Learning Script - 3

Python 2

보안기술 본부 김윤경

2018.11.12

openbase

The subject comes first, the medium second.

주제가 제일 중요하고 도구는 그 다음이다.

- Richard Prince

스크립트는 단지 하나의 도구입니다.
가장 중요한 것은 원래 하려는 목표를 정확히 이해하고, 관련 지식을 갖춰 적합한 과정으로 수행하는 것입니다.
스크립트는, 할 일의 프로세스가 결정되고 검증된 후, 그 일에 도움을 주려고 사용하는 도구중 하나입니다.

Python – 추천 강좌 활용

openbase

스크립트 활용에 필요한 핵심 지식:

- 자료형, 제어구조, 입출력

- 점프 투 파이썬(https://wikidocs.net/book/1) 2, 3, 4장 추천

Python - 이전시간 복습

openbase

이전 시간에 간단한 자료형 및 제어구조를 살펴보고, 그것들을 이용해 whois 조회를 해 봤습니다.

- 자료형: list, set, string, number
- 제어구조: for 문
- 함수(function 또는 method):

함수	분류	설명
str()	built-in method	객체를 문자열 형태로 변환하여 리턴
set()	built-in method	주어진 iterable을 set으로 변환하여 리턴, 또는 빈 set 자료형 리턴
sum()	built-in method	주어진 iterable의 요소값의 합을 리턴
sorted()	built-in method	주어진 iterable을 정렬하여 리턴
range()	built-in method	주어진 옵션에 맞는 일련의 숫자 리턴
len()	built-in method	객체의 요소 개수 리턴. 문자열의 경우 문자열 길이.
string.format()	string method	문자열 포맷 지정
string.split()	string method	주어진 구분자로 문자열을 나눠서 문자열 배열을 생성하여 리턴
list.sort()	list method	리스트 요소를 정렬하여 재배열

Python 기본

openbase

Python 버전 선택

- 불가피한 경우를 제외하고 반드시 3을 사용합니다.
- 버전 2를 써야 하는 경우

  (1) 스크립트를 실행해야 할 환경이 Python 버전2 인데 업그레이드 할 수 없다.

  (2) 이미 쓰고 있는 스크립트를 수정하려고 하는데, 그 스크립트가 Python 버전2 이다. -> 버전3으로 수정할 수 있는 분량이라고 하더라도 수정 후 기존 동작을 검증할 자신이 없으면 바꾸지 않는 게 좋음.

인코딩: 스크립트 파일은 UTF-8로 작성합니다.

원칙적으로는 소스 코드의 인코딩 타입을 소스 상단에 아래와 같이 기재해야 합니다.

```
# -*- coding: utf-8 -*-
# -*- coding: euc-kr -*-
```

기본 영문자(라틴문자)만으로 구성된 경우는 인코딩을 명시하지 않아도 문제가 되지 않지만, 영문자 외 문자를 소스코드에 사용한 경우(예: 한글로 주석을 달아 놓은 경우) 실행이 되지 않을 수 있습니다. 그런데, Python v3에서는 기본 인코딩이 UTF-8이므로 UTF-8로 코드를 작성하면 인코딩 타입을 명시하지 않아도 됩니다.

다수의 한글 지원 에디터들이 UTF-8로 파일을 저장하므로 문제가 없지만 윈도우 메모장으로 스크립트를 만들면서 한글을 쓰면 실행되지 않습니다.

openbase

들여쓰기

다른 개발 언어에서 들여쓰기는 가독성을 높이기 위한 권장 사항입니다. 그러나 Python에서는 들여쓰기 자체가 문법의 일부입니다.

if 조건문, while 반복문, for 반복문, 함수 등 특정 영역에 속하는 코드 블럭은 들여쓰기 해야 합니다. 특정 블럭 내 다시 하위 블럭이 있으면 한 단계 더 들여 씁니다.

