Chương 2: NHỮNG VẤN ĐỀ CƠ BẢN CỦA PYTHON

----------//----------

1. Giới thiệu chung

Python là một ngôn ngữ lập trình bậc cao cho các mục đích lập trình đa năng, do Guido van Rossum tạo ra và lần đầu ra mắt vào năm 1991. Python được thiết kế với ưu điểm mạnh là dễ đọc, dễ học và dễ nhớ. Python là ngôn ngữ có hình thức rất sáng sủa, cấu trúc rõ ràng, thuận tiện cho người mới học lập trình

Có 2 phiên bản Python đang được sử dụng phổ biến ở thời điểm hiện tại là Python 2 và Python 3. Về cơ bản, Python 2 và Python 3 có cách viết giống nhau, chỉ khác nhau ở một số thư viện hỗ trợ riêng cho Python 2 hoặc Python 3.

Có rất nhiều cách để viết chương trình Python như phần mềm:

1. Python IDE
2. Visual Studio Code
3. Jupyter notebook- offline
4. ….

Các trình biên dịch online:

1. Datacamp
2. Google Colab- online
3. w3schools
4. …

Các ví dụ trong bài sử dụng Python 3 nền tảng online trên Google Colab và offline với phần mềm Jupyter Notebook để biên dịch.

Chương trình đầu tiên, hello world!

Chương trình đầu tiên khi làm quen với Python là sử dụng lệnh ***print*** in một chuỗi ký tự ra màn hình. Lệnh ***print*** có cú pháp như sau:

```
print(object(s), sep=separator, end=end, file=file, flush=flush)
```

Trong đó:

object(s)	Bất kỳ đối tượng nào, số lượng tùy ý. Nội dung bên trong sẽ được chuyển sang định dạng chuỗi trước khi in ra màn hình.
sep='*separator*'	Không bắt buộc. Định nghĩa cách tách đối tượng nếu có nhiều hơn một đối tượng được in. Mặc định là ‘ ’ (khoảng cách)
end='*end*'	Không bắt buộc. Chỉ ra ký tự được in khi kết thúc. Mặc đinh là ‘\n’ (xuống dòng)
file	Không bắt buộc. Một đối tượng với một phương thức ghi. Mặc định là sys.stdout
flush	Không bắt buộc. Một Boolean, chỉ định xem ngõ ra được giải phóng (True) hay được lưu vào bộ đệm (False). Mặc định là ‘False’

Sinh viên soạn thảo câu lệnh sau trong Python:

```
print("hello world")
```

Chạy chương trình ta được kết quả:

```
hello world
```

Lệnh ***print*** trong Python sử dụng định dạng chuỗi tương tự như ngôn ngữ C. Các định dạng cơ bản bao gồm:

- %s – Định dạng chuỗi ký tự (String)
- %d – Định dạng số nguyên (Integer)
- %f – Định dạng số thực (Float)
- %.<number of digits>f – Định dạng số thực với số lượng ký tự số sau dấu chấm thập phân cố định.
- %x/%X – Định dạng số Hex (%x: Dạng hex viết thường/ %X: Dạng HEX viết hoa)

Ví dụ minh họa:

```
# This prints out "Hello, John!"
name = "John"
print("Hello, %s!" % name)
```

```
Hello, John!
```

```
# This prints out "John is 23 years old."
name = "John"
age = 23
print("%s is %d years old." % (name, age))
```

```
John is 23 years old.
```

```
a = 20.0
b = 3.0
c = a/b
print("The result is: %d" % (c))
print("The result is: %f" % (c))
print("The result is: %.3f" % (c))
```

```
The result is: 6
The result is: 6.666667
The result is: 6.667
```

2. Biến và kiểu dữ liệu

Python hoàn toàn là hướng đối tượng. Các biến trong Python không cần khai báo trước khi sử dụng hoặc khai báo kiểu biến như trong các ngôn ngữ khác. Mọi biến trong Python là một đối tượng.

2.1 Số nguyên:

```
myint = 7
print(myint)
```

```
7
```

2.2 Số thực:

```
myfloat = 7.0
print(myfloat)
myfloat = float(7)
print(myfloat)
```

```
7.0
7.0
```

2.3 Chuỗi:

Có thể khai báo trong dấu nháy đơn hoặc dấu nháy kép:

```
mystring = 'hello'
print(mystring)
mystring = "hello"
print(mystring)
```

```
hello
hello
```

Khi muốn khai báo dấu đơn hoặc dấu nháy kép trong đoạn văn, thì toàn bộ nội dung sẽ nằm trong dấu nháy đơn/kép còn lại:

```
mystring = 'Don"t worry about apostrophes'
print(mystring)
mystring = "Don't worry about apostrophes"
print(mystring)
```

```
Don"t worry about apostrophes
Don't worry about apostrophes
```

+ Các toán tử đơn giản có thể thực hiện trên số và chuỗi:

```
one = 1
two = 2
three = one + two
print(three)

hello = "hello"
world = "world"
helloworld = hello + " " + world
print(helloworld)
```

```
3
```

```
hello world
```

+ Phép gán có thể thực hiện với nhiều biến lần lượt như sau:

```
a, b = 3, 4
print(a,b)
```

```
3 4
```

Không thể thực hiện ghép toán trên cả số và chuỗi. Ví dụ:

```
# This will not work!
one = 1
two = 2
hello = "hello"
print(one + two + hello)
```

```
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Các phép toán trên chuỗi

+ Đếm số ký tự trong chuỗi:

```
astring = "Hello world!"
print(len(astring))
```

```
12
```

+Xác định vị trí ký tự trong chuỗi

```
astring = "Hello world!"
print(astring.find('o')
```

```
4
```

Kết quả trả về là 4 vì thứ tự chuỗi trong Python bắt đầu từ 0 và hàm này chỉ trả về một kết quả đầu tiên.

+Đếm số ký tự trong chuỗi:

```
astring = "Hello world!"
print(astring.count("l"))
```

```
3
```

Lệnh đếm ký tự "l" trong chuỗi "Hello world!". Kết quả trả về là 3, tức là có 3 ký tự "l" trong chuỗi.

+Cắt chuỗi tại vị trí cho trước

```
astring = "Hello world!"
print(astring[3:7])
```

```
lo w
```

Lệnh này sẽ cắt chuỗi tại vị trí đầu tiên (3) đến vị trí cuối -1 (6)

```
astring = "Hello world!"
print(astring[3:7])
print(astring[3:7:1])
print(astring[3:7:2])
```

```
lo w
lo w
l
```

Lệnh [3:7:2] có nghĩa là lấy từ 3 đến 7, cách 2 bước. *[start:stop:step]*.

Index âm/ index ngược (negative indexing)

Python cho phép chỉ định index số âm trong 1 danh sách, tức là đi chiều ngược từ phải qua trái với từng index trong list.

Ví dụ.

```
data = ('a','b','c')
print(data[-1])
print(data[0])
print(data[1])
```

```
C
a
b
```

+Tạo chuỗi ngược.

```
astring = "Hello world!"
print(astring[::-1])
```

```
!dlrow olleH
```

+ Chuyển đổi ký tự thành ký tự in hoa/ ký tự thường

```
astring = "Hello world!"
print(astring.upper())
print(astring.lower())
```

```
HELLO WORLD!
hello world!
```

+Kiểm tra bắt đầu và kết thúc chuỗi có đúng “từ khóa” không

```
astring = "Hello world!"
print(astring.startswith("Hello"))
print(astring.endswith("asdfasdfasdf"))
```

```
True
False
```

+ chia chuỗi theo điều kiện.

