

모듈 만들기

목차

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

[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

- module_basic 디렉터리 만든 후 아래 두 파일 넣기



main.py

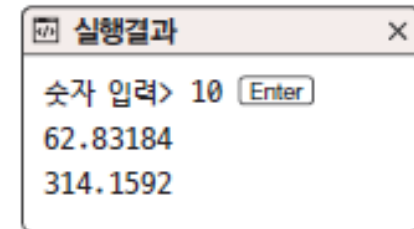


test_module.py

- 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)
 - 복잡하고 구조화된 모듈 만들 때 사용하는 기능

__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()
```

__name__ == “__main__”



```
# 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 엔트리 포인트로 실행되는지 확인
- 예시 – test_module.py (모듈 활용하기)

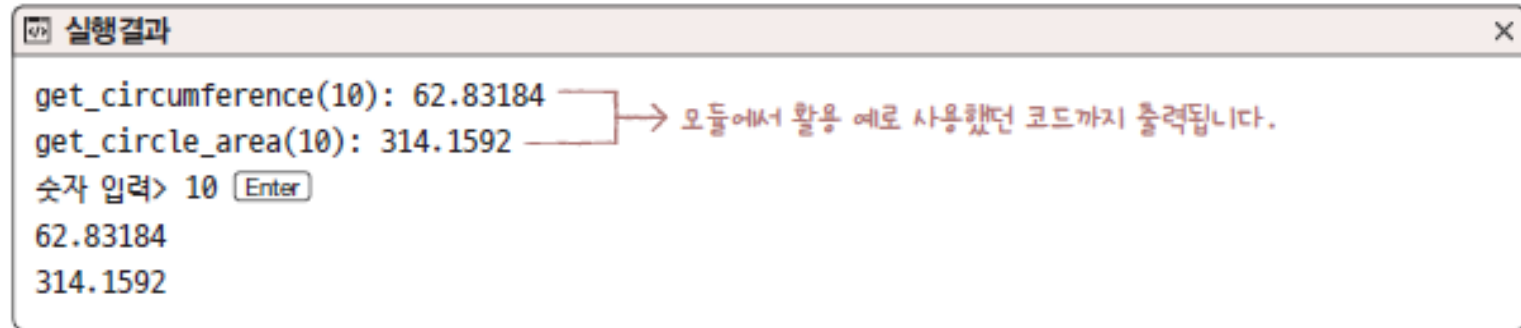
```
01  PI = 3.141592
02
03  def number_input():
04      output = input("숫자 입력> ")
05      return float(output)
06
07  def get_circumference(radius):
08      return 2 * PI * radius
09
10  def get_circle_area(radius):
```

__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))
```



```
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 코드 활용
- 조건문으로 __name__이 "__main__"인지 확인

__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 확인하:
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

- 모듈 (module)
- 패키지 관리 시스템 (Package Management System)
 - pip
 - 모듈이 모여서 구조 이루면 패키지
- 패키지 만들기
 - main.py 파일은 엔트리 포인트로, test_package 폴더는 패키지로 사용



- 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"
```

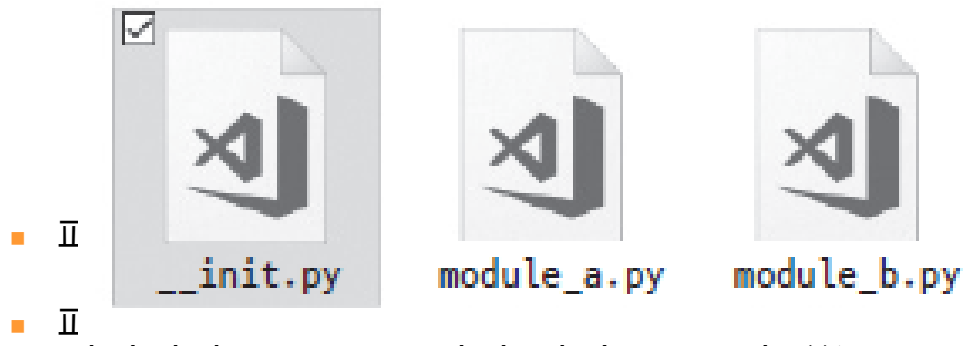
```
01  # 패키지 내부의 모듈을 읽어 들입니다.  
02  import test_package.module_a as a  
03  import test_package.module_b as b  
04  
05  # 모듈 내부의 변수를 출력합니다.  
06  print(a.variable_a)  
07  print(b.variable_b)
```

실행결과

a 모듈의 변수
b 모듈의 변수

- `__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



