WEB STUDY

5 WEEK

YOON - SANG SEOK

[CONTENTS]

Python 공식 문서 : https://docs.python.org/ko/3/tutorial/

0. 과제 피드백

1. Python - function, closure, decorator

[function의 기본 사용] https://wikidocs.net/24 https://github.com/amamov/Pythonic/blob/main/02_syntax/03_function.ipynb

```python
def myFunction(arg1, arg2, *args, **kwargs):
    print("arg1 : ", arg1)
    print("arg2 : ", arg2)

    print("args : ", args)
    print("args type : ", type(args))

    print("kwargs : ", kwargs)
    print("kwargs : ", type(kwargs))
    return arg1 + arg2


if __name__ == "__main__":

    myFunction(1, 2, 3, 4, 5, 6, 7, 8, 9, what=False, why=True, Hello="Hello world!")

"""
[출력 결과]
arg1 :  1
arg2 :  2
args :  (3, 4, 5, 6, 7, 8, 9)
args type :  <class 'tuple'>
kwargs :  {'what': False, 'why': True, 'Hello': 'Hello world!'}
kwargs :  <class 'dict'>
"""
```

[데코레이터]

```python
def decorator(func):
    def inner():
        print("This is emoticon")
        func()

    return inner


if __name__ == "__main__":

    def smile():
        print("^_^")

    smile = decorator(smile)
    smile()

    @decorator
    def confused():
        print("@_@")

    confused()
```

[데코레이터 활용]

```python
import time


def performance_clock(func):

    """ 함수의 퍼포먼스를 체크하는 함수 """

    def performance_clocked(*args):
        start_time = time.perf_counter()
        result = func(*args)
        end_time = time.perf_counter()
        print(f"Duration : {round(end_time - start_time, 8)}")
        return result

    return performance_clocked


if __name__ == "__main__":

    @performance_clock
    def time_func(second):
        time.sleep(second)

    @performance_clock
    def sum_func(*numbers):
        return sum(numbers)

    time_func(3)  # Duration : 3.00169363
    sum_func(1, 2, 3, 4, 5)  # Duration : 1.14e-05
    sum_func(1, 2, 3, 4, 5, 6, 7, 8, 9)  # Duration : 3.24e-06
```

2. Python - class

[class의 기본 사용]

https://wikidocs.net/28

https://github.com/amamov/Pythonic/blob/main/02_syntax/04_class.ipynb

```python
class Robot:
    # Class Variable
    population = 0

    def __init__(self, name, age):  # 생성자
        self.name = name  # Instanse Variable
        self.age = age  # Instanse Variable
        Robot.population += 1

    # Instance Method
    def die(self):
        print(f"{self.name} is being destroyed!")
        Robot.population -= 1
        if Robot.population == 0:
            print(f"{self.name} was the last one.")
        else:
            print(f"There are still {Robot.population} robots working.")

    # Instance Method
    def say_hi(self):
        print(f"Greetings, my masters call me {self.name}.")
        print(f"I am {self.age} years old.")

    # Class Method
    @classmethod
    def how_many(cls):
        print(f"We have {cls.population} robots.")

    # Class Method
    @classmethod
    def is_name_amamov(cls, instance):
        if instance.name == "amamov":
            print("This robot name is amamov")
        else:
            print("This robot name is not amamov")
```

[상속]

https://wikidocs.net/28

https://github.com/amamov/Pythonic/blob/main/02_syntax/04_class.ipynb

```python
class Mother:

    tribe = "human"

    def run(self):
        print("달리는 능력")

    def attack(self):
        print("공격하는 능력")


You, a few seconds ago | 1 author (You)
class Son(Mother):
    def jump(self):
        print("점프하는 능력")

    ## 오버라이딩
    def run(self):
        print("아들만의 달리는 능력")

    ## super()을 이용한 상속
    def mother_run(self):
        super().run()

        print(super())
        # <super: <class 'Son'>, <Son object>>

        print(type(super()))
        # <class 'super'>
```

3. Python - import

[import]

https://github.com/amamov/Pythonic/tree/main/03_import

4. Database（RDBMS）

https://sqlitebrowser.org/dl/

계획

1. 서버 개발을 위한 Python 복습 및 심화, DB의 이해 (2/1)

2. 백엔드에 대한 이해, Flask 기본적인 사용법, Flask Template 엔진으로 웹 개발 (2/8)

3. Flask SQLalchemy를 사용하여 DB 핸들링, 투두리스트 구현 (2/15)

4. 쿠키와 세션에 대한 이해 (로그인 기능), Flask로 회원가입과 로그인 구현 (2/22)

5. Flask로 간단한 API 구현, 백엔드 개발 마무리 (3/1…?)

6~8. 개인 프로젝트 시작 ~ 프로젝트 발표회, Heroku 서버에 배포하기 (3주 정도)

과제

코끼리 여러 마리가 호수를 가로지르는 일 차선 다리를 정해진 순으로 건너려 합니다. 모든 코끼리가 다리를 건너려면 최소 몇 초가 걸리는지 알아내야 합니다. 코끼리는 1초에 1만큼 움직이며, 다리 길이는 bridge_length이고 다리는 무게 weight까지 견딥니다.

※ 코끼리가 다리에 완전히 오르지 않은 경우, 이 코끼리의 무게는 고려하지 않습니다.

예를 들어, 길이가 2이고 10kg 무게를 견디는 다리가 있습니다. 무게가 [7, 4, 5, 6]kg인 코끼리가 순서대로 최단 시간 안에 다리를 건너려면 아래와 같이 건너야 합니다.

경과시간	다리를 지난 코끼리	다리를 건너는 코끼리	대기중
0	[]	[]	[7,4,5,6]
1~2	[]	[7]	[4,5,6]
3	[7]	[4]	[5,6]
4	[7]	[4,5]	[6]
5	[7,4]	[5]	[6]
6~7	[7,4,5]	[6]	[]
8	[7,4,5,6]	[]	[]

따라서, 모든 코끼리가 다리를 지나려면 최소 8초가 걸립니다.
solution 함수의 매개변수로 다리 길이 bridge_length,
다리가 견딜 수 있는 무게 weight, 코끼리별 무게 ele_weights가 주어집니다.
이때 모든 코끼리가 다리를 건너려면 최소 몇 초가 걸리는지
return 하도록 solution 함수를 완성한 후 py파일을 압축해서 보내주세요.
(데코레이터를 사용해서 solution 함수의 속도도 측정해주세요.) [HINT] 스택/큐

제한 조건

- bridge_length는 1 이상 10,000 이하입니다.
- weight는 1 이상 10,000 이하입니다.
- ele_weights의 길이는 1 이상 10,000 이하입니다.
- 코끼리의 무게는 1 이상 weight 이하입니다.

입출력 예

bridge_length	weight	ele_weights	return
2	10	[7,4,5,6]	8
100	100	[10]	101
100	100	[10,10,10,10,10,10,10,10,10,10]	110

```python
@performance_clock
def solution(bridge_length, weight, ele_weights):
    time = 0
    # Code
    return time
```