1)
print("Set myset2 :",myset2)
print("Set myset1 - myset2 :",myset1-myset2)
```

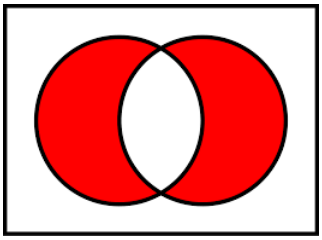


```
Set myset1 : {10, 80, 50, 20, 120, 30}
Set myset2 : {100, 70, 50, 20, 120, 30}
Set myset1 - myset2 : {80, 10}
```

56

Sets - Symmetric Difference

```
# Demo 02-36

myset1 = set((10,10,20,30,50,80,120))
myset2 = set((20,30,50,70,100,120))

print("Set myset1 :",myset1)
print("Set myset2 :",myset2)
print("Set myset1 ^ myset2 :",myset1^myset2)
```



```
Set myset1 : {10, 80, 50, 20, 120, 30}
Set myset2 : {100, 70, 50, 20, 120, 30}
Set myset1 ^ myset2 : {80, 100, 70, 10}
```

57

Arithmetic Operators

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

58

Assignment Operators

Operator	Name	Example
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3

59

Assignment Operators (Con.t)

Operator	Name	Example
**=	x **= 3	x = x ** 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

60

Comparison Operators

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

61

Logical Operators

Operator	Name	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

62

Identity Operators

Operator	Name	Example
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

63

Membership Operators

Operator	Name	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

64

Bitwise Operators

Operator	Name	Description	Example
&	AND	Sets each bit to 1 if both bits are 1	x & y
|	OR	Sets each bit to 1 if one of two bits is 1	x | y
^	XOR	Sets each bit to 1 if only one of two bits is 1	x ^ y
~	NOT	Inverts all the bits	~x
<<	Zero fill left shift	Shift left by pushing zeros in from the right and let the leftmost bits fall off	x << 2
>>	Signed right shift	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off	x >> 2

65

if statements

```
# Demo 02-37

myname = input("Enter name :")
price = float(input("Enter price of %s :" %myname))

if price >= 1000:
    temp = price * 0.05
    price = price - temp

vat = price * 0.07
price = price + vat

print("The price of %s (inc VAT 7%%) is %.2f %s" %(myname, price, "Baht."))
```

```
Enter name : Arsenal Home Jersey
Enter price of  Arsenal Home Jersey :1500
The price of  Arsenal Home Jersey (inc VAT 7%) is 1524.75 Baht.
```

66

if-else statements

```
# Demo 02-38

mynum = int(input("Enter Integer Number : "))

if mynum % 2 != 0:
    print(mynum, " is Odd.")
else:
    print(mynum," is Even.")

print("End of Program.")
```

```
Enter Integer Number : 5
5  is Odd.
End of Program.
```

```
Enter Integer Number : 10
10  is Even.
End of Program.
```

67

if-elif-else statements

```
# Demo 02-39

myscore = float(input("Enter your score : "))
mymsg = "Your grade is "

if myscore >= 80:
    print(mymsg + "A")
elif myscore >= 70:
    print(mymsg + "B")
elif myscore >= 60:
    print(mymsg + "C")
elif myscore >= 50:
    print(mymsg + "D")
else:
    print(mymsg + "F")

print("End of Program.")
```

```
Enter your score : 75
Your grade is B
End of Program.
```

68

Nested if

```
# Demo 02-40

myvar = 200

if myvar < 500:
    print("Expression value is less than 500")
    if myvar == 400:
        print("Which is 400")
    elif myvar == 300:
        print("Which is 300")
    elif myvar == 200:
        print("Which is 200")
elif myvar < 200:
    print("Expression value is less than 200")
else:
    print("Flase Expression.")
```

```
Expression value is less than 500
Which is 200
```

69

Condition Example

```
# Demo 02-41

mynum1 = float(input("Enter first number : "))
mynum2 = float(input("Enter second number : "))
mynum3 = float(input("Enter third number : "))

if mynum1 < mynum2:
    if mynum1 < mynum3:
        minnum = mynum1
    else:
        minnum = mynum3
elif mynum2 < mynum3:
    minnum = mynum2
else:
    minnum = mynum3

print("Minimum Number is : ",minnum)
```

```
Enter first number : 40
Enter second number : 20
Enter third number : 50
Minimum Number is :  20.0
```

70

While Loop

```
# Demo 02-42

count = 0
while (count < 10):
    print ('The count is:', count)
    count = count + 1

print ("The end of program")
```

```
The count is: 0
The count is: 1
The count is: 2
The count is: 3
The count is: 4
The count is: 5
The count is: 6
The count is: 7
The count is: 8
The count is: 9
The end of program
```

71

While with else

```
# Demo 02-43

count = 0
while count < 9:
    print(count, " is less than 9.")
    count = count + 1
else:
    print(count," is not less than 9.")
```

```
0  is less than 9.
1  is less than 9.
2  is less than 9.
3  is less than 9.
4  is less than 9.
5  is less than 9.
6  is less than 9.
7  is less than 9.
8  is less than 9.
9  is not less than 9.
```

72

While Loop Example