```
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.")
```

```
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)
```

실행결과

test_package를 읽어 들였습니다.

a 모듈의 변수

b 모듈의 변수

- 텍스트 데이터 (text data)
 - 우리가 쉽게 읽을 수 있는 형태의 데이터
 - 컴퓨터는 내부적으로 모든 처리를 0과 1로 이루어진 이진숫자로 수행
 - 텍스트 데이터로 쉽게 편집

“Hello Python Programming”의 이진 데이터

```
01001000 01100101 01101100 01101100 01101111 00100000 01010000 01111001 01110100  
01101000 01101111 01101110 00100000 01010000 01110010 01101111 01100111 01110010  
01100001 01101101 01101101 01101001 01101110 01100111
```

“Hello Python Programming”의 이진 데이터를 10진수로 표기한 형태

```
72 101 108 108 111 32 80 121 116 104 111 110 32 80 114 111 103 114 97 109 109 105  
110 103
```

Ctrl	Dec	Hex	Char	Code	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
^@	0	00		NUL	32	20	!	64	40	@	96	60	'
^A	1	01		SOH	33	21	!	65	41	A	97	61	a
^B	2	02		STX	34	22	"	66	42	B	98	62	b
^C	3	03		ETX	35	23	#	67	43	C	99	63	c
^D	4	04		EOT	36	24	\$	68	44	D	100	64	d
^E	5	05		ENQ	37	25	%	69	45	E	101	65	e
^F	6	06		ACK	38	26	&	70	46	F	102	66	f
^G	7	07		BEL	39	27	'	71	47	G	103	67	g
^H	8	08		BS	40	28	(72	48	H	104	68	h
^I	9	09		HT	41	29)	73	49	I	105	69	i
^J	10	0A		LF	42	2A	*	74	4A	J	106	6A	j
^K	11	0B		VT	43	2B	+	75	4B	K	107	6B	k
^L	12	0C		FF	44	2C	,	76	4C	L	108	6C	l
^M	13	0D		CR	45	2D	-	77	4D	M	109	6D	m
^N	14	0E		SO	46	2E	.	78	4E	N	110	6E	n
^O	15	0F		SI	47	2F	/	79	4F	O	111	6F	o
^P	16	10		DLE	48	30	0	80	50	P	112	70	p
^Q	17	11		DC1	49	31	1	81	51	Q	113	71	q
^R	18	12		DC2	50	32	2	82	52	R	114	72	r
^S	19	13		DC3	51	33	3	83	53	S	115	73	s
^T	20	14		DC4	52	34	4	84	54	T	116	74	t
^U	21	15		NAK	53	35	5	85	55	U	117	75	u
^V	22	16		SYN	54	36	6	86	56	V	118	76	v
^W	23	17		ETB	55	37	7	87	57	W	119	77	w
^X	24	18		CAN	56	38	8	88	58	X	120	78	x
^Y	25	19		EM	57	39	9	89	59	Y	121	79	y
^Z	26	1A		SUB	58	3A	:	90	5A	Z	122	7A	z
^[27	1B		ESC	59	3B	:	91	5B	[123	7B	[
^\ ^]	28 29	1C 1D		FS GS	60 61	3C 3D	< = > ?	92 93 94 95	5C 5D 5E 5F	\] ^ _	124 125 126 127	7C 7D 7E 7F	\] ^ _
^^ ^-	30 31	1E 1F	▲ ▼	RS US	62 63	3E 3F							

- 바이너리 데이터 (binary data)
 - 텍스트 에디터로 열었을 때 의미를 이해할 수 없는 데이터
 - 이미지, 동영상 등

3바이트를 차지하는 텍스트 데이터 100[49 48 48]

00110001 00110000 00110000

1바이트를 차지하는 바이너리 데이터 100[100]

01100100



덤프NG

IHDR , ?팩 tEXtSoftware Adobe ImageReadyq?< /IDATx班] tU?pDFKEA?쵸8?W 톨?Y`?(r)b
滔Pj+4*

켠? Q?+쥬\$%l? !겼□儉율複笏桎M^참鎂삼뵈;3w뽰?延;v?i衲

pXUr?땡??[??? 0貼?7rH????p8rB백fg?*뵈~,|마^잉0뽰뽰?M쑤 0닿塏&sH ?[8保諍Fw?M1?Y헬

잠(u)?남눔S백X꺽烹뽰?ë"c참i?#□? ちJ'4A肥@

?5. \$Zp秕꺽뽰 t뵈r매뽰툏u?_r隘??7뽰嶠w???

...생략...

비교 항목	텍스트 데이터	바이너리 데이터
구분 방법	• 텍스트 에디터로 열었을 때 읽을 수 있습니다.	• 텍스트 에디터로 열어도 읽을 수 없습니다.
장점	• 사람이 쉽게 읽을 수 있습니다. • 텍스트 에디터로 쉽게 편집할 수 있습니다.	• 용량이 적습니다.
단점	• 용량이 큼니다.	• 사람이 쉽게 읽을 수 없습니다. • 일반적으로는 텍스트 에디터로 편집할 수 없습니다.

- 텍스트 데이터를 우리가 읽기 쉬운 글자로 보이기 위해, 혹은 바이너리 데이터를 읽어 이미지로 보이기 위해 데이터를 변환하는 것
 - 디코딩 (decoding)
 - 반대 과정

- 인터넷의 이미지 저장하기
 - 파일 열 때 뒤에 "b" 붙이기
 - 바이너리 데이터로 저장

