BÀI THỰC HÀNH 1: CƠ BẢN VỀ PYTHON

Python là ngôn ngữ lập trình thông dịch cho phép người dùng có thể viết code ngay trên trình thông dịch hoặc viết vào file sau đó chạy chúng. Để bắt đầu Python, trong cửa sổ DOS (Windows) hoặc ở Terminal (Linux) gõ "python" và có thể nhập lệnh trực tiếp từ đây.

```
Python 2.7.12 (default, Nov 19 2016, 06:48:10)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Ngoài ra, chúng ta có thể sử dụng Notepad để soạn thảo code với định dạng đuôi .py và các script *.py của Python có thể được thực hiện tại dòng lệnh như sau:

```
python <tên file>.py
```

(Hoặc sử dụng Pycharm, Spyder, Anaconda, Sublime Text,…)

Lưu ý trong Python:

- Các khối mã lệnh (code block) được tạo ra bằng cách thụt đầu dòng ở mỗi câu lệnh thay vì sử dụng cặp dấu ngoặc { } như các ngôn ngữ khác.
- Chúng ta không cần khai báo biến một cách tường minh. Kiểu dữ liệu sẽ được gán một cách linh động khi khởi tạo biến. Ví dụ: a = 1 thì a là dữ liệu kiểu nguyên, nếu a = 1.0 thì a là kiểu số thực.
- Chú thích trong code sử dụng dấu # (dòng) , "" (khối);
- Sử dụng Escaping Backslash \ để nhập chuỗi có chứa dấu nhấy đơn ' hoặc dấu nháy kép " hoặc xuống dòng khi viết lệnh có chuỗi dài.

  Ví dụ:

```
string = 'I\'m a student the "Can Tho University" '
print(string)
string = 'Toi muon tao ra mot chuoi rat rat dai \
Toi muon tao ra mot chuoi rat rat dai'
print(string)
```

  Kết quả thực hiện:

```
I'm a student the "Can Tho University"
Toi muon tao ra mot chuoi rat rat dai Toi muon tao ra mot chuoi rat rat dai
```

- Sử dụng hàm input để nhập dữ liệu từ bàn phím (Linux: raw_input). Tạo file testhundred.py cho phép kiểm tra một số nhập từ bàn phím có lớn hơn 100 hay không.

```
number = int(input("Nhap mot so nguyen: " ))
if number < 100:
    print ("So da nhap nho hon 100")
else:
    print ("So da nhap lon hon 100" )
```

Kết quả thực hiện:

Nhap mot so nguyen: 25 So da nhap nho hon 100

- Khai báo nhiều giá trị cho nhiều biến trên một dòng.

```
>>>a, b = 45, 54
>>> a 45
>>>b 54
```

1. Một số kiểu dữ liệu cơ bản: int, float, str, bool (True: 1, False: 0)

Ví dụ kiểu chuỗi:

```
mystring = "Hello World"
print(len(mystring)) #do dai chuoi, ket qua: 11
char = mystring[0]
print(char) #Ket qua: H
char = mystring[-1]
print(char) #Ket qua: d
char = mystring[-2]
print(char) #Ket qua: l
substring = mystring[1:5] #chuoi con bat dau tai vi tri 1 den vi tri 5, index bat dau la 0
print(substring) #Ket qua: ello
substring = mystring[:5] #chuoi con ben trai, 5 ky tu
print(substring) #Ket qua: Hello
substring = mystring[6:] #chuoi con ben phai, bat dau tu ky tu thu 6
print(substring) #Ket qua: World
mystring1 = mystring[::-1] #dao nguoc chuoi
print(mystring1) #Ket qua: dlroW olleH
mystring = "abcdefgh ij"
```