```
# Demo 02-44

count = 1
sum = 0.0

print("Exit this program, please type 0 or 0.0 :")

mynum = float(input("Enter a number : "))
while mynum != 0 or mynum != 0.0:
    sum += mynum
    avg = sum / count
    count += 1
    print("Average of number is : ", avg)
    mynum = float(input("Enter a number :"))
```

```
Exit this program, please type 0 or 0.0 :
Enter a number : 5
Average of number is :  5.0
Enter a number :10
Average of number is :  7.5
Enter a number :15
Average of number is :  10.0
Enter a number :0
```

73

break

```
# Demo 02-45

for chr in 'Python':
    if chr == 'h':
        break
    print('Current Character :', chr)

print("---------------------------------------")

myvar = 10
while myvar > 0:
    print('Current variable value :', myvar)
    myvar = myvar - 1
    if myvar == 5:
        break
```

```
Current Character : P
Current Character : y
Current Character : t
------------------------------
Current variable value : 10
Current variable value : 9
Current variable value : 8
Current variable value : 7
Current variable value : 6
```

74

continue

```
# Demo 02-46

for chr in 'Python':
    if chr == 'h':
        continue
    print('Current Character :', chr)

print("---------------------------------------")

myvar = 10
while myvar > 0:
    print('Current variable value :', myvar)
    myvar = myvar - 1
    if myvar == 5:
        continue
```

```
Current Character : P
Current Character : y
Current Character : t
Current Character : o
Current Character : n
---------------------------------------
Current variable value : 10
Current variable value : 9
Current variable value : 8
Current variable value : 7
Current variable value : 6
Current variable value : 5
Current variable value : 4
Current variable value : 3
Current variable value : 2
Current variable value : 1
```

75

pass

- for loops cannot be empty, but if you for some reason have a for loop with no content, put in the pass statement to avoid getting an error.

```
# Demo 02-47

for chr in 'Python':
    if chr == 'h':
        pass
        print('This is pass block')
    print('Current Character :', chr)
```

```
Current Character : P
Current Character : y
Current Character : t
This is pass block
Current Character : h
Current Character : o
Current Character : n
```

76

for Loop

```
# Demo 02-48

mynum = int(input("Enter integer number(odd) : "))
flag = False
num = 1
for i in range(3, mynum+2, 2):
    if flag == False:
        num = num-1 / i
        flag = True
    else:
        num = num + 1 / i
        flag = False
print ("Result of PI/4 = %.4f" %num)
```

```
Enter integer number(odd) : 15
Result of PI/4 = 0.7543
```

77

FUNCTIONS

78

Functions

- A function is a block of code which only runs when it is called.
- You can pass data, known as parameters, into a function.
- A function can return data as a result.

79

range() Functions

```
# Demo 03-01

begin, end = 1, 10
sum = 0

for i in range(begin, end + 1):
    sum += i

print(f'Result of {begin}..{end} = {sum}')

begin, end = 10, 20
sum = 0
for i in range(begin, end + 1):
    sum += i

print(f'Result of {begin}..{end} = {sum}')
```

```
Result of 1..10 = 55
Result of 10..20 = 165
```

80

Functions

```
# Demo 03-02
2 usages
def sumnum(begin, end):
    sum = 0
    for i in range(begin, end + 1):
        sum += i

    print(f'Result of {begin}..{end} = {sum}')

sumnum(1, 10)
sumnum(10, 20)
```

```
Result of 1..10 = 55
Result of 10..20 = 165
```

81

Pass by Reference

```
# Demo 03-03
1 usage
def changelist(mylist):
    mylist.append([100, 200, 300, 400, 500])
    print("Values inside the function : ", mylist)
    return

mylist = [10,20,30]
changelist(mylist)
print ("Values outside the function : ", mylist)
```

```
Values inside the function :  [10, 20, 30, [100, 200, 300, 400, 500]]
Values outside the function :  [10, 20, 30, [100, 200, 300, 400, 500]]
```

82

Local Variable

```
# Demo 03-04
1 usage
def changelist(mylist):
    mylist = [40,50, 60, 70]  # Assign new reference in mylist
    print("Values inside the function : ", mylist)
    return

mylist = [10,20,30]
changelist(mylist)
print ("Values outside the function : ", mylist)
```

```
Values inside the function :  [40, 50, 60, 70]
Values outside the function :  [10, 20, 30]
```

83

Pass Parameter to Functions

```
# Demo 03-05
2 usages
def circlearea(radius):
    if radius <= 0:
        return
    else:
        return 3.14 * radius ** 2

mycircle1 = circlearea(0)
print(mycircle1)

mycircle2 = circlearea(10)
print(mycircle2)
```

```
None
314.0
```

84

Pass Parameter to Functions

```python
# Demo 03-06
1 usage
def printinfo(name, age):
    print ("Name : ", name)
    print ("Age : ", age)
    return

printinfo('Sunisa', 35)
```

```
Name :  Sunisa
Age :  35
```

85

Swapping Arguments

```python
# Demo 03-07
1 usage
def printinfo(name, age):
    print ("Name : ", name)
    print ("Age : ", age)
    return

printinfo(age=40, name='Tissana')
```

```
Name :  Tissana
Age :  40
```

86

Default Arguments

```python
# Demo 03-08
2 usages
def printinfo(name, age=25):
    print ("Name : ", name)
    print ("Age : ", age)
    return

printinfo(name='Somsak')
print('--------------------------------')
printinfo(age=30, name='Veerapong')
```

