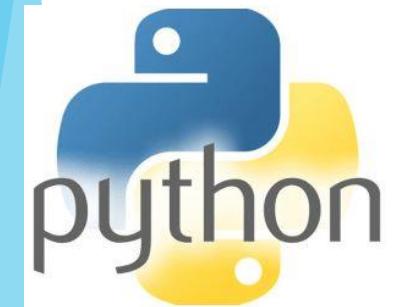


# Python Introduction



自動化開發與認證課

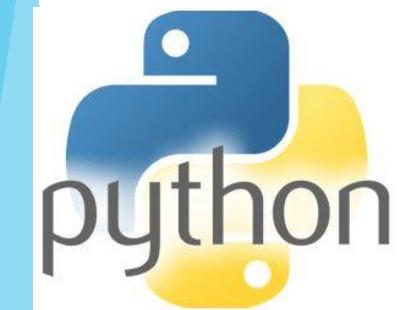
David\_Zhu

2022/5/10

# Directory

- ▶ Python install for Windows
- ▶ Visual Studio Code / Anaconda Spyder
- ▶ Package Install (Pip install)
- ▶ Python
  - ▶ Common
  - ▶ Data Type
  - ▶ Data Structure (Tuple(), List[], Set{}, dict{})
  - ▶ Flow control
  - ▶ Module
  - ▶ Exception
  - ▶ Object-oriented programming
    - ▶ Class
    - ▶ Encapsulation
    - ▶ Inheritance
    - ▶ Abstract Method
    - ▶ Polymorphism
- ▶ Reference

# Python Install



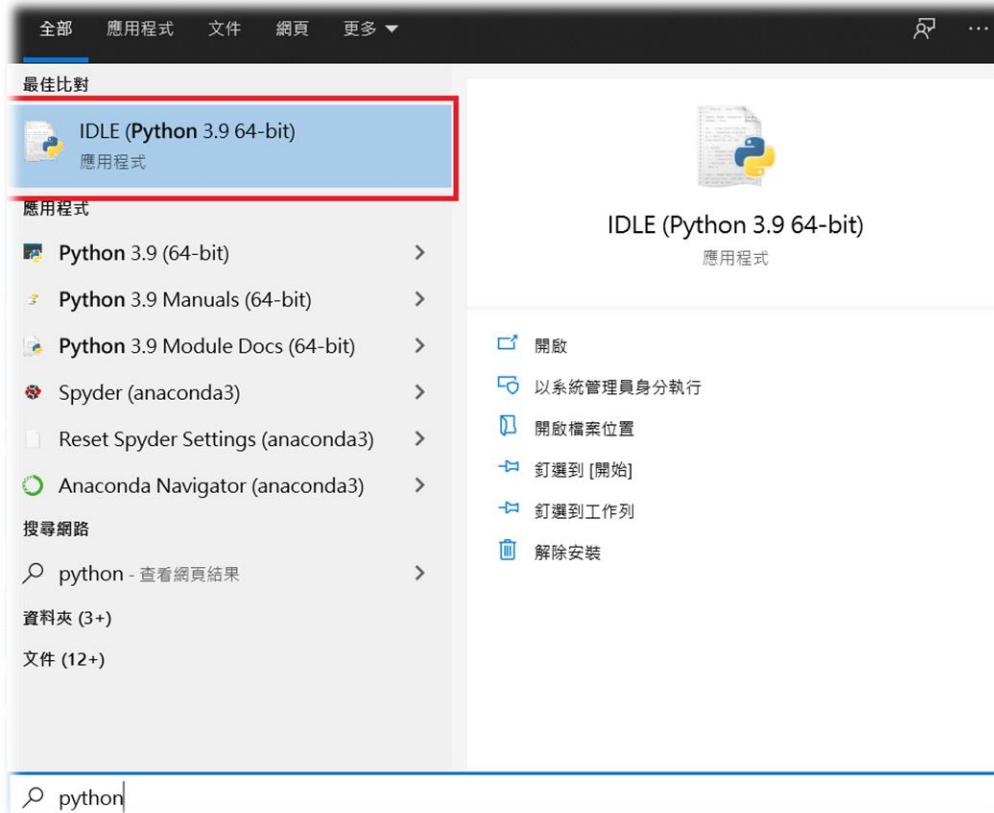
# Python Install

Download : <https://www.python.org/downloads/>

The screenshot shows the Python.org website. At the top, there's a navigation bar with links for About, Downloads (which is highlighted with a red box), Documentation, Community, Success Stories, News, and Events. Below the navigation bar is a search bar with a magnifying glass icon and a 'GO' button. To the right of the search bar is a 'Socialize' button. The main content area features a large Python logo on the left. On the right, there's a section titled 'Compound Data Types' with some explanatory text and a link to 'More about lists in Python 3'. Below this are five numbered buttons (1, 2, 3, 4, 5). At the bottom of the page, there's a quote: 'Python is a programming language that lets you work quickly and integrate systems more effectively.' followed by a 'Learn More' link.



# Python Install

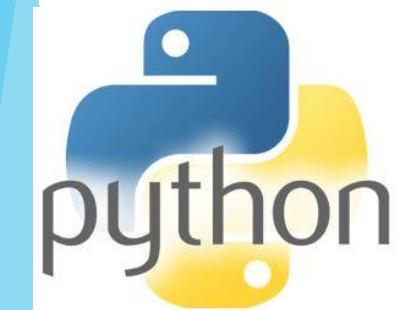


The image shows the IDLE Shell 3.9.2 window. The title bar reads "IDLE Shell 3.9.2". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main window displays the Python shell output:

```
Python 3.9.2 (tags/v3.9.2:1a79785, Feb 19 2021, 13:44:55) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("Hello Word")
Hello Word
>>>
```

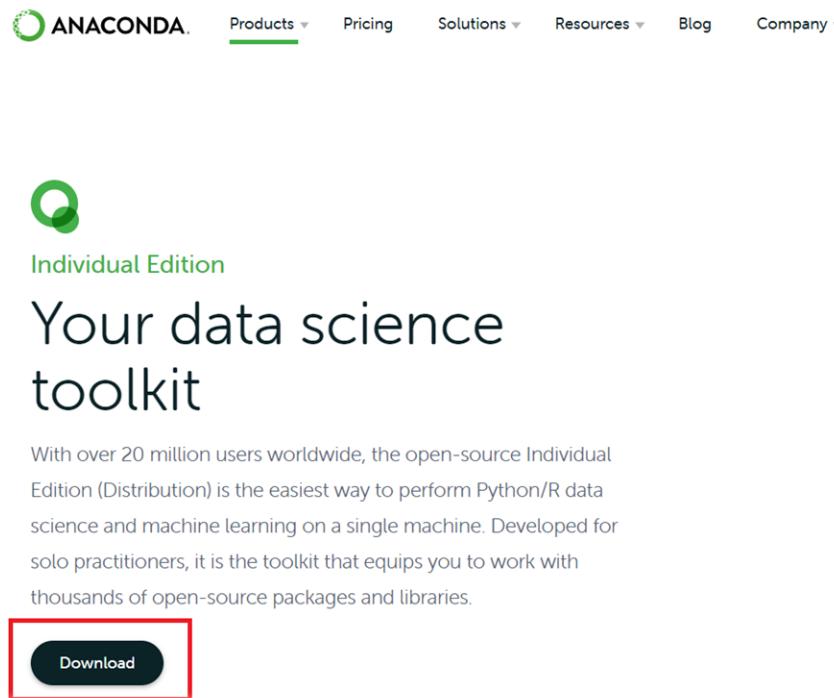
In the bottom right corner of the window, there is a status bar with "Ln: 5 Col: 4".

# Anaconda(Spyder)

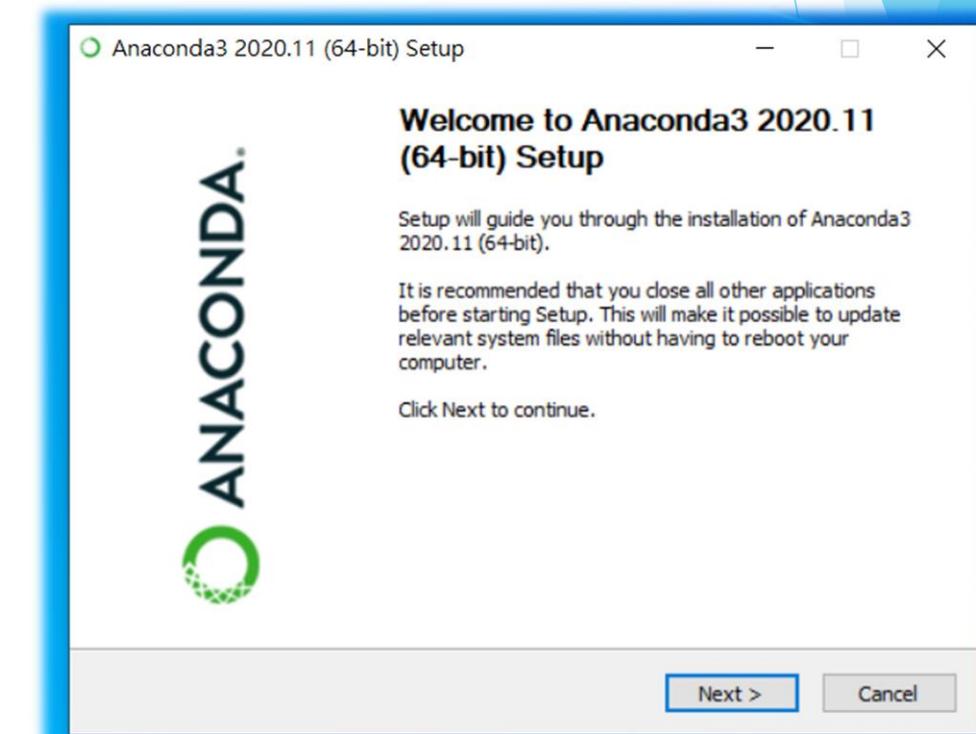


# Anaconda (Spyder)

<https://www.anaconda.com/products/individual>

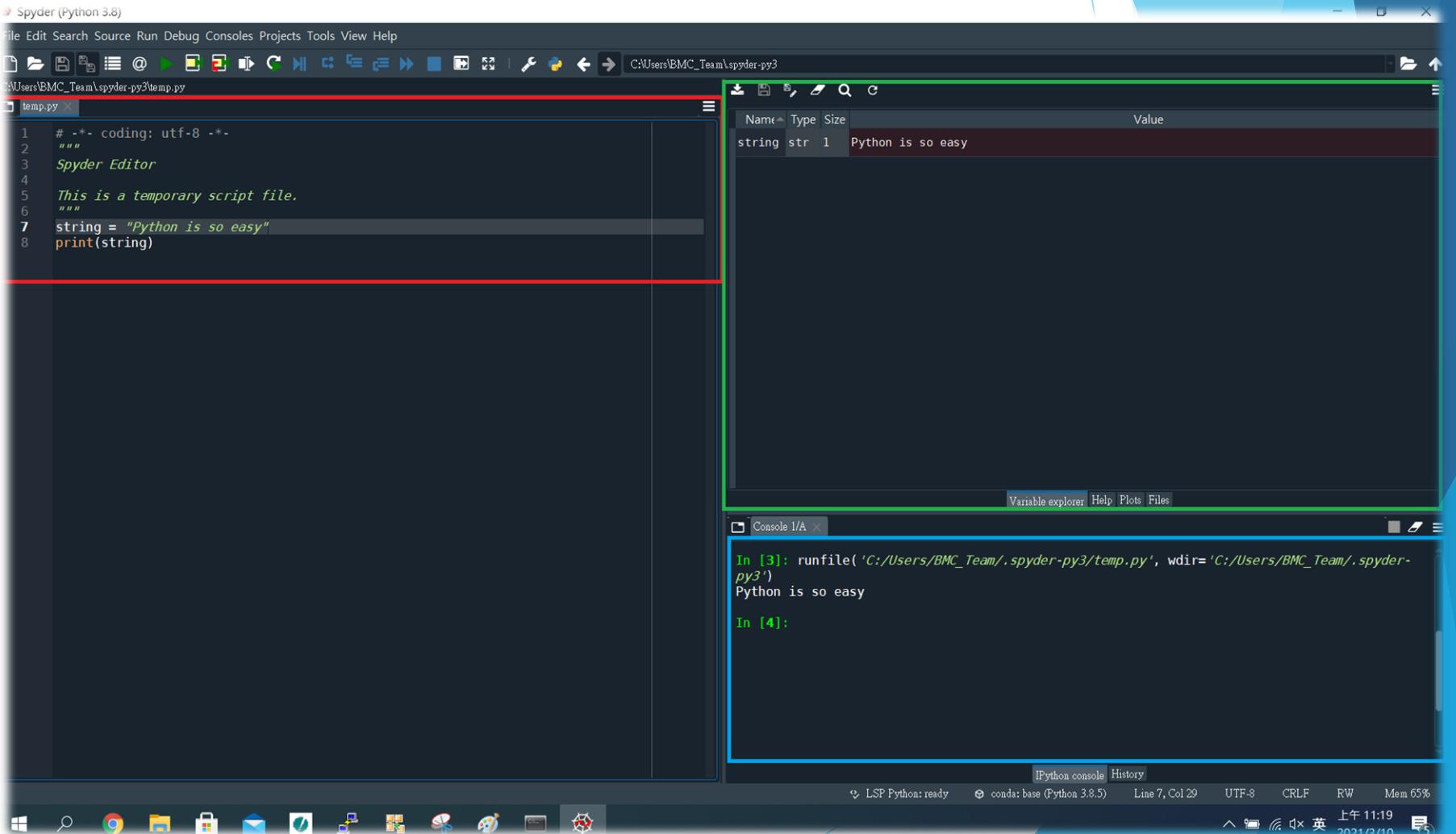


The screenshot shows the Anaconda Individual Edition landing page. At the top, there's a navigation bar with links for Products, Pricing, Solutions, Resources, Blog, and Company. A prominent 'Get Started' button is located in the center of the header. Below the header, there's a large green 'Q' icon followed by the text 'Individual Edition'. The main headline reads 'Your data science toolkit'. A paragraph below explains that with over 20 million users worldwide, the open-source Individual Edition (Distribution) is the easiest way to perform Python/R data science and machine learning on a single machine. It's developed for solo practitioners and equips them to work with thousands of open-source packages and libraries. A red-bordered 'Download' button is at the bottom left.



