



BỘ MÔN KỸ THUẬT MÁY TÍNH – VIỄN THÔNG
CƠ SỞ VÀ ỨNG DỤNG IOTS
MMH: ITFA436064/ ITFA336064

Họ và tên: Vũ Tiến Phát-21151309

Hoàng Thị Diễm Quỳnh-21151326

Nguyễn Nam Huy-21151244

Ngô Xuân Thọ-21119376

Trương Nguyễn Quốc Thắng-22119231

1.

Data Types (liệt kê các kiểu dữ liệu trong Python):

Tên các kiểu dữ liệu trong Python:

int, float, complex, string, integer, dictionary, str, tuple, byte, bool, list, range, dict, set, frozenset, bytearray, memoryview, ...

```
x = 3
print(type(x))
print(x)
print(x + 1)
print(x - 1)    |
print(x * 2)
print(x ** 2)
x += 1
print(x)
x *= 2
print(x)
y = 2.5
print(type(y))
print(y, y + 1, y * 2, y ** 2)
```

```
<class 'int'>
3
4
2
6
9
4
8
<class 'float'>
2.5 3.5 5.0 6.25
> |
```

Giải thích: Trong câu lệnh print trên có chức năng in kết quả ra bàn hình và thực hiện các phép cộng, trừ, nhân, lũy thừa trong ngoặc.

Câu x +=1 thực hiện gán x=x+1. tương tự với câu lệnh x*=2 là phép tính lũy thừa.

Hàm type() để kiểm tra kiểu dữ liệu của biến.

2.

Loop (cấu trúc vòng lặp trong python):

```
*Cấu trúc vòng lặp for
for iterator_var in sequence:
    statement(s)
```

*Cấu trúc vòng lặp while

while expression:

statement(s)

```
animals = ['cat', 'dog', 'monkey']
for animal in animals:
    print(animal)
```

cat

dog

monkey

Giải thích: với vòng lặp for mỗi lần sẽ in một giá trị trong chuỗi animals

```
animals = ['cat', 'dog', 'monkey']
for idx, animal in enumerate(animals):
    print('#%d: %s' % (idx + 1,
animal))
```

#1: cat

#2: dog

#3: monkey

Giải thích:

+ Trong vòng lặp for, mỗi lần sẽ in ra một giá trị trong chuỗi animals và biến idx sẽ nhận giá trị của chỉ số biến vừa được in ra.

+ %d dùng để định dạng số nguyên, %s dùng để định dạng cho chuỗi.

+ idx + 1 là do trong chuỗi giá trị đầu tiên sẽ là 0 nên cộng thêm 1 để có số thứ tự đếm bắt đầu từ 1.

3.

Functions (cách định nghĩa hàm)

Trong Python, bạn có thể định nghĩa hàm bằng cách sử dụng từ khóa def.

Cú pháp:

```
def ten_ham(tham_so1, tham_so2, ...):
    # Khối mã bên trong hàm
    # Các câu lệnh thực hiện một
    return ket_qua # (không bắt buộc)
```

```
def sign(x):
    if x > 0:
        return 'positive'
    elif x < 0:
        return 'negative'
    else:
        return 'zero'

for x in [-1, 0, 1]:
    print(sign(x))
```

negative

zero

positive

Giải thích:

+ def sign(x) là hàm định nghĩa của sign và nhận x làm tham số. Hàm này xác định dấu của tham số x. Nếu x > 0 hàm sẽ trả về giá trị 'positive'. Nếu x < 0 hàm sẽ trả về giá trị 'negative'. Nếu x = 0 hàm sẽ trả về giá trị 'zero'.

+ Trong vòng lặp for, mỗi lần sẽ in ra một giá trị. Với x = -1 sẽ in ra 'negative'. Với x = 0 sẽ in ra 'zero'. Với x = 1 sẽ in ra 'positive'.

4.

Numpy (thư viện numpy có đặc điểm gì?):

_ Mảng Nhiều Chiều: NumPy giới thiệu một cấu trúc dữ liệu mới là "mảng" (array), cho phép lưu trữ và làm việc với dữ liệu đa chiều (ví dụ: ma trận, tensor) một cách hiệu quả.

_ Hiệu Năng Cao: NumPy được viết bằng ngôn ngữ C, nên có hiệu năng cao. Các phép toán trên mảng NumPy được thực hiện nhanh chóng hơn so với việc sử dụng danh sách (list) trong Python thông thường.