```
Name :  Somsak
Age :  25
--------------------
Name :  Veerapong
Age :  30
```

87

Lambda Functions

```python
# Demo 03-09

myfunc1 = lambda mynum1 : mynum1 + 50
print(myfunc1(100))
print('--------------------------------')

myfunc2 = lambda mynum1, mynum2 : mynum1 * mynum2
print(myfunc2(10, 5))
print('--------------------------------')

1 usage
def myfunc3(n):
  return lambda mynum3 : mynum3 * n

mydouble = myfunc3(2)
print(mydouble(10))
```

```
150
--------------------
50
--------------------
20
```

88

Return

```
# Demo 03-10
1 usage
def sum(arg1, arg2):
    total = arg1 + arg2
    print("Result inside the function : ", total)
    return total

total = sum(20, 50)
print ("Result outside the function : ", total)
```

```
Result inside the function :  70
Result outside the function :  70
```

89

Functions Example

```
# Demo 03-11
1 usage
def menu():
    print("Welcome to Calculator Program")
    print("Your options are :")
    print("****************************")
    print("1-Addition")
    print("2-Subtraction")
    print("3-Multiplication")
    print("4-Division")
    print("5-Exit Program")
    print("****************************")
    return int(input("Choose your option : "))
```

90

Functions Example (Cont.)

```
1 usage
def add(mynum1, mynum2):
    print("You chose the Addition")
    print("Result of ", mynum1, "+", mynum2, "=", mynum1 + mynum2)
    return mynum1 + mynum2

1 usage
def sub(mynum1, mynum2):
    print("You chose the Subtraction")
    print("Result of ", mynum1, "-", mynum2, "=", mynum1 - mynum2)
    return mynum1 - mynum2
```

91

Functions Example (Cont.)

```
1 usage
def mul(mynum1, mynum2):
    print("You chose the Multiplication")
    print("Result of ", mynum1, "*", mynum2, "=", mynum1 * mynum2)
    return mynum1 * mynum2

1 usage
def div(mynum1, mynum2):
    print("You chose the Division")
    if mynum2 != 0:
        print("Result of ", mynum1, "/", mynum2, "=", mynum1 / mynum2)
        return mynum1 / mynum2
    else:
        print("Can't divide by zero !!!")
        return False
```

92

Functions Example (Cont.)

```
1 usage
def main():
    loop = 1
    choice = 0
    while loop == 1:
        choice = menu()
        if choice == 1:
            add(int(input("Number 1 : ")), int(input("Number 2 : ")))
        elif choice == 2:
            sub(int(input("Number 1 : ")), int(input("Number 2 : ")))
        elif choice == 3:
            mul(int(input("Number 1 : ")), int(input("Number 2 : ")))
        elif choice == 4:
            div(int(input("Number 1 : ")), int(input("Number 2 : ")))
        elif choice == 5:
            loop = 0
    else:
        print("The End of Program.")

if __name__ == "__main__":
    main()
```

93

Functions Example (Cont.)

```
Welcome to Calculator Program
Your options are :
*****************************
1-Addition
2-Subtraction
3-Multiplication
4-Division
5-Exit Program
*****************************
Choose your option : 4
Number 1 : 5
Number 2 : 0
You chose the Division
Can't divide by zero !!!
```

```
Welcome to Calculator Program
Your options are :
*****************************
1-Addition
2-Subtraction
3-Multiplication
4-Division
5-Exit Program
*****************************
Choose your option : 3
Number 1 : 5
Number 2 : 10
You chose the Multiplication
Result of  5 * 10 = 50
```

```
Welcome to Calculator Program
Your options are :
*****************************
1-Addition
2-Subtraction
3-Multiplication
4-Division
5-Exit Program
*****************************
Choose your option : 5
The End of Program.
```

94

MODULES AND PACKAGES

95

Modules

- A module is a file consisting of Python code.
- A module can define functions, classes and variables. It can also include runnable code.
- A module allows you to logically organize your Python code.
- A module is a Python object with arbitrarily named attributes that you can bind and reference.

96

Create Modules

```python
# Create Module : CalculateAreaRectangle
def rectangle(width, height):
    return width * height

def squre(width1, width2):
    return width1 * width2

def parallelogram(height, base):
    return height * base

def trapezoid(sumofpararell, height):
    return 0.5 * sumofpararell * height
```

97

Call Modules

```python
# Demo 04-01
import CalculateAreaRectangle

print("Area of Rectangle : ",CalculateAreaRectangle.rectangle(10, 15))
print("Area of Squre : ",CalculateAreaRectangle.squre(5, 5))
print("Area of Parallelogram : ",CalculateAreaRectangle.parallelogram(5.5, 10.5))
print("Area of Trapezoid :",CalculateAreaRectangle.trapezoid(5, 2.5))
```

```
Area of Rectangle :  150
Area of Squre :  25
Area of Parallelogram :  57.75
Area of Trapezoid : 6.25
```

98

Packages

A package is a hierarchical file directory structure that defines a single Python application environment that consists of modules and subpackages and sub-subpackages, and so on.

phone

__init__.py (required to make Python treat *phone* as a package)

g3.py

isdn.py

pots.py

99

Packages

