기초빅데이터프로그래밍

표준 모듈(라이브러리)

서강대학교
SOGANG UNIVERSITY

목차

1 os

- os 모듈은 운영체제에서 제공하는 정보를 제공하거나 운영체제의 기능을 사용할 수 있는 방법을 제공한다.
- os 패키지는 내부에 많은 패키지와, 모듈, 함수, 클래스들로 이루어져 있다.

1.1 os.name

- 파이썬이 실행되는 운영체제의 이름을 표시한다.
- 다음의 이름들 중 하나가 표시 된다
  - ‘posix’, ‘nt’, ‘ce’, ‘java’.

```
>>> import os
>>> os.name
>>> ‘nt’
```

1.2 os.getcwd()

- 현재 파이썬의 작업 디렉터리를 가져온다.

```
>>> import os
>>> os.getcwd()
>>> 'D:₩₩working₩₩test'
```

1.3 os.chdir("새_경로")

- 현재 파이썬이 작업 디렉터리를 "새_경로"로 바꾼다.

```
>>> import os
>>> os.chdir("d:₩₩working")
>>> os.getcwd( )
'd:₩₩working'
```

1.4 os.path.join(경로1, 경로2, ...)

- 주어진 경로 요소들을 합해서 하나의 경로를 만들어 준다.
  - 윈도우즈 경우 디렉터리간의 구분을 "**₩₩**"로 하는 반면, 리눅스 계열은 '**/**'로 한다.
  - os.path.join 함수는 **해당 운영체제에 맞는 경로를** 생성해 준다.
  - 윈도우즈에서 실행한 경우.

```
>>> import os
>>> os.path.join( os.getcwd( ), "test")
'd:₩₩working₩₩test'
```

  - 리눅스에서 실행한 경우

```
>>> import os
>>> os.path.join( os.getcwd( ), "test")
'/home/kang/working/test'
```

1.5 os.listdir(["경로"])

- "경로"내에 포함된 **파일과 디렉터리를 리스트 형태로 반환한다**.
- "경로"가 지정되지 않으면 현재 작업 디렉터리 내의 파일과 디렉터리를 리스트 형태로 반환한다.

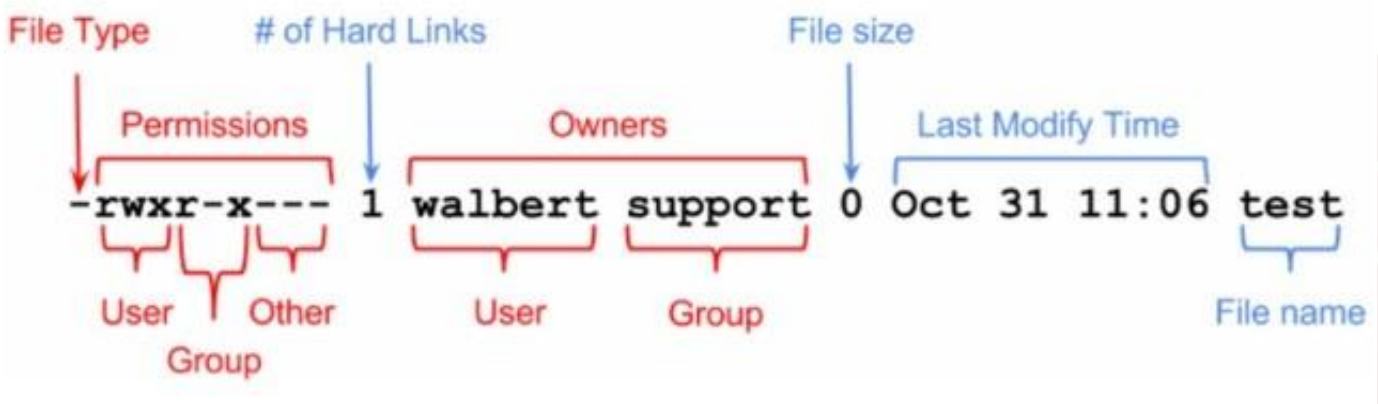
```
>>> import os
>>> dir_list = os.listdir("d:₩₩working₩₩test")
>>> print(dir_list)
['car.py', 'student.py']
```

1.6 os.mkdir("경로" [, mode=0777])

- 디렉터리를 생성한다.

```
>>> import os
>>> os.mkdir( os.path.join( os.getcwd( ), "test"))
>>> os.listdir( )
['car.py', 'student.py', 'test']
```

- 현재 디렉터리에 test라는 디렉터리를 만든다.