# Anaconda (Spyder)

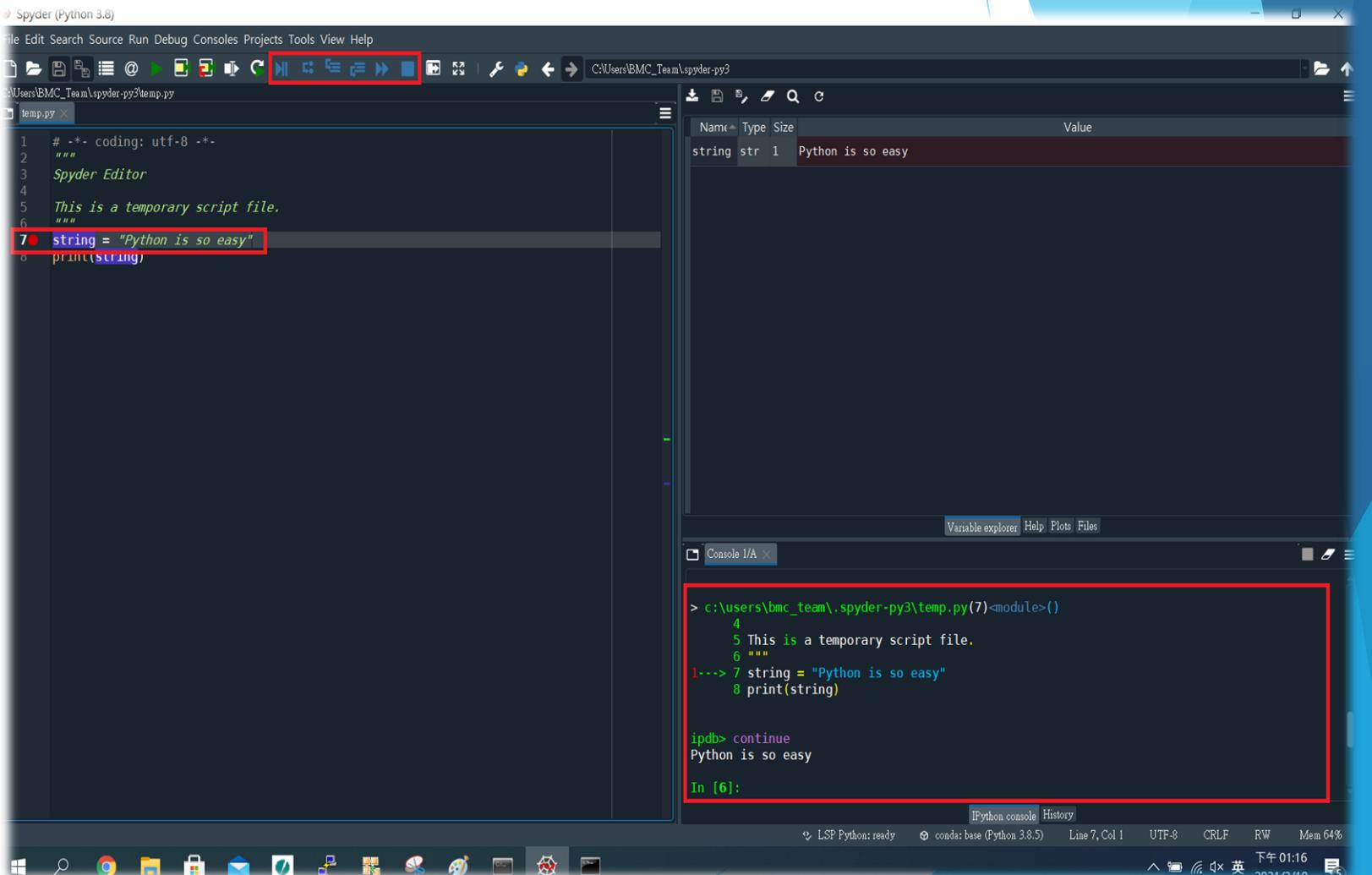
- Browse tabs
- Variables explorer
- Console



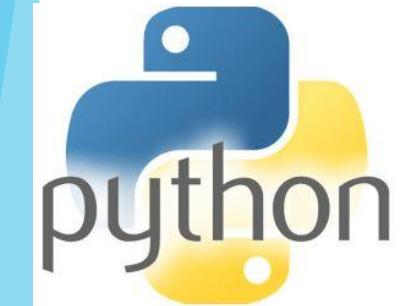
# Anaconda (Spyder)

## Debug mode

- By step to execute
- View all of the variables

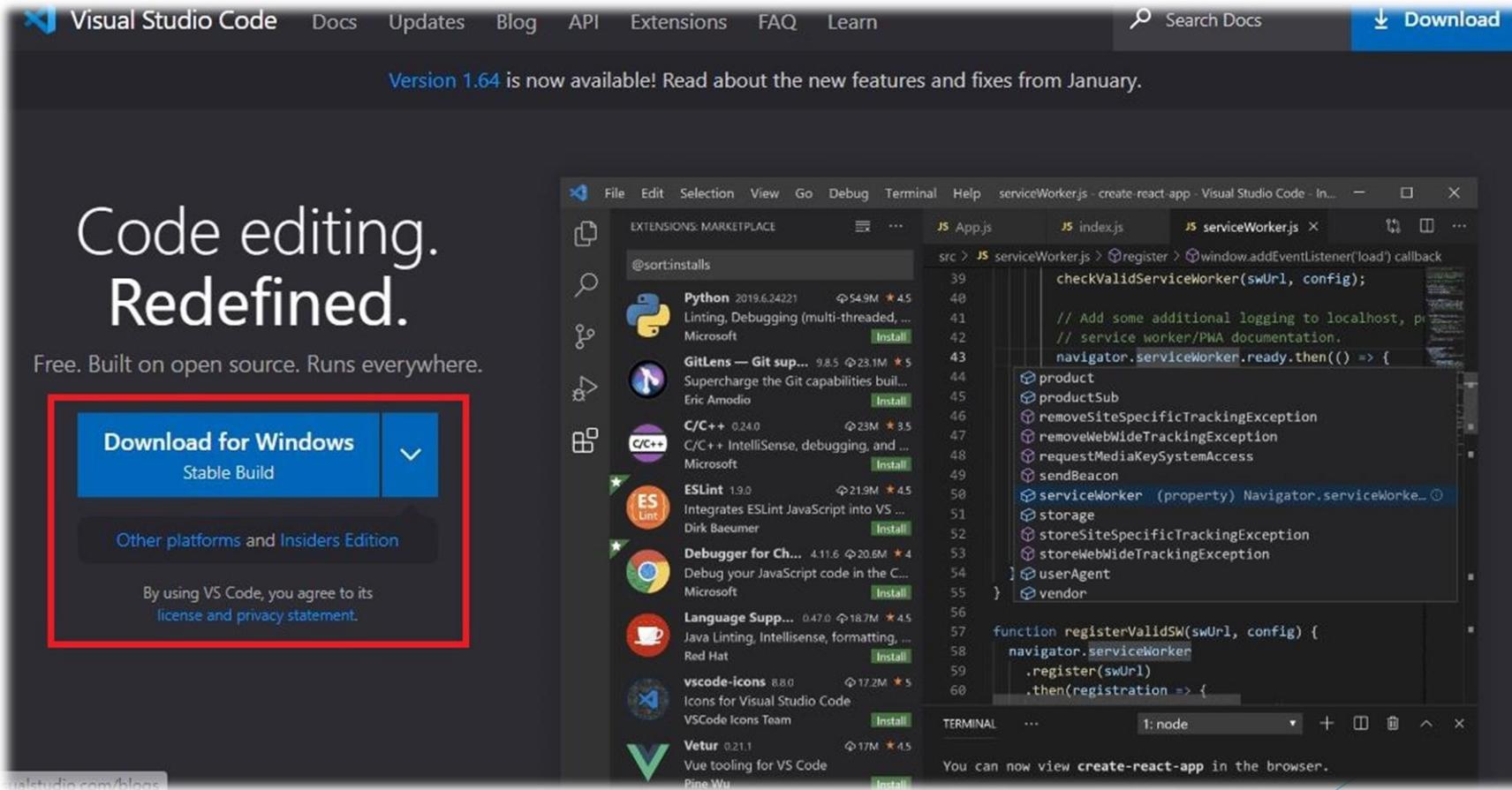


# Visual Studio Code

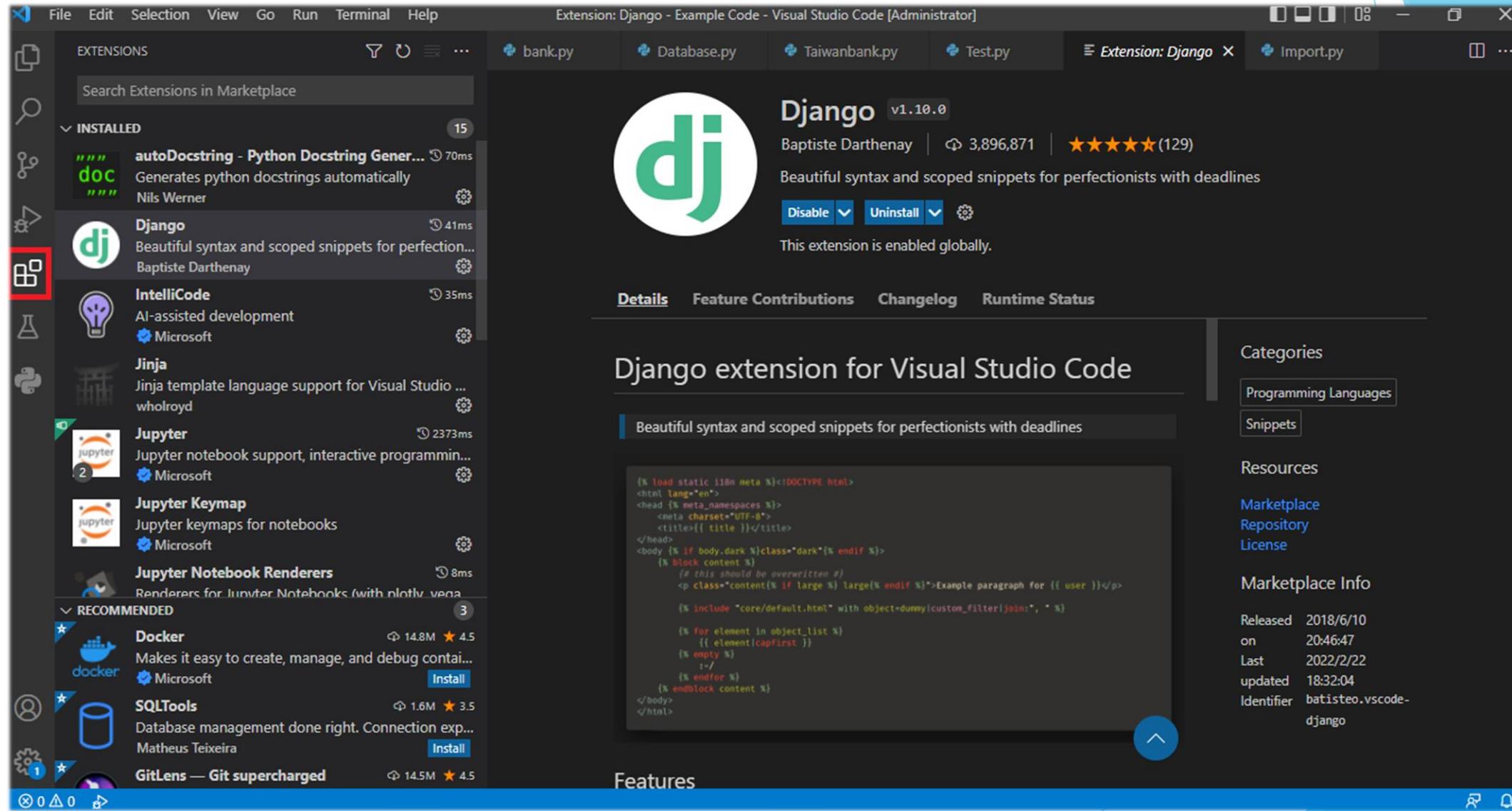


# Visual Studio Code

Download : <https://code.visualstudio.com/>



# Visual Studio Code (Extensions)



# Visual Studio Code

- Explorer
- Code
- Console

The screenshot displays the Visual Studio Code interface with several features highlighted:

- Explorer**: The left sidebar shows the file structure. A red box highlights the "OPEN EDITORS" section, which contains "Training.py".
- Code Editor**: The main area shows a Python script named "Training.py". The code defines a class "baseAnimal" with methods \_\_init\_\_ and animal, and creates an instance X of the class.
- Terminal**: The bottom pane shows a Windows PowerShell window running the script. The output is:

```
PS C:\Users\user\Desktop> & C:/Users/user/AppData/Local/Microsoft/WindowsApps/python3.9.exe c:/Users/user/Desktop/Training.py
LaLa
Cat
Is a animal
PS C:\Users\user\Desktop>
```

# Visual Studio Code

## Debug mode

- By step to execute
- View all of the variables

The screenshot shows the Visual Studio Code interface in debug mode. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The title bar indicates "Training.py - Desktop - Visual Studio Code". The main area displays Python code:

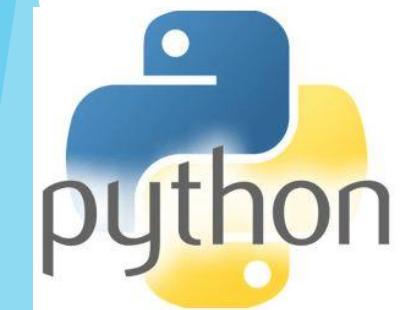
```
File Edit Selection View Go Run Terminal Help
No Configurable Training.py X
Training.py > ...
1 < class baseAnimal:
2
3     def __init__(self, name, species):
4         self.name = name
5         self.species = species
6
7     def animal(self):
8         print("Is a animal")
9
10 X = baseAnimal('LaLa', 'Cat')
11 print(X.name)
12 print(X.species)
13 X.animal()
14
```

The left sidebar features a "VARIABLES" section with a tree view. A red box highlights the "Locals" node, which contains a "X: <\_\_main\_\_.baseAnimal>" entry. This entry is expanded, showing "name: 'LaLa'" and "species: 'Cat'". Below this is a "WATCH" section. The "CALL STACK" section at the bottom shows "PAUSED ON ... <module> Training.py".

The right side of the interface includes a "Python Debug Console" tab with the following output:

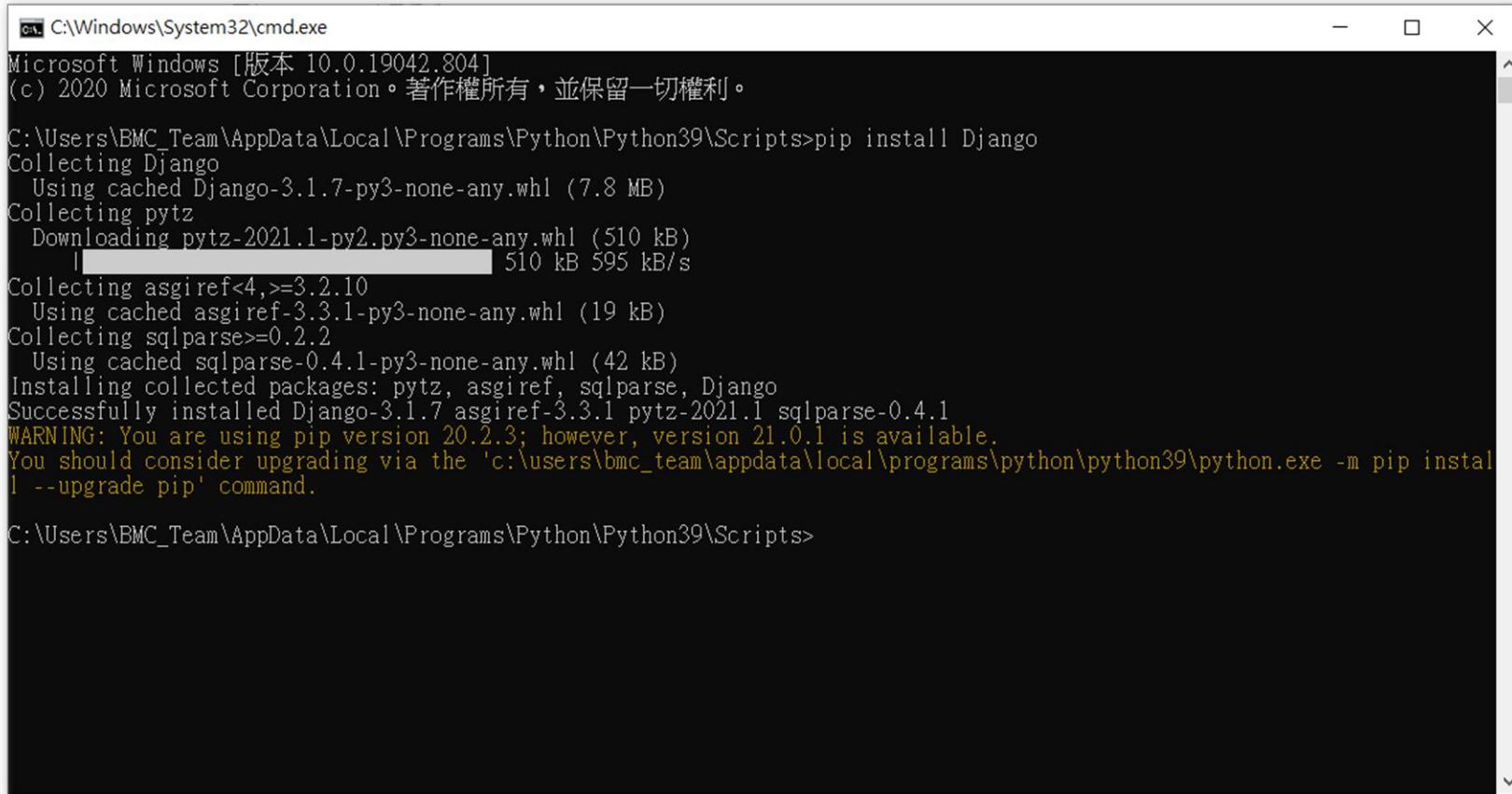
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Windows PowerShell
Copyright (C) Microsoft Corporation. 著作權所有，並保留一切權利。
請嘗試新的跨平台 PowerShell https://aka.ms/pscore6
PS C:\Users\user\Desktop> & 'C:\Users\user\AppData\Local\Microsoft\WindowsApps\python3.9.exe' 'c:\Users\user\.vscode\extensions\ms-python.python-2022.0.18145
23869\pythonFiles\lib\python\debugpy\launcher' '59433' '--' 'c:\Users\user\Desktop\Training.py'
LaLa
```

# Package Install



# Package Install (Python)

\$ pip install \$PackageName

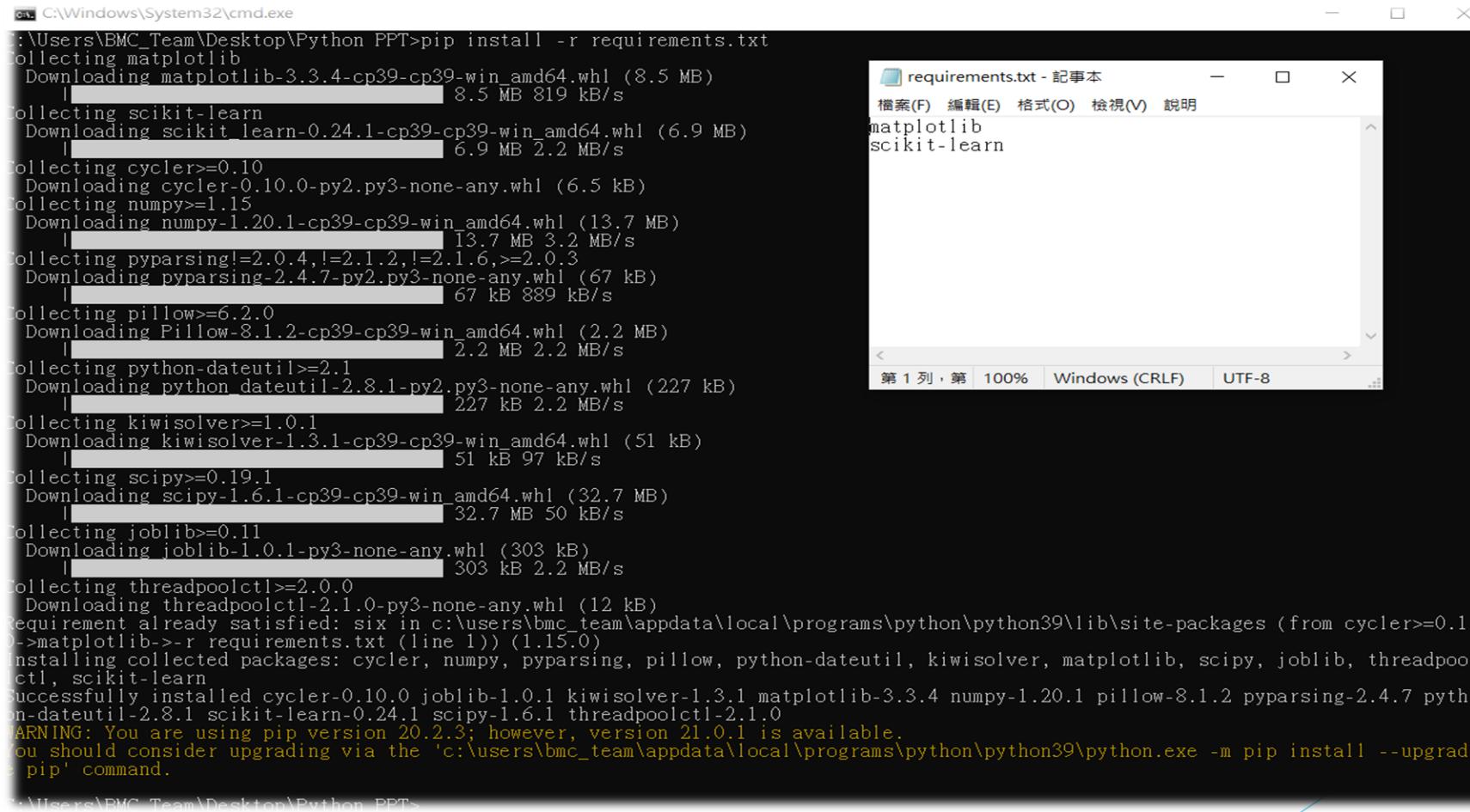


```
C:\Windows\System32\cmd.exe
Microsoft Windows [版本 10.0.19042.804]
(c) 2020 Microsoft Corporation。著作權所有，並保留一切權利。
C:\Users\BMC_Team\AppData\Local\Programs\Python\Python39\Scripts>pip install Django
Collecting Django
  Using cached Django-3.1.7-py3-none-any.whl (7.8 MB)
Collecting pytz
  Downloading pytz-2021.1-py2.py3-none-any.whl (510 kB)
    |██████████| 510 kB 595 kB/s
Collecting asgiref<4,>=3.2.10
  Using cached asgiref-3.3.1-py3-none-any.whl (19 kB)
Collecting sqlparse>=0.2.2
  Using cached sqlparse-0.4.1-py3-none-any.whl (42 kB)
Installing collected packages: pytz, asgiref, sqlparse, Django
Successfully installed Django-3.1.7 asgiref-3.3.1 pytz-2021.1 sqlparse-0.4.1
WARNING: You are using pip version 20.2.3; however, version 21.0.1 is available.
You should consider upgrading via the 'c:\users\bmc_team\appdata\local\programs\python\python39\python.exe -m pip install --upgrade pip' command.

