모듈(Module)

- 모듈
  - 함수, 변수와 클래스들을 모아 놓은 파일(라이브러리)
- 모듈을 사용하는 이유 : 프로그램의 재사용이 가능
- 모듈의 종류
  - 표준 모듈(https://docs.python.org/3/py-modindex.html)
  - 서드 파티(3rd party) 모듈(https://pypi.org/)
  - 사용자 정의 모듈
- 파이썬은 오픈 소스를 지향하므로 많은 사람들이 개발하여 공개한 수 많은 모듈이 존재
- 오픈 모듈은 무료로 사용 가능
- 빅데이터, 인공지능 등과 같은 프로그램들은 오픈 모듈을 사용할 수 있기 때문에 파이썬을 주로 사용

모듈 설치 및 사용

- 서드 파티 모듈 설치 - 명령어창에서 설치

  c:/>pip install modulename [--upgrade]

- 모듈 불러오기
  - 프로그램 외부에 존재하는 모듈을 프로그램 내부로 불러와서 사용하기 위해 필요한 명령(import)
  - `import module_name`
- 사용 방법

```
>>> import socket
socket 모듈에 포함된 속성(변수, 메소드, 클래스)를 사용할 수 있다
dir(socket)  - socket 모듈에 들어 있는 변수, 메소드, 클래스를 보여줌
dir(socket.socket) - socket 모듈의 socket 클래스에 포함된 변수, 메소드를 보여 줌
```

모듈 임포트 방법 몇가지 예

- import module
- import module as m
- from module import *
- from module import fn1, fn2

datetime 모듈 : 시간 관련된 정보를 제공하는 모듈

```
>>> import datetime as dt
>>> today = dt.date.today()
>>> today
datetime.date(2025, 6, 4)
>>> print(today)
2025-06-04
>>> today.year
2025
>>> today.month
6
>>> today.day
4
>>> d1 = dt.date(2000, 1, 1)
>>> d2 = dt.date(2025, 6, 4)
>>> diff = d2 -d1
>>> diff.days
9286
>>> d2.weekday() # 0은 월, 1은 화, ..., 6은 일요일
2
>>> d2.isoweekday() # 1은 월, 2는 화, ..., 7은 일요일
3
```

time

- time 모듈은 시간 관련된 정보를 제공하는 모듈
- time()은 1970년 1월1일 0시부터 지금까지 흐른 시간을 초로 표시하는 함수. 유닉스 운영체제에서 표준으로 사용되는 이 시간체계를 에포크(epoch)시간이라 함.

```
>>> import time
>>> time.time()
1613219051.9306018
>>> time.asctime() # asctime()은 현재의 날짜, 시간을 문자열로 표시하는 함수
'Sat Feb 13 21:24:18 2021'
# time.localtime()은 time.time()이 리턴한 실숫값을 사용해서 연, 월, 일, 시, 분, 초, ... 의 형태로 바꾸어 주는 함수
>>> time.localtime(time.time())
time.struct_time(tm_year=2025, tm_mon=6, tm_mday=4, tm_hour=17, tm_min=22, tm_sec=47, tm_wday=2, tm_yday=155, tm_isdst=0)
# time.asctime()은 리턴된 튜플 형태의 값을 인수로 받아서 날짜와 시간을 알아보기 쉬운 형태로 리턴하는 함수
>>> time.asctime(time.localtime(time.time()))
'Wed Jun  4 17:23:13 2025'
# 현재 시각을 리턴
>>> time.ctime()
'Wed Jun  4 17:23:27 2025'
# time.strftime()은 시간에 관계된 것을 세밀하게 표현하는 여러 가지 포맷 코드를 제공
>>> time.strftime('%c', time.localtime(time.time()))
'Wed Jun  4 17:24:26 2025'
```

- time.sleep 함수는 주로 루프 안에서 많이 사용
- 이 함수를 사용하면 일정한 시간 간격을 두고 루프 실행 가능

```
# sleep1.py
import time
for i in range(10):
    print(i)
    time.sleep(1)
```

위 예는 1초 간격으로 0부터 9까지의 숫자를 출력

math

- 수학 관련 함수를 사용할 수 있는 모듈

```
>>> import math
>>> dir(math) # math 모듈에 있는 함수 또는 상수의 이름
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin', 'asinh', 'atan',
'atan2', 'atanh', 'cbrt', 'ceil', 'comb', 'copysign', 'cos', 'cosh', 'degrees', 'dist', 'e', 'erf', 'erfc',
'exp', 'exp2', 'expm1', 'fabs', 'factorial', 'floor', 'fma', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd',
'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'isqrt', 'lcm', 'ldexp', 'lgamma', 'log', 'log10',
'log1p', 'log2', 'modf', 'nan', 'nextafter', 'perm', 'pi', 'pow', 'prod', 'radians', 'remainder', 'sin',
'sinh', 'sqrt', 'sumprod', 'tan', 'tanh', 'tau', 'trunc', 'ulp']
```

- math.gcd(), math.lcm(): 최대공약수, 최소공배수

```
>>> math.gcd(60, 100, 80)
20
>>> math.lcm(15, 25)
75
```