```
# Demo : Create CalculateAreaTriangle.py

def triangle(height, base):
    return 1 / 2 * height * base

def equilateral(width):
    return 3**(1/2) * width * width

def isosceles(base, sidebase):
    return base/4 * 4 * (((sidebase*sidebase) - (base*base)))**(1/2)

def pythagorean(perpendicular):
    return 0.5 * perpendicular * perpendicular
```

100

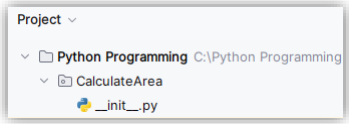
Packages (Create __init__.py)

```
#This initial file for CalculateArea Package (__init__.py)

from CalculateAreaRectangle import rectangle, squre, parallelogram, trapezoid
from CalculateAreaTriangle import triangle, equilateral, isosceles, pythagorean
```

101

Packages

```
# Demo 04-02
import CalculateArea

print("Area of squre : ",CalculateArea.squre(5.5, 5.5))
print("Area of rectangle : ",CalculateArea.rectangle(5.5, 7.5))
print("Area of triangle : ",CalculateArea.triangle(3.5, 2.5))
print("Area of equilateral : ",CalculateArea.equilateral(2.5))
```




```
Area of squre :  30.25
Area of rectangle :  41.25
Area of triangle :  4.375
Area of equilateral :  10.825317547305481
```

102

EXCEPTIONS

103

Exceptions

- An exception is an event, which occurs during the execution of a program that disrupts the normal flow of the program's instructions.
- When a Python script encounters a situation that it cannot cope with, it raises an exception.
- An exception is a Python object that represents an error.
- When a Python script raises an exception, it must either handle the exception immediately otherwise it terminates and quits.

104

Exceptions Handling

```
try:
    normal statement(s)
except Exception 1:
    exception statement(s)
except Exception 2:
    exception statement(s)
    ..........
except Exception n:
    exception statement(s)
else:
    normal statement(s)
```

105

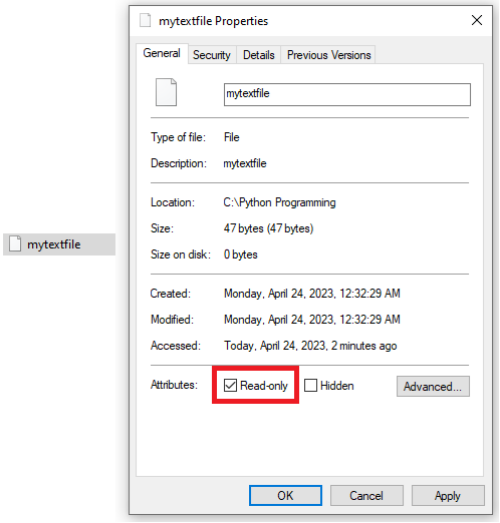
try...except

```
# Demo 05-01
try:
    myfile = open("mytextfile", "W")
    myfile.write("This is my file for Exception Handling Example.")
except IOError:
    print("Error - can not find file or read data.")
else:
    print("Written the file successfully.")
    myfile.close()
```

```
Written the file successfully.
```

106

try...except



107

try...except

```
# Demo 05-01
try:
    myfile = open("mytextfile", "W")
    myfile.write("This is my file for Exception Handling Example.")
except IOError:
    print("Error - can not find file or read data.")
else:
    print("Written the file successfully.")
    myfile.close()
```

```
Error - can not find file or read data.
```

108

Undefined Exception

```
# Demo 05-02
try:
    myfile = open("mytextfile", "w")
    myfile.write("This is my file for Exception Handling Example.")
except:
    print("IO Error with File.")
else:
    print("Written the file successfully.")
    myfile.close()
```

```
IO Error with File.
```

109

Multiple Exceptions

```
# Demo 05-03
try:
    myfile = open('integers.txt')
    mystream = myfile.readline()
    i = int(mystream.strip())
except IOError:
    print("I/O Error.")
except ValueError:
    print("No valid integer.")
except:
    print("Unexpected error.")
```

```
I/O Error.
```

110

Multiple Exceptions

```python
# Demo 05-04
try:
    myfile = open("testfile2")
    myfile.write("This is my file for exception handling!!")
except(IOError, ValueError, SystemError):
    print("Error: can not find file or read data.")
else:
    print("Written file successfully")
    myfile.close()
```

```
Error: can not find file or read data.
```

111

Exception Arguments

```python
# Demo 05-05
1 usage
def temp_convert(var):
    try:
        return int(var)
    except ValueError as Args:
        print("Argument doesn't a numbers\n", Args.args)

temp_convert("Thailand")
```

```
Argument doesn't a numbers
 ("invalid literal for int() with base 10: 'Thailand'",)
```

112

raise

```
# Demo 05-06
2 usages
def functionName(mynum):
    if mynum < 1:
        raise ("Invalid Number!", mynum)
        print("The code would not be executed")
    print("Process Success.")

functionName(10)
functionName(0)
```

```
Process Success.
Traceback (most recent call last):
  File "C:\Python Programming\05-06-Raise.py", line 9, in <module>
    functionName(0)
  File "C:\Python Programming\05-06-Raise.py", line 4, in functionName
    raise ("Invalid Number!", mynum)
TypeError: exceptions must derive from BaseException
```

113

FILE MANAGEMENT

114

File Attributes

```
# Demo 06-01
# Show information about file
myfile = open("MyFile.txt", "w")
print ("Name of file : ", myfile.name)
print ("Closed or not : ", myfile.closed)
print ("Opening mode : ", myfile.mode)
```