다음은 들여쓰기를 두 단계로 한 예시 입니다.

```
result = []
for i in range (0, 20):
    print("current number = {}".format(i))
    if i%4 == 0:
        print("4의 배수 발견: {}".format(i))
        result.append(i)
```

```
1. 1~19까지 숫자를 돌며 찍는 for loop 블럭
2. 그 안에 4의 배수이면 처리 하는 if 블럭
   - % 연산자: 나머지를 구한다.
   - 4의 배수가 발견되면 프린트하고,
   - result라는 리스트에 넣는다.
```

중요!! 들여쓰기를 잘못해서 코드가 다른 블럭에 포함되지 않도록 주의해야 합니다.

openbase

들여쓰기

들여쓰기는 일정한 간격으로 하면 어떤 폭으로 해도 상관없습니다. 보통 탭, space 2개, space 4개 중 하나로 하는데, space 4를 권장합니다.

공백 2칸

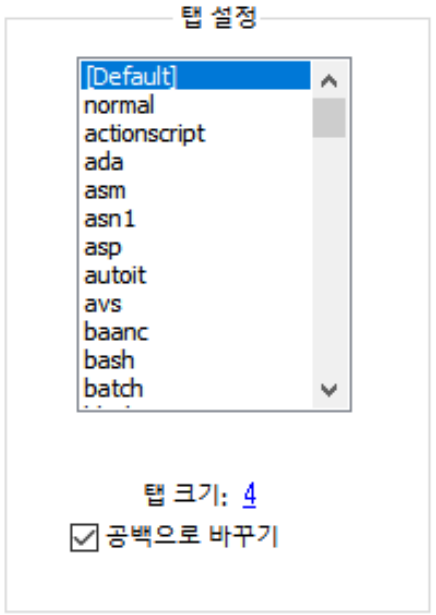
```
if a == 10:
  print('10입니다.')
```

공백 4칸

```
if a == 10:
    print('10입니다.')
```

탭 1칸

```
if a == 10:
	print('10입니다.')
```



* 다수의 에디터 들이 탭키 입력을 공백으로 대체하는 옵션을 갖고 있습니다.

들여쓰기

참고로 자바에서의 들여쓰기를 잠시 보면, 들여쓰기를 어떻게 해도 블럭 구조만 맞으면 문제가 되지 않습니다.
아래 코드는 둘다 1~5사이 정수를 더하고 나서 결과를 출력하는 코드입니다.
좌측은 예쁘게 들여쓰기가 됐고 우측은 들쭉날쭉합니다만, for loop의 { } 블럭 구성에 문제가 없기 때문에 결과는 같습니다.

```
int sum = 0;
for (int i = 1; i <= 5; i++) {
    sum += i; //sum = sum + i
}
System.out.println(sum); //합계를 출력
```

```
int sum = 0;
for (int i = 1; i <= 5; i++) {
sum += 0; }
     System.out.println(sum); //합계를 출력
```

Python 기본

openbase

주석

- 한 줄 주석: # ~~~
- 여러 줄 주석 : ''' ~~~ ''' 또는 """ ~~~ """

```
import datetime

"""
def print_current_time():
    cur_time = datetime.datetime.now()
    print("current time: {}".format(cur_time))
"""

'''
def print_yesterday():
    yesterday = datetime.datetime.now() - datetime.timedelta(days=1)
    print("yesterday: " + yesterday.strftime("%Y/%m/%d"))
'''

#print_current_time()
```

openbase

함수, import의 위치

함수, import의 정의는 어느 곳에 할까요. 사용하기 전에만 하면 됩니다. 함수는 그 함수를 호출하기 전에 하고, import도 import 한 모듈을 쓰기 전에 하면 됩니다. 그러나 보통 import는 스크립트 시작부분에 합니다. 함수는 메인 스크립트 사이사이에 두지 않고 앞쪽 부분에 모아놓습니다.

왼쪽 보다는 오른쪽과 같이 작성합니다.

그리고 왼쪽과 같이 impor를 함수 안에 하면 다른 블럭에서 사용시 또 import 해야 합니다.

```
def print_current_time():
    import datetime
    cur_time = datetime.datetime.now()
    print("current time: {}".format(cur_time))

print_current_time()

def print_yesterday():
    import datetime
    yesterday = datetime.datetime.now() - datetime.timedelta(days=1)
    print("yesterday: " + yesterday.strftime("%Y/%m/%d"))

print_yesterday()
```

```
import datetime

def print_current_time():
    cur_time = datetime.datetime.now()
    print("current time: {}".format(cur_time))


def print_yesterday():
    yesterday = datetime.datetime.now() - datetime.timedelta(days=1)
    print("yesterday: " + yesterday.strftime("%Y/%m/%d"))


print_current_time()
print_yesterday()
```

openbase

```
#!/usr/bin/python

print("main 1")

def function1():
    print("function1")
    function3()

print("main 2")

def function2():
    print("function2")

function2()
print("main 3")

def function3():
    print("function3")

function1()
```

실행 순서

스크립트의 실행 순서를 알아 봅시다.

