Buổi 1. Giới thiệu Python

1 Thông tin chung

Mục tiêu buổi học

- Giới thiệu ngôn ngữ lập trình Python
- Hướng dẫn cài đặt các ví dụ cơ bản về lập trình Python: biến, các kiểu dữ liệu, hàm, ...

Kiến thức và kỹ năng đạt được

- Nắm vững các khái niệm trong lập trình Python.
- Áp dụng cài đặt được các bài tập thực hành.

Công cụ thực hành

- Ngôn ngữ lập trình: Python
- Công cụ thực hành: Anaconda, colab

Thời gian thực hành: 3 tiết

2 Nội dung lý thuyết

Giới thiệu

Python là một ngôn ngữ lập trình cấp cao do Guido van Rossum tạo ra năm 1990. Đây là một trong các ngôn ngữ lập trình phổ biến nhất thế giới.

Một số ưu điểm của Python:

- Mã nguồn đơn giản, dễ hiểu
- Kiểu dữ liệu linh động
- Hỗ trợ nhiều thư viện xử lý toán học, dữ liệu, xử lý ảnh, ...

Lịch sử phát triển của ngôn ngữ Python

- Python 1: 1994
- Python 2: 2000
- Python 3: 2008

Công cụ lập trình

- Tự cài đặt Python https://www.python.org
- Anaconda https://www.anaconda.com/
- Google Colab (online) https://colab.research.google.com/

3 Nội dung thực hành

3.1 Biến, kiểu dữ liệu

- Trong ngôn ngữ Python, biến không cần khai báo kiểu dữ liệu trước khi sử dụng. Biến có thể thay đổi kiểu dữ liệu động tùy theo giá trị được gán.
- Kiểu dữ liệu nguyên thủy:
  - NoneType: None
  - bool: True, False
  - số: số nguyên, số thực, số phức
- Kiểu trừu tượng lưu trữ dữ liệu *(container)*: chuỗi, danh sách, tập hợp, ...

[1]:
```
# Ví dụ. In thông báo ra màn hình
print("Hello World")
```

```
Hello World
```

[2]:
```
# Ví dụ. In thông báo ra màn hình
msg = "Hello World"
print(msg)
```

```
Hello World
```

[3]:
```
# Ví dụ. Tính tổng 2 số
a = 5
b = 0.5
c = a + b

print("c = ", c)
c
```

```
c =  5.5
```

[3]: 5.5

[4]:
```
# Ví dụ. Tính tổng 2 số
a, b = 5, 0.5
c = a + b

print("c = ", c)
c
```

```
c =  5.5
```

[4]: 5.5

[5]:
```
# Ví dụ. Hoán vị 2 số a và b
a, b = 5, 10
print(a, b)

a, b = b, a
print(a, b)
```

```
5 10
10 5
```

[6]:
```
# Ví dụ. Biến thay đổi kiểu dữ liệu động tùy theo giá trị được gán.
var = 10
print(type(var))

var = 0.5
print(type(var))

var = "text"
print(type(var))

var = "text"
print(type(False))
```

```
<class 'int'>
<class 'float'>
<class 'str'>
<class 'bool'>
```

3.2 Các cấu trúc điều khiển

3.2.1 Cấu trúc rẽ nhánh

[7]:
```
# Ví dụ. Kiểm tra n là số chẵn hay lẻ
n = 1
if n % 2 == 0:
    print("even")
else:
    print("odd")
```

```
odd
```

[8]:
```
# Ví dụ. Tìm số lớn nhất giữa 3 số a, b, c
a, b, c = 5, 10, 1
max = a
if max < b:
    max = b
if max < c:
    max = c
```

```
max
```

[8]: 10

3.2.2 Cấu trúc lặp

[9]:
```
sum = 0
n = 5
for i in range(1, n + 1):
    sum = sum + i
    print(i)
print("sum =", sum)
```

```
1
2
3
4
5
sum = 15
```

[10]:
```
# Ví dụ. Đếm các số chẵn trong n số nguyên dương đầu tiên.
count = 0
n = 10

i = 1
while i != n :
    if i % 2 == 0:
        count = count + 1
    i = i + 1

print(count)
```

```
4
```

3.3 Kiểu chuỗi

[11]:
```
# Ví dụ. Một số thao tác với kiểu chuỗi.
greeting = "Hello world!"

#greeting.lower()
#greeting.title()
#greeting.strip()
greeting.strip("!")
```

[11]: 'Hello world'

3.4 Các kiểu dữ liệu trừu tượng

3.4.1 List

Trong Python, danh sách *(list)* là một cấu trúc dữ liệu có thể lưu trữ các phần tử có nhiều kiểu dữ liệu khác nhau. Một danh sách có thể chứa một hay nhiều danh sách con.

[12]:
```
# Ví dụ. Khai báo và khởi tạo danh sách
empty = []
letters = ['a', 'b', 'c', 'd']
numbers = [1, 3, 5, 7, 9]

print(empty)
print(letters)
print(numbers)

# Danh sách có thể chứa nhiều kiểu dữ liệu khác nhau
mixed = [1, 2, 0.7, "text", True]
print(mixed)

# Tạo danh sách từ chuỗi
msg = "Hello World"
lst = msg.split()
lst

# Tạo chuỗi từ danh sách
new_msg = " ".join(lst) # ký tự khoảng trắng
new_msg
```

```
[]
['a', 'b', 'c', 'd']
[1, 3, 5, 7, 9]
[1, 2, 0.7, 'text', True]
```

[12]:
```
'Hello World'
```

[13]:
```
# Ví dụ. Danh sách chứa nhiều danh sách con
lst = [[0, 2, 4, 6, 8], [1, 3, 5, 7, 9]]
print(lst)
```

```
[[0, 2, 4, 6, 8], [1, 3, 5, 7, 9]]
```

Truy xuất phần tử trong danh sách

Sử dụng toán tử [] để truy xuất trực tiếp đến phần tử của danh sách.