```
astring = "Hello world!"
afewwords = astring.split(" ")
print(afewwords)
```

```
['Hello', 'world!']
```

2.4 Toán tử

Giống như bất kỳ trình biên dịch khác, Python cũng có các phép toán cộng, trừ, nhân, chia. Thứ tự thực hiện phép toán cũng giống như các phép toán thông thường thực hiện.

+ Toán tử

Toán tử (Operator)	Tên phép toán	Ví dụ
+	Cộng	x + y
-	Trừ	x - y
*	Nhân	x * y
/	Chia	x / y
%	Chia lấy dư	x % y
**	Lấy lũy thừa	x ** y
//	Chia lấy phần nguyên	x // y

+ Toán tử gán

Toán từ gán (Assignment Operators)	Ví dụ	Ý nghĩa
=	x = 10	Gán giá trị 10 vào biến x
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

+ Toán tử thao tác bit (Bitwise Operators):

Toán tử	Ý nghĩa	Ý nghĩa
&	AND	Lệnh AND các bit

|	OR	Lệnh OR các bit
^	XOR	Lệnh XOR các bit
~	NOT	Đảo bit
<<	Dịch sang trái	Dịch trái, thêm zero vào bên phải
>>	Dịch sang phải	Dịch phải, thêm giá trị giống bit ngoài cùng bên trái vào bên trái

Ví dụ

```
number = 1 + 2 * 3 / 4.0
print(number)
```

```
2.5
```

```
remainder = 11 % 3
print(remainder)
```

```
2
```

```
squared = 7 ** 2
cubed = 2 ** 3
print(squared)
print(cubed)
```

```
49
8
```

```
helloworld = "hello" + " " + "world"
print(helloworld)
```

```
hello world
```

```
lotsofhellos = "hello" * 10
print(lotsofhellos)
```

```
hellohellohellohellohellohellohellohellohellohello
```

+ Toán tử điều kiện

Toán tử điều kiện	Ý nghĩa	Ví dụ
==	Bằng	x == y
!=	Không bằng	x != y
>	Lớn hơn	x > y
<	Nhỏ hơn	x < y
>=	Lớn hơn hoặc bằng	x >= y
<=	Nhỏ hơn hoặc bằng	x <= y

Python sử dụng các biến boolean để đánh giá các điều kiện. Các giá trị boolean sẽ là True/False được trả về khi một biểu thức được so sánh hoặc đánh giá. Ví dụ:

```
x = 2
print(x == 2) # prints out True
print(x == 3) # prints out False
print(x < 3) # prints out True
```

```
True
False
True
```

Lưu ý rằng việc gán biến được thực hiện bằng toán tử dấu bằng đơn "=", trong khi việc so sánh giữa hai biến được thực hiện bằng toán tử dấu bằng kép "==". Toán tử "không bằng" được đánh dấu là "! =".

+ Toán tử logic (Logical Operators)

Toán tử logic	Ý nghĩa	Ví dụ
and	Trả về **True** nếu cả 2 mệnh đề là đúng	x < 5 and x < 10
or	Trả về **True** nếu một trong 2 mệnh đề là đúng	x < 5 or x < 4
not	Trả về **False** nếu mệnh đề đúng	not(x < 5 and x < 10)

Ví dụ minh họa

```
name = "John"
age = 23
if name == "John" and age == 23:
    print("Your name is John, and you are also 23 years old."
)

if name == "John" or name == "Rick":
    print("Your name is either John or Rick.")
```

```
Your name is John, and you are also 23 years old.
Your name is either John or Rick.
```

```
name = "John"
if name in ["John", "Rick"]:
    print("Your name is either John or Rick.")
```

```
Your name is either John or Rick.
```

```
print(not False) # Prints out True
print((not False) == (False)) # Prints out False
```

```
True
False
```

+ Toán từ định danh (Identity Operators) được dùng để xác định xem hai biến có đang trỏ tới cùng một đối tượng hay không. Với các kiểu dữ liệu như int, str, float,... thì toán tử này tương đương với toán tử ==

Toán từ định danh	Ý nghĩa	Ví dụ
is	Trả về **True** nếu cả 2 cùng đối tượng	x is y
is not	Trả về **True** nếu 2 biến không cùng đối tượng	x is not y

```
a = 6
b = 6
print(a is b)
```

```
True
```

```
statement = False
another_statement = True
if statement is True:
    print("do something")
elif another_statement is True: # else if
    print("do something else")
else:
    print("do another thing")
```

```
do something else
```

+Toán tử Membership (Membership Operators)

in	Trả về **True** nếu một chuỗi với giá trị được chỉ định có trong đối tượng	x in y
not in	Trả về **True** nếu một chuỗi với giá trị được chỉ định không có trong đối tượng	x not in y

Python sử dụng dấu ":" (2 chấm) và các thụt đầu dòng cho các khối thay vì sử dụng dấu ngoặc nhọn như trong C. Có thể sử dụng Tab hoặc Space đều được. Lưu ý phải thống nhất các thụt đầu dòng cho toàn bộ chương trình. Ví dụ:

```
x = 1
if x == 1:
  # indented two spaces
  print("x is 1.")
```

```
x is 1.
```

```
x = 1
if x == 1:
  # indented one tab
  print("x is 1.")
```

```
x is 1.
```

```
x = 1
if x == 1:
    # indented four spaces
    print("x is 1.")
```

```
x is 1.
```

2.5 Cấu trúc if - else

Cấu trúc chỉ có if:

```
if(điều kiện):
    Các câu lệnh nếu điều kiện đúng
    …
Lệnh này không nằm trong if
…
```

Cấu trúc if-else

```
if(điều kiện):
    Các câu lệnh nếu điều kiện đúng
    …
else:
    Các câu lệnh nếu điều kiện không đúng
    …
Lệnh này không nằm trong if-else
…
```

Cấu trúc if-elif-else

```
if(điều kiện 1):
    Các câu lệnh nếu điều kiện 1 đúng
    …
elif(điều kiện 2):
    Các câu lệnh nếu điều kiện 2 đúng
    …
else:
    Các câu lệnh nếu các điều kiện trên không đúng
    …
Lệnh này không nằm trong if-elif-else
```

Python hỗ trợ các điều kiện logic thông thường từ toán học:

- Bằng: a == b
- Không bằng: a! = b
- Nhỏ hơn: a <b
- Nhỏ hơn hoặc bằng: a <= b
- Lớn hơn: a> b
- Lớn hơn hoặc bằng: a> = b

Các điều kiện này có thể được sử dụng theo một số cách, phổ biến nhất là trong "câu lệnh if" và vòng lặp.

Một "câu lệnh if" được viết bằng cách sử dụng từ khóa if.

```
a = 33
b = 200
if b > a:
  print("b is greater than a")
```

```
b is greater than a
```

+elif

Từ khóa elif nghĩa là "nếu điều kiện trước đó không đúng, thì hãy thử điều kiện này".

```
a = 33
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
```

```
a and b are equal
```

+else: các điều kiện còn lại

```
x = 2
if x == 2:
    print("x equals two!")
else:
    print("x does not equal to two.")
```

```
x equals two!
```

2.6 Cấu trúc lặp

Cú pháp:

+ Vòng lặp for:

Vòng lặp **for** được sử dụng để lặp qua một chuỗi liên tục (đó là một danh sách, một bộ, một từ điển, một tập hợp hoặc một chuỗi). Cú pháp:

```
for [biến lặp lại] in [Mảng]:
   Các câu lệnh
```

Ví dụ có một mảng primes gồm 4 phần tử, thực hiện vòng lặp for in lần lượt các phần tử trong mảng.

```
primes = [2, 3, 5, 7]
for prime in primes:
    print(prime)
```

```
2
3
5
7
```

Vòng lặp for không yêu cầu đặt trước biến lập chỉ mục.