File Type
# of Hard Links
File size
Permissions
Owners
Last Modify Time
-rwxr-x--- 1 walbert support 0 Oct 31 11:06 test
User
Other
Group
User
Group
File name

1.7 os.remove("파일_경로")

- "파일_경로"의 파일을 삭제한다.

```
>>> import os
>>> os.remove( os.path.join( os.getcwd( ), "car.py"))
>>> os.listdir( )
['student.py', 'test']
```

- 현재 디렉터리의 car.py 파일을 삭제한다.

1.8 os.rmdir("경로")

- 디렉터리를 삭제한다.

```
>>> import os
>>> os.rmdir( os.path.join( os.getcwd( ), "test"))
>>> os.listdir( )
['student.py']
```

- 현재 디렉터리에서 "test" 디렉터리를 삭제한다.

```
import os
os.name
```

```
'nt'
```

```
os.getcwd()
```

```
'C:\\Users\\KyungHee'
```

```
os.chdir("C:\\Catholic\C++")
```

```
os.getcwd()
```

```
'C:\\Catholic\\C++'
```

```
os.chdir("C:\\Users")
```

```
os.path.join(os.getcwd(),"Kyunghee")
```

```
'C:\\Users\\Kyunghee'
```

```
dir_list=os.listdir("C:\\Users")
print(dir_list)
```

```
['All Users', 'Default', 'Default User', 'Default.m
igrated', 'desktop.ini', 'EasySurvey', 'KyungHee',
'Public', 'UpdatusUser']
```

```
os.chdir("C:\\Entry")
os.mkdir(os.path.join(os.getcwd(),"test1"))
os.listdir()
```

```
['content_resources_200_percent.pak',
 'content_shell.pak',
 'd3dcompiler_47.dll',
 'Entry.exe',
 'ffmpeg.dll',
 'icon.ico',
 'icudtl.dat',
 'libEGL.dll',
 'libGLESv2.dll',
 'LICENSE',
 'LICENSES.chromium.html',
 'locales',
 'natives_blob.bin',
 'node.dll',
 'resources',
 'snapshot_blob.bin',
 'test',
 'test1',
 'ui_resources_200_percent.pak',
 'Update.exe',
 'version',
 'xinput1_3.dll',
 '엔트리 제거.exe']
```

glob 모듈

glob() 함수는 경로에 대응되는 모든 파일 및 디렉토리의 리스트를 반환

- 윈도우의 dir, 리눅스의 ls와 유사한 기능제공
- ***와 ? 도 사용가능**

```
import glob
files= glob.glob('*')
for x in files:
    if os.path.isdir(x):
        print("{} <DIR>".format(x))
    else:
        print(x)
```

```
content_resources_200_percent.pak
content_shell.pak
d3dcompiler_47.dll
Entry.exe
ffmpeg.dll
icon.ico
icudtl.dat
libEGL.dll
libGLESv2.dll
LICENSE
LICENSES.chromium.html
locales <DIR>
natives_blob.bin
node.dll
resources <DIR>
snapshot_blob.bin
test <DIR>
test1 <DIR>
ui_resources_200_percent.pak
Update.exe
version
xinput1_3.dll
엔트리 제거.exe
```

2 sys

- sys모듈은 파이썬 인터프리터와 관련된 정보 및 기능들을 제공한다.
- sys 모듈 중에서 가장 많이 쓰이는 요소 중 하나는 명령행 인자를 받아 오는 요소인 **sys.argv**이다.

2.1 sys.argv

- 파이썬 프로그램을 실행할 때 주어진 명령행 인자를 담고 있는 리스트 이다.