- [i] dựa vào chỉ mục/thứ tự phần tử thứ i
- [i:] lấy phần tử từ i đến cuối danh sách
- [:j] lấy phần tử từ j-1 đến đầu danh sách
- [i : j] lấy phần tử từ vị trí i đến vị trí j - 1

[14]:
```python
# Ví dụ. Truy xuất phần tử
mixed = [1, 2, 0.7, "text", True]
print(mixed[0])
print(mixed[2])
print(mixed[-2])

# 0, 1, 2, 3
pets = ["tom", "jerry", "spike", "donald"]
print(pets[1:])
print(pets[:2])
print(pets[1:3])

for p in pets:
    print(p)
```

```
1
0.7
text
['jerry', 'spike', 'donald']
['tom', 'jerry']
['jerry', 'spike']
tom
jerry
spike
donald
```

Các thao tác với danh sách

Python cung cấp rất nhiều hàm xử lý kiểu dữ liệu danh sách:

- append: thêm một phần tử vào cuối danh sách.
- insert: thêm một phần tử vào vị trí i xác định.
- remove: xóa một phần tử khỏi danh sách.
- clear: xóa danh sách.
- extend: thêm danh sách l2 vào danh sách l1.
- sort: sắp xếp thứ tự danh sách.
- reverse: đảo ngược thứ tự các phần tử trong danh sách.
- count: đếm số lần xuất hiện của một phần tử.

[15]:
```python
# Ví dụ. Các thao tác với danh sách trong Python
pets = ["jerry", "tom", "spike"]
print(pets)

pets.append("donald")
print(pets)

pets.sort()
print(pets)
```

```
['jerry', 'tom', 'spike']
['jerry', 'tom', 'spike', 'donald']
['donald', 'jerry', 'spike', 'tom']
```

3.4.2 Tuple

Bộ *(tuple)* là dãy các phần tử liên tiếp nhau. Giá trị của các phần tử này không được thay đổi.

[16]:
```
fruits = ('apple', 'banana', 'cherry', 'orange', 'kiwi')

print(fruits)
```

```
('apple', 'banana', 'cherry', 'orange', 'kiwi')
```

Truy xuất phần tử

[17]:
```
print(fruits[1])
print(fruits[-1])
print(fruits[2:5])

for t in fruits:
    print(t)
```

```
banana
kiwi
('cherry', 'orange', 'kiwi')
apple
banana
cherry
orange
kiwi
```

- Để thay đổi giá trị của tuple, ta cần chuyển tuple thành list để thay đổi dữ liệu. Sau đó, thực hiện thao tác chuyển ngược lại từ list thành tuple.

[18]:
```
l = list(fruits)
print('list:\n', l)

l[2] = 'berry'
fruits = tuple(l)
print('tuple:\n', fruits)
```

```
list:
 ['apple', 'banana', 'cherry', 'orange', 'kiwi']
tuple:
 ('apple', 'banana', 'berry', 'orange', 'kiwi')
```

3.4.3 Dictionary

Từ điển *(dictionary)* là dữ liệu được mô tả bởi nhiều cặp khóa-giá trị.

[19]:
```
pets = { 'name' : 'Tom', 'age' : 10, 'gender' : True , 'weight' : 10.5 }

print ( pets )
print ( pets['name'])
```

```
{'name': 'Tom', 'age': 10, 'gender': True, 'weight': 10.5}
Tom
```

3.4.4 Set

Tập hợp *(set)* là dãy các phần tử không trùng nhau

[20]:
```
a = { 1, 3, 5, 7, 9 }
b = { 1, 2, 4, 6, 8, 10}

print (a | b)
print (a & b)
print (a - b)
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
{1}
{9, 3, 5, 7}
```

3.5 Hàm trong Python

[21]:
```
# Ví dụ. Hàm hiển thị thông điệp Hello World
def show():
    print("Hello World")
show()
```

```
Hello World
```

[22]:
```
# Ví dụ. Hàm tính tổng n số nguyên dương đầu tiên.
def sum(n):
    s = 0
    for i in range(1, n + 1):
        s = s + i
    return s

n = 5
sum(n)
```

[22]: 15

[23]:
```
# Ví dụ. Hàm đệ quy tính n!
def factorial(n):
    if n == 0:
        return 1
```

```
    return n * factorial(n - 1)

n = 3
factorial(n)
```

[23]: 6

4 Bài tập

- Nhập vào một năm, viết hàm cho biết năm đó thuộc thế kỷ thứ mấy?
- In số lớn nhất trong ba số a, b và c.
- Nhập vào một số nguyên dương n từ 0-99 và in ra số n bằng chữ. Ví dụ: n = 27 in ra màn hình dòng chữ "Hai muoi bay".
- Nhập vào một mật khẩu, kiểm tra mật khẩu đó có an toàn hay không? Mật khẩu an toàn phải thỏa các điều kiện: chiều dài từ 8-20 ký tự; phải có chữ hoa, chữ thường và số.
- Mật mã Caesar là một mật mã dịch chuyển. Mỗi ký tự trong bản rõ được thay thế bằng một ký tự cách nó một đoạn trong bảng chữ cái để tạo thành bản mã. Giả sử chọn n = 3, ký tự A sẽ được thay bằng D. Cài đặt hàm mã hóa và giải mã chuỗi ký tự nhập từ bàn phím.