```
Name of file :  MyFile.txt
Closed or not :  False
Opening mode :  w
```

115

Read File

```
#Demo 06-02
FilePath = "C:\Python Programming\README.txt"
try:
    f = open(FilePath)
    str = f.read()
    print(str)
    print("*********************************************")

    f = open(FilePath)
    str = f.readlines(15)
    print(str)
    print("*********************************************")

    f = open(FilePath)
    while 1:
        line = f.readline()
        if len(line):
            print(line)
        else:
            break
except IOError as err:
    print("Cann't open file because :",err.args)
else:
    print("*********************************************")
    print("This file was closed!")
    f.close()
```

```
This is Python Programming
==================================
Copyright (c) 2001, 2002, 2003, 2004, 2005,
*********************************************
['This is Python Programming\n']
*********************************************
This is Python Programming

==================================

Copyright (c) 2001, 2002, 2003, 2004, 2005,
*********************************************
This file was closed!
```

116

Linecache (.getline)

```
# Demo 06-03
import linecache

FilePath = "C:\Python Programming\README.txt"

for line in range(4):
    print(linecache.getline(FilePath, line))
linecache.clearcache()
```

```
This is Python Programming

====================================

Copyright (c) 2001, 2002, 2003, 2004, 2005,
```

117

split

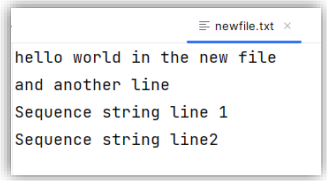
```
# Demo 06-04
FilePath = "C:\Python Programming\README.txt"
wordTemp = []
wordCount = 0
file = open(FilePath, 'r')

for line in file:
    for word in line.split():
        wordTemp.append(word)
        wordCount = wordCount + 1

print(wordTemp)
print(wordCount)
```

```
['This', 'is', 'Python', 'Programming', '====================================', 'Copyright', '(c)', '2001,', '2002,', '2003,', '2004,', '2005,']
12
```

118

write and writelines

```
# Demo 06-05
string1 = "hello world in the new file\n"
string2 = "and another line\n"
sequence = ["Sequence string line 1\n", "Sequence string line2"]

file = open("newfile.txt", "w")
file.write(string1)
file.write(string2)
file.writelines(sequence)
file.close()
```



119

OBJECT ORIENTED PROGRAMMING

120

Create Class

```
# Demo 07-01
2 usages
class Car:
    # attributes
    color = "Unknow"
    brand = "Unknow"
    numberofseats = 4
    numberofwheels = 4
    maxSpeed = 0
    regnumber = 0

    def __init__(self, color, brand, numberofseats, numberofwheels, maxSpeed):
        self.color = color
        self.brand = brand
        self.numberofseats = numberofseats
        self.numberofwheels = numberofwheels
        self.maxSpeed = maxSpeed
        self.regnumber += 1
```

121

Create Class

```
    # methods
    def setColor(self, x):
        self.Color = x

    def setBrand(self, x):
        self.brand = x

    def setNumberOfSeats(self, x):
        self.numberofseats = x

    def setNumberOfWeels(self, x):
        self.numberofwheels = x

    def setMaxSpeed(self, x):
        self.maxSpeed = x

    3 usages
    def printInfo(self):
        print("The color of this car is :", self.color)
        print("The car was brand by :", self.brand)
        print("The number of seats is :", self.numberofseats, "seats.")
        print("The number of wheels is :", self.numberofwheels, "wheels.")
        print("The maximum speed is :", self.maxSpeed, "km/hour")
        print("The registration number is :", self.regnumber)
        print("============================================================")
```

122

Create Class

```
# Creating instance and use it
mycar1 = Car('Black','Toyota',4,4,150)
mycar1.printInfo()
mycar1.color = 'Blue'
mycar1.printInfo()

mycar2 = Car('Silver','Honda',4,4,170)
mycar2.printInfo()
```

```
The color of this car is : Black
The car was brand by : Toyota
The number of seats is : 4 seats.
The number of wheels is : 4 wheels.
The maximum speed is : 150 km/hour
The registration number is : 1
==========================================
The color of this car is : Blue
The car was brand by : Toyota
The number of seats is : 4 seats.
The number of wheels is : 4 wheels.
The maximum speed is : 150 km/hour
The registration number is : 1
==========================================
The color of this car is : Silver
The car was brand by : Honda
The number of seats is : 4 seats.
The number of wheels is : 4 wheels.
The maximum speed is : 170 km/hour
The registration number is : 1
==========================================
```

123

Create Class

```
#Demo 07-02
4 usages
class Employee:
    empCount = 0

    def __init__(self, name, salary):
        self.name = name
        self.salary = salary
        Employee.empCount += 1

    1 usage
    def displayCount(self):
        print("Total Employee =%d" % Employee.empCount)

    2 usages
    def displayEmployee(self):
        print("Name : ", self.name, ", Salary: ", self.salary)
```

124

Create Class

```
#Creating instance objects
employee1 = Employee("Phanwadee", 75000)
employee2 = Employee("Sunisa", 85000)

employee1.displayEmployee()
employee2.displayEmployee()

employee1.displayCount()
```

```
Name :  Phanwadee , Salary:  75000
Name :  Sunisa , Salary:  85000
Total Employee =2
```

125

Destroy the object