_ **Phép Toán Số Học và Ma Trận:** NumPy cung cấp một loạt các phép toán số học (cộng, trừ, nhân, chia, căn bậc hai, luỹ thừa, logarit, v.v.) và phép toán ma trận (nhân ma trận, chuyển vị, định thức, giải hệ phương trình tuyến tính, v.v.).

_ **Hỗ Trợ Đa Dạng Kiểu Dữ Liệu:** NumPy cho phép bạn làm việc với nhiều kiểu dữ liệu khác nhau như số nguyên, số thực, số phức, boolean, và nhiều kiểu dữ liệu tùy chỉnh.

_ **Đối Tượng Mảng Đồng Nhất:** Mảng NumPy chứa các phần tử cùng kiểu dữ liệu, đảm bảo tính đồng nhất và hiệu quả trong lưu trữ và tính toán dữ liệu.

_ **Hỗ Trợ Trong Khoa Học Dữ Liệu:** NumPy là một phần quan trọng của hệ sinh thái khoa học dữ liệu Python. Nó làm việc tốt với các thư viện và frameworks như SciPy, pandas, Matplotlib, scikit-learn, TensorFlow và PyTorch.

_ **Cộng Đồng Lớn và Phát Triển Liên Tục:** NumPy có một cộng đồng phát triển lớn và được duy trì chặt chẽ, đảm bảo tính ổn định và phát triển của thư viện.

_ **Công Cụ Tùy Chỉnh:** Bạn có thể định nghĩa và sử dụng các cấu trúc dữ liệu tùy chỉnh bằng NumPy, cho phép bạn làm việc với dữ liệu theo cách bạn muốn.

_ **Hỗ Trợ Đa Dạng Nhiều Nền Tảng:** NumPy hoạt động trên nhiều nền tảng, bao gồm Windows, macOS và Linux.

_ **Tích Hợp Dễ Dàng:** NumPy tích hợp tốt với ngôn ngữ Python và có thể được sử dụng trong các dự án Python một cách dễ dàng.

```
import numpy as np

a = np.array([1, 2, 3])
print(type(a))
print(a.shape)
print(a[0], a[1], a[2])
a[0] = 5
print(a)

b = np.array([[1,2,3],[4,5,6]])
print(b.shape)
print(b[0, 0], b[0, 1], b[1, 0])
```

```
<class 'numpy.ndarray'>
//print(type(a)) in ra kiểu dữ liệu
của biến ra. Trong bài kết quả in ra
cho biết 'a' là một mảng của Numpy.
(3,) //print(a.shape) in ra độ dài của
mảng 'a'. Kết quả được in ra trong
bài cho biết độ dài của mảng 'a' là
3.
1 2 3 //print(a[0], a[1], a[2]) in ra các
giá trị trong mảng theo vị trí 0, 1, 2.
[5 2 3] //a[0]=5 thay đổi giá trị của
a[0]. Sau đó in ra các giá trị trong
mảng 'a' với a[0]=5, a[1]=2, a[2]=3.
(2, 3) //print(b.shape) in ra độ dài
của mảng 'b'. Kết quả được in ra
trong bài cho biết độ dài của mảng
'b' là ma trận 2x3.
1 2 4 //print(b[0, 0], b[0,1 ], b[1, 0])in
ra các giá trị trong ma trận b. Với b[0, 0]
= 1, b[0,1 ] = 2, b[1, 0] = 4.
```

```

import numpy as np
x = np.array([[1,2],[3,4]], dtype=np.float64)
y = np.array([[5,6],[7,8]], dtype=np.float64)
print(x + y)
print(np.add(x, y))
print(x - y)
print(np.subtract(x, y))
print(x * y)
print(np.multiply(x, y))
print(x / y)
print(np.divide(x, y))
print(np.sqrt(x))

