

# Tài liệu hướng dẫn thực hành Buổi 1

## Môn : Máy học ứng dụng

Nội dung chính : Giới thiệu ngôn ngữ lập trình Python

### 1. Giới thiệu

Python là một ngôn ngữ lập trình dạng thông dịch do Guido Van Rossum tạo ra năm 1990. Python được viết từ nhiều ngôn ngữ lập trình khác nhau và tạo ra những bản hiện thực khác nhau. Trong đó bản hiện thực chính là CPython được viết bằng C.

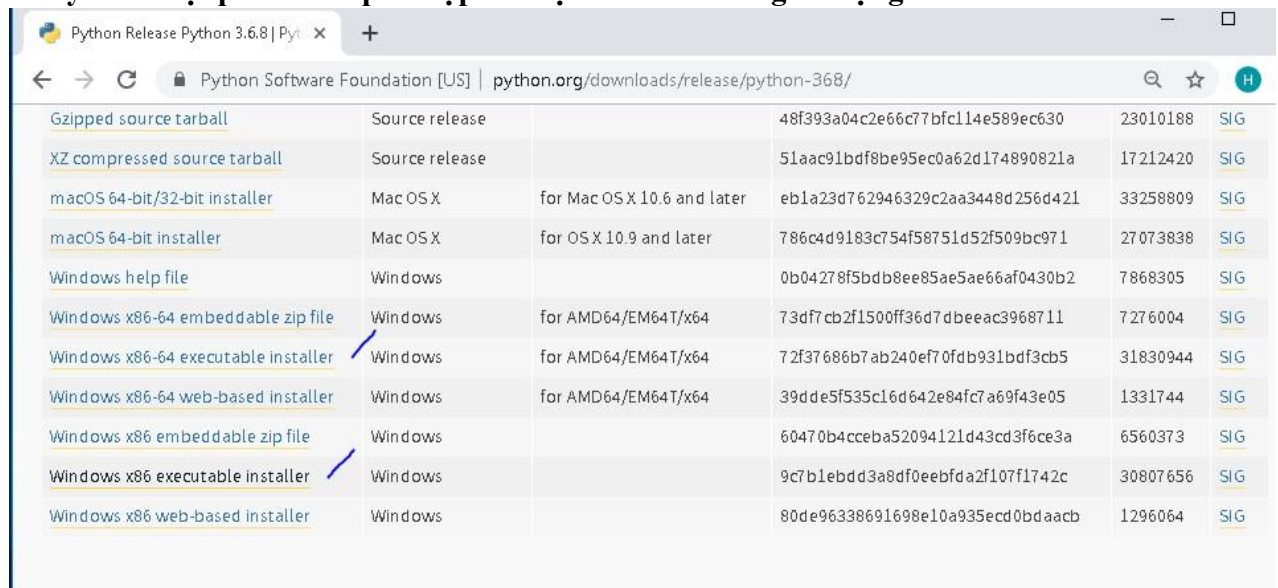
Python có tính tương thích lớn trên nhiều môi trường phát triển và cũng được ứng dụng rộng rãi trên nhiều lĩnh vực.

Điểm khác biệt đặc trưng giữa C và Python là Python sử dụng cơ chế cấp phát bộ nhớ tự động và hoàn toàn tạo kiểu động (khác với C phải khai báo biến và cấp phát bộ nhớ tương ứng với kiểu của biến). Điều này giúp tối thiểu hóa số lần gõ phím để viết mã lệnh và giúp cho cấu trúc có hình thức gọn gàng sáng sủa rất thuận tiện cho người mới học lập trình.

### 2. Hướng dẫn cài đặt Python 3.6

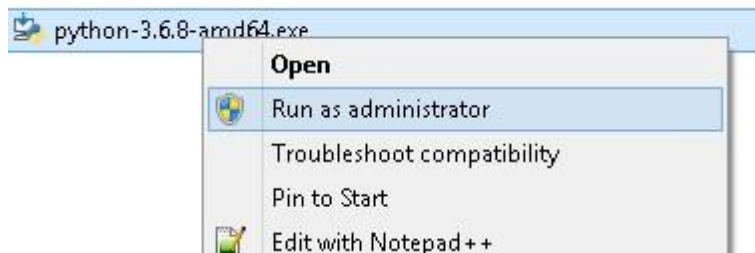
Download Python : Truy cập vào trang web : <https://www.python.org/> để download Python