```
# Demo 07-03
1 usage
class Car:
    def __init__(self, color='red', speed=100):
        self.color = color
        self.speed = speed

    # Destructor method
    def __del__(self):
        class_name = self.__class__.__name__
        print(class_name, "object has destroyed")

mycar1 = Car()
mycar2 = mycar1
mycar3 = mycar1

print ("ID[mycar1]=",id(mycar1),", ID[mycar2]=",id(mycar2),", ID[mycar3]=",id(mycar3))

del mycar1
del mycar2
del mycar3
```

```
ID[mycar1]= 2152509035664 , ID[mycar2]= 2152509035664 , ID[mycar3]= 2152509035664
Car object has destroyed
```

126

Inheritance

```python
# Demo 07-04
4 usages
class ClassicCar:
    regnumber = 0
    color = 'white'

    def __init__(self):
        ClassicCar.regnumber += 1
        print("ClassicCar: Constructor")

    2 usages
    def ClassicCarGetReg(self):
        print("ClassicCar: Register number is ",self.regnumber)

    2 usages
    def ClassicCarSetColor(self, color):
        self.color = color
        print("ClassicCar: Set color")

    4 usages
    def ClassicCarGetColor(self):
        print("ClassicCar: Get color is ", ClassicCar.color)

    def __del__(self):
        class_name = self.__class__.__name__
        print("ClassicCar: ", class_name, "object has destroyed")
```

```python
1 usage
class ToyotaCar(ClassicCar):
    def __init__(self):
        self.color = 'gold blond'
        print("Toyota Car: Constructor")

    1 usage
    def ToyotaCarGetColor(self):
        print('Toyota Car: Get color is ', self.color)

1 usage
class HondaCar(ClassicCar):
    def __init__(self):
        self.color = 'black'

    1 usage
    def HondaCarGetColor(self):
        print('Honda Car: Get color is ', self.color)
```

127

Inheritance

```python
mycar1 = ToyotaCar()
mycar1.ClassicCarGetReg()
mycar1.ClassicCarGetColor()
mycar1.ClassicCarSetColor('green')
mycar1.ClassicCarGetColor()
mycar1.ToyotaCarGetColor()

print("----------------------------------")

mycar2 = HondaCar()
mycar2.ClassicCarGetReg()
mycar2.ClassicCarGetColor()
mycar2.ClassicCarSetColor('yello')
mycar2.ClassicCarGetColor()
mycar2.HondaCarGetColor()

print("----------------------------------")

del mycar1
del mycar2
```

```
Toyota Car: Constructor
ClassicCar: Register number is  0
ClassicCar: Get color is  white
ClassicCar: Set color
ClassicCar: Get color is  white
Toyota Car: Get color is  green
----------------------------------
ClassicCar: Register number is  0
ClassicCar: Get color is  white
ClassicCar: Set color
ClassicCar: Get color is  white
Honda Car: Get color is  yello
----------------------------------
ClassicCar:  ToyotaCar object has destroyed
ClassicCar:  HondaCar object has destroyed
```

128

Overriding methods

```
#Demo_07-05
2 usages
class ClassicCar:
    def brake(self):
        print("ClassicCar: Normal brake!")

1 usage
class ToyotaCar(ClassicCar):
    1 usage
    def brake(self):
        print("ToyotaCar: Disk brake pad!")

1 usage
class HondaCar(ClassicCar):
    1 usage
    def brake(self):
        print ("HondaCar: Drum brake pad!")

mycar1 = ToyotaCar()
mycar1.brake()

mycar2 = HondaCar()
mycar2.brake()
```

```
ToyotaCar: Disk brake pad!
HondaCar: Drum brake pad!
```

129

DATABASE

130

Python MySQL

- Python can be used in database applications.
- One of the most popular databases is MySQL.
- To be able to experiment with the code examples in this tutorial, you should have MySQL installed on your computer.
- You can download a MySQL database at https://www.mysql.com/downloads/.

131

Install MySQL Driver

- Python needs a MySQL driver to access the MySQL database.
- In this tutorial we will use the driver "MySQL Connector".
- We recommend that you use PIP to install "MySQL Connector".
- PIP is most likely already installed in your Python environment.

```
python -m pip install mysql-connector-python
```

132

Create Connection

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword"
)

print(mydb)
```

133

Create Database

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword"
)

mycursor = mydb.cursor()

mycursor.execute("CREATE DATABASE mydatabase")
```

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword",
  database="mydatabase"
)
```

134

Create Table

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

mycursor.execute("CREATE TABLE customers (name VARCHAR(255), address VARCHAR(255))")
```

135

Insert Data

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

sql = "INSERT INTO customers (name, address) VALUES (%s, %s)"
val = ("John", "Highway 21")
mycursor.execute(sql, val)

mydb.commit()

print(mycursor.rowcount, "record inserted.")
```