```
mystring1 = mystring[::2] #lay ky tu dau va cach khoang 2 ky tu
print(mystring1) #Ket qua: aceg j
mystring1 = mystring[::-2] #lay ky tu cuoi va cach khoang 2 ky tu
print(mystring1) #Ket qua: j geca
str3 = str1 + str2 #nối 2 chuỗi
….
```

2. Các toán tử (operator)

a. **Toán tử số học (arithmetic operator):** các toán tử khác ngoài các toán tử +, -, *, /

%: trả về phần dư khi chia toán tử bên trái cho toán tử bên phải (x % y)

//: Trả về phần nguyên khi chia toán tử bên trái cho toán tử bên phải (x // y)

: Toán hạng bên trái lũy thừa toán hạng bên phải (xy, = x^y)

b. **Toán tử so sánh (comparison operator)**

==: Trả về True nếu cả 2 toán hạng bằng nhau và ngược lại trả về False (x == y)

!=: Trả về True nếu cả 2 toán hạng không bằng nhau và ngược lại trả về False (x != y)

c. Toán tử logic (logical operator): and, or, not

d. **Toán tử trên bit (bitwise operaor):** thực hiện trên từng bit của toán hạng & (AND), | (OR), ~ (NOT), ^ (XOR), >> (dịch phải chuỗi bit), << (dịch trái chuỗi bit). Toán hạng sẽ được tự động được chuyển đổi để biểu diễn dưới dạng bit rồi thực hiện các toán tử trên bit.

Ví dụ:

```
x = 10 (nhị phân là 0000 1010)
y = 4   (nhị phân là 0000 0100
x = 4
y = 10
print(x & y)
print(x | y)
print(~x)
print(x ^ y)
print(x >> 2) # dịch chuyển qua phải 2 bit
print(x << 2) # dịch chuyển qua trái 2 bit
```

Kết quả thực hiện tương ứng: 0, 14, -5, 14, 1, 16

e. **Toán tử khác:**

- **is:** Trả về **True** nếu các toán hạng có giá trị được lưu trong cùng một vùng nhớ (tham chiếu đến cùng một object)
- **is not:** Trả về **True** nếu các toán hạng có giá trị không được lưu trong cùng một vùng nhớ (không tham chiếu đến cùng một object)

Ví dụ:

```
x = 10
```

```
y = 4
print(x is y)
print(x is not y)
```

Kết quả thực hiện tương ứng: False, True

3. Một số kiểu dữ liệu đặc biệt trong Python:

a. **Danh sách (list):** là kiểu dữ liệu có cấu trúc. Các giá trị trong danh sách cách nhau bởi dấu phẩy (,) và nằm trong dấu ngoặc vuông. Danh sách có thể chứa bất kỳ kiểu dữ liệu nào, ví dụ:

```
>>> a = [1, 2, 3, 'India', 'Fedora']
>>> a
[1, 2, 3, 'India', 'Fedora']
>>> a[0] 1
>>> a[4]
'Fedora'
```

Sử dụng hàm len để xác định độ dài của danh sách

```
>>>print(len(a))
>>> 5
```

Kiểm tra một phần tử thuộc danh sách sử dụng in. Ví dụ:

```
>>> a = ['Fedora', 'is', 'cool']
>>> 'cool' in a
>>> True
```

Duyệt qua danh sách trong Python bằng vòng lặp:

```
>>> a = ['Fedora', 'is', 'powerful']
>>>for x in a:
... print (x )
...
Fedora is powerful
```

Cấu trúc dữ liệu kiểu danh sách trong Python được hỗ trợ đầy đủ các phương thức: thêm phần tử vào danh sách tại ví trí bất kỳ (insert(<index>,element)), thêm vào cuối danh sách (append(element)), đếm số lần xuất hiện của một phần tử trong danh sách (count(element)), xóa một phần tử trong danh sách (del list(index); list.remove(element)), sắp xếp danh sách (list.sort())

```
L = [1, 2, 1, 'India', 'Fedora']
```

```
print(len(L))
L.insert(3, 4) # thêm phần tử 4 tại vị trí số 3
L.append("Linux")
print(L.count(2)) #đếm số lần xuất hiện của phần tử số 2 trong danh sách
del L[1] #xoá phần tử tại index 1 trong danh sách
del L[2:4] #xoá phần tử từ index 2 đến index 4 – 1
del L[:] #xoá hết phần tử trong list
L.remove(1) #xoá phần tử đầu tiên có giá trị 1 trong danh sách
L.clear() #xoá toàn bộ phần tử có trong list và làm rỗng L
L1=[8, 5, 2, 7, 9, 1]
L1.sort() #sắp xếp giá trị số theo chiều tăng
print(L1) #kết quả: >>>[1, 2, 5, 7, 8, 9]
```

***Xóa phần tử trong list python theo điều kiện**

