python

prepare

get python

python official site

settings

- Editors
  - Visual Studio Code
  - Vim
  - ~~Sublimetext~~
  - ~~pyCharm~~

```
$ python3 hello_world.py
```

variables and types

- 변수 이름은 snake_case로
- 문자열은 “This is a string”, 'This is also a string.' 모두 가능
- {string_variable}.title() → 첫 글자 대문자로
- {string_variable}.upeer()
- {string_variable}.lower()
- f-string
  - full_name = f“{first_name} {last_name}”
  - full_name = “{} {}”.format(fist_name, last_name)
- escape characters; \', \\, \n, \r, \t, \b, \f, \ooo, \xhh
- {string_variable}.rstrip() → 오른쪽 공백 삭제, .strip() → 양쪽 공백 제거, .lstrip() → 왼쪽 공백 제거
- numeric; integer, floating point number, underscored number → 15_000_000_000
- 상수는 대문자로; MAX_CONNECTIONS = 5000
- comments; #

```
message = "Hello Python"
print(message)
```

```
>>> import this
```

list

- 대괄호 []와 콤마 ,
- 이름을 복수형으로
- 인덱스는 0부터, [-1]은 끝에서부터의 인덱스

```
bicycles = ['trek', 'cannondale', 'redline', 'specialized']
print(bicycles)

print(bicycles[0])
print(bicycles[0].title())

bicycles[0] = 'mini bell'  # change value
bicycles.append('trek')  # append value
bicycles.insert(0, 'strida')  # insert value

del bicycles[0]  # delete
popped_bicycle = bicycles.pop()  # pop value (stack)
#popped_bicycle = bicycles.pop(0)  # pop value by index
bicycles.remove('redline')  # remove by value

bicycles.sort()  # sort permanently by alphabet
bicycles.sort(reverse=True) # reverse sort
print(sorted(bicycles))  # sort temporarily
bicycles.reverse()  #

len(bicycles)
```

list handling

```
magicians = ['alice', 'david', 'carolina']
for magician in magicians:
    print(magician)
for value in range(1, 5):  # 1 ~ 4
    print(value)
numbers = list(range(1, 6))  # numbers = [1, 2, 3, 4, 5]
even_numbers = list(range(2, 11, 2))  # even_numbers = [2, 4, 6, 8, 10]

squares = []
for value in range(1, 11):
    square = value ** 2
    squares.appen(square)
```

```
# squares = [value ** 2 for value in range(1,11)]
print(squares)

digits = [1, 2, 3, 4, 5, 6, 7, 8, 9, 0]
min(digits)
max(digits)
sum(digits)
```

- slice; 원래 리스트 유지
  - players[0:3] # index 0에서부터 3개까지 자름 [:3]
  - players[2:] # index 2부터 끝까지
  - players[-3:] # 끝에서 셋
- 복사
  - my_friends = your_friends[:]
  - my_friends = your_friends는 포인터 같은 역할
- tuple(immutable)
  - 대괄호 대신 소괄호 ()와 콤마 ,
  - 한 개 항목의 튜플이더라도 콤마 필요; my_t = (3,)
- Coding style; PEP(Python Enhancement Proposal) 8
  - 들여쓰기 공백 네 칸
  - 행 길이 79자
  - PEP 8 -- Style Guide for Python Code

if state

```
cars = ['audi', 'bmw', 'subaru', 'toyota']

for car in cars:
    if car == 'bmw':
        print(car.upper())
    else:
        print(car.title())
```

- == ; equals to
- != ; not equals
- <, ⇐, >, >=
- and, or
- in; 'mushrooms' in requested_toppings
- not in;
- boolean expression; True, False
- if, if-else, if-elif-else

```
if age < 4:  # GOOD
```

```
if age<4:  #bad
```

dictionary

- key-value pair
- 중괄호 {}

```
alien_0 = {'color': 'green', 'points': 5}

print(alien_0['color'])
print(alien_0['points'])

alien_0['x_position'] = 0
alien_0['y_position'] = 25

dict_0 = {}  # 빈 값으로 딕셔너리 선언

del alien_0['points']  # points key 제거

favorite_languages = {
    'jen': 'python',
    'sarah': 'c',
    'edward': 'ruby',
    'phil': 'python',
    }
point_value = alien_0.get('points', 'No point value assigned.')  # key가 없을 경우의 default 값을 지정하여 error 방지

user_0 = {
    'username': 'efermi',
    'first': 'enrico',
    'last': 'fermi',
    }

for key, value in user_0.items():
    print(f"\nKey: {key}")
    print(f"Value: {value}")
for name in favorite_languages.key():
    print(name.title())
for language in favorite_languages.values():
    print(language.title())
for language in set(favorite_languages.values()):  # set는 중복을 제거한 고유한 데이터 형식
    print(language.title())
```

- nesting

    - 딕셔너리 안에 리스트, 리스트 안에 딕셔너리, 딕셔너리 안에 딕셔너리…

user input and while loop

- 사용자 입력

```
message = input("Tell me something and I will repeat it back to you: ")
print(message)

prompt = "If you tell us who you are we can personalize the messages you
see."
prompt += "\nWhat is you first name? "

name = input(prompt)
print(f"\nHello, {name}!")
```

- 형변환; int(string)
- modulo operator %
- while loop

```
current_number = 1

while currnet_number <= 5:
    print(current_number)
    current_number += 1
```

```
message = ""

while message != 'quit':
  message = input(prompt)
  print(message)
```

