정규세션 1주차

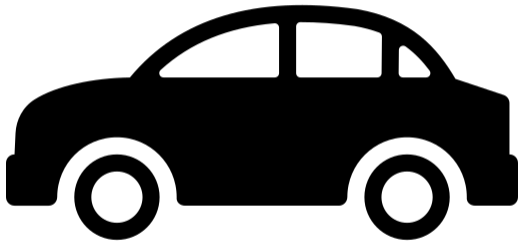
ToBig's 11기 이소라

CLASS IN PYTHON

contents

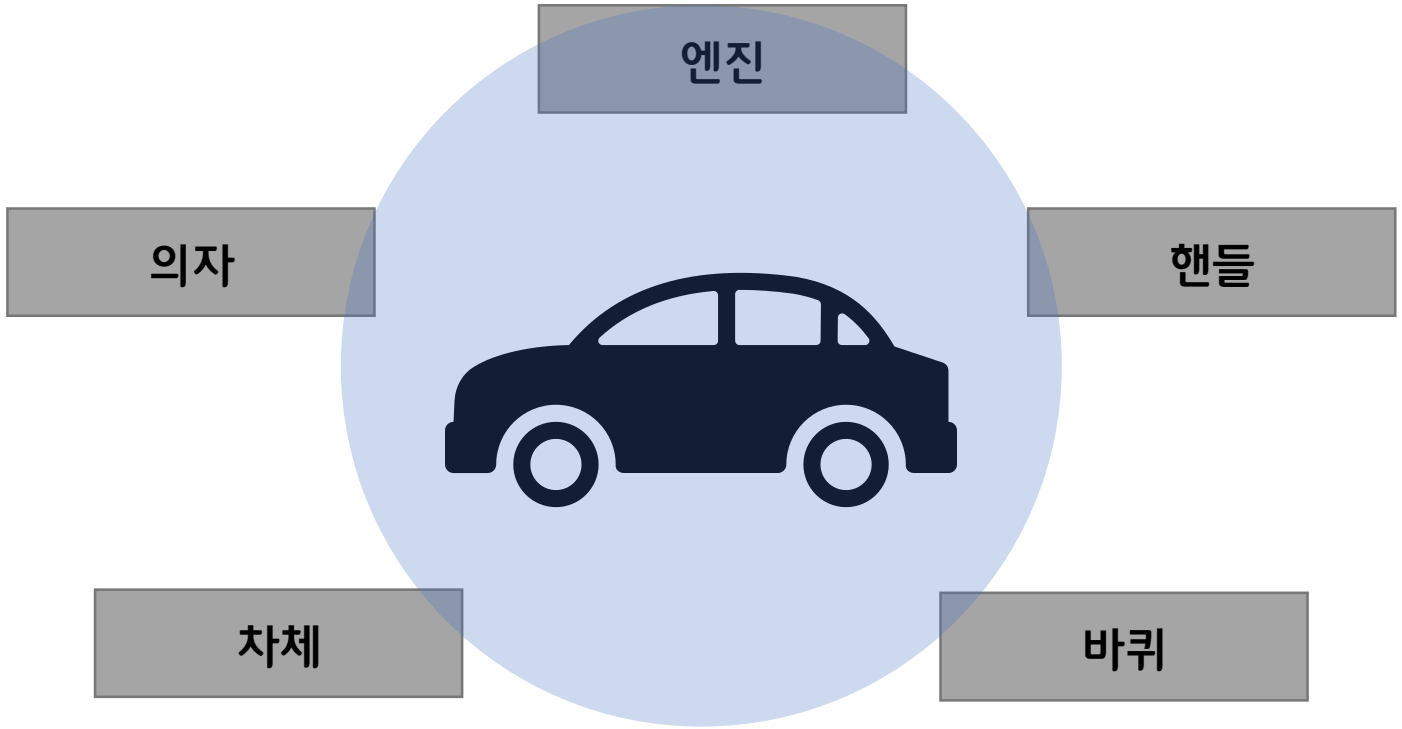
Unit 01 | POP (Procedual Oriented Programming