왼쪽 스크립트의 실행 결과 출력은 어떻게 될까요?

Python 기본

openbase

```
#!/usr/bin/python

print("main 1")

def function1():
    print("function1")
    function3()

print("main 2")

def function2():
    print("function2")

function2()
print("main 3")

def function3():
    print("function3")

function1()
```

실행 순서

스크립트의 실행 순서를 알아 봅시다.
왼쪽 스크립트의 실행 결과 출력은 어떻게 될까요?
함수는 호출되지 않으면 실행되지 않습니다.

```
main 1
main 2
function2
main 3
function1
function3
```

openbase

```
#!/usr/bin/python

print("main 1")

def function1():
    print("function1")

print("main 2")

def function2():
    print("function2")
    function3() # function1에서 이동

function2()
print("main 3")

def function3():
    print("function3")

function1()
```

실행 순서

왜 function2()에서 부르면 오류가 날까요?

```
main 1
main 2
function2

NameError: name 'function3' is not defined
```

스크립트는 읽으면서 실행됩니다. function1()은 호출위치가 맨 끝이므로 function3()을 알고 있는 상태에서 호출됩니다.

반면에 function2()호출시에는 아직 function3()을 알 수 없습니다.

Python - 딕셔너리 & JSON

openbase

딕셔너리(Dictionary)

- key: value로 구성된 자료구조입니다.
- 기본적으로 key를 이용해서 value를 액세스합니다.
- 표현형태상(외관상)으로는 json과 유사합니다. 실제로 json 데이터와 딕셔너리 사이의 변환이 가능합니다.
  (* 대체적으로 가능하지만 변환이 불가능한 경우도 있습니다.)
- 상세한 설명은 다음 페이지로 대체
  https://wikidocs.net/16#_1

Python - 딕셔너리 & JSON

openbase

JSON (JavaScript Object Notation) 모듈과 딕셔너리

- 딕셔너리 -> JSON : dumps(), dump()
- JSON -> 딕셔너리 : loads(), load()

```
import json

dic1 = {'name':'ykkim', 'birth':'1002', 'age': 33}
print('dict original ==> name: {}, birth: {}'.format(dic1.get('name'), dic1.get('birth')))

json_string = json.dumps(dic1)
print('json_string: ' + json_string)

dic_comeback = json.loads(json_string)
print('dic_comeback  ==> name: {}, birth: {}'.format(dic_comeback.get('name'), dic_comeback.get('birth')))
```

- 결과

```
dict original ==> name: ykkim, birth: 1002
json_string: {"name": "ykkim", "birth": "1002", "age": 33}
dic_comeback  ==> name: ykkim, birth: 1002
```

Python - 딕셔너리 & JSON

openbase

JSON (JavaScript Object Notation) 모듈과 딕셔너리

- 파일에서 read/write: load() & dump()

```
import json



with open('mydata.json') as f:
    dic_data = json.load(f)

print('dic_data  ==> name: {}, birth: {}'.format(dic_data.get('name'), dic_data.get('birth')))

with open('back.json', 'w') as of:
    # json.dump(dic_data, of)
    json.dump(dic_data, of, indent = 4)
```

실습 - whois 조회

openbase

- 파일로 주어진 다수의 도메인을 읽어서,
- WHOIS 정보를 조회하여,
- 필요한 항목만 추출하고
- csv 파일로 저장하는

실습입니다.

이번 시간 실습은 다음 페이지를 참고합니다.

https://secuwave.github.io/secure3/learn_script/03/exercise00.html

(참고-스터디메인: https://secuwave.github.io/secure3/learn_script/main)


자료 :
https://secuwave.github.io/secure3/learn_script/main
openbase