Lab 4: Python cơ bản

Yêu cầu:

- **Laptop đã cài đặt Python**

Thực hiện: kiểm tra phiên bản version trên máy tnh cá nhân, mở Window + R gõ cmd nhập "py"hoặc "python" hoặc "python -- version" kiểm tra như hình dưới thì đã cài đặt Python

```
Microsoft Windows [Version 10.0.19044.1741]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>py
Python 3.10.4 (tags/v3.10.4:9d38120, Mar 23 2022, 23:13:41) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()

C:\Users\Admin>python
Python 3.10.4 (tags/v3.10.4:9d38120, Mar 23 2022, 23:13:41) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()

C:\Users\Admin>python --version
Python 3.10.4

C:\Users\Admin>
```

Để thoát khỏi mode python sử dụng lệnh quit() hoặc exit() hoặc nhấn ctrl+Z

>>> **quit()/ exit()> Enter để thoát**

Hướng dẫn:

Phần 1: Phép toán tử cơ bản và hàm print()

Operation	Math	Syntax
Cộng	a+b	a+b
Trừ	a-b	a-b
Nhân	axb	a*b
Chia	a÷b	a/b

Mũ	a^b	a**b

Ví dụ 1: số phép toán sử dụng cú pháp của python

```
>>> 2+3
5
>>> 10-4
6
>>> 2*4
8
>>> 20/5
4.0
>>> 3**2
9
```

- **Chuỗi** là tập hợp các **ký tự bất kỳ**, chẳng hạn như **chữ cái**, **số**, **ký** hiệu **hoặc dấu chấm câu**. Trình thông dịch tương tác sẽ trực tiếp xuất văn bản mà bạn nhập dưới dạng chuỗi với điều kiện bạn kèm theo chuỗi trong dấu nháy đơn (') hoặc dấu nháy kép (").
- Sử dụng hàm **print()** để in chuỗi ký tự ra màn hình người dùng.

```
>>> "Hello World!"
'Hello World!'
>>> 'Hello World!'
'Hello World!'
>>> print("Hello World!")
Hello World!
>>> text=("Hello World!")
```

```
>>> print(text)
Hello World!
```

- Có thể ghép 2 hay nhiều chuỗi lại với nhau như cú pháp sau:

```
>>> str1="Cisco"
>>> str2="Networking"
>>> str3="Academy"
>>> space=" "
>>> print(str1+space+str2+space+str3)
Cisco Networking Academy
```

- Khi một chuỗi ký tự nhân (*) với một số n thì sẽ in ra chuỗi đó với n lần.

```
>>> x=3
>>> x*5
15
>>> "Cisco"*x
'CiscoCiscoCisco'
```

Phần 2: Các loại dữ liệu, biến.

- **Integer** – **Số nguyên**. VD: 1,3,5,7,….
- **Float** – **Số thực**. VD: 3.14159.
- **String** – **Chuỗi ký tự**. VD: ‘Hello World’
- **Boolean** – **Toán tử logic** gồm 2 giá trị là **True - False.**

Có thể dùng hàm **type**() để xác định loại dữ liệu của biến đó.

```
>>> type(98)
<class 'int'>
>>> type(98.6)
<class 'float'>
>>> type("Hi!")
<class 'str'>
>>> type(True)
<class 'bool'>
```

Phần 3: Các phép toán so sánh:

Dấu	Ý nghĩa
>	Lớn hơn

Dấu	Ý nghĩa
<	Bé hơn
==	Bằng
!=	Không bằng
>=	Lớn hơn hoặc bằng
<=	Bé hơn hoặc bằng

Ví dụ:

```
>>> 1<2
True
>>> 1>2
False
>>> 1==1
True
>>> 1!=1
False
>>> 1>=1
True
>>> 1<=1
True
```

- Ép kiểu dữ liệu từ loại dữ liệu này sang dữ liệu khác

Ví dụ:

```
>>> x=3
```

```
>>> print("The value of X is " + x)
Traceback (most recent call last):
 File "<pyshell#27>", line 1, in <module>
 Print("The value of X is " + x)
TypeError: must be str, not int
```

- Sử dụng hàm **str()** để biến đổi dữ liệu thành kiểu string:

Ví dụ:

```
>>> print("The value of x is " + str(x))
The value of x is 3
>>> type(x)
<class 'int'>
```

Phần 4: List và Dictionary

List sẽ dùng dấu ngoặc vuông [ ] và chứa các item bên trong, có thể dùng type() để kiểm tra loại dữ liệu, len() để kiểm tra số lượng phần tử bên trong List.

Ví dụ:

```
>>> hostnames=["R1","R2","R3","S1","S2"]
>>> type(hostnames)
<class 'list'>
>>> len(hostnames)
5
>>> hostnames
['R1', 'R2', 'R3', 'S1', 'S2']
```

- Giá trị cuối của List sẽ là [-1]

Dictionary sẽ dùng dấu {} để chứa các cặp key:value bên trong, các cặp key:value được cách nhau bởi dấu phẩy (,), và key:value có dấu nháy kép (“) nghĩa là đó là string

Ví dụ:

```
>>> ipAddress =
{"R1":"10.1.1.1","R2":"10.2.2.1","R3":"10.3.3.1"}
>>> type(ipAddress)
<class 'dict'>
```

- Không giống như List là dùng vị trí của item để gọi giá trị ra, Dictionary sẽ sử dụng key để gọi giá trị value. Có thể thêm cặp key:value bằng cách tự định nghĩa key mới bằng value mới.

```
>>> ipAddress
{'R1': '10.1.1.1', 'R2': '10.2.2.1', 'R3':
'10.3.3.1'}
>>> ipAddress["R1"]
'10.1.1.1’
>>> ipAddress["S1"]="10.1.1.10"
>>> ipAddress
{'R1': '10.1.1.1', 'R2': '10.2.2.1', 'R3':
'10.3.3.1', 'S1': '10.1.1.10'}
>>> "R3" in ipAddress
True
```

Phần 5: Input, Output của người dùng

Để có thể nhận giá trị input của người dùng nhập vào ta dùng hàm input(), để lấy giá trị output sử dụng hàm print()

```
>>> firstName = input("What is your first name? ")
```

```
What is your first name? Bob
>>> print("Hello " + firstName +"!")
Hello Bob!
```

Phần 6: Vòng lập for

Khi có một List gồm n phần tử bên trong, để có thể in ra một phần tử trong đó ta sẽ dùng lệnh *print(x)*. Ở đây **x** sẽ là vị trí của phần tử muốn in ra. Vậy để in hết tất cả các phần tử bên trong đó ta sẽ sử dụng vòng lặp **for.**

Ví dụ 1: Sử dụng for để duyệt danh sách

Bên dưới đây ta sẽ khai báo một danh sách **devices** gồm những phần tử lần lượt là **"R1","R2","R3","S1","S2"** sau đó ta sử dụng vong lặp for để duyệt những phần tử bên trong danh sách devices.

```
>>> devices=["R1","R2","R3","S1","S2"]
>>> for item in devices:
        print(item)



R1
R2
R3
S1
S2
```

Ở đây **item** là một biến có thể tự đặt được, **item** sẽ được gán giá trị lần lượt là R1, R2, R3,…