Pytuto (Python Tutorial) 2nd assignment.

Hyungwon Yang
Hyung8758@gmail.com
EMCS Lab

This task was designed for those who just started python. I wish them try to complete the task and improve their python skill. Please search as many solutions as you can through the internet and ask questions to me and your teammates to tackle down the tasks. However, DO NOT share codes. Good luck.

Introduction

Flow Control

There are a number of ways to control computers. Let's say you want to make computer print 10 sentences. In this case, you can write 10 sentences in a script as follows.

```
>>> print 'Welcome'; print 'Welcome'; print 'Welcome'; print 'Welcome'; print 'Welcome'; print 'Welcome'; print 'Welcome'; print 'Welcome'; print 'Welcome'; print 'Welcome';
```

Or you can assign the sentence as a variable.

```
>>> mySentence = 'Welcome'
>>> print mySentence * 10
```

However, what if sentences contain consecutive numbers or the work that you want to do is much more complicated. The most powerful and efficient way to handle this kind of task is using loops.

Loops, such as for and if, are very useful tools when computer programmers assign tasks to computer. Therefore, they are ubiquitous in scripts, modules, packages, and functions. Why they are useful? Let me give you examples to answer that question. You can do the task above with for loop as follows.

```
>>> for i in 10:
>>>     print 'Welcome'
```

- Loop syntax
  1. When you start any loop, you have to mark ':' at the end so that the computer would notice that indentation will start.
  2. Identical numbers of indented lines (if 3 lines are indented with 1 tab key, then that 3 lines have identical numbers of indentation) belong to the line in

which colon starts. If a line didn't start with indentation or has different numbers of indentation from the above, it is independent or dependent on higher line.

ex) 1.for A:
for B:
C : B belongs to A but C is independent from A and B.

2. for A:
for B:
C
D : Even though D has a different numbers of indentation from the above, it is dependent on A, B (higher lines.) and C

That's it! It is pretty simple and intuitive. But loops are more prominent when you do more complicated tasks such as putting consecutive numbers, or displaying multiple numbers.

```
>>> for first_count in range(1,11):
>>>     for second_count in range(1,11):
>>>         print 'I say %d and %d' %(first_count,second_count)
```

* Please run this in python command line and see the result.
(More loop information is here: https://docs.python.org/2/tutorial/controlflow.html)

Multiple loop combination (2 for loops and 1 if loop) is also possible. However please be careful when you use multiple loops in your script because it loses script readability (which means your script is hard to understand) and sometimes you will be confused with the script you wrote. If you have to write many loops in your script please insert '#' comment lines and separate loops as many as possible.

There is one thing you have to remember in python is indentation. Indentation is one of the syntax rules in python. You may use space bar or tab key to put indentation and python will understand it as long as your indentation usage is consistent.

```
>>> for i in range(10):
>>>     print 'Welcome to Python'
>>>     print 'I love Python'
>>> print 'Do you like Python?'
```

* Use tab key to put indentation. You can also use space bar.

As you can see above, it follows indentation rule correctly. First 3 lines involve in for loop but last 4th line is not belonged to for loop. 4th line print 'Do you like Python?' is separated line from the for loop and it is decided by indentation.

Task

A) Simple flow control (for and if loops)

- 1. Generate a string variable ‘Korea university’.
  2. Use for loop so as to print each letter in the screen.
  ex) K, o, r, …, t, y.

- 1. Make a vector that contains 1000 random numbers. (normally distributed)
  2. Select only even number index values from the vector.
  ex) [1,2,3,4,5,6,7,8,9,10] => use only: 2,4,6,8,10
  3. Compare first and second values and save the larger one.
  ex) compare 2nd and 4th values, 6th and 8th values and so on.