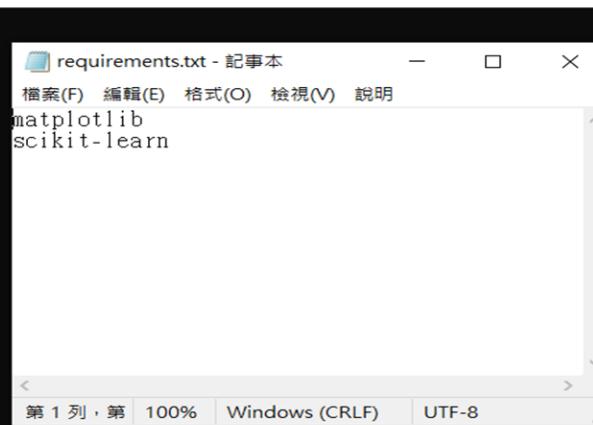
C:\Users\BMC_Team\AppData\Local\Programs\Python\Python39\Scripts>
```

# Package Install (requirements.txt)

\$ pip install -r requirements.txt

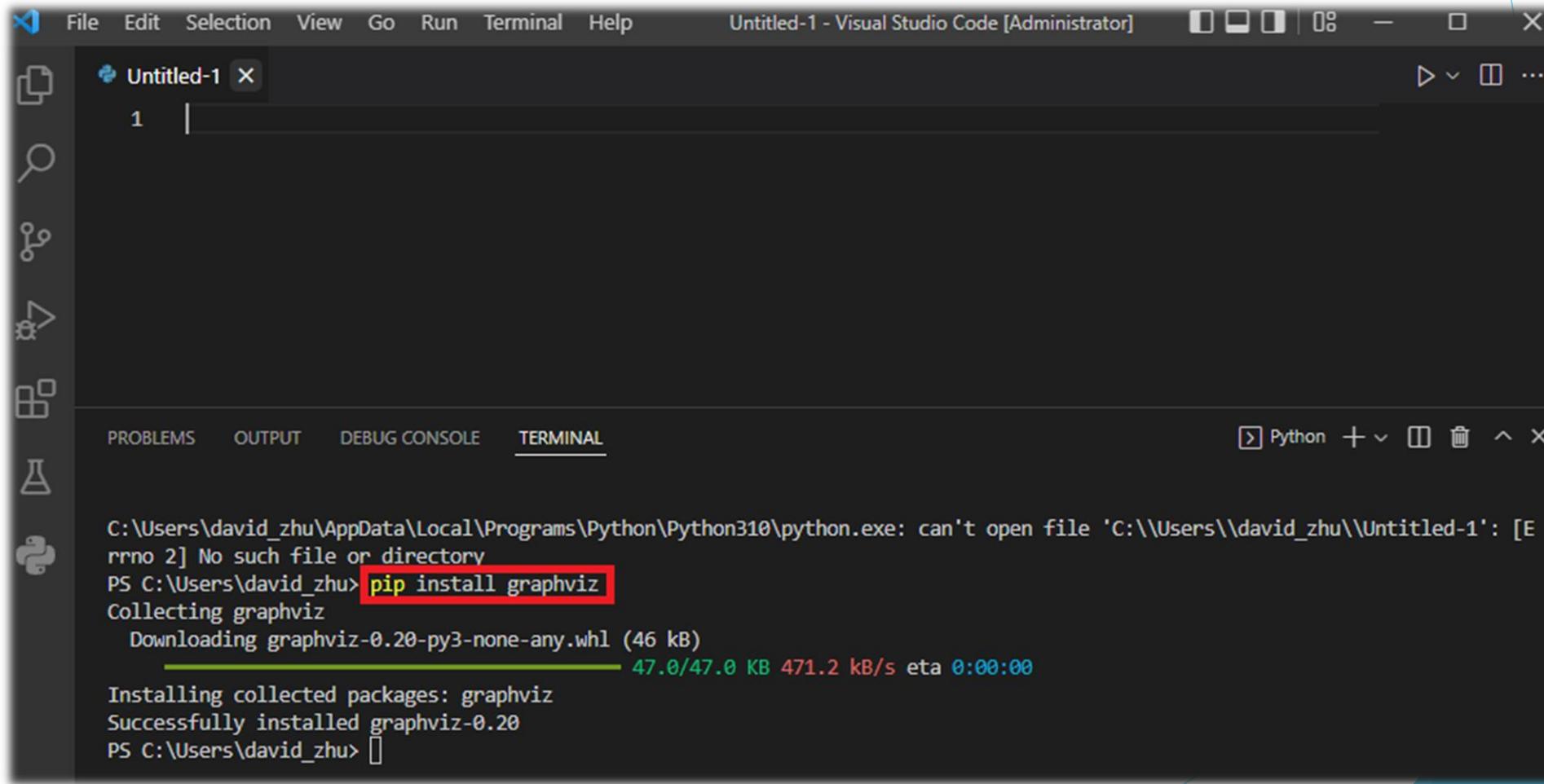


```
C:\Windows\System32\cmd.exe
C:\Users\BMC_Team\Desktop\Python PPT>pip install -r requirements.txt
Collecting matplotlib
  Downloading matplotlib-3.3.4-cp39-cp39-win_amd64.whl (8.5 MB)
    |██████████| 8.5 MB 819 kB/s
Collecting scikit-learn
  Downloading scikit_learn-0.24.1-cp39-cp39-win_amd64.whl (6.9 MB)
    |██████████| 6.9 MB 2.2 MB/s
Collecting cycler<=0.10
  Downloading cycler-0.10.0-py2.py3-none-any.whl (6.5 kB)
Collecting numpy<=1.15
  Downloading numpy-1.20.1-cp39-cp39-win_amd64.whl (13.7 MB)
    |██████████| 13.7 MB 3.2 MB/s
Collecting pyparsing!=2.0.4,!>=2.1.2,!>=2.1.6,>=2.0.3
  Downloading pyparsing-2.4.7-py2.py3-none-any.whl (67 kB)
    |██████████| 67 kB 889 kB/s
Collecting pillow>=6.2.0
  Downloading Pillow-8.1.2-cp39-cp39-win_amd64.whl (2.2 MB)
    |██████████| 2.2 MB 2.2 MB/s
Collecting python-dateutil>=2.1
  Downloading python_dateutil-2.8.1-py2.py3-none-any.whl (227 kB)
    |██████████| 227 kB 2.2 MB/s
Collecting kiwisolver>=1.0.1
  Downloading kiwisolver-1.3.1-cp39-cp39-win_amd64.whl (51 kB)
    |██████████| 51 kB 97 kB/s
Collecting scipy>=0.19.1
  Downloading scipy-1.6.1-cp39-cp39-win_amd64.whl (32.7 MB)
    |██████████| 32.7 MB 50 kB/s
Collecting joblib>=0.11
  Downloading joblib-1.0.1-py3-none-any.whl (303 kB)
    |██████████| 303 kB 2.2 MB/s
Collecting threadpoolctl>=2.0.0
  Downloading threadpoolctl-2.1.0-py3-none-any.whl (12 kB)
Requirement already satisfied: six in c:\users\bmc_team\appdata\local\programs\python\python39\lib\site-packages (from cycler>=0.1
->matplotlib->-r requirements.txt (line 1)) (1.15.0)
Installing collected packages: cycler, numpy, pyparsing, pillow, python-dateutil, kiwisolver, matplotlib, scipy, joblib, threadpool
ctl, scikit-learn
Successfully installed cycler-0.10.0 joblib-1.0.1 kiwisolver-1.3.1 matplotlib-3.3.4 numpy-1.20.1 pillow-8.1.2 pyparsing-2.4.7 pyth
on-dateutil-2.8.1 scikit-learn-0.24.1 scipy-1.6.1 threadpoolctl-2.1.0
WARNING: You are using pip version 20.2.3; however, version 21.0.1 is available.
You should consider upgrading via the 'c:\users\bmc_team\appdata\local\programs\python\python39\python.exe -m pip install --upgrad
e pip' command.
C:\Users\BMC_Team\Desktop\Python PPT>
```



# Package Install (VS code)

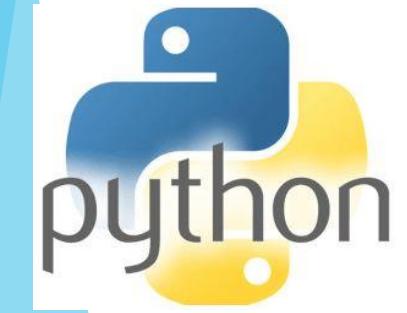
\$ pip install \$PackageName



A screenshot of the Visual Studio Code interface, specifically focusing on the Terminal tab. The terminal window shows the command `pip install graphviz` being run. The output indicates that the package is being downloaded and successfully installed.

```
C:\Users\david_zhu\AppData\Local\Programs\Python\Python310\python.exe: can't open file 'C:\\\\Users\\\\david_zhu\\\\Untitled-1': [Errno 2] No such file or directory
PS C:\\\\Users\\\\david_zhu> pip install graphviz
Collecting graphviz
  Downloading graphviz-0.20-py3-none-any.whl (46 kB)
    47.0/47.0 KB 471.2 kB/s eta 0:00:00
Installing collected packages: graphviz
Successfully installed graphviz-0.20
PS C:\\\\Users\\\\david_zhu>
```

# Common



# Common(Keyword)

False	class	finally	is	Return
None	continue	for	lambda	Try
True	def	from	nonlocal	While
and	del	global	not	With
as	elif	if	or	Yield
assert	else	import	Pass	
break	except	in	Raise	

\* *Don't used to name the variable*

# Common(Built-in functions)

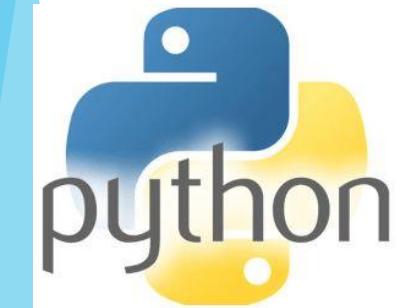
abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	hex()	next()	slice()
ascii()	divmod()	id()	object()	sorted()
bin()	enumerate()	input()	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	<u>__import__()</u>
complex()	hasattr()	max()	round()	

\* *Don't used to name the function*

# Common(Variable name)

1. First word should be *English* or \_.
2. After the first word that can used *English*, *numeric* or \_ .

# Data Type



# Data Type

## 1. Numeric type

- Integer (int)
- Floating-point (float)
- Boolean (bool)

## 2. Text type

- String (str)

## 3. Sequence type

- List[]
- Tuple()
- Dict{}
- Set{}

# Data Type (Integer)

```
x = 1          # Assign 1 to x
print(x)       # print x
print(type(x)) # print x type
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_
thon.exe "c:/Users/david_zhu/Desktop/Python_Introduction-main/
1
<class 'int'>
```

# Data Type (Float)

```
x = 1.5          # Assign 1.5 to x
print(x)         # print x
print(type(x))  # print x type
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main\Example
thon.exe "c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/
1.5
<class 'float'>
```

# Data Type(Arithmetic operator)

+	加	$a+b$
-	減	$a-b$
*	乘	$a*b$
/	除	$a/b$
%	取餘數	$a \% b$

# Data Type(Arithmetic operator)

```
a = 6                                # Assign 6 to a  
b = 3                                # Assign 3 to b  
  
print("a + b = ", a + b)      # a + b  
print("a - b = ", a - b)      # a - b  
print("a * b = ", a * b)      # a * b  
print("a / b = ", a / b)      # a / b  
print("a mod b = ", a % b)     # a mod b
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main> python3.10\python.exe "c:/Users/david_zhu/AppData/Local/Programs/Python/Python310/python.exe" "c:/Users/david_zhu/Desktop\Python_Introduction-main\Example Code\Test.py"  
a + b =  9  
a - b =  3  
a * b =  18  
a / b =  2.0  
a mod b =  0
```

# Data Type(Comparison operator)

<	小於	$a < b$
>	大於	$a > b$
$\leq$	小於等於	$a \leq b$
$\geq$	大於等於	$a \geq b$
$=$	相等	$a == b$
$\neq$	不相等	$a != b$

\* *It will return the Boolean type*

# Data Type(Comparison operator)

```
a = 6                                # Assign 6 to a  
b = 3                                # Assign 3 to b  
  
print("a < b ", a < b)                # a < b ?  
print("a > b ", a > b)                # a > b ?  
print("a <= b ", a <= b)              # a <= b ?  
print("a >= b ", a >= b)              # a >= b ?  
print("a = b ", a == b)                # a = b ?  
print("a != b ", a != b)               # a != b ?
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-zhu\AppData\Local\Programs\Python\Python310\python.exe "c:/Users/david_zhu/hon_Introduction-main/Example Code/Test.py"  
a < b    False  
a > b    True  
a <= b   False  
a >= b   True  
a = b    False  
a != b   True
```

# Data Type (Str)

```
x = "string"          # Assign "string" to x
print(x)              # print x
print(type(x))        # print x type
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main\Example
thon.exe "c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/
String
<class 'str'>
```

# Data Type (bool)

```
x = True          # Assign “True” to bool x  
y = False         # Assign “False” to bool y  
  
print(x)          # print x  
print(type(x))    # print x type  
print(y)          # print x  
print(type(y))    # print x type
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main\Example  
thon.exe "c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/  
True  
<class 'bool'>  
False  
<class 'bool'>
```

# Data Type (Float transfer to Integer)

```
x = 10.5          # float type  
print(x)         # print x  
print(type(x))  # print x type
```

```
x = int(x)       # float transfer to integer  
print(x)         # print x  
print(type(x))  # print x type
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main\Exampl  
thon.exe "c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main  
10.5  
<class 'float'>  
10  
<class 'int'>
```

# Data Type (Integer transfer to String)

```
x = 10          # Integer type  
print(x)        # print x  
print(type(x)) # print x type
```

```
x = str(x)      # Integer transfer to string  
print(x)        # print x  
print(type(x)) # print x type
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main\Example  
thon.exe "c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/  
10  
<class 'int'>  
10  
<class 'str'>
```

# Data Type (String transfer to Integer)

```
x = "10"          # String type of the Integer  
print(x)          # print x  
print(type(x))    # print x type
```

```
x = int(x)        # String transfer to Integer  
print(x)          # print x  
print(type(x))    # print x type
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main\Example  
thon.exe "c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/  
10  
<class 'str'>  
10  
<class 'int'>
```

# Data Type (String transfer to Integer)

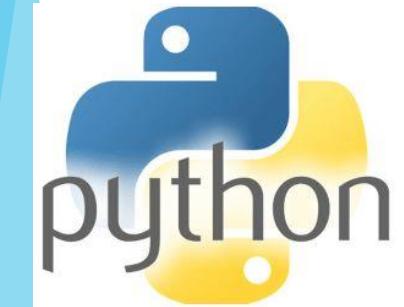
```
x = "string"          # String type  
print(x)              # print x  
print(type(x))        # print x type
```

```
x = int(x)            # String transfer to Integer  
print(x)              # print x  
print(type(x))        # print x type
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main\Example Code> & C:/  
thon.exe "c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/Example Code/  
string  
<class 'str'>  
Traceback (most recent call last):  
  File "c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/Example Code/T  
    x = int(x)          # String transfer to Integer  
ValueError: invalid literal for int() with base 10: 'string'
```

\* Only String type of the Integer can be transfer

# Data Structure



# Data Structure (Tuple)