136

Insert Data

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

sql = "INSERT INTO customers (name, address) VALUES (%s, %s)"
val = [
  ('Peter', 'Lowstreet 4'),
  ('Amy', 'Apple st 652'),
  ('Hannah', 'Mountain 21'),
  ('Michael', 'Valley 345'),
  ('Sandy', 'Ocean blvd 2'),
  ('Betty', 'Green Grass 1'),
  ('Richard', 'Sky st 331'),
  ('Susan', 'One way 98'),
  ('Vicky', 'Yellow Garden 2'),
  ('Ben', 'Park Lane 38'),
  ('William', 'Central st 954'),
  ('Chuck', 'Main Road 989'),
  ('Viola', 'Sideway 1633')
]

mycursor.executemany(sql, val)

mydb.commit()

print(mycursor.rowcount, "was inserted.")
```

137

Select Data

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

mycursor.execute("SELECT name, address FROM customers")

myresult = mycursor.fetchall()

for x in myresult:
  print(x)
```

138

Filter Data

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

sql = "SELECT * FROM customers WHERE address LIKE '%way%'"

mycursor.execute(sql)

myresult = mycursor.fetchall()

for x in myresult:
  print(x)
```

139

Order Data

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

sql = "SELECT * FROM customers ORDER BY name"

mycursor.execute(sql)

myresult = mycursor.fetchall()

for x in myresult:
  print(x)
```

140

Delete Data

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

sql = "DELETE FROM customers WHERE address = 'Mountain 21'"

mycursor.execute(sql)

mydb.commit()

print(mycursor.rowcount, "record(s) deleted")
```

141

Update Data

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

sql = "UPDATE customers SET address = 'Canyon 123' WHERE address = 'Valley 345'"

mycursor.execute(sql)

mydb.commit()

print(mycursor.rowcount, "record(s) affected")
```

142

Drop Table

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

sql = "DROP TABLE customers"

mycursor.execute(sql)
```

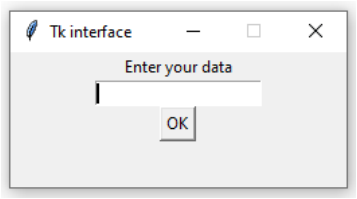
143

GUI

144

Create Basic GUI

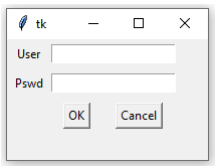
```
# 09-01
from tkinter import *

window = Tk()
window.title('Tk interface')
window.geometry('250x100')
window.minsize(200, 80)
window.resizable(0, 0)

label = Label(text='Enter your data')
entry = Entry()
button = Button(text='OK')

label.pack()
entry.pack()
button.pack()

window.mainloop()
```



145

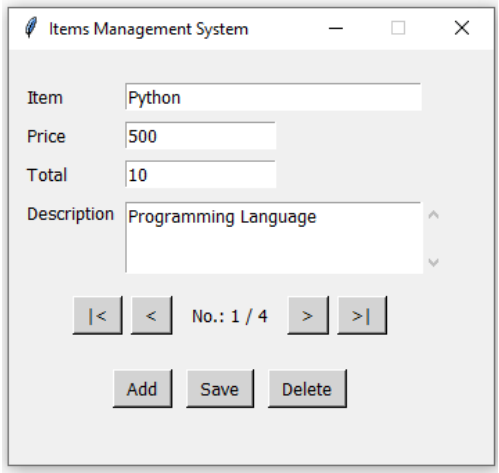
Create Basic GUI

```
#Demo 09-02
from tkinter import *

window = Tk()
window.geometry('200x120')

Label(text='User').grid(row=0, column=0, padx=5)
Entry().grid(row=0, column=1, columnspan=2, pady=5)

Label(text='Pswd').grid(row=1, column=0, padx=5)
Entry().grid(row=1, column=1, columnspan=2, pady=5)

Button(text='OK').grid(row=2, column=1, pady=5)
Button(text='Cancel').grid(row=2, column=2, pady=5)

window.mainloop()
```



146

GUI Example Demo



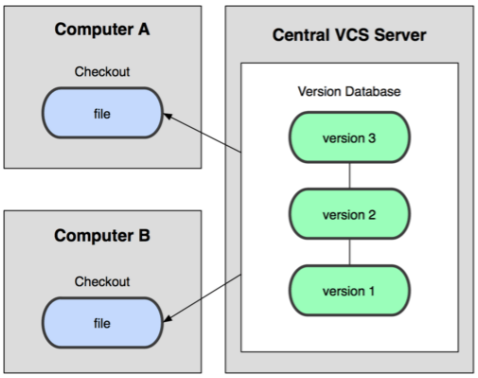
147

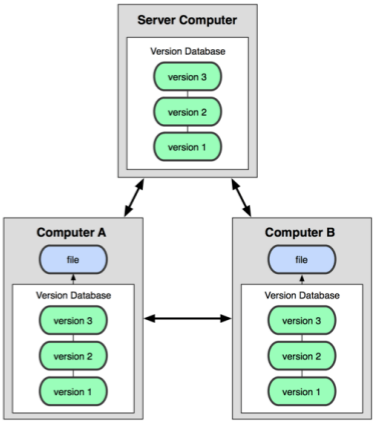
GIT VERSION CONTROL

148

Git

Central Vs. Distributed





149

Git Setup

```
$ git config --global user.name
$ git config --global user.email
$ git config --global color.ui true
```

150

Create a new Repository

```
$ cd project/

$ git init
$ git add .
$ git commit
$ git rm --cached <file>...
$ git reset HEAD <file>
$ git init project002
```

151

Git Clone