```
mynum = list(range(10))
print(mynum) #kết quả >>> [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
xóa_số_lẻ = [num for num in mynum if num % 2 = 0]
print(xóa_số_lẻ) #kết quả >>> [0, 2, 4, 6, 8]
xóa_số_chẵn = [num for num in mynum if num % 2 = 1]
print(xóa_số_chẵn) #kết quả >>> [1, 3, 5, 7, 9]
```

…

b. **Kiểu dữ liệu tuple:** tương tự như danh sách nhưng tuples không bị thay đổi như danh sách và tuple sử dụng dấu ngoặc đơn trong khai báo trong khi danh sách sử dụng ngoặc vuông. Tuple có 2 phương thức nổi bật là đếm số lần xuất hiện của l phần tử (tuple.count(element)), và trả về phần tử tại vị trí index (tuple.index(element))

```
>>> data = (“Tom”, “England”, “Python”)
>>> name, country, language = data
>>> name # kết quả:>>> ‘Tom’
>>> country #kết quả:>>> ‘England’
>>> language # kết quả:>>> ‘Python’
 >>> print(name,country, language,sep='\n') # in xuống dòng cùng lúc 3 biến
>>>print(data.count("Ton")) #kết quả: 0
```

```
>>>print(data.count("Tom")) #kết quả: 1
>>>print(data.index("Python")) #kết quả: 2
#thêm phần tử vào trong tuple, sử dụng toán tử +=,
#ví dụ: new_tuple +=(value,)
data += (1,)
print(data) #kết quả: ('Tom', 'England', 'Python', 1)
data += ("abc",)
print(data)#kết quả: ('Tom', 'England', 'Python', 1, 'abc')
```

c. **Tập hợp (Set):** là kiểu cấu trúc dữ liệu có đặc điểm

1. **Không có thứ tự:** mỗi phần tử sẽ ở thứ tự khác nhau khi truy cập vào nó, không có chỉ số để xác định vị trí của phần tử và không thể truy cập thông qua chỉ số.
2. **Bất biến:** nghĩa là không thể thay đổi giá trị của các phần tử (không thể sửa các nội dung của phần tử), giống với Tuple nhưng Set cho phép thêm hoặc loại bỏ các phần tử khỏi nó.
3. **Không có sự lặp lại của các phần tử**

- Tạo tập hợp set: sử dụng dấu {} hoặc sử dụng hàm set()

```
set1 = {"Python", 1, 2, 3, 4}
print(set1) #kết quả:{1, 2, 3, 4, 'Python'}
set2 = set(("C++", 4,6,8)) #ép kiểu set từ kiểu tuple
print(set2) #kết quả: {8, 4, 'C++', 6}
set3 = set([4, 1,2,3,1,2,3,4]) #ép kiểu set từ kiểu list
print(set3) #kết quả: {1, 2, 3, 4}
a = set('abcthabcjwethddda')
print(a) #kết quả: {'b', 'e', 'c', 'a', 'w', 'j', 'h', 'd', 't'}
```

- Xác định số phần tử trong set:

```
print(len(a)) #, kết quả: 9
```

- Truy cập và duyệt các phần tử trong tập hợp: thực hiện vòng lặp duyệt qua toàn bộ các phần tử

