



염희균

목차

- 시작하기 전에
- 모듈 만들기
- 패키지
- `__name__=="__main__"`
- 키워드로 정리하는 핵심 포인트
- 확인문제

시작하기 전에

[핵심 키워드] : 엔트리 포인트, `__name__=="__main__"`, 패키지

[핵심 포인트]

모듈을 만드는 방법을 알면 직접 모듈을 만드는 것은 물론이고 다른 사람이 만든 모듈을 분석할 수도 있다

시작하기 전에

- module_basic 디렉터리 만든 후 아래 두 파일 넣기



`main.py`



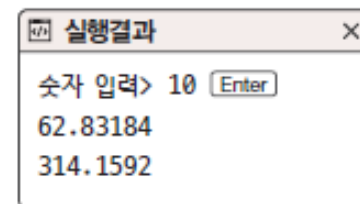
`test_module.py`

모듈 만들기

- module_basic 디렉터리 만든 후 아래 두 파일 저장하고 main.py 파일 실행

```
01  # test_module.py 파일
02  PI = 3.141592
03
04  def number_input():
05      output = input("숫자 입력> ")
06      return float(output)
07
08  def get_circumference(radius):
09      return 2 * PI * radius
10
11  def get_circle_area(radius):
12      return PI * radius * radius
```

```
01  # main.py 파일
02  import test_module as test
03
04  radius = test.number_input()
05  print(test.get_circumference(radius))
06  print(test.get_circle_area(radius))
```



패키지(package)

- 모듈 (module)을 여러 개 모아둔 집합을 패키지라고 부름
- 패키지 관리 시스템 (Package Management System)
 - pip
 - 모듈이 모여서 구조 이루면 패키지
- 패키지 만들기
 - mian.py 파일은 엔트리 포인트로, test_package 폴더는 패키지로 사용



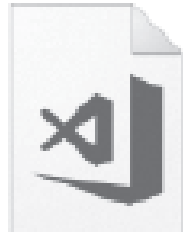
test_package



main.py

패키지

- test_package 폴더 내부에 module_a.py 파일과 module_b.py 파일 생성



module_a.py



module_b.py

- 두 파일에 아래와 같이 입력

```
01  # ./test_package/module_a.py의 내용
02  variable_a = "a 모듈의 변수"
```

```
01  # ./test_package/module_b.py의 내용
02  variable_b = "b 모듈의 변수"
```

패키지

```
01  # 패키지 내부의 모듈을 읽어 들입니다.  
02  import test_package.module_a as a  
03  import test_package.module_b as b  
04  
05  # 모듈 내부의 변수를 출력합니다.  
06  print(a.variable_a)  
07  print(b.variable_b)
```

실행결과

a 모듈의 변수
b 모듈의 변수

패키지

- `__init__.py` 파일
 - 패키지 읽을 때 어떤 처리를 수행해야 하거나 패키지 내부의 모듈들을 한꺼번에 가져오고 싶을 때 사용
 - test_package 폴더 내부에 `__init__.py` 파일 추가



- 패키지 읽어들이는 때 `__init__.py`를 가장 먼저 실행
- 패키지와 관련된 초기화 처리 등 할 수 있음

패키지

__init__.py

```
01  # "from test_package import *"로
02  # 모듈을 읽어 들일 때 가져올 모듈
03  __all__ = ["module_a", "module_b"] → *사용 시 읽어들일 모듈의 목록
04
05  # 패키지를 읽어 들일 때 처리를 작성할 수도 있습니다.
06  print("test_package를 읽어 들였습니다.")
```

main.py

```
01  # 패키지 내부의 모듈을 모두 읽어 들입니다.
02  from test_package import *
03
04  # 모듈 내부의 변수를 출력합니다.
05  print(module_a.variable_a)
06  print(module_b.variable_b)
```

패키지

실행결과

X

test_package를 읽어 들였습니다.

a 모듈의 변수

b 모듈의 변수

모듈 내 직접 실행 시 : `__name__=="__main__"`

- `__name__`
 - 엔트리 포인트 (entry point) / 메인 (main)
 - 프로그램의 진입점
 - 메인 내부에서의 `__name__`은 `"__main__"`

```
>>> __name__  
'__main__'
```

- 모듈의 `__name__`
 - 엔트리 포인트 아니지만 엔트리 포인트 파일 내에서 import 되었기 때문에
모듈 내 코드가 실행
 - 모듈 내부에서 `__name__` 출력하면 모듈의 이름 나타냄

__name__=="__main__"

- 예시 - 모듈 이름을 출력하는 모듈 만들기

```
01  # main.py 파일
02  import test_module
03
04  print("# 메인의 __name__ 출력하기")
05  print(__name__)
06  print()
```

```
01  # test_module.py 파일
02  print("# 모듈의 __name__ 출력하기")
03  print(__name__)
04  print()
```