**Chú ý : Cài đặt phiên bản phù hợp với hệ điều hành đang sử dụng**

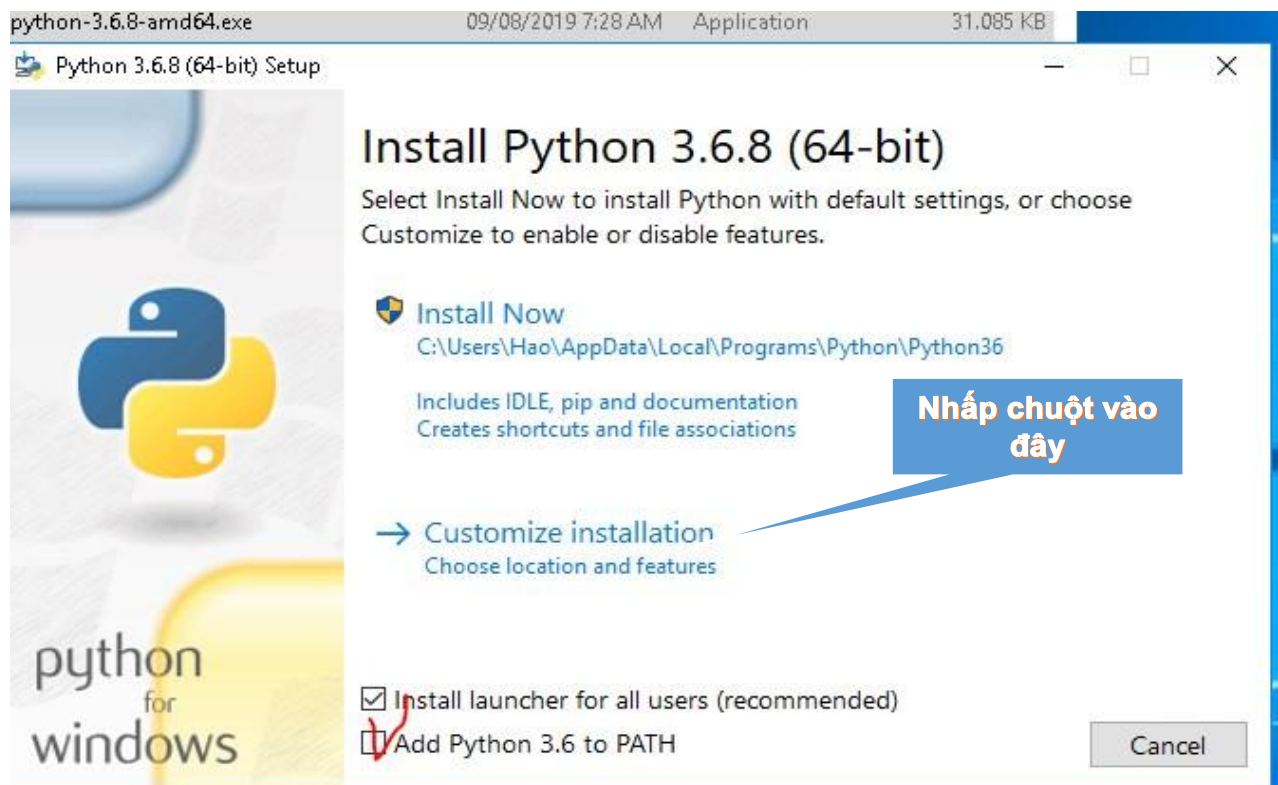


<a href="#">Gzipped source tarball</a>	Source release		48f393a04c2e66c77bfc114e589ec630	23010188	<a href="#">SIG</a>
<a href="#">XZ compressed source tarball</a>	Source release		51aac91bdf8be95ec0a62d174890821a	17212420	<a href="#">SIG</a>
<a href="#">macOS 64-bit/32-bit installer</a>	Mac OS X	for Mac OS X 10.6 and later	eb1a23d762946329c2aa3448d256d421	33258809	<a href="#">SIG</a>
<a href="#">macOS 64-bit installer</a>	Mac OS X	for OS X 10.9 and later	786c4d9183c754f58751d52f509bc971	27073838	<a href="#">SIG</a>
<a href="#">Windows help file</a>	Windows		0b04278f5bdb8ee85ae5ae66af0430b2	7868305	<a href="#">SIG</a>
<a href="#">Windows x86-64 embeddable zip file</a>	Windows	for AMD64/EM64T/x64	73df7cb2f1500ff36d7dbecac3968711	7276004	<a href="#">SIG</a>
<a href="#">Windows x86-64 executable installer</a>	Windows	for AMD64/EM64T/x64	72f37686b7ab240ef70fdb931bdf3cb5	31830944	<a href="#">SIG</a>
<a href="#">Windows x86-64 web-based installer</a>	Windows	for AMD64/EM64T/x64	39dde5f535c16d642e84fc7a69f43e05	1331744	<a href="#">SIG</a>
<a href="#">Windows x86 embeddable zip file</a>	Windows		60470b4cceba52094121d43cd3f6ce3a	6560373	<a href="#">SIG</a>
<a href="#">Windows x86 executable installer</a>	Windows		9c7b1ebdd3a8df0eebfda2f107f1742c	30807656	<a href="#">SIG</a>
<a href="#">Windows x86 web-based installer</a>	Windows		80de96338691698e10a935ecd0bdaacb	1296064	<a href="#">SIG</a>