```
# Prints out the numbers 0,1,2,3,4
for x in range(5):
    print(x)
# Prints out 3,4,5
for x in range(3, 6):
    print(x)
# Prints out 3,5,7
for x in range(3, 8, 2):
    print(x)
```

```
0
1
2
3
4
3
4
5
3
5
7
```

+ Lặp qua một chuỗi

```
for x in "banana":
  print(x)
```

```
b
a
n
a
n
a
```

+Vòng lặp while:

Cú pháp:

```
While [điều kiện]:
   Lệnh A
   Lệnh B
```

```
# Prints out 0,1,2,3,4
count = 0
while count < 5:
    print(count)
    count += 1  # This is the same as count = count + 1
```

```
0
1
2
3
4
```

+ Break và continue:

```
# Prints out 0,1,2,3,4
count = 0
while True:
    print(count)
    count += 1
    if count >= 5:
        break

# Prints out only odd numbers - 1,3,5,7,9
for x in range(10):
    # Check if x is even
    if x % 2 == 0:
        continue
    print(x)
```

```
0
1
2
3
4
1
3
5
7
```

```
9
```

Với câu lệnh break, chúng ta có thể dừng vòng lặp trước khi nó lặp qua tất cả các mục:

Với câu lệnh continue, chúng ta có thể dừng lần lặp hiện tại của vòng lặp và tiếp tục với phần tiếp theo:

+Từ khóa else ở vòng lặp chỉ định một khối mã sẽ được thực thi khi vòng lặp kết thúc

```
# Prints out 0,1,2,3,4 and then it prints "count value reache
d 5"

count=0
while(count<5):
    print(count)
    count +=1
else:
    print("count value reached %d" %(count))
```

```
0
1
2
3
4
count value reached 5
```

```
# Prints out 1,2,3,4
for i in range(1, 10):
    if(i%5==0):
        break
    print(i)
else:
    print("this is not printed because for loop is terminated
 because of break but not due to fail in condition")
```

```
1
2
3
4
```

```
# Prints out 1,2,3,4
for i in range(1, 5):
    print(i)
else:
    print("this line will be printed")
```

```
1
2
```

```
3
4
this line will be printed
```

Bài tập Biến và kiểu dữ liệu

1. Sử dụng lệnh print, in dòng chữ “Good morning” lên màn hình.
2. Tạo các biến với giá trị là tên SV và MSSV. Sử dụng lênh print, in tên và MSSV lên màn hình.
3. Tạo ra các biến dạng số nguyên, thực hiện các phép toán ‘+’, ‘-’, ‘*’, ‘/’ và in các kết quả đó ra màn hình.

5. Viết chương trình Python in ra giá trị họ tên ngược lại giống như sau:

- Data = (‘họ’,’tên đệm’,’tên’)
- In ra màn hình kết quả: ‘tên’ ‘tên đệm’ ‘họ’

Bài tập phần 3

6. Viết chương trình Python tính khoảng cách từ điểm (x1,y1) đến điểm (x2,y2)

7. Viết chương trình Python tính tổng các phần tử số trong mảng cho trước gồm cả ký tự và số.

8. Viết chương trình Python tìm số lớn nhất trong một dãy số cho trước.

9. Viết chương trình Python tìm số nhỏ nhất trong một dãy số cho trước

10. Viết chương trình Python tìm số lớn thứ 2 trong một dãy số cho trước

11. Viết chương trình Python tìm số nhỏ thứ 2 trong một dãy số cho trước.

12. Cho trước số nguyên N. In ra màn hình các số nguyên tố nhỏ hơn N.

3. Các kiểu dữ liệu mảng trong Python

A) Giới thiệu chung

Có 4 loại Tổng Hợp loại dữ liệu trong ngôn ngữ lập trình Python là:

- List: đây là loại được tạo nên và thay đổi được. Cho phép lập lại các thành viên trong đó
- Tuple: đây là loại được tạo nên và không thay đổi được. Cho phép lập lại các thành viên
- Set: là tổng hợp không được tạo nên và không thay đổi được. Không được phép lập lại các thành viên.
- Dictionary: là tổng hợp không được tạo nên, thay đổi được và có mục chỉ dẫn (mục lục). Không lặp lại các thành viên.

Khi chọn loại tổng hợp, cần phải hiểu biết những đặc tính của loại đó. Chọn đúng bộ dữ liệu đặc thù có ý nghĩa quan trọng làm tang hay giảm hiệu suất và bảo mật.

B) Các kiểu dữ liệu mảng trong Python

3.1 List (Liệt kê trong Python)

3.1.1 Giới thiệu List

List là loại tổn hợp được tạo nên bởi người dùng và thay đổi được. Trong Python, list được viết trong những ngoặc vuông.

Là một cấu trúc dữ liệu cơ bản trong Python, thuộc nhóm sequence.

Gồm nhiều phần tử, mỗi phần tử có một vị trí gọi là index, phần tử đầu tiên có index = 0, phần tử cuối cùng có index = số phần tử - 1

Các phần tử trong list có thể cùng hoặc khác kiểu. Tuy nhiên, cùng kiểu thì dễ xử lý tính toán.

3.1.2 Tạo list

3.1.2.1 Khai báo và gán giá trị các phần tử trong list bằng phép gán

Cách tạo list bằng phép gán "="

Ví dụ về tạo list

```
Thislist =
["apple", "banana", "cherry"]
print(thislist)
```

3.1.2.2Tạo list bằng cách nối các list với nhau

NHẬP 2 list lại làm 1 dùng toán "+"

```
list1 = ["a", "b" , "c"]
list2 = [1, 2, 3]

list3 = list1 + list2
print(list3)
```

Dùng hàm extend để nối 2 list với nhau

```
list1 = ["a", "b" , "c"]
list2 = [1, 2, 3]

list1.extend(list2)
print(list1)
```

3.1.2.3 Tạo list bằng cách lặp lại list đã có

Tạo list bằng cách copy list có sẵn

Dùng hàm copy()

```
thislist =
["apple", "banana", "cherry"]
mylist = thislist.copy()
```

Dùng lệnh list()

```
thislist =
["apple", "banana", "cherry"]
mylist = list(thislist)
```

Lưu ý lệnh list() có thể dùng tạo kiểu dữ liệu mới

```
thislist = list(("apple", "banana", "cherry"))

# note the double round-brackets
print(thislist)
```

Lưu ý nếu dùng cách này tạo kiểu dữ liệu thì có hai dấu ngặc đơn ((

3.1.2.4 Tạo list bằng cách copy phần tử từ vị trí index tới cuối

Để copy từ vị trí index tới cuối ta dùng toán tử slicing như sau

```
# mixed list
list =  ['cat', 0, 6.7]
 # copying a list using slicing
new_list =  list[:]
 # Adding element to the new list
new_list.append('dog')
```

3.1.3 Truy xuất phần tử trong list

Muốn truy cập sử dụng dữ liệu tương ứng thì:

```
thislist = ["apple", "banana", "cherry"]
print(thislist[1])
```

Truy xuất phần tử có thể dùng chỉ số âm

Chỉ Số Âm: nghĩa là bắt đầu đếm ngược từ phần cuối của list

"-1" nghĩa là phần tử vị trí cuối cùng của List
"-2" nghĩa là phần tử kế cuối

```
thislist = ["apple", "banana", "cherry"]
print(thislist[-1])
```

Truy Xuất Một Phần Của List: bạn có thể chỉ ra một khoảng của List bằng cách dùng dấu ":". Ví dụ:

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[2:5])
```

Nghĩa là in ra màn hình các thành phần trong khoảng từ 2 tới 4. LƯU Ý vị trí số 5 sẽ không xuất ra. VỊ TRÍ ĐẦU TIÊN CÓ CHỈ SỐ 0

Truy xuất từ vị trí đầu tiên tới khoảng nào đó:

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[:4])
```

Truy xuất từ vị trị nào đó tới cuối

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[2:])
```

Truy xuất dùng khoảng chỉ số âm

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[-4:-1])
```

3.1.4 Cập nhật giá trị cho phần tử trong list

Thay đổi 1 giá trị thành phần nào đó trong list

```
thislist = ["apple", "banana", "cherry"]
thislist[1] = "blackcurrant"
print(thislist)
```

3.1.5 Vòng lặp trong List

Bạn có thể dùng vòng lặp for trong LIST như để in hết từng thành phần từng bước 1

```
thislist = ["apple", "banana", "cherry"]
for x in thislist:
  print(x)
```

Dùng vòng lặp kiểm tra có 1 thành phần nào đó trong list hay không

```
thislist = ["apple", "banana", "cherry"]
if "apple" in thislist:
  print("Yes, 'apple' is in the fruits list")
```

để kiểm tra có thành phần apple trong list hay không?

3.1.6 <u>Các phương thức cơ bản của list</u>

3.1.6.1 Lấy chiều dài số phần tử trong list

Dùng hàm len()