```
$ git clone git://github.com/sengopal/simplegit.git
Initialized empty Git repository in
/private/tmp/simplegit/.git/
remote: Counting objects: 100, done.
remote: Compressing objects: 100% (86/86), done.
remote: Total 100 (delta 35), reused 0 (delta 0)
Receiving objects: 100% (100/100), 9.51 KiB, done.
Resolving deltas: 100% (35/35), done.
$ cd simplegit/
$ ls
```

152

Git Status

```
$ git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
# new file:   README
# new file:   hello.py
#
# Changed but not updated:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in
working directory)
#
# modified:   README
#
```

153

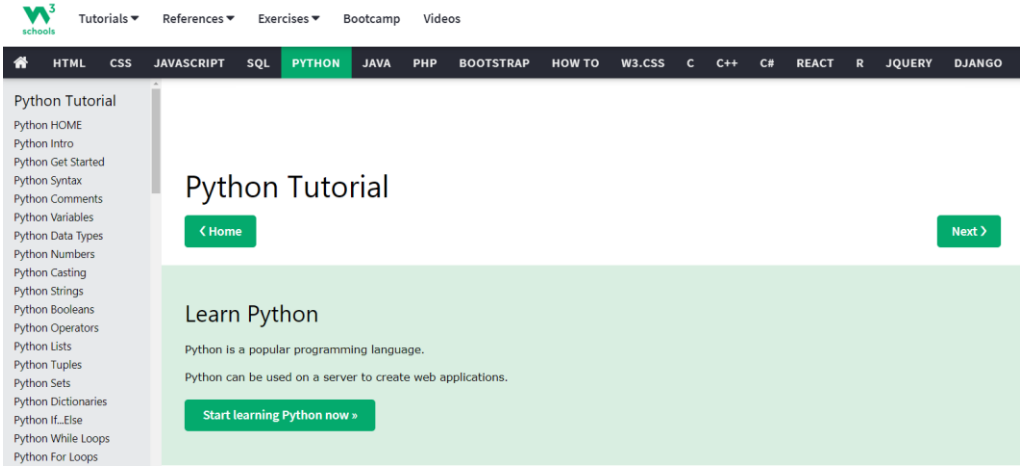
REFERENCE

154

Book

155

Online Content

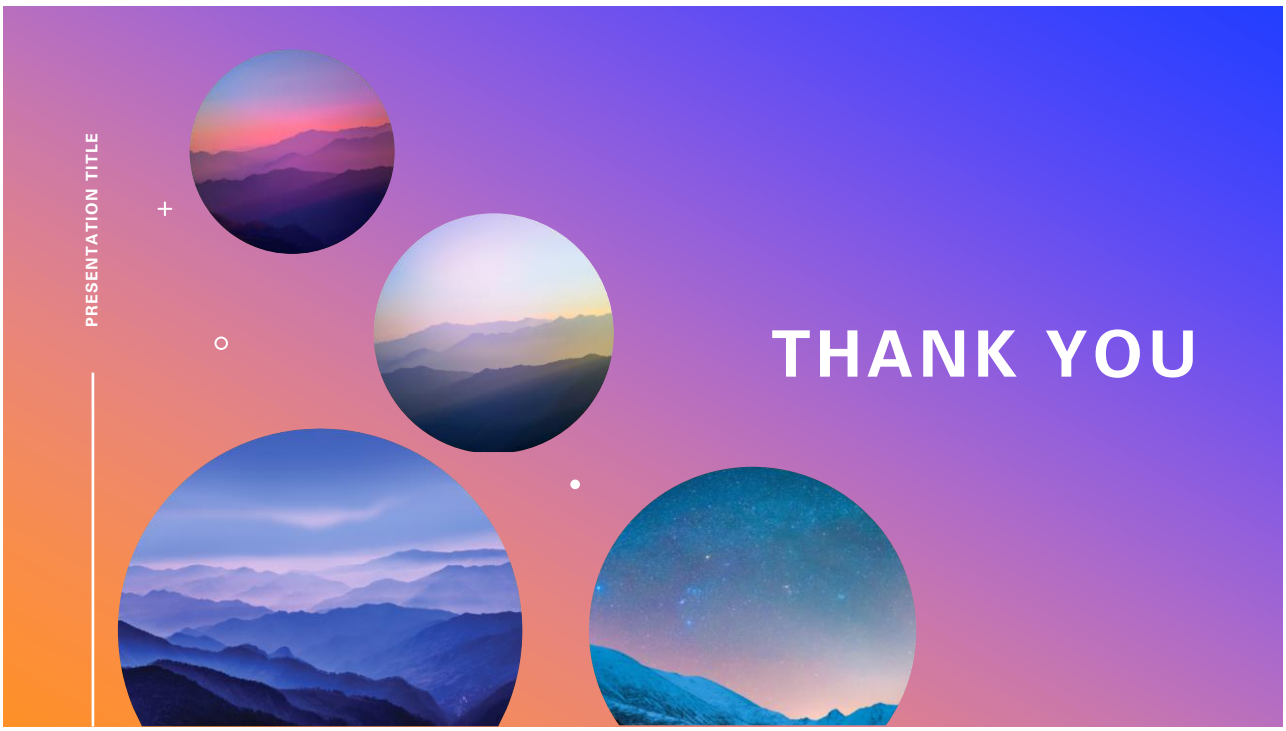


156

Online Content

https://www.slideshare.net/sengopal/git-and-github-101

157



158