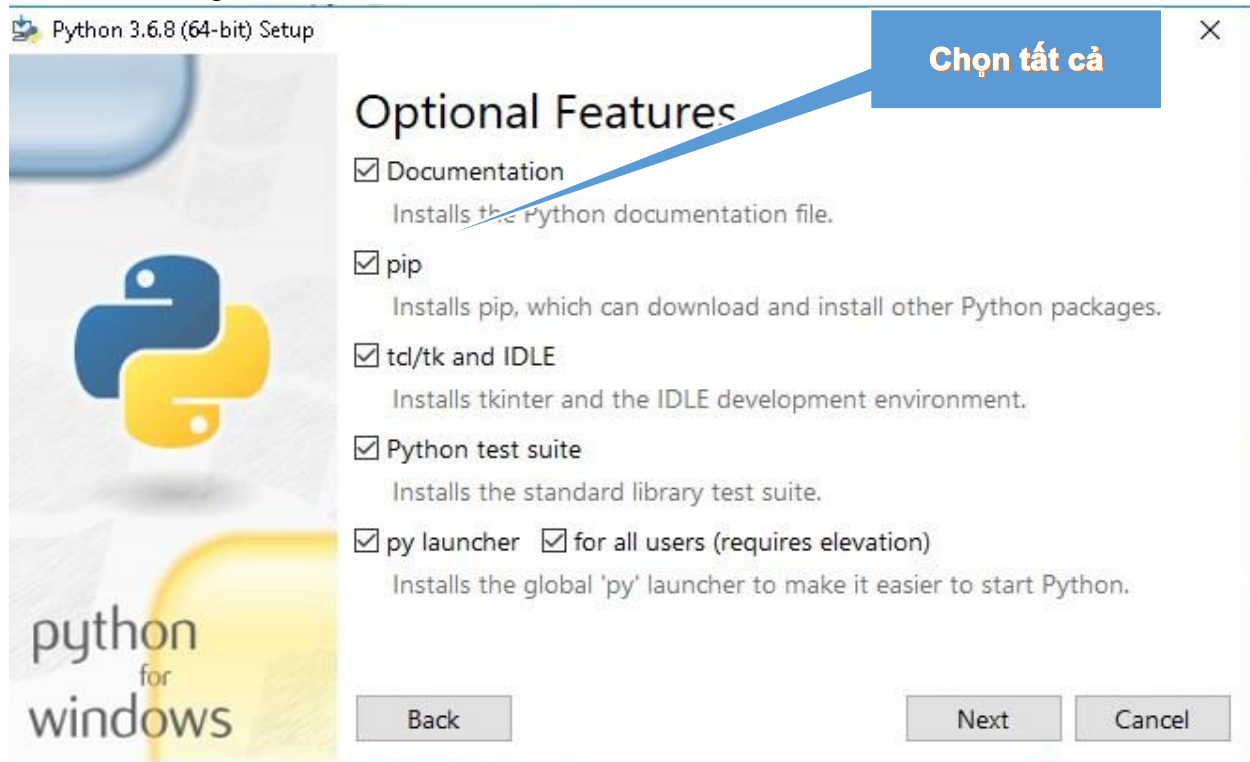
Ví dụ cài trên hệ điều hành Windows 64 bits. Ta download và thực thi tập tin **python-3.6.8amd64.exe** với quyền quản trị (Right Click - Chọn Run as administrator như hình



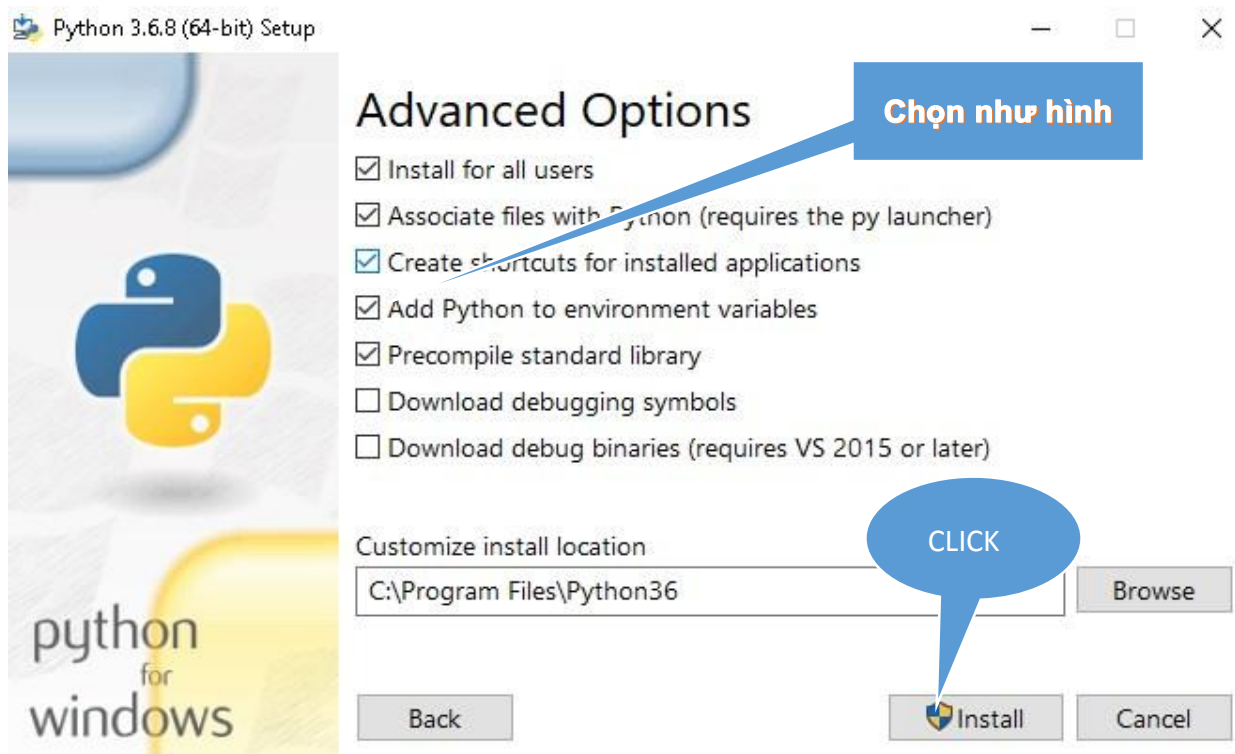
Chương trình cài đặt có giao diện như sau:



Giao diện kế tiếp sẽ là:




Giao diện kế tiếp sẽ là



Chọn Install thì chương trình sẽ cài đặt cho đến khi hoàn thành



### Kiểm tra chương trình

- Mở cửa sổ terminal (bấm tổ hợp  + R, gõ vào *cmd*).
- Trên cửa sổ terminal gõ lệnh `python` sau đó enter

```
C:\Windows\system32\cmd.exe - python
Microsoft Windows [Version 10.0.16299.967]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Hao>python
Python 3.6.8 (tags/v3.6.8:3c6b436a57, Dec 24 2018, 00:16:47) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> _
```

- Thực hiện các dòng lệnh của python. Khi thực hành nên các dòng lệnh chương trình để kiểm tra được từng dòng lệnh.
- Để kết thúc chương trình gõ `exit()` hoặc `Ctrl + Z` và nhấn Enter

### 3. Hướng dẫn soạn thảo

Có thể sử dụng phần mềm bất kỳ để soạn thảo và lưu lại với tên chương trình có phần mở rộng là **.py**

#### a. Sử dụng notepad++

Tạo file với phần mở rộng **.py** và soạn thảo.

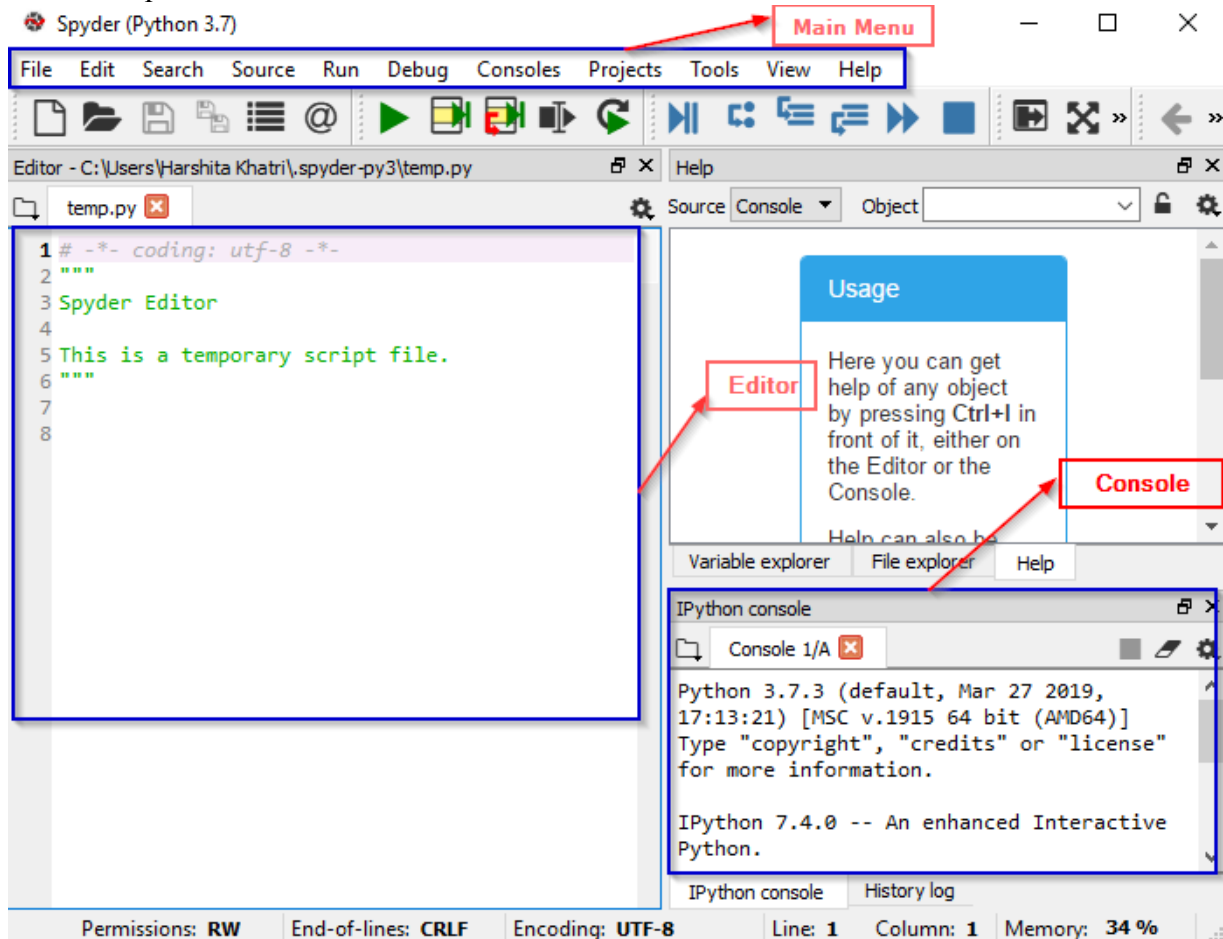
Copy từng dòng lệnh ở cửa sổ soạn thảo notepad++ và nhấp phải chuột, chọn “past” vào python trên terminal để biên dịch từng dòng lệnh.