```
thislist = ["apple", "banana", "cherry"]
print(len(thislist))
```

3. 1.6.2 Thêm phần tử vào cuối list

Thêm thành phần dùng append(), chèn vào vị trí cuối cùng

```
thislist = ["apple", "banana", "cherry"]
thislist.append("orange")
print(thislist)
```

3.1.6.3 Chèn thêm vào trước thành phần nào đó trong List

```
thislist = ["apple", "banana", "cherry"]
thislist.insert(1, "orange")
print(thislist)
```

Trong ví dụ chèn thêm "orange" phía trước vị trí số 1 (chính là "banana")

3.1.6.4 Xóa một phần tử trong list (del, remove, pop)

Dùng lệnh remove

```
thislist = ["apple", "banana", "cherry"]
thislist.remove("banana")
print(thislist)
```

Dùng lệnh pop()

```
thislist = ["apple", "banana", "cherry"]
thislist.pop(2)
print(thislist)
```

Lưu ý pop() để remove chỉ số đặc biệt (để trống là vị trí cuối cùng)

Dung del để bớt thành phần đặc biệt chỉ định

```
thislist = ["apple", "banana", "cherry"]
del thislist[0]
print(thislist)
```

Dùng lệnh del để xóa hết các thành phần trong list

```
thislist = ["apple", "banana", "cherry"]
del thislist
```

Xoá hết clear() toàn bộ list

```
thislist = ["apple", "banana", "cherry"]
thislist.clear()
print(thislist)
```

3.1.6.5 Đảo ngược list

```
list01 = ["apple", "banana", "cherry"]
print('list01 is', list01)

list01.reverse()

print('Reverse of list01 is', list01)
print(thislist)
```

Phương thức này có tác dụng sắp xếp lại các phần tử trong list theo một thứ tự xác định.

Cú pháp:

```
mylist.sort(reverse, key)
```

Trong đó:

- mylist là list mà các bạn muốn sắp xếp.
- reverse là một boolean cấu hình kiểu sắp xếp. Nếu reverse = True thì list sẽ được sắp xếp từ lớn đến bé, nếu reverse = False thì list sẽ được sắp xếp theo thứ tự từ bé đến lớn. Mặc định thì reverse = False.
- key là callback def để xử lý list hoặc là một lamda function (thường được dùng để sắp xếp các list tuple hoặc dictionary).

Ví dụ:

```
list = ['A', 'C', 'B', 'E', 'D']

list.sort()

print(list)

# Kết quả: ['A', 'B', 'C', 'D', 'E']

list.sort(reverse=True)

print(list)

# Kết quả: ['E', 'D', 'C', 'B', 'A']

def custom_sort(elem):

    return elem[1]

list = [(1, 2), (5, 7), (7, 100), (4, 4)]

list.sort(key=custom_sort)

print(list)

# Kết quả: [(1, 2), (4, 4), (5, 7), (7, 100)]
```

3.1.7 Tổng hợp các hàm dùng trong LIST

Phương pháp	Mô tả cách sử dụng
append()	Thêm một thành phần tại cuối của list
clear()	Bỏ hết hoàn toàn thành phần trong list. Clear trắng.
copy()	Trả về bản sao của list
count()	Trả về số các thành phần có trong list với giá trị chỉ định
extend()	Thêm nhiều thành phần vào cuối list hiện tại với giá trị chỉ định
index()	Trả về chỉ số đầu tiên với giá trị chỉ định
insert()	Thêm một thành phần trước vị trí chỉ định
pop()	Xoá bớt thành phần tại vị trí chỉ định
remove()	Xoá tại vị trí chỉ định
reverse()	Đảo thứ tự trong list
sort()	Xắp xếp theo trình tự nào đó trong list

3.2 Tuples trong Python

3.2.1 Giới thiệu Tuples

Tuple là kiểu tập hợp được định sẵn và không thay đổi được. Tuple sử dụng dấu ngoặc đơn

3.2.2 Tạo một Tuples

Tạo một Tuples dùng phép gán

```
thistuple = ("apple", "banana", "cherry")
print(thistuple)
```

Dùng hàm tuple()

```
thistuple = tuple(("apple", "banana", "cherry")) # note
the double round-brackets
print(thistuple)
```

```
["apple", "banana", "cherry", "orange", "kiwi", "melon", "m
ango"]
```

3.2.3Truy xuất các thành phần trong Tuples

3.2.3.1 Truy xuất thành phần dùng ngoặc vuông có chỉ số

Bạn có thể truy xuất các thành phần của tuple bằng cách dùng chỉ số trong ngoặc vuông.

```
thistuple = ("apple", "banana", "cherry")
print(thistuple[1])
```

Nghĩa là print thành phần thứ 2 trong tuple

3.2.3.2 Truy xuất dùng chỉ số âm

Có thể dùng chỉ số âm -1 chỉ vị trí cuối cùng hoặc -2 chỉ vị trí kế cuối

```
thistuple = ("apple", "banana", "cherry")
print(thistuple[-1])
```

3.2.3.3 Dùng khoảng chỉ số

Có thể dùng khoảng 2:5 để truy xuất như sau:

```
thistuple =
("apple", "banana", "cherry", "orange", "kiwi", "melon
", "mango")
print(thistuple[2:5])
```

Trả về vị trí thứ 3, 4 và 5.

LƯU Ý: bắt đầu từ vị trí 2 tới vị trí 4 (không bao gồm vị trí 5)

3.2.3.4 Dùng khoảng chỉ số âm

```
thistuple =
("apple", "banana", "cherry", "orange", "kiwi", "melon", "m
ango")
print(thistuple[-4:-1])
```

Trả về bao gồm vị trí -4 và không bao gồm vị trí -1.

3.2.3 Vòng lặp của Tuples

Tích hợp với thành phần để in ra chuỗi giá trị tương ứng

```
thistuple = ("apple", "banana", "cherry")
for x in thistuple:
  print(x)
```

Phần hàm for sẽ được học trong Chương vòng lặp của Python

3.2.4 Các phương thức cơ bản trong Tuples

3.2.4.1Thay đổi giá trị của Tuples

```
x = ("apple", "banana", "cherry")
y = list(x)
y[1] = "kiwi"
x = tuple(y)
print(x)
```

Vì giá trị của Tuple không thể thay đổi nên phải đổi kiểu sang List và lưu vào Y. Sau đó đổi giá trị trong list Y và từ đó gán đổi kiểu Tuple ngược lại vào X.

3.2.4.2 Đo chiều dài của Tuples

Để xác định chiều dài của Tuple sử dụng len() như sau:

```
thistuple = ("apple", "banana", "cherry")
print(len(thistuple))
```

3.2.4.3 Thêm giá trị cho Tuples

Thêm giá trị cho tuples là không thể một khi Tuple được tạo

```
thistuple = ("apple", "banana", "cherry")
thistuple[3] = "orange" # This will raise an error
print(thistuple)
```

phần code này sẽ đưa ra lỗi.

3.2.4.4 Tạo tuple với 1 thành phần, lưu ý dấu phẩy

```
thistuple = ("apple",)
print(type(thistuple))
#NOT a tuple
thistuple = ("apple")
print(type(thistuple))
```

Xoá thành phần trong tuple: là KHÔNG THỂ
Nhưng có thể xoá toàn bộ tuple

```
thistuple = ("apple", "banana", "cherry")
del thistuple
print(thistuple) #this will raise an error because
the tuple no longer exists
```

3.2.4.5 Nhập hai tuples

```
tuple1 = ("a", "b" , "c")
tuple2 = (1, 2, 3)
tuple3 = tuple1 + tuple2
print(tuple3)
```

3.2.6. Tổng hợp các hàm dùng trong Tuple

Phương pháp	Mô tả
count()	Trả về số lần của một giá trị đặc biệt trong Tuple
index()	Search trong tuple cho giá trị chỉ định và trả về vị trí nơi được tìm thấy

3.3 Dictionary (từ điển trong Python)

3.3.1 Giới thiệu Dictionary

Dictionaries là tập các thành phần chưa được set up sẵn, có thể thay đổi và chỉ số truy cập. Trong Python dictionaries dùng dấu ngoặc nhọn, và có những quy định cho dữ liệu chữ và giá trị số.

3.3.2 *Tạo Dictionary*

Dùng phép gán để tạo dữ liệu kiểu dictionary

Ví du:

```
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
print(thisdict)
```

3.3.3 Truy xuất các phần tử trong Dictionary

Để truy cập các thành phần bên trong dictionaries thì có thể dùng key name bên trong dấu ngoặc vuông:

```
x = thisdict["model"]
print (‘model of car is’, x)
# Kết quả x là “Mustang”
```

```
x = thisdict.get("model")
print (‘model of car is’, x)
# Kết quả x là “Mustang”
```

3.3.4 Cập nhật các giá trị phần tử trong dictionary

Cập nhật phần tử bằng phép gán "="

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict["year"] = 2018
```

Trong dữ liệu kiểu Dictionary thì có 2 thành phần key "year" và giá trị đi kèm với nó là "1964". Trong phần ví dụ ta thay đổi giá trị 1964 của key name "year" thành 2018.