B) Advanced flow control (for, if and while loops)

- 1. Use numpy.random.randint to make a vector that contains 100 random integers. (set min=1, max=50)
  2. Add up those integers from the 1st to the end consecutively.
  3. While adding up those integers, if the sum is over 100 then get the indices from the first and the last random integers and then start adding up again from the next integers
  ex) There are 10 random integers=[20,30,10,10,10,50,20,20,30,40]. Add up from the first 20+30 and it is 50. You have to find first and last indices that are 1 and 2. Start again. 10+10+10+50 and it is 80. (you cannot get 50 exactly in this case) You have to find first and last indices that are 3 and 6. Start again until you get through the all numbers.
  4. Print the indices.
  look) numpy.random

  (You don’t need to use regular expression on this task.)
- 1. Use open function to import ‘Obama_Address.txt’.
  2. Split whole sentences into words.
  ex) ‘My name is hyungwon Yang’ => ‘My’,’Name’,’is’,’hyungwon’,’Yang’
  3. Use len function to get the each word length and make length groups such as a two-characters word group, a five-characters word group.
  4. Classify words based on their length and put them into identical length groups.
  ex) ‘cat’, ’hat’ and ‘power’, ’stake’ are two and five characters groups respectively.
  5. Print the list and make sure to close the file that was imported.
  look) Search how to import text file in python. ‘with’ function is useful.

If you finish your assignment, please send the code to 'hyung8758@gmail.com'. The script name should be '?th Assignment_YourName.py'. In the code script, the assignment number (e.g, 1st assignment), your name and email address should be written in the first line. Ask questions and give comments to me. It is always welcome.

Weekly Tips

- 리스트 list : 리스트는 값들의 나열이다. 순서가 존재하며, 여러 종류의 값을 담을 수 있다. 또한 문자열과 마찬가지로 0 부터 인덱스를 가지고 있으며 슬라이싱도 가능하다.
  1. colors = ['red','green','gold']
  2. colors.append('blue'); colors; ['red','green','gold','blue']
  3. colors.insert(1,'black') insert 를 통해 원하는 위치에 넣을 수 있다.
  4.colors.count('blue') 해당 아규먼트의 갯수값을 반환한다.
  5.colors.pop()을 하면 맨 뒤에서부터 값을 뽑아서 보여준다. 그와동시에 그 값은 리스트에서 사라진다.

- 셋트 set : 수학 집합과 비슷하다. 셋트는 리스트와 마찬가지로 값들의 모임이며, 순서는 없다. 값이 중복되지 않는다.
  1. a = {1,2,3}
  2. b = {3,4,5}
  3. 합집합 a.union(b) or a | b ; {1,2,3,4,5}
  4. 교집합 a.intersection(b) or a & b; {3}
  5. 차집합 a - b ; {1,2}
  5. d = {1,2,3,3,4,5}; {1,2,3,4,5}

- 튜플 tuple : 리스트와 유사하나, 리스트와 달리 [] 대신 ()로 묶어서 표현하며, 읽기전용이다. 읽기 전용인 만큼 제공되는 함수도 리스트에 비해 적지만, 속도는 그만큼 빠르다.
  1. t = (1,2,3)
  2. a,b = 1,2
  3. (a,b) = (1,2)
  4. a,b = b,a

- 사전 dictionary : 알아두면 정말 편리한 자료구조이다. 사전은 키(keys)와 값(values)의 쌍으로 이루어져 있다. 키값을 통해서 값을 가져올 수 있으며, 인덱스는 지원하지 않는다. 사전이 값을 추가하려면 새로운 키와 값을

할당하면 되고, 변경도 변경하려는 항목의 키에 변경할 값을 할당하면 된다. 사전의 내용을 얻기 위해서는 items(), keys(), values() 메서드를 사용하면 된다.

1. d = dict(a=1, b=2, c=3); d; {'a'=1, 'c'=3, 'b'=2}
2. color = {'apple='red', 'banana'='yellow', 'watermelon'='green'}
3. color['apple']; 'red'

- 부울 bool : 부울은 참과 거짓을 나타내는 자료형으로 가능한 값은 True 와 False 뿐이다.

  논리연산자 : and(&), or(|), not

  수치를 논리연산자에 사용하는 경우, 0 은 False 로 간주하고 음수를 포함한 다른 값을 모두 True 로 간주한다. 문자열을 논리연산자에 사용하는 경우에도 " 빈문자열만 False 로 본다. 값이 없는 상태를 나타나는 None 의 경우에도 False 로 간주한다.

  1. bool(0); False
  2. bool(-1) : True
  3. bool('test'); True
  4. bool(None) ; False