- flag; bool 변수를 이용하여 while 처리
- break; 루푸 빠져 나가기
- continue; 루프의 처음으로
- 무한 루프 주의

function

```

```

```
def greet_user():  # 함수 정의
  """간단한 환영 인사를 표시합니다"""  # document string 혹은 docstring 따옴표 세개"""
로 둘러쌈
    print("Hello")
greet_user()

def describe_pet(animal_type, pet_name='dog'):  # 기본 값이 있는 파라메터는 뒤쪽에
  """반려동물에 관한 정보를 출력합니다"""
    print(f"\nI have a {animal_type}.")
    print(f"My {animal_type}'s name is {pet_name.title()}.")
    #return 값
describe_pet('hamster', 'harry)
describe_pet(animal_type='hamster', pet_name='harry')
describe_pet(pet_name='harry', animal_type='hamster')
describe_pet('willie')
```

- function에 리스트를 넘길 때 function_name(list_name[:])로 사본을 넘기면 원래 값이 유지 (Call by value), 리스트를 넘기면 원래 리스트가 수정됨 (Call by reference).
- 함수 정의/호출시 기본값 지정할 때 공백을 쓰지 않는다

```
def make_pizza(*toppings):  # 여러 개의 매개변수 빈 튜플
def build_profile(first, last, **user_info):  # 빈 딕셔너리
```

- 함수를 모듈로 저장
    - 소문자

```
import pizza  # pizza.py 파일을 열고 그 파일에 있는 모든 함수 사용

pizza.make_pizza()  # pizza 모듈 안의 make_pizza() 함수 호출

# 모듈에서 일부 함수만 사용
from module_name import function_name
from module_name import function_0, function_1, function_3

# 별칭; 함수의 별칭으로 호출
from pizza import make_pizza as mp
from module_name import function_name as fn

# 모듈의 별칭
import pizza as p
import module_name as mn

# 모든 함수 비추천
from pizza import *
from module_name import *
```

class

```
class Dog:  # 대문자로 시작
  """개를 모델화하는 시도"""  # docstring
    def __init__(self, name, age):  # 생성자 메서드, 반드시 구현
        ""name과 age 속성 초기화"""
        self.name = name
        self.age = age
    def sit(self):
   """명령에 따라 앉는 개"""
        print(f"{self.name} is now sitting.")
    def roll_over(self):
   """명령에 따라 구르는 개"""
        print(f"{self.name} rolled over!")
my_dog = Dog('Willie', 6)
my_dog.sit()
my_dog.roll_over()
```

- super() 메서드로 superclass에 접근
- override.
- CamelCase

```
from car import Car  # car.py 안에 Car 클래스 하나만 있을 경우
from car import ElectricCar  # car.py 안에 여러 클래스 중 ElectricCar 만 임포트
from car import Car, ElectricCar  # car.py 안에 Car, ElectricCar 클래스 임포트
import car  # car.py 안에 있는 모든 클래스 임포트

from module_name import *

from electric_car import ElectricCar as EC  # electric_car.py의 ElectricCar 클래스를 EC로 임포트
```

- python standard library

```
from random import randint
randint(1, 6)

from random import choice
choiced_value = choice(list_values)
```

file and exception

```
# 기본적인 읽기
file_path = '/home/..../openfile.txt'
with open(file_path) as file_object:
    contents = file_object.read()
print(contents)

# 라인씩 읽기
with open(filename) as file_object:
    for line in file_object:
        print(line)
filename = 'pi_digits.txt'
with open(filename) as file_object:
    lines = file_object.readlines()
for line in lines:
    print(line.rstrip())
print(f"{pi_string[:52]...")  # 52번째 캐릭터까지만

# write a file
filename = 'prg.txt'

with open(filename, 'w') as file_object:  # w: write mode, r: read mode, a:
append mode, r+; read and write. default is 'r'
    file_object.write("I love programming.")
#open(filename, encoding='utf-8')
```

```
try:
    print(5/0)
except ZeroDivisionError:  # FileNotFoundError,
    print("You can't divide by zero!")
    #pass  # 조용히 실패하기, except 블럭 안에 아무것도 하지 않고 pass만.
else:
    print("try block success.")
```

- string.replace(find_string, replace_string)
- string.count(find_string) returns count

```
import json

json.dump(contents, file_object)  # contents를 file_object에 저장, with
open(filename, 'w') 사용
loaded_object = json.load(file_object)  # with open(filename)
```

- refactoring; 코드를 더 사용하기 쉽게 재구성

code test

- unit test,
- test cases; unit test의 묶음

```
import unittest

class NamesTestCase(unittest.TestCase): #unittest.TestCase를 상속 받는 클래스 생성

    def setUp(self):  # 각 메서드를 실행하기 전에
....
....
    def test_first_last_name(self):
        ....
        ....
        self.assertEqual(formatted_name, 'Janis Joplin')  # 예상한 결과가 일치하는 지 단언assert.
if __name__ == '__main__':  # 다른 프로그램에서 임포트하지 않고 직접 실행하면 __name__의 값은 '__main__'이 된다.
  unittest.main()
```

method	**example**
assertEqual(a,b)	a == b임을 확인
assertNotEqual(a,b)	a != b임을 확인
assertTrue(x)	x가 True임을 확인
assertFalse(x)	x가 False임을 확인
assertIn(item, list)	item이 list 안에 있음을 확인
assertNotIn(item, list)	item이 list 안에 있지 않음을 확인

game

data visualization

web application

From:
https://theta5912.net/ - **reth**

Permanent link:
https://theta5912.net/doku.php?id=public:computer:python&rev=1627921238

Last update: **2021/08/03 01:20**