## b. Sử dụng Spyder

Hướng dẫn cài đặt:

<https://www.edureka.co/blog/spyder-ide/#installation>

Kết quả cài đặt như sau:



Cách sử dụng Spyder để viết một chương trình đơn giản bằng Python :

- Tạo một thư mục trong ổ D:\ và đặt tên là Python (Tạo thư mục để lưu trữ chương trình. Tại phòng thực hành thì lưu vào trong đĩa D:\)
- Mở Spyder, từ menu File chọn New File, ta thấy một cửa sổ mới được tạo ra, gõ tên file hello.py
- Bấm tổ hợp Ctrl + S để save file đó vào thư mục Python vừa tạo ở ổ đĩa D:\
- Trên cửa sổ soạn thảo của Spyder viết vào dòng lệnh sau : `print ("Hello Python")` và bấm Ctrl + S để save lại
- Bấm nút Run trên menu để chạy chương trình.

## 4. Các cú pháp cơ bản

Python phân biệt các ký tự thường và hoa, đồng thời tăng cường sử dụng các từ khóa tiếng Anh, hạn chế sử dụng các ký hiệu và cấu trúc cú pháp so với các ngôn ngữ khác Một số từ khóa thông dụng của Python :

- |          |         |       |
|----------|---------|-------|
| • and    | exec    | not   |
| • as     | finally | or    |
| • assert | for     | pass  |
| • break  | from    | print |

- class            global            raise
- continue       if                return
- def             import            try
- del             in                while
- elif            is                with
- else            lambda            yield
- except          continue

Các toán tử cơ bản :

+ - \* / // (chia làm tròn) % (phần dư) \*\* (lũy thừa)  
 ~ (not) & (and) | (or) ^ (xor)  
 << (left shift) >> (right shift)  
 == (bằng) <= >= != (khác)

Cấu trúc khối lệnh : sử dụng canh lề để bao các khối lệnh của hàm, lớp, hoặc luồng điều khiển. Số khoảng trắng dùng để canh lề có thể tùy chọn nhưng các lệnh trong cùng một khối phải được canh lề như nhau.

Trong Python 3.x, việc nhập liệu từ bàn phím có thể được thực hiện qua hàm input(), hiển thị ra màn hình (shell) với hàm print(tham số) Ví dụ:

```
print ("Nhập vào tên:",end=" ")#thêm end=" " de không xuống hàng
ten=input();
print ("Xin chào",ten)
```

Ví dụ :

```
a=5
b=3
if a>b:
    a=a*2+3
    b=b-6
    c=a/b
    print (c)
```

Dòng lệnh dài viết trên nhiều dòng sử dụng ký tự \ Ví dụ :

```
c=a+b+\
    10*a-b/4-\
    5+a*3
print (c)
```

Lệnh nằm trong các cặp dấu ngoặc : [] {} () không cần sử dụng ký tự \ để tiếp tục dòng  
 Dấu ; để cách nhiều lệnh trên cùng 1 dòng

## 5. Lệnh và cấu trúc điều khiển Lệnh

if :

```
if biểu_thức_điều_kiện:
    # lệnh...
if biểu_thức_điều_kiện:
    # lệnh... else:
    # lệnh...
```

Ví dụ :



```
a=5
b=3
if a>b:
    print ("True")
    print (a)
else:
    print ("Fasle")
    print (b)
```

Lệnh while :

```
while biểu_thức_đúng:
    # lệnh...
```

Ví dụ :

```
a=1
b=10
while a<b:
    a+=1
    print (a)
```

Lệnh for :

```
for phần_tử in dãy:
    # lệnh...
```

Ví dụ :

```
for i in range (1,10):
    print (i)
```

Khai báo hàm :

```
def tên_hàm (tham_biến_1, tham_biến_2, tham_biến_n):
    # lệnh...
    return giá_trị_hàm
```

Ví dụ :

```
def binhphuong(number):
    return number*number
print (binhphuong(5))
```

## 6. Các kiểu dữ liệu

Python gồm các kiểu dữ liệu cơ bản như sau :

Kiểu số – Number : Kiểu dữ liệu numbers chứa giá trị là một con số, nó được tạo ra khi ta gán các giá trị là một con số cho biến đó.

Python hỗ trợ 4 kiểu dữ liệu numbers khác nhau:

- int
- long
- float
- complex

Ví dụ :

```
a = 5
b = -7
c = 1.234
```

Kiểu chuỗi – String : Kiểu dữ liệu strings được xác định khi giá trị được gán cho nó nằm giữa cặp dấu ' ' hoặc " "

Ví dụ :

```
str1 = "Hello"  
str2 = "welcome"  
str3 = "abcdef12345"
```

Kiểu danh sách – List : Một list trong Python là một nhóm có thứ tự các đối tượng. Điều quan trọng cần phải nhấn mạnh là những đối tượng đó không cần phải là cùng loại. Nó có thể là một hỗn hợp tùy ý của các đối tượng như số, chuỗi hoặc có thể là một danh sách khác.