```
b= {"Python", "C++", "Java"}
for i in b:
    print(i)
```

- Thêm phần tử vào trong Set: sử dụng phương thức add() để thêm 1 phần tử và update() để thêm nhiều phần tử, phương thức update có thể lấy Tuple, List, String, Set khác làm đối số, nếu giá trị giống nhau sẽ bị loại bỏ.

  b.add(3) #kết quả: {'Python', 3, 'Java', 'C++'}

  b.update([1,2],{4, 5, 7,1}) #kết quả: {1, 'Python', 3, 'C++', 2, 4, 5, 7, 'Java'}

- Xoá phần tử khỏi Set: discard() giữ nguyên một Set nếu phần tử không tồn tại, hoặc remove() đưa ra thông báo lỗi nếu phần tử không tồn tại; xoá bỏ toàn bộ phận tử clear(), del; xoá và trả lại 1 phần tử bất kỳ pop()

  b.discard(1)

  b.discard("Java")

  b.remove(2)

  b.clear()

  del b

  pt=a.pop()

- Sao chép đối tượng trong Set:

  c=b.copy()

  d=set(b)

- Set hỗ trợ đầy đủ các phép toán trên tập hợp bao gồm: phép toán or (union()), phép toán and (intersection()), phần bù (difference()), kiểm tra tập con (issubset()),

  a={1,2,3,4}

  b={2,4,6,8,10}

  c= a.union(b)

  d=a.intersection(b)

  d=a.difference(b)

  print(a.issubset(b) # a là tập con của b thì trả về True

  ...

d. Tự điển (Dictionary)

Kiểu cấu trúc dữ liệu tự điển là tập các cặp giá trị <key:value> cách nhau bởi dấu phẩy (,) đặt trong dấu ngoặc {}, không theo thứ tự và key là duy nhất.

Lưu ý:

- Khóa của Dictionary có thể là chuỗi hoặc số, khóa này là duy nhất ở từng cấp.

- Giá trị của Dictionary có thể chuỗi, số hoặc các cấu trúc List, Tuple, Set hoặc thậm chí là Dictionary.

Ví dụ:

>>> data = {'kushal':'Fedora', 'kart_':'Debian', 'Jace':'Mac'}

>>> data

{'kushal': 'Fedora', 'Jace': 'Mac', 'kart_':'Debian'}

>>> data[‘kart_’] ‘Debian’

Để kiểm tra một key có trong tự điển hay không, dùng in

>>> ‘Jace’ in data

>>> True

Cấu trúc kiểu tự điển có thể được tạo ra từ hàm dict() với đối số là tuple của key và value

>>> dict((('Indian','Delhi'),('Bangladesh','Dhaka')))

{'Indian': 'Delhi', 'Bangladesh':'Dhaka'}

Để duyệt qua tự điển, dùng phương thức items()

>>> data

{'Kushal': 'Fedora', 'Jace':'Mac', 'kart_':'Debian', 'parthan':'Ubuntu'}

>>> for x, y in data.items():

... print ("%s uses %s" % (x, y))

...

Kushal uses Fedora Jace uses Mac kart_ uses Debian

parthan uses Ubuntu

Thêm một key vào tự điển

```
1  d = {0:10, 1:20}
2  print(d)
3  d.update({2:30})
4  print(d)
```

Kết quả

```
{0: 10, 1: 20}
{0: 10, 1: 20, 2: 30}
```

Ghép các tự điển thành một

```
dic1={1:10, 2:20}
dic2={3:30, 4:40}
dic3={5:50,6:60}
dic4 = {}
for d in (dic1, dic2, dic3): dic4.update(d)
print(dic4)
```

```
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
```

Kiểm tra key đã có trong tự điển chưa

```
d = {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
def is_key_present(x):
  if x in d:
      print('Key is present in the dictionary')
  else:
      print('Key is not present in the dictionary')
is_key_present(5)
is_key_present(9)
```

```
Key is present in the dictionary
Key is not present in the dictionary
```

Xoá một key khỏi tự điển

```
1  myDict = {'a':1,'b':2,'c':3,'d':4}
2  print(myDict)
3  if 'a' in myDict:
4      del myDict['a']
5  print(myDict)
```

```
{'c': 3, 'b': 2, 'd': 4, 'a': 1}
{'c': 3, 'b': 2, 'd': 4}
```

Ghép 2 danh sách thành tự điển

```
1  keys = ['red', 'green', 'blue']
2  values = ['#FF0000','#008000', '#0000FF']
3  color_dictionary = dict(zip(keys, values))
4  print(color_dictionary)
```

```
{'green': '#008000', 'blue': '#0000FF', 'red': '#FF0000'}
```

Tham khảo một số hàm của một số kiểu dữ liệu đặc biệt trong Python: https://docs.python.org/3/tutorial/datastructures.html

Bài tập: Viết chương trình loại bỏ các phần tử trùng trong tự điển

2. Hàm:

Sử dụng từ khóa def để định nghĩa hàm.

def functionname(params):
statement1 statement2

Ví dụ:

def tinhtong(a, b):
return a + b

Thực thi:

>>> res = tinhtong(23, 3)

>>> res 26

3. Lớp

Cú pháp khai báo class như sau:

```
class nameoftheclass(parent_class):
    statement1
    statment2
    statement3
```

__init__(2 dấu '_' liên tiếp) là một phương thức đặc biệt trong class Python, dùng để khởi tạo một lớp. Phương thức này được gọi khi một đối tượng được khởi tạo. Ví dụ sau tạo ra một lớp Student với tên, ngành học và năm