- 원주율 π(파이) 값, 자연로그 밑 상수 e값 등과 ceil(), floor(), trunc(), fabs(), sin(), cos(), tan() 등의 여러 수학 함수가 정의됨
- 삼각함수에서 사용하는 각은 호도(radian)이기 때문에 우리에게 익숙한 각도(degree)를 radians() 함수를 이용해 호도(radian)로 바꾸어야 함
- radian을 degree로 바꾸려면 degrees() 사용

```
>>> math.pi # 원주율
3.141592653589793
>>> math.e # 자연로그 밑 e
2.718281828459045
>>> math.sqrt(4) # 제곱근 계산
2.0
>>> math.radians(180)
3.141592653589793
>>> math.degrees(math.pi/2)
90.0
```

random module

- random 모듈은 난수가 필요하거나, 리스트의 요소를 임의로 섞거나 선택할 때 사용
- randint(start, end)는 정수 start부터 end까지 범위의 난수가 필요할 때 사용
- 비슷하지만 randrange(start, end)는 정수 start부터 end-1까지 범위의 난수가 필요할 때 사용
- random()은 0부터 1사이의 실수의 난수를 반환

```
>>> import random as rd
>>> rd.randint(1,5)
3
>>> rd.randrange(1,10)
8
>>> rd.random()
0.7237440037454024
```

- choice(seq)는 주어진 리스트 seq의 요소 중 하나를 임의로 선택
- sample(nlist, k)은 주어진 리스트 nlist에서 k개 길이의 리스트를 반환
- shuffle()은 리스트의 항목을 임의로 뒤섞는다.

```
>>> L=[0,1,2,3,4]
>>> rd.choice(L)
0
>>> rd.sample(L, 2)
[3, 4]
>>> rd.shuffle(L)
>>> L
[4, 2, 3, 0, 1]
>>>
```

pickle module

- 리스트나 딕셔너리와 같은 객체도 파일에 기록하는 방법 - pickle 모듈의 dump()와 load() 메소드를 사용
- pickle 모듈은 객체를 파일에 저장하거나 네트워크에 전송할 수 있도록 바이트 스트림으로 변환하는 작업(직렬화 또는 pickling)을 해주는 dump() 함수
- 바이트 스트림을 객체로 변환하는 작업(역직렬화 또는 unpickling)을 해주는 load() 함수를 제공
- pickling을 사용하여 바이너리 IO를 수행하기 위해서는 rb 또는 wb 모드를 사용
- 어떤 자료형이든 저장하고 불러올 수 있다.

사용 예

- 리스트 사용 예

```
>>> import pickle
>>> f = open('list.bin', 'wb')
>>> list = [1,2,3,4,5]
>>> pickle.dump(list, f)
>>> f.close()
>>> f = open('list.bin', 'rb')
>>> list = pickle.load(f)
>>> print(list)
[1, 2, 3, 4, 5]
```

- 딕셔너리 사용 예

```
import pickle
Pres = { "Kennedy": 35, "Obama": 44}

# 딕셔너리를 피클 파일에 저장
pickle.dump(Pres, open("pres.pic", "wb"))

# 피클 파일에 딕셔너리를 로딩
Pres2 = pickle.load(open("pres.pic", "rb"))
Pres2['Trump'] = 45
pickle.dump(Pres2, open("pres2.pic", "wb"))
print(Pres)
print(Pres2)
```

os module

- 환경 변수나 디렉터리, 파일 등의 OS 자원을 제어할 수 있게 해 주는 모듈

메소드명	설명
os.chdir(폴더)	현재 폴더 위치 변경
os.getcwd()	현재 폴더의 위치를 받음
os.mkdir(폴더)	폴더 만들기
os.rmdir(폴더)	폴더 삭제
os.listdir(폴더)	해당 폴더에 있는 파일의 리스트
os.remove(파일)	파일 삭제
os.rename(src, dst)	src 파일 이름을 dst로 바꿈
os.system('dir')	시스템의 dir 명령어 사용
os.popen('dir')	시스템 dir 명령어를 실행한 결괏값을 읽기 모드 형태의 파일 객체로 리턴
os.path.isfile(파일)	주어진 파일의 존재여부 검사
os.path.exists(파일)	주어진 파일의 존재여부 검사
os.path.isdir(filename)	filename이 디렉터리인지 파일인지 구별
os.path.join(dir, file)	디렉터리와 파일 이름을 이어 주는 함수
os.path.splitext(filename)	파일 이름을 확장자를 기준으로 두 부분으로 나눔

사용 예

```
>>> import os
>>> os.environ             # os.environ은 현재 시스템의 환경 변숫값을 리턴
environ({'ALLUSERSPROFILE': 'C:\\ProgramData', ... 생략 })
>>> os.environ['PATH']    # 시스템의 PATH 환경 변수 내용
'C:\\Program Files\\Python313\\Scripts\\; ... 생략 })
>>> os.system('dir')
 C 드라이브의 볼륨에는 이름이 없습니다.
 볼륨 일련 번호: 5029-0858
 C:\zzz 디렉터리
2025-06-04  오후 06:29    <DIR>          .
2025-06-04  오후 06:28                 0 a.txt
2025-06-04  오후 06:29                 0 b.txt
               2개 파일                   0 바이트
               1개 디렉터리  369,672,740,864 바이트 남음
0
>>> f = os.popen('dir')
>>> print(f.read())
 C 드라이브의 볼륨에는 이름이 없습니다.
 볼륨 일련 번호: 5029-0858
 C:\zzz 디렉터리
2025-06-04  오후 06:29    <DIR>          .
2025-06-04  오후 06:28                 0 a.txt
2025-06-04  오후 06:29                 0 b.txt
               2개 파일                   0 바이트
               1개 디렉터리  369,672,749,056 바이트 남음
```

- 만약 주어진 파일의 존재여부를 검사하고 싶으면?

```
import os.path
if os.path.isfile("Lectures.txt"): # os.path.exists("Lectures.txt")도 같은 역할
    print("file exists.")
```