python

prepare

on WSL2

- pip 설치

```
$ sudo apt update & sudo apt upgrade -y
$ sudo apt install python3-pip
$ sudo apt install python3-venv
```

- 가상환경 생성

```
$ python3 -m venv .venv
```

- 가상환경 활성화

```
$ source .venv/bin/activate
  또는
$ . .venv/bin/activate
```

- 가상환경 비활성화

```
$ deactivate
```

python settings · 2022/12/29 16:17 · alex

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

- pygame

```
$ python3 -m pip install --user pygame
```

```
import sys
import pygame
import pygame.font
import pygame.sprite import Sprite
import pygame.sprite import Group


....
        pygame.init()
        pygame.display.set_mode((1200,800))  # ((0,0), pygame.FULLSCREEN)
        pygame.display.set_caption("pygame")
        ...
        ...
        ...
        pygame.image.load('image/ship.bmp')
        ...
        ...
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                sys.exit()
            elif event.type == pygame.KEYDOWN:  # pygame.KEYUP,
pygame.MOUSEBUTTONDOWN,
                if event.key == pygame.K_RIGHT:  # pygame.K_LEFT, pygame.K_q
                    ...
        pygame.display.flip()
        # pygame.Rect(0, 0, self.settings.bullet_width,
        #     self.settings.bullet_height)
        # pygame.draw.rect(self.screen, self.color, self.rect)
        # pygame.sprite.Group()
        # pygame.sprite.goupcollide(
```

```
        #     self.bullets, self.aliens, True, True)
        # if pygame.sprite.spritecollideany(slef.ship, self.aliens):
        #     ...
        # mouse_pos = pygame.mouse.get_pos()
        # pygame.mouse.set_visible(False) / True
        # pygame.font.SysFont(None, 48)
```

- helper method; ex) _check_events(), _update_screen()

data visualization

```
$ python3 -m pip install --user matplotlib
```

```
import matplotlib.pyplot as plt

squares = [1, 4, 9, 16, 25]

plt.style.use('seaborn')  # plt.style.available = ['seaborn-dark', 'seaborn-
darkgrid', 'seaborn-ticks', 'fivethirtyeight', .... ]
fig, ax = plt.subplots()  # 그래프 생성
ax.plot(squares)  # ax.plot(squares, linewidth=3), ax.plot(input_values,
squares, linewidth=3)

ax.set_title("Square Numbers", fontsize=24)
ax.set_xlabel("Value", fontsize=14)
ax.set_ylabel("Square of Value", fontsize=14)

ax.tick_params(axis='both', labelsize=14)  # 눈금 라벨 크기를 정함

plt.show()
```

- ax.scatter(2, 4)
- ax.scatter(2, 4, s=200)
- ax.scatter(x_values, y_values, c='red', s=10)
- ax.scatter(x_values, y_values, c=(0, 0.8, 0), s=10)
- ax.scatter(x_values, y_values, c=y_values, cmap=plt.cm.Blues, s=10)
- ax.tick_params(axis='both', which='major', labelsize=14)
- ax.axis([0, 1100, 0, 11000000]
- https://matplotlib.org
- plt.savefig('squares_plot.png', bbox_inches='tight')
- ax.get_xaxis().set_visible(False)
- ax.get_yaxis().set_visible(False)

```
$ python3 -m pip install --user plotly
```

- https://plot.ly/python

```
from plotly.graph_objs import Bar, Layout
from plotly import offline

...
...
data = [Bar(x=x_values, y=frequencies)]
...

x_axis_config = {'title': 'Result'}
y_axis_config = {'title': 'Frequency of Result'}
my_layout = Layout(title='Results of rolling one D6 1000 times',
    xaxis_x_axis_config, yaxis=y_axis_config)
offline.plot({'data':data, 'layout': my_layout}, filename='d6.html')
```

```
import csv

filename = 'data/aaa.csv'

with open(filename) as f:
    reader = csv.reader(f)
    header_row = next(reader)
    print(header_row)
    for index, column_header in enumerate(header_row):
        print(index, column_header)
...
```

```
import datetime import datetime
first_date = datetime.strptime('2021-08-02', '%Y-%m-%d')
print(first_date)
```

- Python strftime cheatsheet
- ax.fill_between(dates, highs, lows, facecolor='blue', alpha=0.1)

```
$ python3 -m pip install --user requests
```

```
import requests

# API 호출을 보내고 응답을 저장
url =
'https://api.github.com/search/repositories?q=language:python&amp;sort=stars
'
headers = {'Accept': 'application/vnd.github.v3+json'}
r = requests.get(url, headers=headers)
```

```
print(f"Status code: {r.status_code}")

# API 응답을 변수에 저장
response_dict = r.json()

# 결과 처리
print(response_dict.keys())
```