Dưới đây là một số thuộc tính của list:

- List là một tập hợp có thứ tự.
- Dữ liệu trong một list có thể thuộc nhiều kiểu khác nhau.
- Các phần tử trong list có thể được truy xuất thông qua chỉ số của chúng.
- Kích thước của một list có thể thay đổi.
- Chúng ta có thể thay đổi một list, ví dụ như việc thêm mới, xóa hoặc cập nhật phần tử trong list.

Ví dụ : Khai báo một danh sách gồm 5 phần tử kiểu chuỗi :

```
cats = ['Tom', 'Snappy', 'Kitty', 'Jessie', 'Chester']
```

Chúng ta có thể truy xuất các phần tử của list thông qua chỉ số của chúng như sau:

```
print (cats[2]) # in ra phần tử có chỉ số là 2  
// Kitty  
print (cats[-1]) # in ra phần tử cuối cùng của list  
// Chester  
  
print (cats[1:3]) # in ra các phần tử có chỉ số từ 1 đến 3 //['Snappy',  
'Kitty']
```

Vì list trong Python có thể thay đổi nên chúng ta có thể thêm mới, sửa hoặc xóa bỏ phần tử trong list:

```
print (cats)  
['Tom', 'Snappy', 'Kitty', 'Jessie', 'Chester']  
  
cats.append('Jerry') # thêm mới phần tử vào list  
print (cats)  
['Tom', 'Snappy', 'Kitty', 'Jessie', 'Chester', 'Jerry']  
  
cats[-1] = 'Jerrt Cat' # gán lại giá trị cho phần tử cuối cùng của list  
print (cats)  
['Tom', 'Snappy', 'Kitty', 'Jessie', 'Chester', 'Jerrt Cat']  
  
del cats[1] # xóa bỏ phần tử có số thứ tự 1 khỏi list  
print (cats)  
['Tom', 'Kitty', 'Jessie', 'Chester', 'Jerrt Cat']
```



Bảng dưới đây liệt kê các hàm dùng để làm việc với list:

- `len(list)`: Lấy số lượng phần tử trong list.
- `max(list)`: Trả về phần tử có giá trị lớn nhất trong list.
- `min(list)`: Trả về phần tử có giá trị nhỏ nhất trong list.
- `list.append(obj)`: Thêm một phần tử vào list.
- `list.count(obj)`: Đếm số lần xuất hiện của phần tử `obj` trong list
- `list1.extend(list2)`: Thêm tất cả các phần tử của `list2` vào `list1`
- `list.index(obj)`: Trả về chỉ số bé nhất mà `obj` xuất hiện trong list
- `list.insert(index, obj)`: Thêm phần tử `obj` vào vị trí `index` trong list.
- `list.pop(index)`: Xóa và trả về phần tử có chỉ số `index` trong list.
- `list.remove(obj)`: Xóa phần tử trong list.
- `list.reverse()`: Đảo ngược các phần tử trong list. □ `list.sort()`: Sắp xếp các phần tử trong list.

Kiểu bộ – Tuple : Một Tuple có thể được hiểu là một danh sách không thay đổi, điều này có nghĩa là bạn không thể thay đổi một tuple một khi nó đã được tạo ra. Một tuple được định nghĩa tương tự như list ngoại trừ việc tập hợp các phần tử được đặt trong dấu ngoặc đơn thay vì dấu ngoặc vuông như list, ngoài ra quy tắc về chỉ số phần tử của tuple cũng tương tự như list.

Vậy tại sao Python lại định nghĩa một kiểu dữ liệu tương tự như list? Dưới đây là các ưu điểm của tuple:

- Tốc độ truy xuất, xử lý dữ liệu của tuple nhanh hơn so với list.
- Trong lập trình, có những dữ liệu không được thay đổi, trong trường hợp đó bạn nên dùng tuple thay vì list, điều này sẽ đảm bảo được dữ liệu của bạn, chống lại những sự thay đổi ngẫu nhiên đối với dữ liệu đó.

Bảng dưới đây liệt kê các hàm dùng để làm việc với list:

- `len(tuple)`: Lấy số lượng phần tử trong tuple.
- `max(tuple)`: Trả về phần tử có giá trị lớn nhất trong tuple.
- `min(tuple)`: Trả về phần tử có giá trị nhỏ nhất trong tuple.
- `tuple(list)`: Chuyển một list sang tuple.

Kiểu từ điển – Dictionary : Một Dictionary trong Python có thể được hiểu là một tập các cặp key-value. Mỗi key được phân cách với giá trị của nó bằng dấu hai chấm (:), các phần tử trong dictionary được phân cách bằng dấu phẩy. Một dictionary rỗng, không chứa bất kì phần tử nào được định nghĩa bằng hai dấu ngoặc nhọn như sau: `{}`.

Key trong dictionary phải là duy nhất nhưng điều này không cần thiết với các value, nghĩa là trong một dictionary có thể có nhiều key có cùng một value. Các value trong dictionary có thể thuộc bất kì một kiểu nào còn các key thì chỉ được cố định trong một số kiểu như: string, number hoặc tuple.