```
class Student(object):
# tra ve doi tuong Student voi ten, nganh hoc, va nam
def   __ini__(self, name, branch, year):
   self.name = name
   self.branch = branch
   self.year = year
   print ("A student object is created" )
def print_details(self):
   #in thong tin sinh vien
   print ("Name:", self.name )
   print ("Branch: ", self.branch )
   print ("Year: ", self.year )
>>> std1 = Student('Kushal','CSE','2005')
   A student object is created
>>> std1.print_details()
   Name: Kushal
   Branch: CSE
   Year: 2005
```

Self là một biến đặc biệt trong Python, dùng chỉ đối tượng hiện tại (giống this trong C++)

BÀI TẬP

1. Viết hàm hoán đổi 2 ký tự đầu tiên của 2 chuỗi đơn và sinh ra chuỗi mới từ 2 chuỗi kết quả (cách nhau bằng khoảng trắng).

Ví dụ:

Input: ‘abc’ ‘efg’

Output: ‘efc abg’

2. Viết hàm loại bỏ các ký tự tại vị trí index chẵn của một chuỗi cho trước Ví dụ:

Input: ‘python’

Output: ‘yhn’

3. Viết hàm đếm số lần xuất hiện của các từ trong câu Ví dụ:

Input: ‘no pain no gain’

Output: {‘no’: 2, ‘pain’: 1, ‘gain’: 1}

4. Viết hàm mã hóa một chuỗi bằng cách dịch chuyển các ký tự sang trái 3 bước. Ví dụ: d->a, e->b, f->c

Input: ‘def’

Output: ‘abc

5. Viết hàm kiểm tra một chuỗi hợp lệ hay không? (có được sinh từ bộ chữ cái nhập) Ví dụ: bộ chữ cái nhập S = {0,1,2}

Các chuỗi hợp lệ: 0011, 12, 012, 122…

Các chuỗi không hợp lệ: 123, 24,…

6. Viết hàm chuyển một chuỗi thành danh sách Ví dụ:

Input: ‘This is a list’

Output: [‘This’, ‘is’, ‘a’, ‘list’]

7. Viết hàm in ra ký tự không lặp lại đầu tiên trong chuỗi Ví dụ:

Input: ‘abcdef’

Output: ‘a’

Input: ‘abcabcdef’

Output: ‘d’

Input: ‘aabbcc’

Output: ‘None’

8. Viết hàm loại bỏ khoảng trắng trong chuỗi cho trước Ví dụ:

Input: ‘a b c’

Output: ‘abc’

9. Viết hàm in ra từ được lặp lại đầu tiên trong chuỗi Ví dụ:

Input: ‘ab ca bc ca ab bc’

Output: ‘ca’

Input: ‘ab ca bc ab’

Output: ‘ab’

10. Viết hàm tìm độ dài tối đa của chuỗi con gồm các số 0 liên tiếp trong chuỗi nhị phân cho trước.

Ví dụ:

Input: ‘1110001110000’

Output: 4

BÀI TẬP NÂNG CAO

Bài tập 1

Kiểm tra 2 danh sách khác nhau có thể sinh ra cùng một chuỗi hay không?

Input danh sách A = (s1, s2, …, sn), danh sách B = (t1, t2, ...tm)

Output: Có, nếu tồn tại một chuỗi được sinh ra từ danh sách A và danh sách B

Ví dụ:

Input: A = (110,0011,0110) và B = (110110,00,110).

Output: có vì s2s3s1 = t2t3t1

Kiểm tra các trường hợp sau:

1. A = (0011,11,1101), B = (101,011,110)
2. A = (100,0,1), B = (1,100,0)

(Gợi ý: sử dụng itertools (import itertools))

Bài tập 2:

Chuỗi DNA được tạo thành từ sự kết hợp của các codon, là 3 ký tự khác nhau bất kỳ thuộc bộ ký hiệu {A,C,G,T}, ví dụ ACT, ACG, TCG... Một đoạn gen là tập hợp của ít nhất 3 codon bắt đầu là ATG và kết thúc bằng TAA, TAG hoặc TGA.

Hãy viết chương trình kiểm tra 1 đoạn gen có hợp lệ hay không?

Ví dụ:

Input: ATGCCCTAG

Output: không hợp lệ

Input: ATGCGTTGA

Output: hợp lệ