3.3.5 Vòng lặp trong Dictionary

Có thể sử dụng phòng lặp trong dữ liệu Dictionary bằng sử dụng vòng for như sau:

```
for x in thisdict:
  print(x)
```

Ví dụ trên in ra tất cả key name trong dữ liệu Dictionary

Có thể in ra giá trị trong Dict bằng vòng lặp:

```
for x in thisdict:
  print(thisdict[x])
```

Có thể in ra key name + giá trị trong Dict bằng vòng lặp:

```
for x, y in thisdict.items():
  print(x, y)
```

3.3.6 Các phương thức cơ bản trong Dictionary

3.3.6.1 Kiểm tra xem một key name có tồn tại không?

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
if "model" in thisdict:
  print("Yes, 'model' is one of the keys in
the thisdict dictionary")
```

3.3.6.2 Xác định có bao nhiêu thành phần trong dictionary bằng hàm len().

```
print(len(thisdict))
```

3.3.6.3 Thêm thành phần vào Dictionary

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict["color"] = "red"
print(thisdict)
```

Thêm key name color và giá trị red vào dữ liệu Dictionary

3.3.6.4 Xoá những thành phần trong Dictionary (pop, del, clear).

Dùng hàm pop()

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.pop("model")
print(thisdict)
```

Dùng del để xoá thành phần với key name chỉ định.

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
del thisdict["model"]
print(thisdict)
```

Dùng clear() để xoá toàn bộ dữ liệu trong Dictionary

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.clear()
print(thisdict)
```

3.3.6.5 Copy và xây dựng một bản sao từ Dictionary có sẵn

Không thể copy đơn giản bằng dấu = như dict2= dict1 một cách bình thường như vậy. Bởi vì dict2 chỉ là tham khoả cho dict1 và có thể làm những thay đổi cho dict1 sẽ tự động thay đổi cho dict2.

Có nhiều cách để copy, một trong cách đó là dùng hàm copy().

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
mydict = thisdict.copy()
print(mydict)
```

Dùng cách khác tạo copy là dùng chính hàm được xây dựng sẵn dict().

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
mydict = dict(thisdict)
print(mydict)
```

Một Dict có thể chứa bên trong nhiều Dict khác gọi là Nest Dictionaries.

```
myfamily = {
  "child1" : {
    "name" : "Emil",
    "year" : 2004
  },
  "child2" : {
    "name" : "Tobias",
    "year" : 2007
  },
  "child3" : {
    "name" : "Linus",
    "year" : 2011
  }
}
```

Có thể tạo nest từ 3 Dicts sẵn có

```
child1 = {
  "name" : "Emil",
  "year" : 2004
}
child2 = {
  "name" : "Tobias",
  "year" : 2007
}
child3 = {
  "name" : "Linus",
  "year" : 2011
```

```
}
myfamily = {
  "child1" : child1,
  "child2" : child2,
  "child3" : child3
}
```

Xây dựng một dict() mới

```
thisdict = dict(brand="Ford",
model="Mustang", year=1964)
# note that keywords are not string
literals
# note the use of equals rather than
colon for the assignment
print(thisdict)
```

3.3.7 Tổng hợp các hàm dùng kèm với Dictionary

Phương Pháp	Mô Tả
clear()	Xoá hết tất cả thành phần của từ điển
copy()	Trả về bản copy của một Dict
fromkeys()	Trả về một dict với key chỉ định và giá trị
get()	Trả về giá gị của key chỉ định
items()	Trả về list chứa tuple và mỗi key + giá trị
keys()	Trả về list chứa các key của Dict
pop()	Xoá thành phần với key chỉ định
popitem()	Xoá cặp key-giá trị cuối cùng
setdefault()	Trả về giá trị của key chỉ định. Nếu key không tồn tại: thêm vào key và với giá trị chỉ định
update()	Cập nhật Dict với cặp key-giá trị chỉ định
values()	Trả về list tất cả giá trị trong dict

3.4 Set trong Python

3.4.1 Giới thiệu Set trong Python

Set là tập hợp những thành phần chưa thiết lập và chỉ định sẵn. Trong python, sets được viết trong ngoặc nhọn.

Set trong python là một tập các giá trị không có thứ tự. Mỗi giá trị trong set là duy nhất, không thể lặp lại và bất biến(tức bạn không thể thay đổi giá trị các phần tử trong set).

Tuy nhiên các set có thể thay đổi. Chúng ta có thể thêm, xóa các phần tử trong set.

Các set có thể được sử dụng để thực hiện các phép tập hợp như phép giao, hợp

3.4.2 Cách tạo một Set

Ví dụ tạo một set:

```
thisset = {"apple", "banana", "cherry"}
print(thisset)
```

Tạo một set bằng chính hàm set()

```
thisset = set(("apple", "banana", "cherry")) # note
the double round-brackets
print(thisset)
```

3.4.3 Truy xuất thành phần trong Set

Vì set là tập giá trị không có thứ tự nên việc truy xuất 1 một phần trong set là không thể và báo lỗi. Ta chỉ có thể kiểm tra xem phần tử có tồn tại trong set bằng cách sau:

```
thisset = {"apple", "banana", "cherry"}
print("banana" in thisset)
```

3.4.4 Các phương thức cơ bản của Set

3.4.4.1 Để thêm 1 thành phần vào Set thì dùng phương pháp add()

```
thisset = {"apple", "banana", "cherry"}
thisset.add("orange")
print(thisset)
```

3.4.4.2 Thêm nhiều thành phần dùng update() như sau:

```
thisset = {"apple", "banana", "cherry"}
thisset.update(["orange", "mango", "grapes"])
print(thisset)
```

3.4.4.3 Đo chiều dài của set

```
thisset = {"apple", "banana", "cherry"}
print(len(thisset))
```

3.4.4.4 Thêm bớt thành phần trong set (remove, discard, clear)

Xoá bớt thành phần trong set dùng remove() hoặc discard() như sau:

```
thisset = {"apple", "banana", "cherry"}
thisset.remove("banana")
print(thisset)
```

Xoá thành phần banana trong set

LƯU Ý: *nếu thành phần banana không tồn tại sẽ phát sinh ra lỗi.*

Còn với discard() thì sẽ không báo lỗi nếu thành phần không tồn tại.

Có thể dùng pop() với chỉ số để trống sẽ xoá thành phần cuối cùng.