Dictionary không dùng chỉ số của các phần tử để truy xuất dữ liệu, thay vào đó, chúng ta có thể truy xuất dữ liệu thông qua key như ví dụ sau:

```
dict1 = {'Name' : 'Zyra', 'Age' : 7, 'Class' : 'A5'} # định nghĩa 1 dictionary
print (dict['Name']) # lấy giá trị có key = 'Name'
'Zyra'
print (dict['Age']) # lấy giá trị có key = 'Age' 7
```

Chúng ta có thể chỉnh sửa một Dictionary như việc thêm mới, xóa, hoặc chỉnh sửa phần tử như ví dụ dưới đây:

```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
dict['Age'] = 8 # cập nhật một phần tử trong dict
dict['School'] = "DPS School" # thêm mới một phần tử vào dict
print (dict['Age'])
// 8
print (dict['School']) //
DPS School
```

Dưới đây là danh sách các hàm để làm việc với Dictionary trong Python:

- `len(dict)`: Lấy số lượng phần tử trong dict.
- `str(dict)`: Tạo ra một chuỗi có thể in được của dict
- `type(variable)`: Trả về kiểu của biến được truyền vào, nếu biến được truyền vào là kiểu dictionary hàm sẽ trả về kiểu dictionary. □ `dict.clear()`: Xóa tất cả các phần tử của dict
- `dict.copy()`: Trả về một bản sao của dict
- `dict.get(key, default=None)`: Trả về giá trị tương ứng với key hoặc trả về giá trị default nếu key không tồn tại trong dict
- `dict.has_key(key)`: Trả về true nếu key tồn tại trong dict, ngược lại trả về false.
- `dict.items()`: Trả về một list các cặp key/value của dict.
- `dict.keys()`: Trả về một list các key của dict
- `dict.setdefault(key, default=None)`: Hàm này gần giống với hàm `get()` nhưng sẽ gán `dict[key]=default` nếu key chưa tồn tại trong dict.
- `dict1.update(dict2)`: Thêm các phần tử của dict2 vào dict1.
- `dict.values()`: Trả về danh sách value của dict dưới dạng một list.

Tham khảo các kiểu dữ liệu đặc biệt trong Python:  
<https://docs.python.org/3/tutorial/datastructures.html>

## 7. Cài đặt các thư viện cần thiết

Python hỗ trợ rất nhiều các gói thư viện hỗ trợ cho nhiều yêu cầu sử dụng khác nhau từ xử lý đồ họa, bảo mật, thiết kế giao diện hay nâng cao hiệu năng tính toán. Trong nội dung môn học Nguyên Lý Máy Học, chúng ta cần cài đặt và sử dụng các thư viện sau :

NumPy : Đây là một thư viện cơ bản hỗ trợ rất mạnh mẽ cho việc tính toán trên những ma trận dữ liệu đa chiều


SciPy : Đây là thư viện tổng hợp chứa các giải thuật, thuật toán và các công cụ giúp cho việc xử lý tín hiệu, tối ưu hóa, thống kê, và nhiều lĩnh vực khác

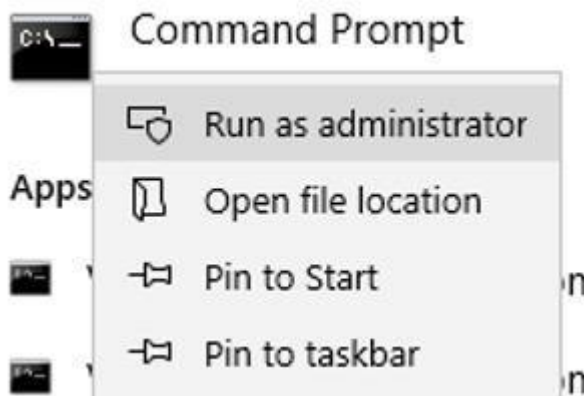
Matplotlib : Đây là thư viện hỗ trợ cho việc hiển thị các biểu đồ, đồ thị trong không gian 2 chiều hoặc 3 chiều.

Pandas: Đây là thư viện cung cấp các cấu trúc dữ liệu và các thao tác để thao tác các bảng số và chuỗi thời gian,.....

**Sklearn: thư viện chứa các giải thuật máy học: bayes thư ngây, cây quyết định, hồi quy,...**

Để cài đặt thư viện sử dụng cho python

- Mở cửa sổ terminal bằng quyền quản trị hệ thống (bấm tổ hợp  + S, gõ vào `cmd`, Right click cửa sổ Command Prompt chọn Run as administrator).



- Cập nhật công cụ pip bằng lệnh `python -m pip install --upgrade pip`
- Sử dụng công cụ **pip** để cài đặt theo cú pháp **pip install tên thư viện**. Hệ thống sẽ tự động download thư viện về máy và cài đặt.