- 두 개의 그래프 그리기 → 두 그래프 사이 칠하기
- 에러 체크; except ValueError:
- 세계 지도 만들기 → 지도에 표시 → 마커 크기 조절 → 마커 색깔 → 다른 컬러 스케일 → 텍스트 추가
- 커스텀 툴팁 추가
- 그래프에 클릭할 수 있는 링크 추가

web application

virtual environment

```
$ python3 -m venv ll_env  # ll_env란 이름으로 가상 환경 만들기
$ source ll_env/bin/activate  # ll_env의 가상 환경 활성화
(ll_env)$ deactivate  # ll_env 가상 환경 활성화 상태에서 사용 중지
```

install django

```
(ll_env)$ pip install django  # ll_env 가상 환경에서 장고 설치
(ll_env)$ django-admin startproject learning_log . # learning_log라는 이름으로 프로젝트 생성 마지막에 . 반드시 입력 -> learning_log 디렉토리 안에 settings.py, urls.py, wsgi.py 생성
(ll_env)$ python manage.py migrate # 데이터베이스 생성 -> 기본적으로 db.sqlite3 생성
(ll_env)$ python manage.py startapp learning_logs  # learning_logs라는 이름의 앱 생성 -> model.py, admin.py, views.py
(ll_env)$ python manage.py runserver # 프로젝트 실행
```

모델 정의

```
from django.db import models

class Topic(models.Model):
    """....."""
    ...
    def __str__(self):
        ...
```

- settings.py 파일

```
INSTALLED_APPS = [
  # 내 앱
    'learning_logs',
  # 장고 기본 앱
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
...
```

- model을 수정하면 makemigrations를 먼저 실행하고 → migrate 실행

```
(ll_env)$ python manage.py makemigrations learning_logs
(ll_env)$ python manage.py migrate
```

- superuser 생성

```
(ll_env)$ python manage.py createsuperuser
```

- 관리자 사이트에서 모델 등록
- 주제 추가

- 장고 셸; quit to Ctrl-D or Ctrl-Z(on Windows)

```
(ll_env)$ python manage.py shell
```

- urls.py

```
from django.contrib import admin
from django.urls import path

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('learning_logs.urls')),
]
```

- views.py

```
from django.shortcuts import render
```

```
# view here
```

- decorator; @login_required on views.py
- other references;
    - bootstrap library
    - heroku; PaaS
    - Git;

```
(ll_env)$ pip install django-bootstrap4  # bootstrap4 설치
```

```
(ll_env)$ pip install psycopg2==2.7.*
(ll_env)$ pip install django-heroku
(ll_env)$ pip install gunicorn
```

- requirements.txt 파일; 작성한 프로젝트에 필요한 패키지들 모음

```
(ll_env)$ pip freeze > requirements.txt
```

- runtime.txt

```
python-3.7.2
```

- Procfile

```
web: gunicorn learning_log.wsgi --log-file -
```

- git

```
(ll_env)$ git --version
```

- .gitignore

```
ll_env/
__pycache__/
*.sqlite3
```

- 헤로쿠 배포
    - 헤로쿠 계정
    - 헤로쿠 cli 설치
    - 필수 패키지 설치
    - requirements.txt 생성
    - 파이썬 런타임 명시; runtime.txt
    - 헤로쿠에서 쓸 수 있도록 settings.py 수정
    - Procfile 만들기

- git을 사용해 프로젝트 파일 추적; 깃 설치 → 설정 → .gitignore 생성 → 프로젝트 커밋
- 헤로쿠에 올리기
- 헤로쿠 데이터 베이스 세팅
- 헤로쿠 배포 과정 개선; 헤로쿠에 슈퍼유저 생성 → 사용하기 쉬운 url 만들기
- 프로젝트 보안; settings.py의 DEBUG 플래그 설정
- 커밋과 푸시
- 헤로쿠에서 환경 변수 세팅하기
- 커스텀 에러 페이지 만들기; 커스텀 템플릿 만들기 → 로컬에서 에러 페이지 보기 → 헤로쿠에 변경 내용 올리기 → get_object_or_404() 메서드
- SECRET_KEY 세팅
- 헤로쿠에서 프로젝트 삭제

Python Keywords and internal functions

keywords

- False
- None
- True
- and
- as
- assert
- async
- await
- break
- class
- continue
- def
- del
- elif
- else
- except
- finally
- for
- from
- global
- if
- import
- in
- is
- lambda
- nonlocal
- not
- or
- pass
- raise
- return
- try

- while
- with
- yield

python internal functions

- abs()
- all()
- any()
- ascii()
- bin()
- bool()
- breakpoint()
- bytearray()
- bytes()
- callable()
- chr()
- classmethod()
- compile()
- complex()
- delattr()
- dict()
- divmod()
- enumerate()
- eval()
- exec()
- filter()
- float()
- format()
- frozenset()
- getattr()
- globals()
- hasattr()
- hash()
- help()
- hex()
- id()
- input()
- int()
- isinstance()
- issubclass()
- iter()
- len()
- list()
- locals()
- map()
- max()
- memoryview()
- min()
- next()

- object()
- oct()
- open()
- ord()
- pow()
- print()
- property()
- range()
- repr()
- reversed()
- round()
- set()
- setattr()
- slice()
- sorted()
- staticmethod()
- str()
- sum()
- super()
- tuple()
- type()
- vars()
- zip()
- import()