```
thisset = {"apple", "banana", "cherry"}
x = thisset.pop()
print(x)
print(thisset)
```

Giá trị mà hàm pop() trả về chính là giá trị mà pop() đã xoá -> cherry

Dùng clear() để xoá toàn bộ set

```
thisset = {"apple", "banana", "cherry"}
thisset.clear()
print(thisset)
```

3.4.4.5 Gia nhập 2 sets

Gia nhập hai sets dùng lệnh union() hoặc update()

```
set1 = {"a", "b" , "c"}
set2 = {1, 2, 3}
set3 = set1.union(set2)
print(set3)
```

3.4.5 Tổng hợp các hàm dùng với Set

Phương pháp	Mô tả
add()	Thêm thành phần vào set
clear()	Xoá tất cả thành phần từ set
copy()	Trả về giá trị copy của
difference()	Trả về giá trị khác nhau giữa hai sets
difference_update()	Xoá những thành phần trong set bao gồm những thành phần khác với set chỉ định.
discard()	Xoá thành phần chỉ định
intersection()	Trả về một set, với phần giao nhau giữa hai sét
intersection_update()	Trả về những thành phần đã loại bỏ phần giao nhau với set chỉ định
isdisjoint()	Trả về bất cứ giá trị không giao nhau giữa hai set
issubset()	Trả về tất cả những thành phần kể cả không chứa trong set
issuperset()	Trả về bất kể set này chứa set khác hoặc không
pop()	Xoá yếu tố từ set với số chỉ định
remove()	Xoá yếu tố chỉ định của set
symmetric_difference()	Trả về set với sự khác biệt đối xứng của 2 sets
symmetric_difference_update()	Thêm vào những khác biệt đối xứng từ set này và set khác
union()	Trả về nội dung gộp 2 set
update()	Update set với tổ hợp set này với những cái khác

Bài tập:

Phần bài tập các kiểu dữ liệu List

1. Sử dụng kiểu dữ liệu List in ra màn hình tam giác được tạo thành từ những dấu * với chiều cao 4 đơn vị.
2. Sử dụng kiểu List viết chương trình có dữ liệu điểm trung bình môn học của học sinh Nguyễn Văn A như sau:
    - Nguyễn Văn A: Toán 7.4 Lý 6.5 Hoá 7.0
    - Hiển thị màn hình hướng dẫn nhập liệu search thông qua nhập tên học sinh. Ví dụ nhập Nguyễn Văn A thì nghĩa là hiển thị xuất ra tên và điểm trung bình môn học.

    GỢI Ý: Áp dụng các kiểu dữ liệu List cho tên và từng môn học tương ứng cho Học Sinh như sau

    Ten= [“Nguyen Van A”, “Nguyen Van B”, “Nguyen Van C”]
    Toan= [“7.4”, “6.5”, “7.0”]

    Tương tự cho các môn còn lại. Ví dụ search Nguyen Van A nghĩa là chọn kiểu list vị trí số 1 thì xuất ra màn hình Ten[0], Toan[0], Ly[0], Hoa[0].
3. Sử dụng kiểu List viết chương trình có kiểu dữ liệu là các loại trái cây từ apple tới mango như trên. Xuất ra màn hình tên và giá tiền/ kg của các loại trái cây từ cherry tới melon. Dữ liệu giá tiền tự chọn
4. Thêm phần check tên học sinh có tồn tại hay không ở phần đầu đoạn code và xuất ra màn hình. THÊM VÀO CODE BÀI SỐ 2
5. Chỉnh sửa phần BÀI THỰC HÀNH SỐ 3, bỏ bớt thành phần cuối cùng. Chèn thêm trước trái kiwi thêm hai loại trái cây khác tự chọn. LƯU Ý TRÁI CÂY DÙNG TIẾNG ANH.
6. Nhập từ bàn phím và lưu dữ vào từng list với các trường như sau: Mã số Sinh Viên, Họ và Tên, Ngày Tháng Năm Sinh, Tên Lớp, Điểm Thường Kỳ 1,2,3, Điểm Thi Giữa Kỳ, Điểm Thi Cuối Kỳ, Điểm trung bình tổng kết, Kết quả đậu rớt.

    LƯU Ý: *chương trình phải có sự lựa chọn cho người dùng muốn nhập liệu hay search. VÀ search dựa trên MSSV, Tên, hoặc nhóm lớp tương ứng như DH14A, DH14B.*

Phần bài tập của Dictionary

7. Thiết lập dữ liệu Dictionary của 1 sinh viên gồm STT, Họ và Tên, MSSV, Ngày tháng năm sinh, quê quán, Học Phí, Sở Thích.

    + In ra màn hình Sở thích của Sinh viên

+ Sau đó thay đổi giá trị sở thích và in ra lại lần nữa.

8. Chỉnh sửa lại code bài 7

+ **In** ra tất cả key name

+ **Dùng** key name đã in ra để truy xuất 1 giá trị nào đó

+ **In** ra tất cả giá trị theo key name đó.

+ **In** ra tất cả key name và giá trị đồng thời

9. Chỉnh sửa code phần bài 7

+ **Kiểm tra** và in ra số lượng thành phần trong Dictionry

+ **Thêm** vào Ngoại ngữ và Dân Tộc, và in ra kết quả tất cả thành phần trong Dictionary

+ **Xoá** key name và giá trị "sở thích" và sau đó in ra tất cả thành phần lại lần nữa.

10. Thiết lập 3 bộ dữ liệu sinh viên gồm: Tên, MSSV, lớp.

+ **Nest** lại 3 dict của 3 sinh viên trên thành một dict mới tên lớp DHDT12B.

+ **In** ra key name tất cả thành phần trong dict lớp DHDT12B

+ **Search** và xuất ra bộ dữ liệu dict của sinh viên thứ 2

+**Copy** dict lớp DHDT12B và tạo ra dict mới là tên môn học ANHVAN01

+ **Thêm** 2 dict của 2 sinh viên lớp DHDT12A vào lớp ANHVAN01.

11. Chỉnh sửa lại phần code bài 10

+ **Thêm** vào 2 sinh viên cho lớp DHDT12B thành tổng 5 sinh viên.
+ **Thêm** vào lớp ANHVAN01 hai sinh viên mới lớp DHDT12B
+ **Chỉnh** sửa toàn bộ dict DHDT12B, lớp ANHVAN01 và 2 sinh viên lớp DHDT12A bằng cách thêm thành phần quê quán, trình độ anh văn điểm Toiec hiện tại, tình trạng học phí.
+ **Xoá 1** sinh viên lớp DHDT12A do chưa đóng học phí khỏi lớp ANHVAN01

Phần bài tập Tuples

12. Sử dụng kiểu Tuple viết chương trình có dữ liệu điểm trung bình môn học của học sinh Nguyễn Văn A như sau:

Nguyễn Văn A: Toán 7.4 Lý 6.5 Hoá 7.0

Hiển thị màn hình hướng dẫn nhập liệu search thông qua nhập tên học sinh. Ví dụ nhập Nguyễn Văn A thì nghĩa là hiển thị xuất ra tên và điểm trung bình môn học.

Sau đó thay đổi giá trị môn Toán của HS Nguyễn Văn A là 8.0

13. Sử dụng kiểu Tuple viết chương trình có kiểu dữ liệu là các loại trái cây từ `["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]`. Xuất ra màn hình tên và giá tiền/ kg của các loại trái cây từ cherry tới melon. Dữ liệu giá tiền tự chọn.

    LƯU Ý sinh viên dùng vòng lặp để xuất giá trị ra màn hình.

14. Tạo hai Tuple với thông số sau"

    Tuple 1: `["apple", "banana", "cherry"]`

    `Tuple 2: ["kiwi", "melon", "mango"]`

    NHẬP 2 tuple lại làm 1 và in ra kết quả

    LƯU Ý: dùng hàm tạo tuples.

15. Tạo ra dữ liệu sẵn có về 10 loại trái cây và giá tiền tương ứng dùng cấu trúc Tuples.

    Cho phép nhập liệu tên trái cây search ra giá tiền.

    + Sau đó cập nhật lại giá tiền của 5 loại trái cây trong tuple đó. Search lại lần nữa các loại trái cây và xuất ra màn hình

    LƯU Ý: *chương trình phải có sự lựa chọn cho người dùng muốn search tên một loại trái cây hay xuất ra toàn bộ trái cây và giá tiền tương ứng.*

Bài tập về Set trong python

16. Tạo một set 10 loại trái cây và in ra màn hình

    + Kiểm tra một loại trái cây có trong set hay không? In ra kết quả Yes or No

17. Từ phần bài 16 chỉnh sửa code thêm 5 loại trái cây khác vào Set.

18. Tạo 2 set trái cây (mỗi set 5 loại trái cây và giá tiền tương ứng).

    + Search trái cây xuất ra màn hình Tên trái cây, giá tiền, và thuộc về set 1 hay 2
    + Nhập hai set làm 1 và search lại xuất lần nữa

19. Tạo 2 set trái cây (mỗi set 5 loại trái cây).

    + Tìm phần giống nhau giữa hai set này và xuất ra màn hình

    + Tìm phần set 1 khác set 2 và xuất ra màn hình

    + Tìm phần set 2 khác set 1 và xuất ra màn hình

    + Xuất ra màn hình set tổ hợp cả hai bộ set trái cây

    +Xuất ra màn hình set 1 có thêm phần khác từ set 2

    +Xuất ra màn hình set 2 có thêm phần khác từ set 1

4. Module và Packet trong Python

4.1. Giới thiệu

Modules có phần code giống như phần code của một thư viện. Một file chứa tập hợp các hàm bạn muốn bao gồm trong ứng dụng của chính bạn.

4.2. Cách tạo một Module

Để tạo một module bạn cần lưu cái code bạn muốn thành một file với đuôi ".py".
Ví dụ lưu một code với tên gọi mymodule.py có nội dung sau:

```
def greeting(name):
    print("Hello, " + name)
```

4.3. Cách sử dụng một module

Để sử dụng một module mà vừa tạo, tôi đi học app gô ta có thể dùng hàm import như sau:

```
import mymodule
mymodule.greeting("Jonathan")
```

Ví dụ trên truyền tên "Jonathan" vào hàm greeting trong file module.

4.4. Các loại biến có trong module

Một module có thể chứa nhiều hàm, như mô tả trên có thể nhiều loại biến như: arrays, dictionaries, objects.
Ví dụ lưu vào file mymodule.py nội dung sau.

```
person1 = {
  "name": "John",
  "age": 36,
  "country": "Norway"
}
```

Ví dụ trên dùng kiểu dictionaries trong module. Bạn có thể import keyword của thành phần trong dict của module bằng hàm import như sau:

```
import mymodule
a = mymodule.person1["age"]
print(a)
```

4.5.Đặt tên và đổi tên module.

Đặt tên module bất cứ tên gì bạn muốn nhưng đuổi bắt buộc phải là .py.
Đổi tên bằng cách tạo liên kết khi bạn import một module bằng sử dụng keyword as như sau:

```
import mymodule as mx
a = mx.person1["age"]
print(a)
```

4.6.Cách xây dựng một Module

Có nhiều cách xây dựng những module trong Pthon

4.6.1. Import vào Module bất cứ thứ gì bạn muốn.

Ví dụ: Import và sử dụng module platfom

```
import platform
x = platform.system()
print(x)
```

4.6.2. Dùng hàm dir()

Sử dụng hàm dir() được xây dựng sẵn để liệt kê các hàm hoặc biến trong một module. Hàm dir() sử dụng như sau:

Ví dụ: liệt kê tất cả các tên thuộc về module platform

```
import platform
x = dir(platform)
print(x)
```

Lưu ý rằng hàm dir() có thể sử dụng trong tất cả module, cũng có thể trong những module bạn tự tạo ra.

4.6.3. Import từ module

Bạn có thể chọn import một phần từ một module bằng sử dụng keyword:
Ví dụ tên hàm mymodule có một hàm và một dictionary

```
def greeting(name):
 print("Hello, " + name)
person1 = {
  "name": "John",
  "age": 36,
  "country": "Norway"
}
```

Ví dụ: Import chỉ duy nhất kiểu dict tên person1 từ module

```
from mymodule import person1

print (person1["age"])
```

Lưu ý: Khi đang import sử dụng keyword from, không được dùng tên module khi cần tham khảo những yếu tố trong module. Ví dụ

`person1["age"]`, **not** ~~`mymodule.person1["age"]`~~

4.7.**Packet trong Python**

4.7.1. Giới thiệu

Trong bài trước bạn đã tìm hiểu về module, cách tạo và gọi một module trong Python. Ở bài này, chúng ta sẽ học cách phân chia code thành những module hiệu quả, rõ ràng, sử dụng các package trong Python. Thêm nữa là cách để nhập và sử dụng package của riêng bạn, hoặc package bạn tải về từ đâu đó vào chương trình Python.

Package trong Python là gì?

Thông thường người dùng không lưu trữ tất cả các file của mình trên máy tính ở cùng một thư mục, vị trí, mà sử dụng hệ thống phân cấp thư mục để truy cập dễ dàng hơn. Các tệp

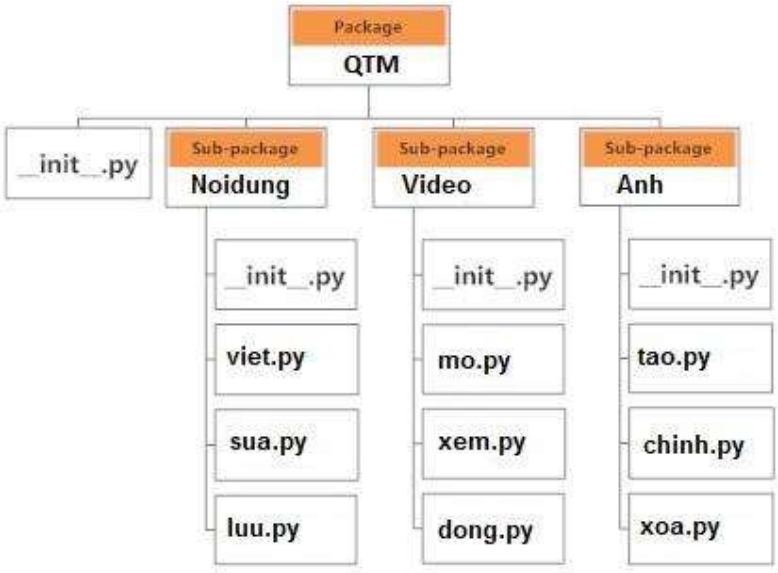
tương tự hoặc cùng liên quan đến một chủ đề nào đó sẽ được để trong cùng một thư mục, chẳng hạn các bài học về hàm Python sẽ lưu trong thư mục hàm Python. Tương tự như vậy, Python có các package cho thư mục và module cho file.

Khi chương trình đang code ngày càng lớn với rất nhiều module, chúng ta sẽ đặt những module giống nhau vào một package, và những nhóm module khác vào package khác. Điều này giúp dễ dàng quản lý chương trình hơn và nó cũng dễ hiểu hơn.

Nếu như thư mục có thể chứa các thư mục con thì package cũng vậy, trong một package có thể có package con và các module khác.

Một thư mục phải chứa file có tên __init__.py để Python hiểu thư mục này là một package. File này có thể để trống, nhưng thông thường các lập trình viên thường đặt code khởi tạo cho package ở đây.

Giả sử, ta đang phát triển chương trình có tên QTM, với các package con, module như sau:



4.7.2. Nhập module từ package trong Python

Ta có thể nhập các module từ package sử dụng toán tử dấu chấm (.).

Ví dụ, nếu muốn nhập module mo.py trong ví dụ trên, bạn làm như sau:

```
    import QTM.Video.mo
```

Nếu trong module mo.py chứa hàm có tên là chon_video(), bạn sẽ phải sử dụng tên đầy đủ để tham chiếu tới nó:

```
    import QTM.Video.mo.chon_video(video1)
```

Nếu cấu trúc trên dài quá, bạn có thể nhập module mà không cần tiền tố package như sau:

```
from QTM.Video import mo
```

Ta có thể gọi hàm đơn giản như sau:

```
mo.chon_video(video1)
```

Ngoài ra, còn một cách nữa để chỉ nhập hàm được yêu cầu (lớp hoặc biến) từ module trong 1 package như sau:

```
from QTM.Video.mo import chon_video
```

Sau đó, trực tiếp gọi hàm này:

```
chon_video(video1)
```

Dù dễ dàng hơn, nhưng cách nhập sau cùng này không được khuyến khích sử dụng. Việc dùng tên đầy đủ sẽ giúp giảm tình trạng nhầm lẫn và tránh bị trùng giữa những định danh giống nhau.

Trong khi nhập các package, Python sẽ tìm kiếm danh sách thư mục được định nghĩa trong sys.path, giống như đường dẫn tìm kiếm module.

4.7.3. Sử dung package trong Python

Trong Python, một số module trong cùng thư mục (và thư mục con) có thể kết hợp lại tạo ra một package. Package giúp đơn giản hóa hơn nữa việc sử dụng nhiều module có liên quan.

Thực hiện ví dụ sau:

Tạo thư mục my_package. Trong thư mục này tạo ra 2 file module1.py và module2.py với code như sau:

module1.py | module2.py

```
1. print('This is the module 1')
2.
3. def func1():
4.     print('Module 1 - func 1')
```

Nếu dừng lại ở đây sẽ có 2 module riêng rẽ. Khi sử dụng bạn cần import từng module.

Giờ hãy tạo file __init__.py trong thư mục my_package. Lưu ý có hai ký tự _ ở trước và ở sau init. Viết code sau cho __init__.py:

```
from my_package.module1 import *
from my_package.module2 import *

#hoặc cũng có thể import như thế này:
#import my_package.module1
#import my_package.module2
```

Đây là hai lệnh import thông thường mà bạn đã học. Bạn có thể sử dụng kiểu import nào cũng được. Tuy nhiên cần lưu ý cách viết tên module:

my_package.module1. Bạn cần ***chỉ định tên thư mục,*** nếu không Python sẽ không tìm được module tương ứng.

Tên file __init__.py có ý nghĩa đặc thù trong Python. Nếu Python nhìn thấy thư mục nào có file với tên gọi __init__.py, nó sẽ tự động coi thư mục này như một package, chứ không còn là thư mục bình thường nữa.

Bên ngoài thư mục package hãy tạo module main.py và viết code như sau:

```
import my_package as mp

mp.module1.func1()
mp.module2.func2()

from my_package import *
module1.func1();
```

Để ý rằng lệnh import giờ không còn hoạt động nữa với từng module mà là với tên thư mục my_ package

Tất cả các thư mục chưa file __init__.py sẽ trở thành một package. Tên thư mục trở thành tên packet và có thể sử dụng cùng lệnh import hoặc from/ import.

Khi import một packet bạn sẽ đồng thời import tất cả các module của nó (được chỉ định trong __init__.py).

Nếu sử dụng lệnh **import <tên_package>** , bạn sẽ truy cập vào các hàm của từng module qua cú pháp <**tên_package>.<tên_module>.<tên_hàm>().**

Nếu sử dụng from <tên_package> import <tên_hàm), bạn cũng có thể sử dụng tên ngắn ngọn của hàm như bình thường

Lưu ý: trong __init__.py và trong module sử dụng package ***nên dùng cùng một cách import.*** *Ví dụ, cùng dùng import hoặc cùng dùng from .. import.*

Bài tập:

1. Viết chương trình tạo module infoSV.py trong đó có hai hàm tên, và hàm mssv. Nhập tên và mssv vào module thông qua hai hàm trên. Có xuất ra màn hình.
2. Tạo module dùng biến dạng dicts với thông tin nhập vào là:

   + Số thứ tự
   + Họ và Tên
   + MSSV
   + Ngày tháng năm sinh
   + Quê quán
   + Lớp
3. Tạo module Lớp 12A và 12B với 5 sinh viên kiểu dữ liệu dictionary gồm: Tên, mssv, ngày tháng năm sinh, quê quán.

   + Tạo module lopLapTrinhMang
   + Import 3 sinh viên từ lớp 12A và 2 sinh viên từ 12B
   + Xuất ra màn hình danh sách tất cả sinh viên lớp Lập Trình Mạng
4. Viết chương trình và in giá trị theo công thức cho trước: Q = √([(2 * C * D)/H]) (bằng chữ: Q bằng căn bậc hai của [(2 nhân C nhân D) chia H]. Với giá trị cố định của C là 50, H là 30. D là dãy giá trị tùy biến, được nhập vào từ giao diện người dùng, các giá trị của D được phân cách bằng dấu phẩy.

Ví dụ: Giả sử chuỗi giá trị của D nhập vào là 100,150,180 thì đầu ra sẽ là 18,22,24.

Gợi ý:

- Nếu đầu ra nhận được là một số dạng thập phân, bạn cần làm tròn thành giá trị gần nhất, ví dụ 26.0 sẽ được in là 26.
- Trong trường hợp dữ liệu đầu vào được cung cấp cho câu hỏi, nó được giả định là đầu vào do người dùng nhập từ giao diện điều khiển.

5. Viết một chương trình có 2 chữ số, X, Y nhận giá trị từ đầu vào và tạo ra một mảng 2 chiều. Giá trị phần tử trong hàng thứ i và cột thứ j của mảng phải là i*j.

   Lưu ý: i=0,1,…,X-1; j=0,1,…,Y-1.
   Ví dụ: Giá trị X, Y nhập vào là 3,5 thì đầu ra là: [[0, 0, 0, 0, 0], [0, 1, 2, 3, 4], [0, 2, 4, 6, 8]]
   Gợi ý:
   Viết lệnh để nhận giá trị X, Y từ giao diện điều khiển do người dùng nhập vào.
6. Viết một chương trình chấp nhận chuỗi từ do người dùng nhập vào, phân tách nhau bởi dấu phẩy và in những từ đó thành chuỗi theo thứ tự bảng chữ cái, phân tách nhau bằng dấu phẩy.

Giả sử đầu vào được nhập là: without,hello,bag,world, thì đầu ra sẽ là: bag,hello,without,world.

Gợi ý:

Trong trường hợp dữ liệu đầu vào được nhập vào chương trình nó nên được giả định là dữ liệu được người dùng nhập vào từ giao diện điều khiển.

7. Viết một chương trình chấp nhận chuỗi là các dòng được nhập vào, chuyển các dòng này thành chữ in hoa và in ra màn hình. Giả sử đầu vào là:

   Hello world Practice makes perfect

   Thì đầu ra sẽ là:

   HELLO WORLD PRACTICE MAKES PERFECT

   Gợi ý:

   Trong trường hợp dữ liệu đầu vào được nhập vào chương trình nó nên được giả định là dữ liệu được người dùng nhập vào từ giao diện điều khiển.

8. Viết một chương trình chấp nhận đầu vào là một chuỗi các từ tách biệt bởi khoảng trắng, loại bỏ các từ trùng lặp, sắp xếp theo thứ tự bảng chữ cái, rồi in chúng.

   Giả sử đầu vào là: hello world and practice makes perfect and hello world again
   Thì đầu ra là: again and hello makes perfect practice world
   Gợi ý:

   - Trong trường hợp dữ liệu đầu vào được nhập vào chương trình nó nên được giả định là dữ liệu được người dùng nhập vào từ giao diện điều khiển.
   - Sử dụng set để loại bỏ dữ liệu trùng lặp tự động và dùng sorted() để sắp xếp dữ liệu.

9. Viết một chương trình chấp nhận đầu vào là chuỗi các số nhị phân 4 chữ số, phân tách bởi dấu phẩy, kiểm tra xem chúng có chia hết cho 5 không. Sau đó in các số chia hết cho 5 thành dãy phân tách bởi dấu phẩy.

   Ví dụ đầu vào là: 0100,0011,1010,1001

   Đầu ra sẽ là: 1010

   Gợi ý:

Trong trường hợp dữ liệu đầu vào được nhập vào chương trình nó nên được giả định là dữ liệu được người dùng nhập vào từ giao diện điều khiển.

10. Viết một chương trình tìm tất cả các số trong đoạn 1000 và 3000 (tính cả 2 số này) sao cho tất cả các chữ số trong số đó là số chẵn. In các số tìm được thành chuỗi cách nhau bởi dấu phẩy, trên một dòng.

    Gợi ý

    Trong trường hợp dữ liệu đầu vào được nhập vào chương trình nó nên được giả định là dữ liệu được người dùng nhập vào từ giao diện điều khiển.