Cài đặt thư viện Pandas : `pip install pandas`

**Cài đặt thư viện Sklearn :** `pip install sklearn`

Cài đặt thư viện SciPy : `pip install scipy` Cài đặt thư viện

Matplotlib : `pip install matplotlib`

Cài đặt thư viện Numpy : `pip install numpy`

- Gỡ bỏ thư viện ra khỏi hệ thống bằng lệnh **pip uninstall tên thư viện** Khi thực hành nếu thư viện bị lỗi, cần phải gỡ ra và cài đặt lại

## 8. Ví dụ minh họa

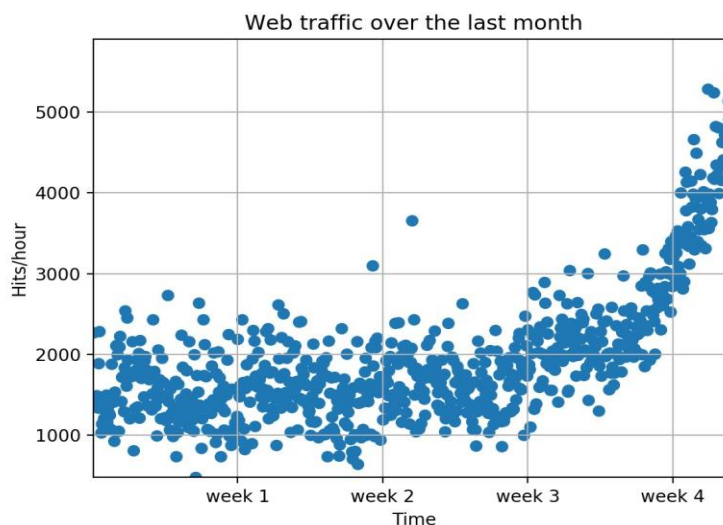
Ví dụ 1 : Thao tác trên mảng với thư viện NumPy

```
10
11 import numpy as np
12
13 #tao mang co 6 phan tu
14 a = np.array([0,1,2,3,4,5])
15 #hien thi mang a
16 print(a)
17 print("So chieu cua mang a: ", a.ndim)
18 print("Kich thuoc ma tran a la: ", a.shape)
19 print("In cac phan tu co gia tri >3 trong mang a: ", a[a>3])
20 #gan lai cac phan tu co gia tri >3 la 10
21 a[a>3] = 10
22
23 #thay doi kich thuoc cua ma tran a
24 b = a.reshape((3,2))
25 print("Mang b la:",b)
26 print("So chieu cua mang b: ", b.ndim)
27 print("Kich thuoc cua ma tran b la: ", b.shape)
28 #Gia tri phan tu 2,1 trong b
29 print(b[2][1])
30 #gan gia tri 50 tai vi tri 2,0
31 b[2][0] = 50
32 print(b)
33
34 c = b*2
35 #Hien thi c
36 print(c)
```

Ví dụ 2 : Đọc và xử lý dữ liệu từ file bên ngoài với thư viện SciPy, hiển thị dữ liệu với thư viện Matplotlib

```
39
40 #ví dụ 2:
41
42 import scipy as sp
43 import numpy as np
44 #đọc dữ liệu từ file web_traffic.tsv và lưu vào biến data
45 data = sp.genfromtxt("web_traffic.tsv",delimiter="\t")
46 #in kích thước dòng, cột của dữ liệu
47 print(data.shape)
48 #(743,2) => dữ liệu gồm 743 dòng và 2 cột
49
50 #lấy dữ liệu ở cột thứ nhất gán cho biến x
51 x = data[:,0]
52 #lấy dữ liệu ở cột thứ 2 gán cho biến y
53 y = data[:,1]
54
55 #đếm số dòng còn lại sau khi xóa bỏ dòng trống ở cột y
56 print(len(~np.isnan(y)))
57
58
59 #loại bỏ các dòng không có dữ liệu trong cột y
60 x = x[~np.isnan(y)]
61
62 y = y[~np.isnan(y)]
63
64
65 import matplotlib.pyplot as plt
66
67 plt.scatter(x,y)
68 plt.title("Web traffic over the last month")
69 plt.xlabel("Time")
70 plt.ylabel("Hits/hour")
71 plt.xticks([w*7*24 for w in range(10)], ['week %i'%w for w in range(10)])
72 plt.autoscale(tight=True)
73 plt.grid()
74 plt.show()
```

Ta thu được kết quả sau :



Ví dụ 3 : Đọc và xử lý dữ liệu từ file bên ngoài với thư viện pandas, dùng “**data.iloc**” (truy xuất cột theo chỉ số) hoặc “**data.loc**” (truy xuất cột theo tên)

```
77 #ví dụ 3:
78 |
79 import pandas as pd
80 import matplotlib.pyplot as plt
81 data = pd.read_csv("play_tennis.csv", delimiter = ",")
82 a=data.head(5) #lay 5 dòng đầu
83 b=data.tail(5) #lay 5 dòng cuối
84 d46=data.iloc[3:7] #hàng thu 3 đến 6, bắt đầu là 0
85 c1 = data.iloc[:,0:1] #cột đầu tiên, cột 0
86 c15=data.iloc[:,0:6] #cột đầu tiên (0) đến cột thu 5
87 c3=data.iloc[:,2] #cột thu 2, bắt đầu từ 0
88 c453=data.iloc[4:6,2] #giá trị dòng 4,5 cột 2 (tính từ 0)
89 print(data.outlook) # in dữ liệu cột outlook
90
91
92
```

Có thể đọc dữ liệu iris từ thư viện có sẵn

```
from sklearn.datasets import load_iris
iris_dt = load_iris()
iris_dt.data[1:5] #dữ liệu x
iris_dt.target[1:5] #nhãn y
```