```

```

//x=np.array([[1,2],[3,4]],dtype=np.float64) lúc này x=[[1. , 2. ] [3. , 4. ]].
y=np.array([[5,6],[7,8]],dtype=np.float64) lúc này y=[[5. , 6.] [7. , 8. ]]
[[ 6.  8.] [10. 12.]] //print(x+y) in ra kết quả x+y
[[ 6.  8.] [10. 12.]]
//print(np.add(x,y)) : in ra kết quả của phép cộng 2 mảng x,y
[[-4. -4.] [-4. -4.]] //print(x-y): in ra kết quả x-y
[[-4. -4.] [-4. -4.]]
//print(np.subtract(x,y)) : in ra kết quả phép trừ 2 mảng x,y
[[ 5. 12.] [21. 32.]] //print(x*y): in ra kết quả x*y
[[ 5. 12.] [21. 32.]]
//print(np.divide(x,y)) : in ra kết quả phép nhân 2 mảng x,y.
[[0.2 0.3333333] [0.42857143
0.5]]//print(x/y): in ra kết quả x/y
[[0.2 0.3333333] [0.42857143
0.5]]//print(np.divide(x,y)) : in ra kết quả phép chia 2 mảng x,y.

[[1. 1.41421356] [1.73205081 2.
]]//print(np.sqrt(x)): in ra kết quả căn bậc 2 của mảng x.

```

6.

Numpy

```

import numpy as np
x = np.array([[1,2],[3,4]])
y = np.array([[5,6],[7,8]])
v = np.array([9,10])
w = np.array([11, 12])
print(v.dot(w))
print(np.dot(v, w))
print(x.dot(v))
print(np.dot(x, v))
print(x.dot(y))
print(np.dot(x, y))

```

219 //v.dot(w) là tích vô hướng của 2 mảng 1 chiều v và w nên kết quả $219 = 9*11+10*12$

219 //np.dot(v,w) được hiểu là tích vô hướng của 2 mảng 1 chiều v,w nên $219 = 9*11+10+12$

[29 67] // mảng x là mảng 2 chiều, mảng v là mảng 1 chiều nên câu lệnh x.dot(v) được hiểu là 1 phép nhân ma trận. Kết quả là
 $[(1*9+2*10)(3*9+4*10)] = [29 67]$

[29 67]// mảng x là mảng 2 chiều, mảng v là mảng 1 chiều nên câu lệnh np.dot(x,v) được hiểu là 1 phép nhân ma trận. Kết quả là
 $[(1*9+2*10)(3*9+4*10)] = [29 67]$

[[19 22] [43 50]]// x và y là 2 mảng 2 chiều nên câu lệnh x.dot(y) là nhân 2

ma trận có kích thước 2×2 . Kết quả:
 $\begin{bmatrix} (1*5+2*7) & (1*6+2*8) \\ (3*5+4*7) & (3*6+4*8) \end{bmatrix} = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$
 $\boxed{\begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}}$ // x và y là 2 mảng 2
 chiều nên câu lệnh `np.dot(x,y)` là
 nhân 2 ma trận có kích thước 2×2 . Kết
 quả: $\begin{bmatrix} (1*5+2*7) & (1*6+2*8) \\ (3*5+4*7) & (3*6+4*8) \end{bmatrix} = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$

7.

Numpy

```
import numpy as np
v = np.array([1,2,3])
w = np.array([4,5])
print(np.reshape(v, (3, 1)) * w)

x = np.array([[1,2,3], [4,5,6]])
print(x + v)

print((x.T + w).T)

print(x + np.reshape(w, (2, 1)))

print(x * 2)
```

```
print(x + np.reshape(w, (2, 1)))
[[ 4  5]
 [ 8 10]
 [12 15]]
[[2 4 6]
 [5 7 9]]
[[ 5  6  7]
 [ 9 10 11]]
[[ 5  6  7]
 [ 9 10 11]]
[[ 2  4  6]
 [ 8 10 12]]
> |
```

Giải thích:

`np.reshape`: Cung cấp hình dạng mới cho
 một mảng mà không thay đổi dữ liệu của
 nó.

`np.array`: Khởi tạo mảng trên numpy

Khởi tạo hai mảng `v` và `w`
`print(np.reshape(v, (3, 1)) * w)`
 Thực hiện đổi kiểu mảng của `v` thành 1
 cột, 3 hàng rồi nhân với mảng `w`.
 Kết quả ra 3 hàng đầu trên hình.

Tiếp theo khởi tạo mảng `x`
`print(x + v)` thực hiện cộng hai mảng
 ra kết quả dòng 4 và 5.