```
01  # 모듈을 읽어 들입니다.  
02  from urllib import request  
03  
04  # urlopen() 함수로 구글의 메인 페이지를 읽습니다  
05  target = request.urlopen("http://www.hanbit.co.kr/images/common/logo_hanbit.  
    png") → 이 코드는 한 줄 코드이니 이어서 입력해야 합니다.  
06  output = target.read()  
07  print(output)  
08
```

```
09 # write binary[바이너리 쓰기] 모드로
10 file = open("output.png", "wb") → 바이너리 형식으로 씁니다.
11 file.write(output)
12 file.close()
```

X

b' \x89PNG\r\n\x1a\n\x00\x00\x00\rIHDR\x00\x00\x01\x04\x00\x00\x00,\x08\x06\x00\x00\x00\x83\x80\xc6\xe5\x00\x00\x00\x19tEXtSoftware\x00Adobe ImageReadyq\xc9e<\x00\x00\x0f/IDATx\nda\xecj]ttU\xc5\x19\xfe\xb3\x11 D\x11\x13\x0cF\x08\x11\x14KEA\xf6M\xd0\xd6\xa3\xb8\xe1\x12W\x8a\xa8\xc7R\xac\xb5

...생략...

- b'로 감싸져 있으므로 바이너리 데이터입니다.

- **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

- pip list 명령어 사용하여 설치된 명령어 확인하고
- pip show <설치된 모듈> 입력하여 모듈 설치된 위치 확인

```
> pip list
astroid (1.5.2)
beautifulsoup4 (4.6.0)
certifi (2017.4.17)
chardet (3.0.4)
...생략...
> pip show beautifulsoup4
Name: beautifulsoup4
Version: 4.6.0
Summary: Screen-scraping library
Home-page: http://www.crummy.com/software/BeautifulSoup/bs4/
...생략...
Location: c:\users\hasat\AppData\Local\programs\python\python36-32\lib\site-
packages

Requires:
```

- 탐색기 사용하여 Location 폴더로 들어가 여러 모듈 설치된 것 확인

BeautifulSoup 모듈의 파일



- 파일을 하나하나 열어보며 찬찬히 분석해보기