```
x = 1 , "string" , False    #Assign 1, “string”, False to bool x  
y = ( 1 , "string" , False) #Assign (1, “string”, False) to tuple y
```

```
print(x)                      #print x  
print(type(x))                #print x type
```

```
print(y)                      #print y  
print(type(y))                #print y type
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main\Example  
thon.exe "c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/B  
(1, 'string', False)  
<class 'tuple'>  
(1, 'string', False)  
<class 'tuple'>
```

# Data Structure (Tuple)

```
x = ( 1 , "string" , False ) #Assign (1, “string”, False) to tuple x  
x[0] = 2 #Assign 2 to x[0]
```

```
print(x) #print x  
print(type(x)) #print x type
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main\Example Code> & C:/Users/david_zhu/AppData/Local  
thon.exe "c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/Example Code/Test.py"  
Traceback (most recent call last):  
  File "c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/Example Code/Test.py", line 3, in <module>  
    x[0] = 2 #Assign 2 to x[0]  
TypeError: 'tuple' object does not support item assignment
```

\* *Tuple object doesn't support item assignment*

# Data Structure (List)

```
x = [ 1 , "string" , False] #Assign [1, “string”, False] to list x  
print(x)                      #print x  
print(type(x))                #print x type
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main\Example  
thon.exe "c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/  
[1, 'string', False]  
<class 'list'>
```

# Data Structure (List)

```
x = [ 1 , "string" , False] #Assign (1, “string”, False) to tuple x  
print(x) #print(x)  
x[0] = 2 #Assign 2 to x[0]  
print(x) #print x  
print(type(x)) #print x type
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main\Example  
thon.exe "c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/l  
[1, 'string', False]  
[2, 'string', False]  
<class 'list'>
```

\* *List object support item assignment*

# Data Structure (list.append)

```
x = [ 1 , "string", False] #Assign [1, “string”,False] to list x  
print(x) #print x  
print(type(x)) #print x type  
  
x.append("Python") #add “python” to list x  
print(x) #print x
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main\Example  
thon.exe "c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/  
[1, 'string', False]  
<class 'list'>  
[1, 'string', False, 'Python']
```

\* *append() function add to end of the List*

# Data Structure (list.insert)

```
x = [ 1 , "string" , False] #Assign [ 1 , "string" , False] to list x  
print(x) #print x  
print(type(x)) #print x type  
  
x.insert(0, "python") #insert “python” to list x[0]  
print(x) #print x
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main\Example  
thon.exe "c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/  
[1, 'string', False]  
<class 'list'>  
['python', 1, 'string', False]
```

# Data Structure (list.remove)

```
x = [ 1 , "string" , False] #Assign [ 1 , "string" , False] to list x  
print(x) #print x  
print(type(x)) #print x type
```

```
x.remove("string") #remove from list x  
print(x) #print x
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main\Example  
thon.exe "c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/  
[1, 'string', False]  
<class 'list'>  
[1, False]
```

# Data Structure (Two-dimensional List)

```
list1 = [[1,2,3],           # Assign 1~9 to two-dimensional list
          [4,5,6],
          [7,8,9]]  
  
print(list1[1][0])        # Print list[1][0]
print(list1[1][1])        # Print list[1][1]
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main\Example
thon.exe "c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/B
4
5
```

# Data Structure (Set)

```
set1 = {"A", "B", "C", 1, 2}  
set2 = {"A", "B", "D", 1, 3}  
  
print("A" in set1)          # "A" in set1 or not  
  
print(set1 | set2)         # Union (聯集)  
  
print(set1 & set2)         # Intersection (交集)  
  
print(set1 - set2)         # Subtraction (差集)  
  
print(set1 ^ set2)         # XOR (互斥或)
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main\Example  
thon.exe "c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/  
True  
{'A', 'B', 2, 1, 3, 'D', 'C'}  
{'A', 'B', 1}  
{'C', 2}  
{'C', 3, 2, 'D'}
```

# Data Structure (Dict)

```
dict = {"Name" : "David", "Age" : 28, "Weight" : 70, "Height" : 176}
```

```
print(dict["Name"])
```

```
print(dict["Age"])
```

```
print(dict["Weight"])
```

```
print(dict["Height"])
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main\Example
thon.exe "c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/
David
28
70
176
```

# Data Structure (dict.clean())

```
dict = {"Name" : "David", "Age" : 28, "Weight" : 70, "Height" : 176}
```

```
print(dict)
```

```
dict.clear()
```

```
print(dict)
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main\Example
thon.exe "c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/
{'Name': 'David', 'Age': 28, 'Weight': 70, 'Height': 176}
{}
```

# Data Structure (dict.copy())

```
dict = {"Name" : "David", "Age" : 28, "Weight" : 70, "Height" : 176}
```

```
dict2 = dict.copy()
```

```
print("dict : ", dict)
```

```
print("dict2 : ", dict2)
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main\Example
thon.exe "c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/
dict : {'Name': 'David', 'Age': 28, 'Weight': 70, 'Height': 176}
dict2 : {'Name': 'David', 'Age': 28, 'Weight': 70, 'Height': 176}
```

# Data Structure (dict.get())

```
dict = {"Name" : "David", "Age" : 28, "Weight" : 70, "Height" : 176}
```

```
print(dict.get("Name"))
```

```
print(dict.get("Age"))
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main\Example
thon.exe "c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/
David
28
```

# Data Structure (dict compare)

```
dict = {"Name" : "David", "Age" : 28, "Weight" : 70, "Height" : 176}
```

```
dict2 = {"Name" : "Ricky", "Age" : 28, "Weight" : 70, "Height" : 176}
```

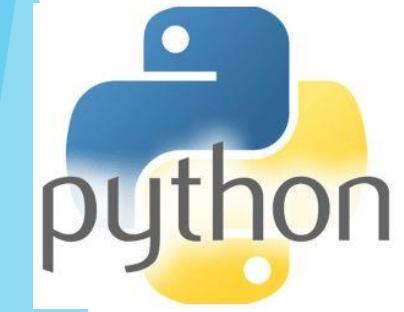
```
print("Compare Result", dict == dict2)
```

```
dict2["Name"] = "David"
```

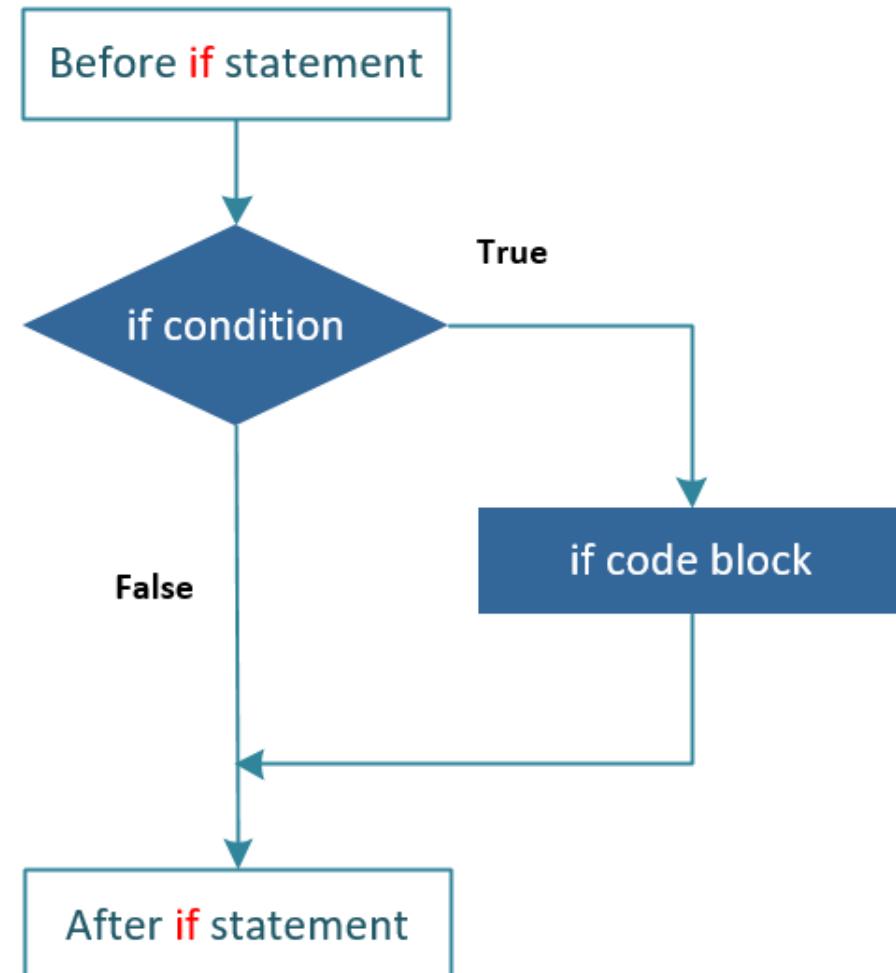
```
print("Compare Result", dict == dict2)
```

```
PS C:\Users\david_zhu\Desktop\test> & C:/Users/david_zhu/AppData/Local/Programs/n310/python.exe c:/Users/david_zhu/Desktop/test/test.py
Compare Result False
Compare Result True
```

# Flow control



# Flow control (if\_else)



# Flow control (if\_else)

```
x = int(input("Please enter an integer: "))    # Assign input number to x

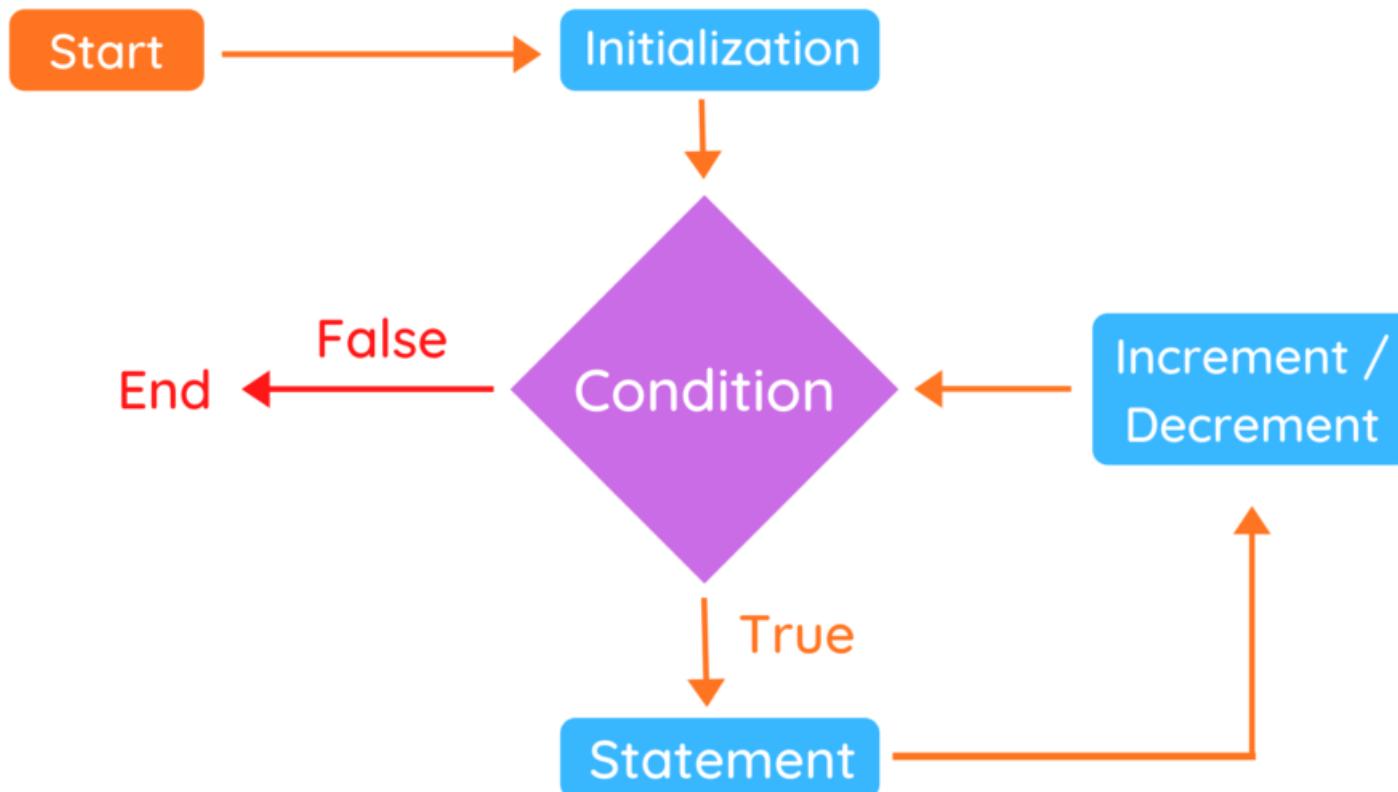
if x < 50:
    print("The Integer is small than 50")      # If x < 50, print

elif x > 50:
    print("The Integer is more than 50")        # If x > 50, print

else:
    print("The Integer is 50")                  # If not meets above, print
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main\Example
thon.exe "c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/
Please enter an integer: 60
The Integer is more than 50
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main\Example
thon.exe "c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/
Please enter an integer: 40
The Integer is small than 50
```

# Flow control (for\_loop)



# Flow control (for\_loop)