`print((x.T + w).T)`: `x.T` là chuyển phép
 chuyển vị nó sẽ hoán đổi biến ma trận
 cột thành ma trận hàng và ngược lại.
`x` sẽ được chuyển thành ma trận cột
 song đó cộng với ma trận `w` rồi lại
 chuyển về ma trận hàng.

`print(x + np.reshape(w, (2, 1)))` là
 cộng mảng `x` với mảng `w` và mảng chuyển
 kiểu mảng thành 2 hàng, 1 cột. Kết
 quả ra dòng 8 và 9.

`print(x * 2)` là nhân các biến trong mảng `x` cho 2 ta được kết quả 2 dòng cuối.

8.

Pandas (Giới thiệu thư viện Pandas)

Pandas là một thư viện phân tích dữ liệu và làm việc với dữ liệu dạng bảng mạnh mẽ trong ngôn ngữ lập trình Python. Nó cung cấp cấu trúc dữ liệu và công cụ để làm việc với dữ liệu dạng bảng, giúp bạn thực hiện các tác vụ phân tích dữ liệu, xử lý dữ liệu, và chuẩn bị dữ liệu cho các mô hình máy học một cách dễ dàng. Một số đặc điểm quan trọng của thư viện Pandas:

- Đọc và Ghi Dữ Liệu
- Xử Lý Dữ Liệu Dạng Chuỗi
- Cộng Đồng Lớn và Phát Triển Liên Tục:

Phân Tích Dữ Liệu

Làm Việc với Dữ Liệu Thiếu

```
1 #Khai báo thư viện pandas và khai báo nó là pd
2 import pandas as pd
3 import numpy as np
4 data = np.array(['a', 'b', 'c', 'd'])
5 s = pd.Series(data, index=[100, 101, 102, 103])
6 print(s)
```

```
100    a
101    b
102    c
103    d
dtype: object
> |
```

import pandas as pd
import numpy as np
Hai dòng này khai báo thư viện pandas và đặt tên nó là pd và np.

data = np.array(['a', 'b', 'c', 'd'])
Tạo một mảng numpy chứa các phần tử 'a', 'b', 'c', 'd'.

s=pd.Series(data, index=[100, 101, 102, 103])
Tạo một đối tượng từ mảng numpy data. Các phần tử trong mảng sẽ trở thành dữ liệu của Series và các số nguyên trong danh sách index sẽ là chỉ mục của các phần tử tương ứng trong Series.

Cuối là in ra giá trị của đối tượng Series 's'.

```

import pandas as pd

read_data = pd.read_excel('BTPython.xlsx')
print(read_data)

```

	A	B	C
1	Ho va ten	MSSV	
2	Vu Tien Phat	21151309	
3	Hoang Thi Diem Quynh	21151326	
4	Ngo Xuan Tho	21119376	
5			

```

          Ho va ten      MSSV
0    Vu Tien Phat   21151309
1  Hoang Thi Diem Quynh  21151326
2    Ngo Xuan Tho  21119376
PS C:\Users\defaultuser0.LAPTOP-POTNR7PE\OneDrive - hcmute.edu.vn\Desktop\LAB> 

```

```

//import pandas as pd: khai báo thư
viện pandas là pd
//read_data
=pd.read_excel('BTPython.xlsx'): đọc dữ liệu
file excel qua câu lệnh pd.read_excel sau đó
gán dữ liệu đã đọc vào biến read_data.
//print(read_data): in biến read_data

```

9.

Đọc file txt

```

with open('BT.txt', 'r') as file:
    BaiTap = file.read()
print(BaiTap)

```

File txt:

```

Vu Tien Phat - 21151309
Hoang Thi Diem Quynh - 21151326
Ngo Xuan Tho - 21119376

```

***Kết quả:**

```

Vu Tien Phat - 21151309
Hoang Thi Diem Quynh - 21151326
Ngo Xuan Tho - 21119376
PS C:\Users\defaultuser0.LAPTOP-POTNR7PE\OneDrive - hcmute.edu.vn\Desktop\LAB> 

```

```

//with open('BT.txt','r') as file:
BaiTap=file.read(): với mode r có chức
năng chỉ mở để đọc file BT.txt sau đó
gán các biến đã đọc vào "file". Đọc
và gán toàn bộ biến ở "file" vào
BaiTap
//print(BaiTap): in chuỗi BaiTap

```

10.
Group photo