실행결과

```
# 모듈의 __name__ 출력하기
test_module

# 메인의 __name__ 출력하기
__main__
```

__name__=="__main__"

- __name__ 활용하기
 - 엔트리 포인트 파일 내부에서 __name__이 "__main__" 값을 가짐을 활용하여 현재 파일이 모듈로 실행되는지 외부에서 (엔트리 포인트)실행되는지 확인
 - 예시 - test_module.py (모듈 활용하기)

```
01  PI = 3.141592
02
03  def number_input():
04      output = input("숫자 입력> ")
05      return float(output)
06
07  def get_circumference(radius):
08      return 2 * PI * radius
09
10  def get_circle_area(radius):
```

__name__=="__main__"

```
11     return PI * radius * radius
```

```
12
```

```
13 # 활용 예
```

```
14 print("get_circumference(10):", get_circumference(10))
```

```
15 print("get_circle_area(10): ", get_circle_area(10))
```

"이런식으로 동작해요!"를 알려주는
활용 예를 넣었습니다.



```
01 import test_module as test → 위 모듈을 읽어 들입니다.
```

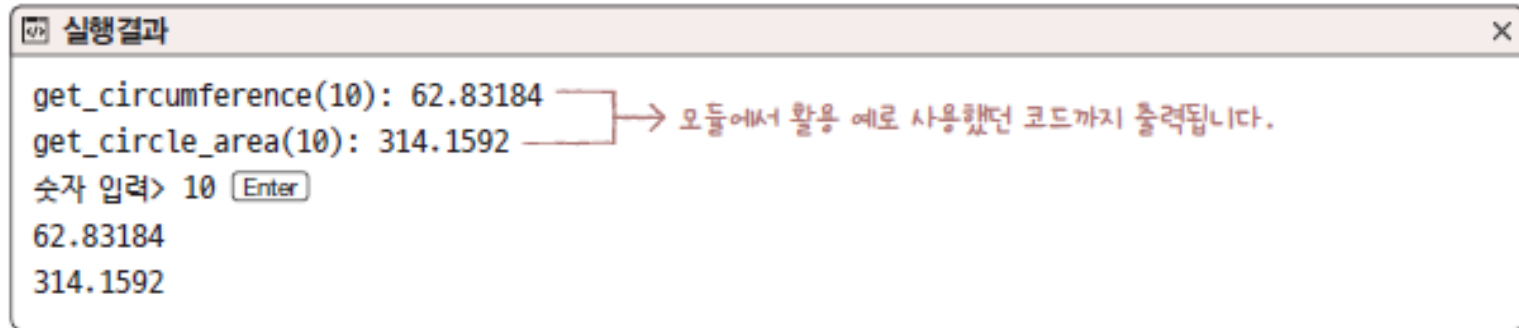
```
02
```

```
03 radius = test.number_input()
```

```
04 print(test.get_circumference(radius))
```

```
05 print(test.get_circle_area(radius))
```

`__name__=="__main__"`



```
실행결과
get_circumference(10): 62.83184
get_circle_area(10): 314.1592
숫자 입력> 10 [Enter]
62.83184
314.1592
```

→ 모듈에서 활용 예로 사용했던 코드까지 출력됩니다.

- 현재 test_module.py 파일에는 동작 설명을 위해 추가한 활용 예시 부분 존재
- 모듈로 사용하고 있는데 내부에서 출력 발생하여 문제
- 현재 파일이 엔트리 포인트인지 구분하는 코드를 활용
- `if __name__=="__main__"`인지의 조건문으로 확인
 - 즉 모듈파일을 직접 실행해서 모듈이 정상적인지 테스트해 볼 때 사용함

__name__=="__main__"

```
01  PI = 3.141592
02
03  def number_input():
04      output = input("숫자 입력> ")
05      return float(output)
06
07  def get_circumference(radius):
08      return 2 * PI * radius
09
10  def get_circle_area(radius):
11      return PI * radius * radius
```

```
12  # 활용 예
```

```
13  if __name__ == "__main__":
14      print("get_circumference(10):", get_circumference(10))
15      print("get_circle_area(10): ", get_circle_area(10))
```

현재 파일이 엔트리 포인트인지 확인하고,
엔트리 포인트일 때만 실행합니다.