- POL (procedure-oriented language) - 절차에 의해 진행되는 언어/프로그래밍을 실행 순서에 의하여 코딩



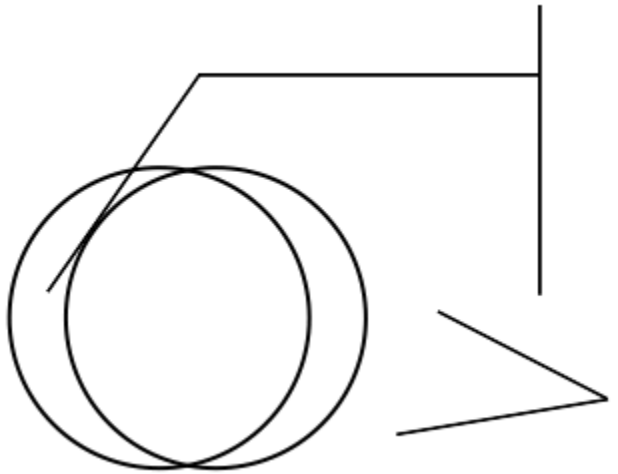
Unit 01 | OOP (Object Oriented Programming)

- OPL (Object Oriented Language)
- 프로그램 / 명령어의 목록 < 객체(독립된 단위) 모임



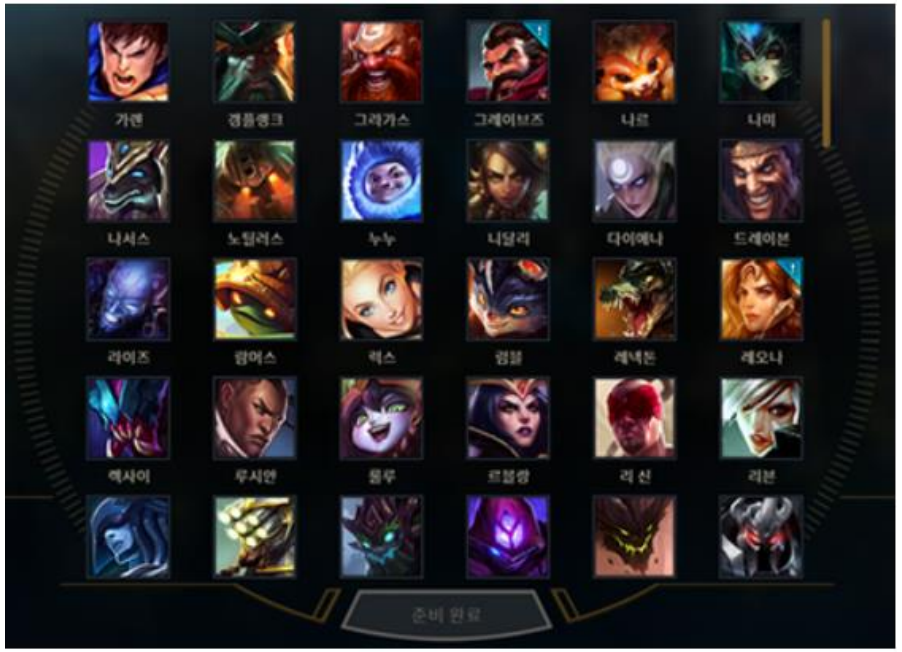
Unit 01 | OOP (Object Oriented Programming)

- OPL (Object Oriented Language)
- 프로그램 / 명령어의 목록 < 객체(독립된 단위) 모임



Unit 01 | OOP (Object Oriented Programming)