```
words = ["Python", "Java", "C++"]      # Assign List[] to words  
  
for w in words:  
    print(w, len(w))                  # Print w in words[] and count it.
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main>  
"c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/Ex  
Python 6  
Java 4  
C++ 3
```

# Flow control (for\_range)

```
for i in range(10):  
    print(i)                      # Print i from 0 to 9
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main\Example Code>  
"c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/Example Code/T  
0  
1  
2  
3  
4  
5  
6  
7  
8  
9
```

# Flow control (nested\_loops)

```
words = ["A", "B", "C", "D"]      #Assign “A”, “B”, “C”, “D” to words

for i in range(5):
    for j in words:
        print(j)                  #Print j
```

```
"c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/Example Code/Test.py"
```

```
A  
B  
C  
D  
A  
B  
C  
D  
A  
B  
C  
D  
A  
B  
C  
D  
A  
B  
C  
D
```

# Flow control (break)

```
for i in range(5):
    print(i)                      #Print I
    if i == 3:                     #if i = 3
        print("Loop Break") #print “Loop Break” then break loop
        break
```

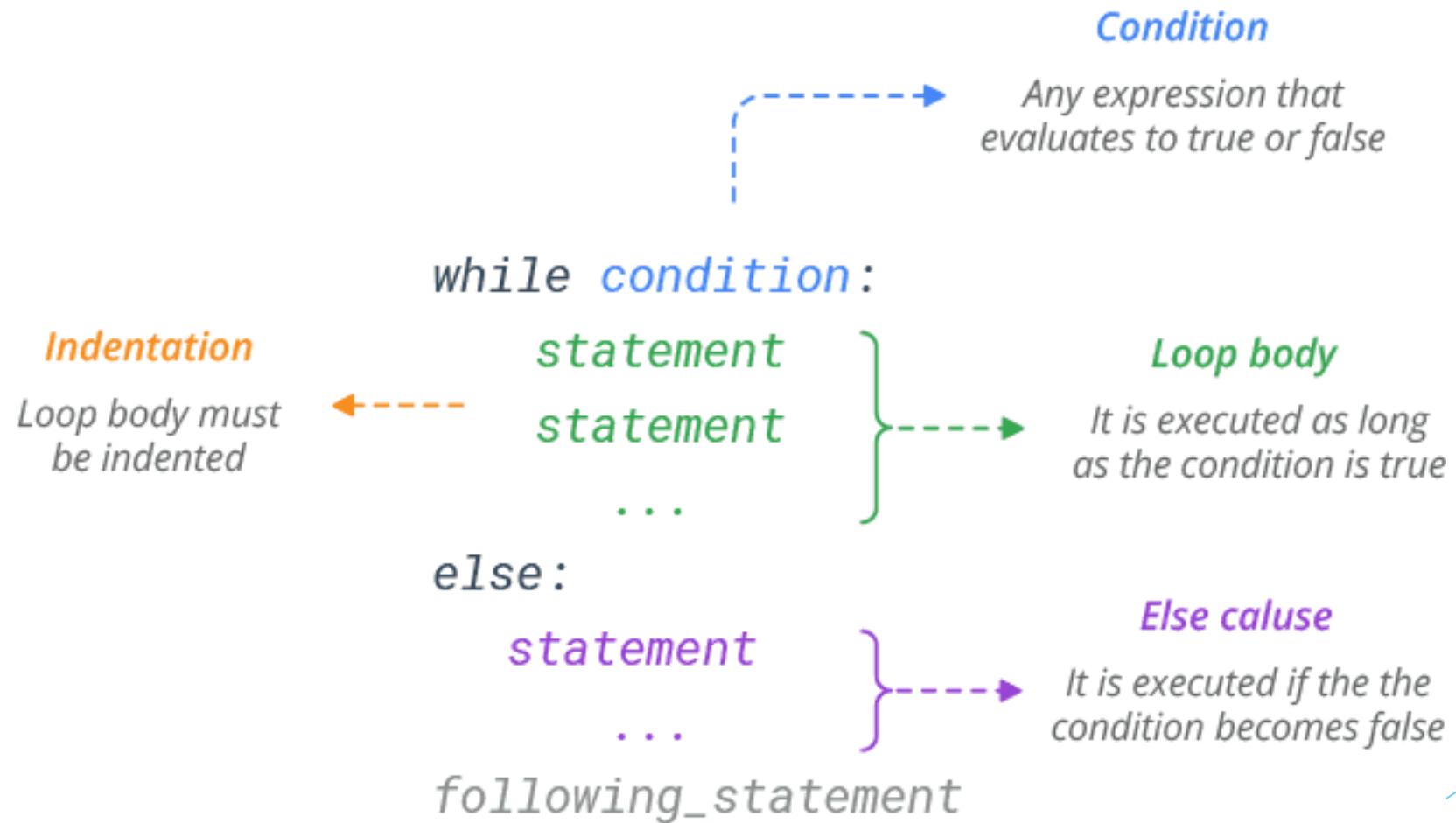
```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main\Example Code>
"c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/Example Code/"
0
1
2
3
Loop Break
```

# Flow control (continue)

```
for i in range(5):
    if i == 3:          #if i = 3
        continue        #Bypass then continue loop
    print(i)            #Print i
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main\Example Code> "c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/Example Code/">
0
1
2
4
```

# Flow control (while\_loop)



# Flow control (while\_loop)

```
i = 0                                # assign 0 to i
while i < 10:                          # if i < 10 is true
    print(i)                            # print i
    i += 1                             # i + 1
else:
    print("finished")                  # if i < 10 is false
```

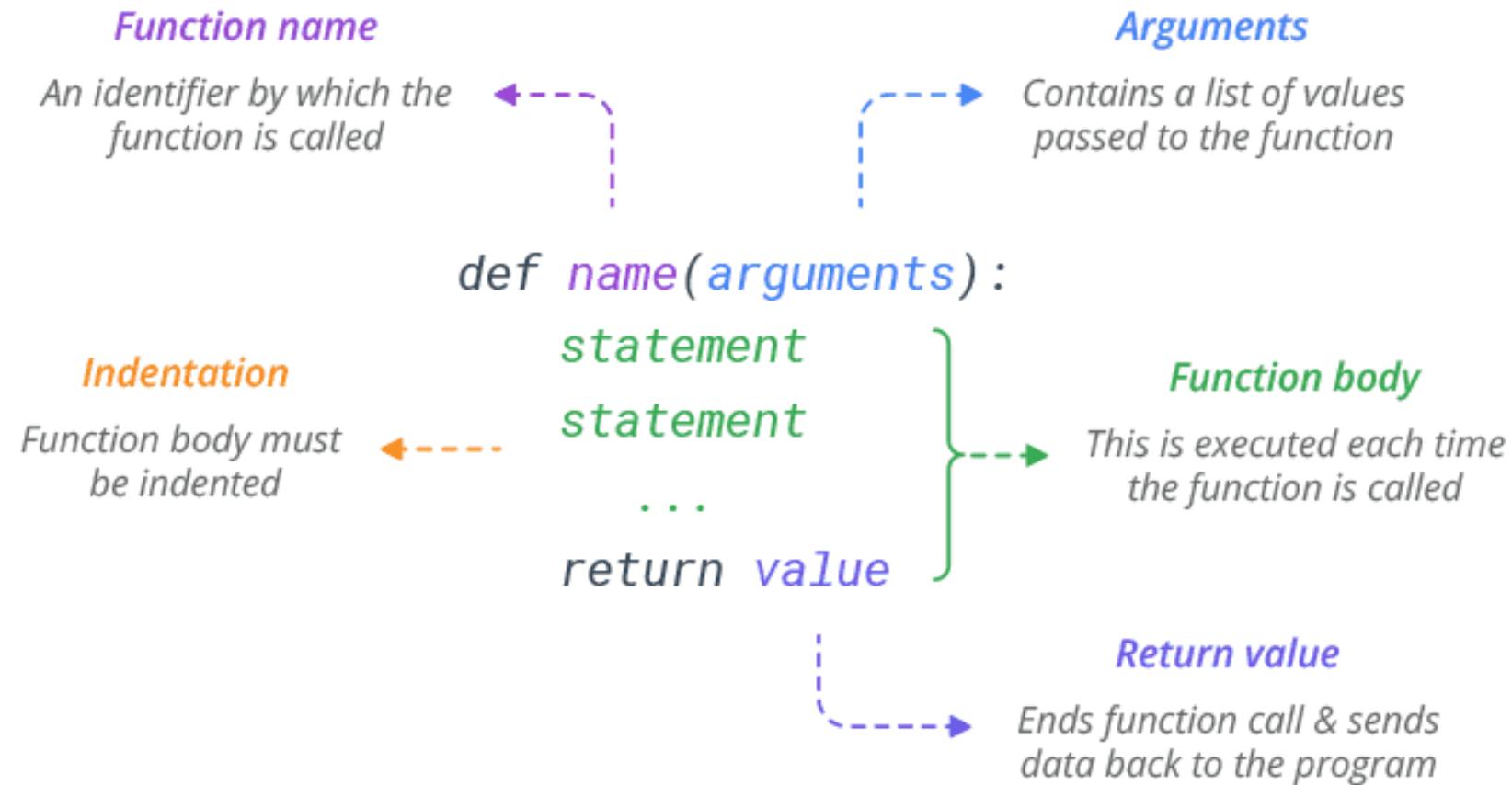
```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main\Example Code>
"c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/Example Code/>
0
1
2
3
4
5
6
7
8
9
finished
```

# Flow control (infinite\_loop)

```
i = 0                                # assign 0 to i
while True:                            # infinite_loop
    print(i)                            # print i
    i += 1                             # i + 1
else:
    print("finished")                 # if i < 10 is false
```

```
9738
9739
9740
9741
9742
9743
9744
9745
9746
9747
9748
9749
9750
9751
9752
9753
9754
9755
9756
9757
9758
9759
```

# Flow control (function)



# Flow control (function)

```
def example(n):      # function example()
    n+=5            # n = n + 5
    print(n)         # print n

example(10)         # call function example()
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main\Example Code>
"c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/Example Code/T
15
```

# Flow control (return)

```
def example(n):      # function example()
    n+=5            # n = n + 5
    return n         # return n to this function

x = example(10)     # assign to x
print(x)
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main\Example Code>
"c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/Example Code/T
15
```

# Flow control (return & print)

```
def example(n):      # function example()
    n+=5            # n = n + 5
    x = 100
    print(n)         # print n on console only
    return x          # return x to this function

result = example(10) # assign to result
print(result + 100)
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main>
"c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/"
15
200
```

- *Return is return the result to function*
- *Print is print the result on console only*

# Flow control (global/local variable)

```
gobal_x = 1                                # Assign 1 to gobal_x
def example():                               # Function example()
    local_x = 2                            # Assign 2 to local_x
    local_y = 6                            # Assign 6 to local_y
    print("global x:",gobal_x)            # print global_x
    print("local x:",local_x)             # print local_x
    print("local y:", local_y)            # print local_y

example()                                    # call function example()
print("local y:", local_y)                  # print local_y
```

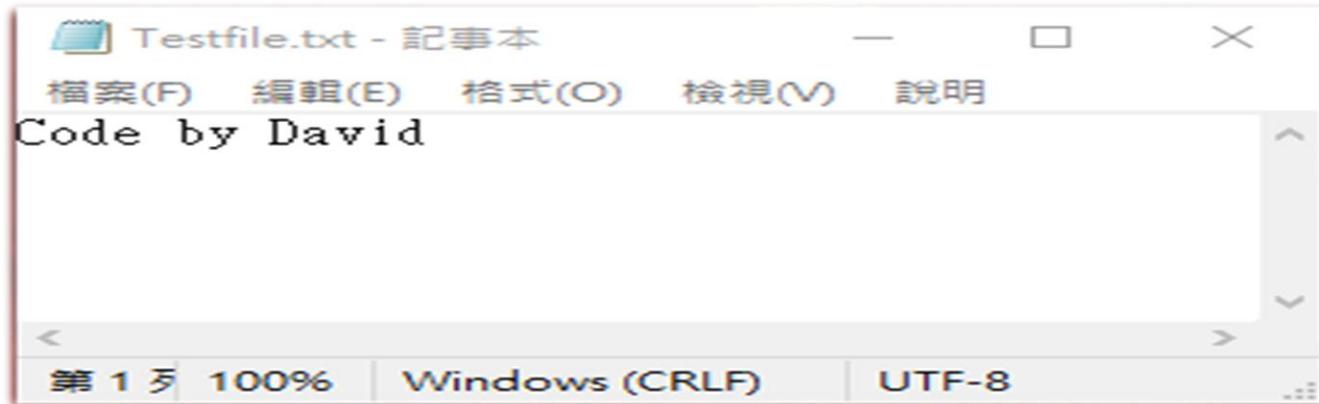
```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main
      "c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/
global x: 1
local x: 2
local y: 6
Traceback (most recent call last):
  File "c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction
      print("local y:", local_y)          # print local_y
NameError: name 'local_y' is not defined. Did you mean: 'locals'?
```

# Flow control (file write)

```
file = open("Testfile.txt","w")      # Open file
```

```
file.write("Code by David")        # Write file
```

```
file.close                         # stream close
```



# Flow control (file read)