C:₩> python command_arguments.py 1 2 “Hello World”

- 위와 같이 파이썬 프로그램을 실행시키면, command_arguments.py, 1, 2, “Hello World”의 4개의 명령행 인자가 **sys.argv**에 담겨진다.

```
# command_arguments.py

import sys

if __name__ == "__main__":
    print("명령행 인자의 갯수
    
    for item in sys.argv:
        print(item)
```

```
%%writefile command_arguments.py
import sys
if __name__ == "__main__":
    print("명령행 인자의 개수:", len(sys.argv))

    for item in sys.argv:
        print(item)
```

```
Writing command_arguments.py
```

```
%run command_arguments.py 1 2 "Hello"
```

```
명령행 인자의 개수: 4
command_arguments.py
1
2
Hello
```

- **sys.copyright**
  - 설치된 파이썬의 저작권 정보를 담고 있다.
- **sys.version**
  - 설치된 파이썬의 버전 정보를 담고 있다.
- **sys.stdin**
  - 파이썬 인터프리터의 표준 입력 객체이다.
- **sys.stdout**
  - 파이썬 인터프리터의 표준 출력 객체이다.
- **sys.stderr**
  - 파이썬 인터프리터의 표준 에러 객체이다.

```
sys.copyright
```

```
'Copyright (c) 2001-2016 Python Software Foundation.\nAll Rights Reserve
d.\n\nCopyright (c) 2000 BeOpen.com.\nAll Rights Reserved.\n\nCopyright
(c) 1995-2001 Corporation for National Research Initiatives.\nAll Rights
Reserved.\n\nCopyright (c) 1991-1995 Stichting Mathematisch Centrum, Ams
terdam.\nAll Rights Reserved.'
```

```
sys.version
```

```
'3.6.0 |Anaconda 4.3.1 (64-bit)| (default, Dec 23 2016, 11:57:41) [MSC
v.1900 64 bit (AMD64)]'
```

```
sys.stdin
```

```
<_io.TextIOWrapper name='<stdin>' mode='r' encoding='cp949'>
```

```
sys.stdout
```

```
<ipykernel.iostream.OutStream at 0x215883bcc88>
```

```
sys.stderr
```

```
<ipykernel.iostream.OutStream at 0x215883bcbe0>
```

3 math

- math 모듈은 수학 관련 함수를 제공하는 모듈이다.

메서드	내용
math.exp(x)	e ** x 값을 구한다.
math.log(x [, base])	log 값을 구한다.
math.pow(x, y)	거듭제곱을 구한다.
math.sin(x)	사인 함수 값을 구한다.
math.cos(x)	코사인 함수 값을 구한다.
math.tan(x)	탄젠트 함수 값을 구한다.

math.asin(x)	아크사인 함수 값을 구한다.
math.acos(x)	아크코사인 함수 값을 구한다.
math.atan(x)	아크탄젠트 함수 값을 구한다.
math.degrees(x)	라디안 각도 x를 디그리 단위로 바꾼다.
math.radians(x)	디그리 각도 x를 라디안 단위로 바꾼다.
math.pi	파이 상수 값을 구한다.
math.e	지수 상수 값을 구한다.

```
>>> import math
>>> math.sin( math.pi / 2)
1.0
>>> math.cos(math.pi / 2)
6.123233995736766e-17
>>> math.tan(math.pi / 2)
1.63312393539537e+16
>>> math.log( math.e )
1.0
>>> math.log( math.pow(math.e, 2))
2.0
>>> math.degrees( math.pi )
180.0
>>> math.radians( 180.0 )
3.141592653589793
```

$$\frac{2000}{\log(20)} * \sin(2\pi)$$

```
import math as m
print((2000/m.log(20)*m.sin(2*m.pi)))#밑이 e인 자연로그
```

```
-1.6351885780427125e-13
```

```
print((2000/m.log(20,10)*m.sin(2*m.pi)))
```

```
-3.765160844035281e-13
```

```
print((2000/m.log10(20)*m.sin(2*m.pi)))#log10():밑이 10인 상용로그
```

```
-3.7651608440352804e-13
```

```
print((2000/m.log(20, 2.718)*m.sin(2*m.pi)))#e=2.718
```

```
-1.6350190347069722e-13
```

```
print((2000/m.log(20, m.e)*m.sin(2*m.pi)))
```

```
-1.6351885780427125e-13
```