__name__=="__main__"

```
01  import test_module as test
02
03  radius = test.number_input()
04  print(test.get_circumference(radius))
05  print(test.get_circle_area(radius))
```

실행결과

숫자 입력> 10

62.83184

314.1592

키워드로 정리하는 핵심 포인트

- **엔트리 포인트** : python 명령어 사용한 첫 진입 파일을 엔트리 포인트라 부른다.
- **`__name__=="__main__"`** : 현재 파일이 엔트리 포인트인지 확인할 때 사용하는 코드
- **패키지** : 모듈이 모인 것

확인문제

- PythonExam 폴더에 영화관 가격 계산 모듈 만들기
 - theater_module.py (3개의 가격 계산 기능을 갖고 있다.)

```
1      # 일반 가격
2      def price(people):
3          print("{0}명 가격은 {1}원 입니다.".format(people, people*10000))
4
5
6      # 조조 할인 가격
7      def price_morning(people):
8          print("{0}명 조조 할인 가격은 {1}원 입니다.".format(people, people*6000))
9
10
11     # 군인 할인 가격
12     def price_soldier(people):
13         print("{0}명 군인 할인 가격은 {1}원 입니다.".format(people, people*4000))
14
```

확인문제

- PythonExam 폴더에 영화관 가격 계산 모듈을 사용하는 파일 만들기
 - 11_practice.py
 - 5가지 방법으로 모듈을 사용함
 - 첫번째 방법

```
1 import theater_module
2 theater_module.price(3) # 3명이 영화보러 갔을때 가격
3 theater_module.price_morning(4) # 4명이 조조할인 영화보러 갔을때 가격
4 theater_module.price_soldier(5) # 5명이 군인이 영화보러 갔을때 가격
```

확인문제

- PythonExam 폴더에 영화관 가격 계산 모듈을 사용하는 파일 만들기
 - 11_practice.py
 - 5가지 방법으로 모듈을 사용함
 - 두번째 방법

```
6 import theater_module as mv #모듈명이 길때
7 mv.price(3)
8 mv.price_morning(4)
9 mv.price_soldier(5)
```

확인문제

- PythonExam 폴더에 영화관 가격 계산 모듈을 사용하는 파일 만들기
 - 11_practice.py
 - 5가지 방법으로 모듈을 사용함
 - 세번째 방법

```
11  from theater_module import *
12  # from random import *
13  price(3)
14  price_morning(4)
15  price_soldier(5)|
```

확인문제

- PythonExam 폴더에 영화관 가격 계산 모듈을 사용하는 파일 만들기
 - 11_practice.py
 - 5가지 방법으로 모듈을 사용함
 - 네번째 방법

```
17     from theater_module import price, price_morning
18     price(5)
19     price_morning(6)
20     price_soldier(7)
```


확인문제

- PythonExam 폴더에 영화관 가격 계산 모듈을 사용하는 파일 만들기
 - 11_practice.py
 - 5가지 방법으로 모듈을 사용함
 - 다섯번 째 방법

```
22     from theater_module import price_soldier as price
23     price(5)
```

패키지 생성 및 사용 문제

- travel 패키지 만들기
 - thailand.py 모듈 생성
 - vietnam.py 모듈 생성
 - __init__.py 파일 생성(현재 자동 생성)

패키지 생성 및 사용 문제

- thailand.py 파일 내용

```
class ThailandPackage:
    def detail(self):
        print("[태국 패키지 3박 5일] 방콕, 파타야 여행 (야시장 투어) 50만원")
```

- vietnam.py 파일 내용

```
1 class VietnamPackage:
2     def detail(self):
3         print("[베트남 패키지 3박 5일] 다낭 효도 여행 60만원")
```

- practice.py 내용

```
26 import travel.thailand
27 trip_to = travel.thailand.ThailandPackage() # ThailandPackage 객체생성
28 trip_to.detail()
```

import 뒤에는 패키지명 또는 모듈
명만 올 수 있다.
(클래스명이나 함수가 올 수 없다.)

패키지 생성 및 사용 문제

- practice.py 내용

클래스명이 오게 할 경우는 from
문으로 시작

```
30 from travel.thailand import ThailandPackage
31 trip_to = ThailandPackage()
32 trip_to.detail()
```

베트남 모듈만 import 할 경우는
from문으로 시작

```
34 from travel import vietnam
35 trip_to = vietnam.VietnamPackage()
36 trip_to.detail()
```

패키지 생성 및 사용 문제

- practice.py (*from 패키지명 import **) 형식으로 패키지 사용시
vietnam, thailand 모듈 접근이 안됨=> `__init__.py`를 수정함

```
40 from travel import *
41 trip_to = vietnam.VietnamPackage()
42 trip_to2 = thailand.ThailandPackage()
43 trip_to.detail()
44 trip_to2.detail()
```

- `__init__.py`

```
1 __all__=["vietnam", "thailand"]
```

패키지 생성 및 사용 문제

- practice.py

```
40     from travel import *
41     trip_to = vietnam.VietnamPackage()
42     trip_to2 = thailand.ThailandPackage()
43     trip_to.detail()
44     trip_to2.detail()
```

[베트남 패키지 3박 5일] 다낭 효도 여행 60만원

[태국 패키지 3박 5일] 방콕, 파타야 여행 (야시장 투어) 50만원

Process finished with exit code 0