```
file = open("Testfile.txt", "r")      # Open file  
  
print(file.read())                  # Read file  
  
file.close                         # stream close
```

```
PS C:\Users\david_zhu\Desktop\AMI DevNet Test Case Transfer to Excel> & C:\Python37\python.exe d:\Users\david_zhu\Desktop\AMI DevNet Test Case Transfer to Excel\FileRW.py"  
Code by David
```

# Flow control (with as)

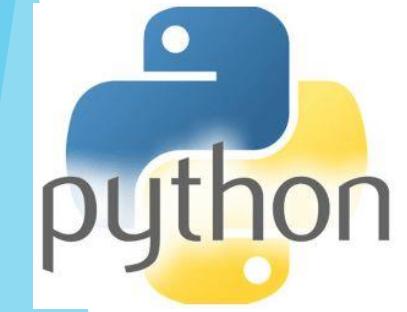
```
txtfile= "Code by David.\n Hello ASUS"
```

```
with open("Testfile.txt","w") as file:      # Open file and stream close  
    file.write(txtfile)                      # Write file
```

```
with open("Testfile.txt","r") as file:  
    print(file.read())                      # Read file
```

```
PS C:\Users\david_zhu\Desktop\AMI DevNet Test Case Transfer to Excel> &  
rs/david_zhu/Desktop/AMI DevNet Test Case Transfer to Excel/FileRW.py"  
Code by David.  
Hello ASUS
```

# Module



# Module(Import)

#util.py

```
def example1():
    print("example1 function")

def example2():
    print("example2 function")
```

#Demo.py

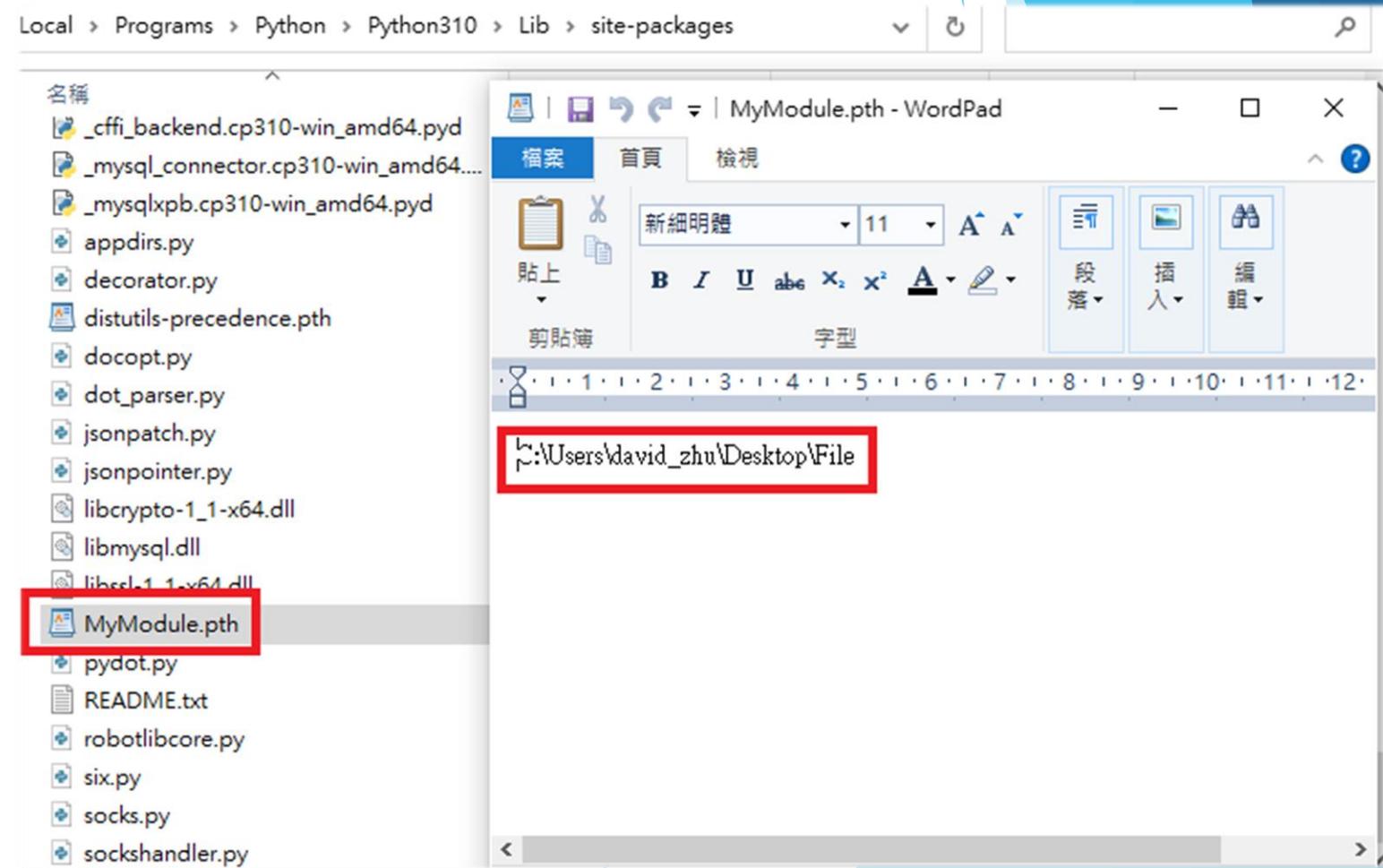
```
Import util
util.example1()
util.example2()
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main\Example
thon.exe "c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/
example1 function
example2 function
```

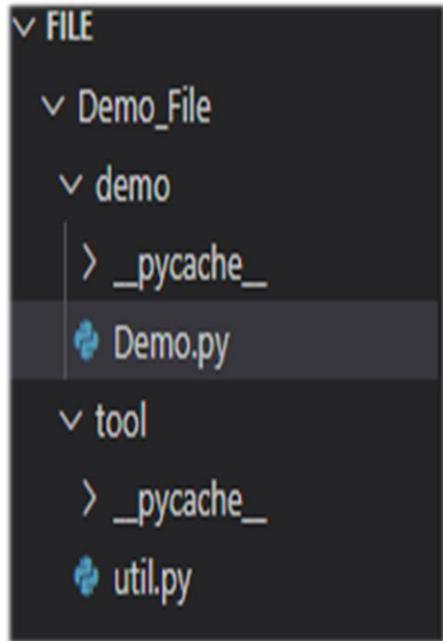
# Module (Different Folder Import)

Add Project package path as below  
with .pth file for python to identify  
your package module

\**AppData\Local\Programs\Python\Python310\*  
*Lib\site-packages*



# Module(Different Folder Import)



```
#util.py
```

```
def example1():
    print("example1 function")

def example2():
    print("example2 function")
```

```
#Demo.py
```

```
from Demo_File.tool import util
util.example1()
util.example2()
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main\Example
thon.exe "c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/
example1 function
example2 function
```

# Module (if `__name__ == "__main__"`:

```
#Test.py
```

```
def example1():
    print("example1 function")
```

```
def example2():
    print("example2 function")
```

```
if __name__ == "__main__":
    print("Under 'Test.py' module")
```

```
#import.py
```

```
import Test
```

```
Test.example1()
```

```
Test.example2()
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main\Example
thon.exe "c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/
example1 function
example2 function
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main\Example
thon.exe "c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/
Under 'Test.py' module
```

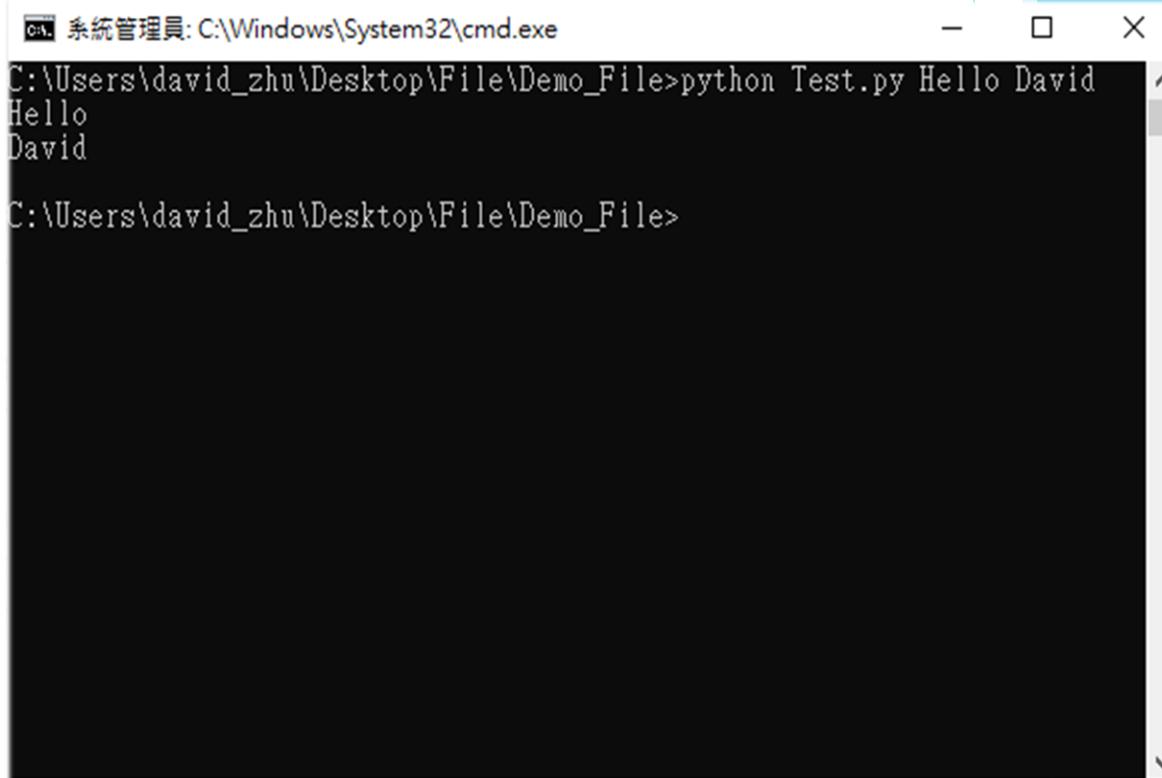
# Module (Library modules)

```
#temp.py
import sys

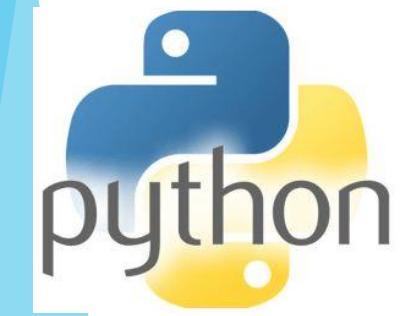
def example1(n):
    print(n)

def example2(m):
    print(m)

if __name__ == "__main__":
    example1(str(sys.argv[1]))
    example2(str(sys.argv[2]))
```



# Exception



# Exception (Error Type)

# Exception (*checking*)

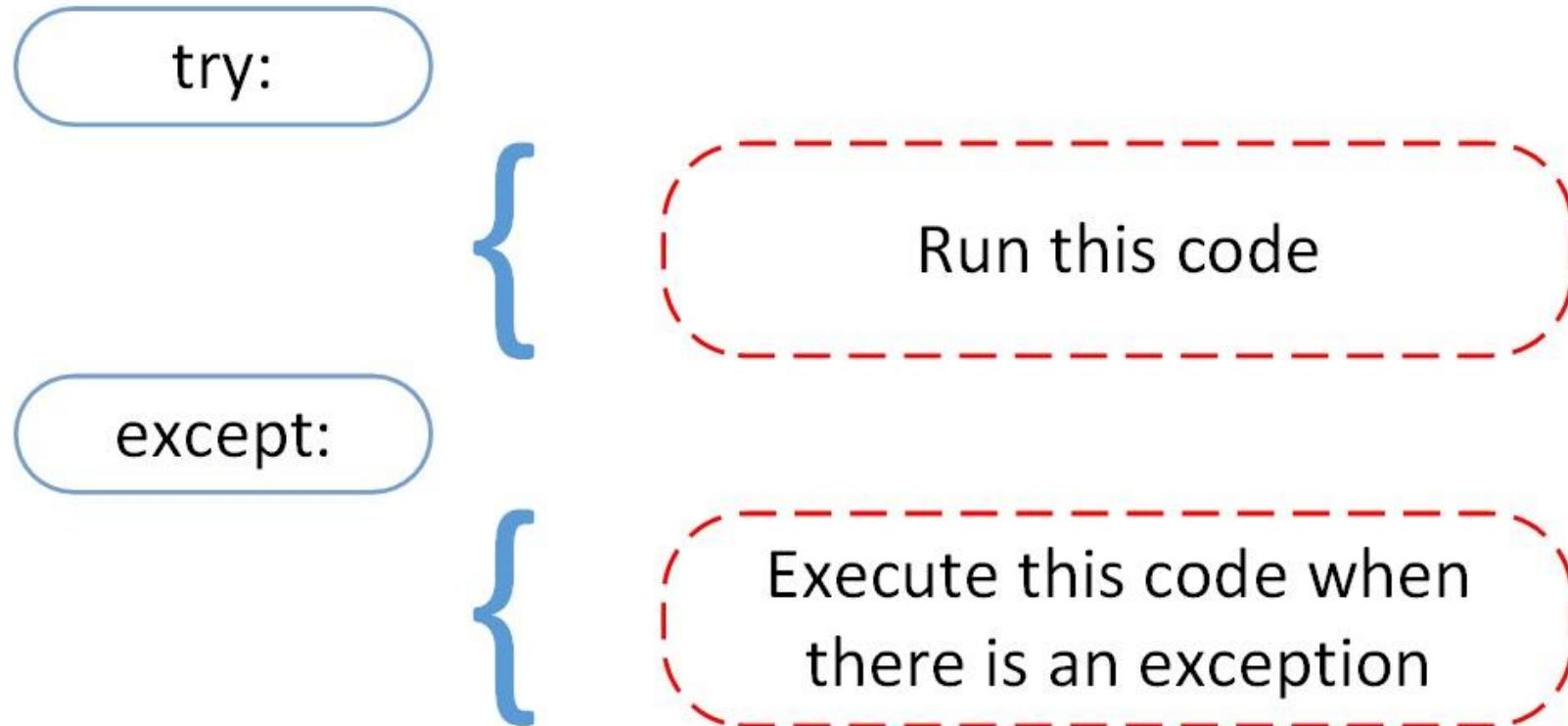
# EX : ValueError, NameError, TypeError

# Runtime Exception (*unchecking*)

# EX : OS crash, HW component error .....

# Can't be catch by python.

# Exception (Error Type)



# Exception (Error)