- 객체 – 주변에 실존하는 모든 사물, 생명체
- 클래스- 객체를 생성하기 위한 틀

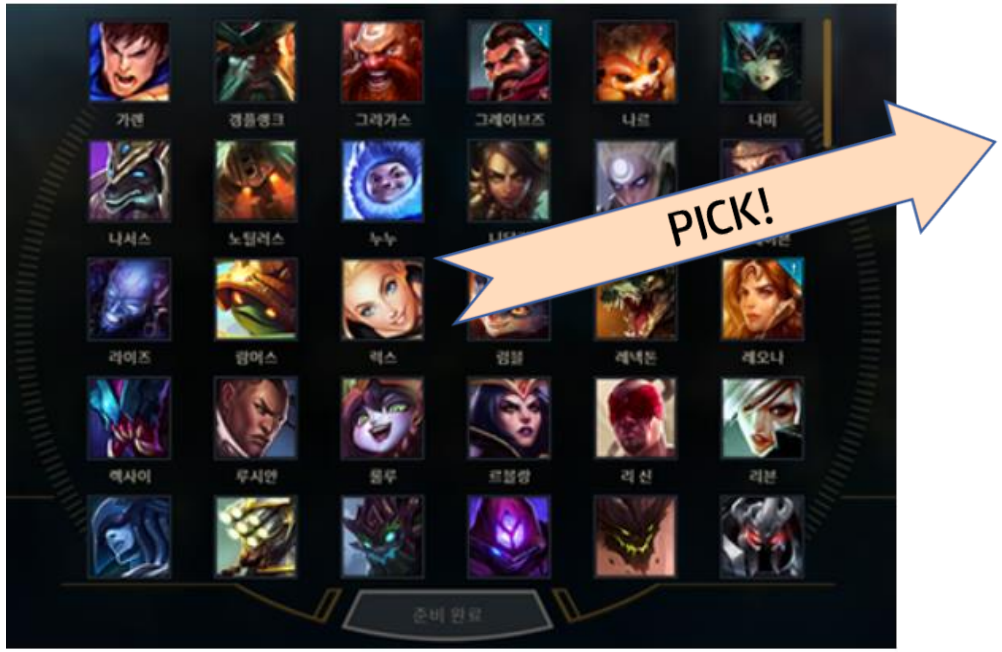


클래스

객체

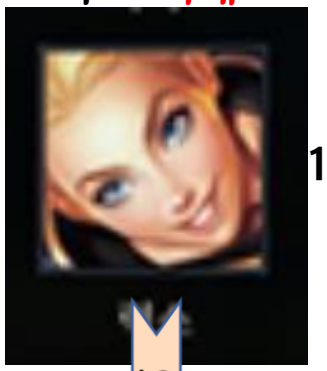
Unit 01 | OOP (Object Oriented Programming)

챔피언 **클래스**



각각의 챔피언은 체력/마나/특성/Q,W,E,R 스킬

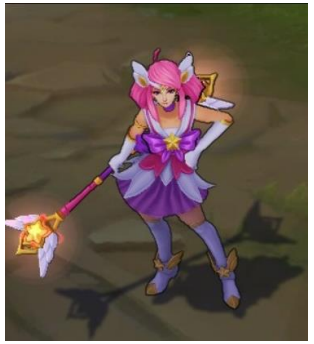
럭스 **객체**



100/50/스킬 사용 후 기본 공격 강화/Q,W,E,R

START

럭스 **인스턴스**



게임 내의 실체화된 정보를 가진 인스턴스

Unit 01 | OOP (Object Oriented Programming)

- 속성 - 객체의 기본 값
- 메소드 - 객체의 행동과 동작을 유발하는 동적 개념

럭스 **객체**

체력 마나 특성	속성	
Q W E R	메소드	

Unit 02 | CLASS

클래스

```
class 챔피언:
    이름=None
    특성=None
    체력=0
    마나=0

    def __init__(self,이름,체력,마나,특성):
        self.이름=이름
        self.체력=체력
        self.마나=마나
        self.특성=특성
    def skill_of_q():
        return q스킬

    def skill_of_w():
        return w스킬

    def sill_of_e():
        return e스킬
```

럭스 객체 선언

```
럭스=Champion(럭스,100,50,스킬 사용 후 기본 공격 강화)
print(럭스.skill_of_q())
```

?

럭스(객체) 고유의 q스킬, w스킬, e스킬 정의가 필요함
상속과 다형성

Unit 02 | CLASS

- 상속
- 다형성

```
class 챔피언:
    이름=None
    특성=None
    체력=0
    마나=0

    def __init__(self,이름,체력,마나,특성):
        self.이름=이름
        self.체력=체력
        self.마나=마나
        self.특성=특성
    def skill_of_q():
        return q스킬

    def skill_of_w():
        return w스킬

    def sill_of_e():
        return e스킬
```

상속

```
class 럭스(챔피언):
    def __init__(self,이름,체력,마나,특성):
        super().__init__(self,이름,체력,마나,특성)

    def skill_of_q():
        return 속박
    def skill_of_w():
        return 보호막

    def skill_of_e():
        return 섬광
```

다형성/오버라이딩

Unit 02 | CLASS

```
class 챔피언:
    이름=None
    특성=None
    체력=0
    마나=0

    def __init__(self, 이름, 체력, 마나, 특성):
        self.이름=이름
        self.체력=체력
        self.마나=마나
        self.특성=특성
    def skill_of_q():
        return q스킬

    def skill_of_w():
        return w스킬

    def sill_of_e():
        return e스킬
```

```
class 럭스(챔피언):
    def __init__(self, 이름, 체력, 마나, 특성):
        super().__init__(self,이름,체력,마나,특성)


    def skill_of_q():
        return 속박
    def skill_of_w():
        return 보호막

    def skill_of_e():
        return 섬광
```

Q.
챔피언1=챔피언(럭스,100,100,기본공격강화)
럭스1=럭스(럭스,100,100,기본공격강화)

Print(챔피언1.skill_of_q())
Print(럭스1.skill_of_q())