```
# Exception error
```

```
Number = int(input( "Please enter a number : "))
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main\Example> python.exe "c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/Exception.py"
Please enter a number : string
Traceback (most recent call last):
  File "c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/Exception.py", line 1, in <module>
    Number = int(input( "Please enter a number : "))
ValueError: invalid literal for int() with base 10: 'string'
```

# Exception (try\_catch)

```
# Exception error

try:
    Number = int(input("Please enter a number: "))

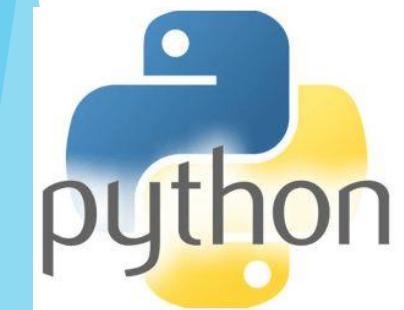
except ValueError:
    print("Oops! That was no valid number. Try again...")

else:
    print("Input successfully")

finally:
    print("Code by David")
```

```
Please enter a number: string
Oops! That was no valid number. Try again...
Code by David
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main\Example
thon.exe "c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/
Please enter a number: 5
Input successfully
Code by David
```

# Object-oriented Programming



# Object-oriented Programming

1. **Encapsulation 封裝**：一種將函式介面的實作細節部份包裝、隱藏起來的方法，它也是一種**防止外界呼叫端，去存取物件內部實作細節的手段**。
2. **Inheritance 繼承**：繼承可以使得**子類具有父類別別的各種屬性和方法**，而不需要再次編寫相同的代碼。
3. **Polymorphism 多型**：指為不同資料類型的實體提供統一的介面，**透過不同的實例化物件來呼叫相同的方法可以達到不同的實作方式**。

# Class (類別)

```
# Class sample  
  
class baseClassName:  
    <statement-1>  
    .  
    .  
    .  
    <statement-N>
```

# Class (類別)

```
# Animal Class
```

```
class baseAnimal:          # baseAnimal class
    string = "Is not a animal"
```

```
def animal(self):
    print("Is a animal")
```

```
X = baseAnimal()          # 實例化
print(X.string)            # Call string
X.animal()                 # Call animal method
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main\Example
thon.exe "c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/
Is not a animal
Is a animal
```

# Class (匿名物件)

```
# Animal Class
```

```
class baseAnimal:          # baseAnimal class
    string = "Is not a animal"

    def animal(self):
        print("Is a animal")

print(baseAnimal().string)  # Call string
baseAnimal().animal()      # Call animal method
```

#以匿名物件的方式省略實例化的動作來使用物件內的方法

```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main\Example
thon.exe "c:/Users/david_zhu/Desktop/Python_Introduction-main/Python_Introduction-main/
Is not a animal
Is a animal
```

# Class ( \_\_init\_\_ 建構子)

```
# Animal Class
```

```
class baseAnimal:
```

```
    def __init__(self, name, species):  
        self.name = name  
        self.species = species
```

```
    def animal(self):  
        print("Is a animal")
```

```
X = baseAnimal('LaLa', 'Cat')  
print(X.name)  
print(X.species)  
X.animal()
```

# \_\_init\_\_ 作為設定物件初始化資料使用

# \_\_init\_\_ 附帶宣告的變數會成為實例化時需要的參數

```
PS C:\Users\david_zhu\Desktop\Python_Intro  
thon.exe "c:/Users/david_zhu/Desktop/Python  
LaLa  
Cat  
Is a animal
```

# Encapsulation (封裝)

```
# Animal Class

class baseAnimal:

    def __init__(self, name, species):
        self.__name = name
        self.__species = species

    def animal(self):
        print("Is a animal")

X = baseAnimal('LaLa', 'Cat')
print(X.__name)
print(X.__species)
X.animal()
```

#透過封裝防止內部資料直接被外部呼叫

```
Traceback (most recent call last):
  File "c:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Intro\encapsulation.py", line 1, in <module>
    X.__name
AttributeError: 'baseAnimal' object has no attribute '__name'
```

# Encapsulation (封裝)

```
# Animal Class

class baseAnimal:

    def __init__(self, name, species):
        self.__name = name
        self.__species = species

    def getAnimalName(getname):
        return getname.__name

    def getAnimalSpecies(getspecies):
        return getspecies.__species

X = baseAnimal("LaLa","Cat")
print(X.getAnimalName())
print(X.getAnimalSpecies())
```

#透過 Method 回傳被封裝的資料

```
PS C:\Users\david_zhu\Desktop\Python_I
thon.exe "c:/Users/david_zhu/Desktop/P
LaLa
Cat
```

# Inheritance (繼承)

# Inheritance

```
class DerivedClassName(BaseClassName):  
    <statement-1>  
    .  
    .  
    .  
    <statement-N>
```

# Inheritance (繼承)

```
# Animal Class

class baseAnimal:

    def __init__(self, name, species):
        self.__name = name
        self.__species = species

    def getAnimalName(self):
        return self.__name

    def getAnimalSpecies(self):
        return self.__species

class cat(baseAnimal):
    def update(self):
        return "喵喵叫"

X = cat("LaLa","Cat")
print(X.getAnimalName())  #透過繼承取得父類別 Method
print(X.getAnimalSpecies())
print(X.update())          #依然可使用自己類別的 Method
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction.exe "c:/Users/david_zhu/Desktop/Python_LaLa
Cat
喵喵叫"
```

# Inheritance (多重繼承)

# Animal Class

```
class baseAnimal:  
    def __init__(self, name):  
        self.__name = name  
    def getAnimalName(self):  
        return self.__name  
  
class baseAnimal2:  
    def __init__(self, species):  
        self.__species = species  
    def getAnimalSpecies(self):  
        return self.__species  
  
class cat(baseAnimal, baseAnimal2):  
    def __init__(self, name, specie):  
        baseAnimal.__init__(self, name)  
        baseAnimal2.__init__(self, specie)  
    def update(self):  
        return "喵喵叫"  
  
X = cat("LaLa", "Cat")  
print(X.getAnimalName()) # 透過多重繼承取得兩個父類別 Method  
print(X.getAnimalSpecies())  
print(X.update()) # 多重繼承後依然可使用自己類別的 Method
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction.exe "c:/Users/david_zhu/Desktop/Python_LaLa  
Cat  
喵喵叫"
```

# Inheritance (覆寫)

```
# Animal Class
```

```
class baseAnimal:
```

```
    def __init__(self, name, species):
```

```
        self.__name = name
```

```
        self.__species = species
```

```
    def getAnimalName(self):
```

```
        return self.__name
```

```
    def getAnimalSpecies(self):
```

```
        return self.__species
```

```
class cat(baseAnimal):
```

```
    def update(self):
```

```
        return "喵喵叫"
```

```
    def getAnimalSpecies(self): #以相同名稱的 Method 覆寫父類別
```

```
        return "我是貓咪"
```

```
X = cat("LaLa","Cat")
```

```
print(X.getAnimalName()) #透過繼承取得父類別 Method
```

```
print(X.getAnimalSpecies())
```

```
print(X.update()) #依然可使用本身類別的 Method
```

```
PS C:\Users\david_zhu\Desktop\Python_Intro  
.exe "c:/Users/david_zhu/Desktop/Python_In  
LaLa  
我是貓咪  
喵喵叫
```

# Inheritance (super ())

```
# Animal Class
```

```
class baseAnimal:  
  
    def __init__(self, name, species):  
        self.__name = name  
        self.__species = species  
  
    def getAnimalName(self):  
        return self.__name  
  
    def getAnimalSpecies(self):  
        return self.__species  
  
class cat(baseAnimal):  
    def update(self):  
        return "喵喵叫"  
  
    def getAnimalSpecies(self):      #以相同名稱的 Method 覆寫父類別  
        return super().getAnimalSpecies() #以 super() 調用父類別 Method  
  
X = cat("LaLa","Cat")  
print(X.getAnimalName())  
print(X.getAnimalSpecies())
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction.exe "c:/Users/david_zhu/Desktop/Python_LaLa  
Cat  
喵喵叫"
```

# Abstract Method (抽象方法)

```
# Abstract Class
```

```
from abc import ABC, abstractmethod
```

```
class baseAnimal(ABC):      # 繼承於 python 內建的抽象類別
    @abstractmethod
    def getAnimalName(getname):
        pass
    @abstractmethod
    def getAnimalSpecies(getspecies):
        pass
```

```
animal = baseAnimal()
# 抽象類別無法被直接實例化
```

```
Traceback (most recent call last):
  File "c:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main\Example Code\Test.py", line 13, in <module>
    animal = baseAnimal()
TypeError: Can't instantiate abstract class baseAnimal with abstract methods getAnimalName, getAnimalSpecies
```

# Abstract Method (抽象方法)

# Abstract Class

```
from abc import ABC, abstractmethod
```

```
class baseAnimal(ABC):
    @abstractmethod
    def getAnimalName(getname):
        pass
    @abstractmethod
    def getAnimalSpecies(getspecies):
        pass
```

```
class cat(baseAnimal):

    def __init__(self, name, species):
        self.name = name
        self.species = species

    def getAnimalName(self): #以相同的 Function 名稱實現 overwrite
        return self.name

    def getAnimalSpecies(self):
        return self.species

cat1 = cat("Lala", "橘貓")
print(cat1.getAnimalName())
print(cat1.getAnimalSpecies())
#抽象類別需透過繼承來實現，並 overwrite 抽象方法
```

```
PS C:\Users\david_zhu\Desktop\Python_Introduction-main\Python_
thon.exe "c:/Users/david_zhu/Desktop/Python_Introduction-main/P
Lala
橘貓
```

# Abstract Method (抽象方法)

# Abstract Class

```
from abc import ABC, abstractmethod
```

```
class baseAnimal(ABC):
    @abstractmethod
    def getAnimalName(getname):
        pass
    @abstractmethod
    def getAnimalSpecies(getspecies):
        pass
```

```
class cat(baseAnimal):

    def __init__(self, name, species):
        self.name = name
        self.species = species
```

```
    def getAnimalName(self):
        return self.name
```

```
cat1 = cat("Lala", "橘貓")
```

# 抽象類別可以強制繼承時所要實作的方法  
# 若沒有在繼承時將抽象類別的方法實作則會產生 error

```
Traceback (most recent call last):
```

```
  File "c:\Users\david_zhu\Desktop\Python_Introduction-main\Python_Introduction-main\E
    ile\Test.py", line 22, in <module>
        cat1 = cat("Lala", "橘貓")
```

```
TypeError: Can't instantiate abstract class cat with abstract method getAnimalSpecies
```

# Polymorphism (多型)

# Polymorphism

```
class cat:  
    def __init__(self, name, species):  
        self.name = name  
        self.species = species  
  
    def getAnimalName(self):  
        print("我叫", self.name)  
  
    def getAnimalSpecies(self):  
        print("我是", self.species)  
  
class dog:  
    def __init__(self, name, species):  
        self.name = name  
        self.species = species  
  
    def getAnimalName(self):  
        print("我叫", self.name)  
  
    def getAnimalSpecies(self):  
        print("我是", self.species)
```

```
cat1 = cat("Lala", "橘貓")  
cat2 = cat("YoYo", "虎斑貓")
```

```
dog1 = dog("RuRu", "柴犬")  
dog2 = dog("QQ", "吉娃娃")
```

```
cat1.getAnimalName() # 透過不同的實例化物件來呼叫 Method  
cat1.getAnimalSpecies()
```

```
cat2.getAnimalName()  
cat2.getAnimalSpecies()
```

```
dog1.getAnimalName() # 透過不同的實例化物件來呼叫 Method  
dog1.getAnimalSpecies()
```

```
dog2.getAnimalName()  
dog2.getAnimalSpecies()
```

# 透過不同的實例化物件來呼叫相同的 Method 可以達到不同的實作方式

```
0/python.exe "c:/Users/david_zhu/  
我叫 Lala  
我是 橘貓  
我叫 YoYo  
我是 虎斑貓  
我叫 RuRu  
我是 柴犬  
我叫 QQ  
我是 吉娃娃
```

# Reference

Python download

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

Visual Studio Code download

<https://code.visualstudio.com/download>

Anaconda download

<https://www.anaconda.com/products/individual>

Python Document

<https://docs.python.org/zh-tw/3/tutorial/index.html>

Python Learn by example

<https://www.learnbyexample.org/python-introduction/>

# Q&A