Unit 02 | CLASS

```python
class 챔피언:
    이름=None
    특성=None
    체력=0
    마나=0

    def __init__(self, 이름, 체력, 마나, 특성):
        self.이름=이름
        self.체력=체력
        self.마나=마나
        self.특성=특성
    def skill_of_q():
        return q스킬

    def skill_of_w():
        return w스킬

    def sill_of_e():
        return e스킬
```

```python
class 럭스(챔피언):
    def __init__(self, 이름, 체력, 마나, 특성):
        super().__init__(self,이름,체력,마나,특성)


    def skill_of_q():
        return 속박
    def skill_of_w():
        return 보호막

    def skill_of_e():
        return 섬광
```

Q.
챔피언1=챔피언(럭스,100,100,기본공격강화)
럭스1=럭스(럭스,100,100,기본공격강화)

Print(챔피언1.skill_of_q()) ➔q스킬
Print(럭스1.skill_of_q()) ➔ 속박

Unit 03 | Magic method

```python
class test:
    data = 10  # 클래스 속성

    def __init__(self, data):
        self.data = data  # 인스턴스 속성


    def printInstance(self):  # 인스턴스 메소드
        print(self.data)


t = test(30)
print(dir(t))
```

['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__',
'__format__', '__ge__', '__getattribute__', '__gt__', '__hash__', '__init__',
'__init_subclass__', '__le__', '__lt__', '__module__', '__ne__', '__new__',
'__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '
__str__', '__subclasshook__',
'__weakref__', 'data', 'printInstance']

Unit 03 | Magic method

- __init__

```
class test:
    data = 10

    def __init__(self, data):
        self.data = data
        print(self.data)



t = test(30)
print(t)
```

- __str__

```
class test:
    data = 10

    def __init__(self, data):
        self.data = data
        print(self.data)

    def __str__(self):
        return 'called by str'



t = test(30)
print(t)
```

```
30
called by str
```

- __lt__

```
class Food():
    def __init__(self, name, price):
        self.name = name
        self.price = price

    def __lt__(self, other):
        if self.price < other.price:
            return True
        else:
            return False


food_1 = Food('아이스크림', 3000)
food_2 = Food('햄버거', 5000)
food_3 = Food('콜라', 2000)

# food_2가 food_1보다 큰지 확인
print(food_1 < food_2)  # 3000 < 5000
print(food_2 < food_3)  # 5000 < 2000
```

```
True
False
```

- 클래스
- 객체
- 인스턴스
- 속성
- 메소드
- 상속
- 다형성
- 매직 메소드

Unit 04 | Assignment

- Data

Apartment.txt

10 개의 아파트
207동 1105호/ 평수 /방의 개수

```
10
207-1105 53 4
205-503 34 3
216-701 84 5
202-1804 63 4
201-404 26 2
209-1201 53 5
211-301 45 4
207-1205 53 4
202-606 63 4
205-801 34 3
```

Vile.txt

3개의 주택
주소 / 평수 / 방의 개수 / 마당 유무

```
3
유현로51 53 4 T
장기로34 88 5 F
풍무로1 140 4 F
```

가장 큰 평수를 가진 아파트/ 주택에 대한 정보를 각각 출력해주세요

Unit 04 | Assignment

- Data

Apartment.txt

10 개의 아파트
207동 1105호/ 평수 /방의 개수

```
10
207-1105 53 4
205-503 34 3
216-701 84 5
202-1804 63 4
201-404 26 2
209-1201 53 5
211-301 45 4
207-1205 53 4
202-606 63 4
205-801 34 3
```

Vile.txt

3개의 주택
주소 / 평수 / 방의 개수 / 마당 유무

```
3
유현로51 53 4 T
장기로34 88 5 F
풍무로1 140 4 F
```

Unit 04 | Assignment

Q .Assignment.py – 절차지향적으로 작성된 코드/
이 코드의 흐름을 보고 객체지향적으로 코딩해주세요 ! (클래스를 만들어주세요)

How to
1. House.py에 있는 House함수를 상속받는 Apartment, Vile 클래스 만들기
2. Main2.py에서 Apartment와 Vile 파일을 import해서 사용하기

Output

```
216동 701호의 5개의 방이 있는 84평의 아파트입니다.
216동 701호의 5개의 방이 있는 84평의 아파트입니다.
풍무로1에 위치해 있는 4개의 방이 있고 마당은 없는 140평의 주택입니다.
```

```
for j in range(N-1):
    """
    class magic method인 lt함수 활용
    최대 평수를 가진 Aparment 클래스 찾기
    """
```

참고 자료

- https://docs.python.org/ko/3/tutorial/classes.html
- https://programmers.co.kr/learn/courses/2/lessons/330#

들어주셔서 감사합니다.