4 time

용어	내용
타임스탬프 (**TimeStamp**)	컴퓨터에서 시간을 측정하는 방법으로 **1970년 1월 1일 자정** 이후로 측정한 절대시간이다. 절대적인 시간은 용도에 맞도록 변환 되어서 사용된다.
협정세계시 (**UTC**, Universal Time Coordinated)	1972년부터 시행된 국제 표준시를 나타낸다. 세슘 원자의 진동수에 의거한 초의 길이가 기준이 된다.
그리니치 평균시 (**GMT**, Greenwich Mean Time)	런던 그리니치 천문대의 자오선상에서의 평균 태양시를 나타낸다. (1972년부터 협정세계시를 사용하지만, 동일한 표현으로 널리 쓰인다.)
지방 표준시 (**LST**, Local Standard Time)	UTC를 기준으로 경도 15도마다 1시간 차이가 발생하는 시간이다. (한국은 동경 135도를 기준으로 UTC보다 9시간 빠르다.)
일광절약 시간제 (**DST**, Daylight Saving Time)	흔히 서머타임으로 알고 있는 것으로, 에너지 절약을 목적으로 시간을 한 시간씩 앞당기거나 뒤로 미루는 제도에 의해 표현된 시간.

한 시점의 시간을 표현하는 정보들을 묶어 struct_time sequence object 으로 표현한다.

strcut_time 시퀀스 속성	내용
tm_year	년도 (1 ~ 9999)
tm_mon	월 (1 ~ 12)
tm_mday	일 (1 ~ 31)
tm_hour	시 (0 ~ 23)
tm_min	분 (0 ~ 59)
tm_sec	초 (0 ~ 61) #윤초 61을 도입해서 조정
tm_wday	각 요일을 숫자로 나타낸다. (0~6)
tm_yday	1월 1일부터 오늘까지 누적된 날짜 값이다. (1 ~ 366)
tm_isdst	일광절약 시간제(서머타임)을 나타낸다 (0, 1, -1) # 실시안함, 실시, 모름

time module 함수

- time.time()
  - **1970년 1월 1일 자정 이후로 누적된 초를 float 단위**로 반환한다.

- time.sleep(초)
  - 현재 동작중인 프로세스를 주어진 초만큼 정지(sleep) 시킨다.

- time.gmtime([secs])
  - 입력된 초를 변환하여, **UTC 기준의 struct_time** 시퀀스 객체로 반환한다.
  - 인자가 입력되지 않은 경우, time()을 이용하여 현재 시간을 반환한다.
- time.localtime([secs])
  - 입력된 초를 변환하여, **지방 표준시 기준의 struct_time** 시퀀스 객체를 반환
  - 인자가 입력되지 않은 경우, time()을 이용하여 현재 시간을 변환한다.

- time.asctime([t])
  - struct_time 시퀀스 객체를 인자로 받아서 'Mon Sep 1 5:55:28:2014'와 같은 문자열형태로 반환한다.
- time.mktime(t)
  - 지방 표준시인 struct_time 시퀀스 객체를 인자로 받아서 time()과 같은 누적된 초를 반환한다.
- time.strftime(format [, t])
  - struct_time 시퀀스 객체를 **사용자가 정의한 형식으로** 변경하여 문자열로 반환한다.

```
In [1]: import time
        time.time()
```

```
Out[1]: 1523781615.1958458
```

```
In [4]: t2= time.gmtime(1523781615.1958458)
        print(t2)
```

```
time.struct_time(tm_year=2018, tm_mon=4, tm_mday=15, tm_hour=8,
tm_min=40, tm_sec=15, tm_wday=6, tm_yday=105, tm_isdst=0)
```

```
In [3]: time.localtime()
```

```
Out[3]: time.struct_time(tm_year=2018, tm_mon=4, tm_mday=15, tm_hour=1
7, tm_min=41, tm_sec=32, tm_wday=6, tm_yday=105, tm_isdst=0)
```

```
In [5]: time.asctime(t2)
```

```
Out[5]: 'Sun Apr 15 08:40:15 2018'
```

```
In [6]: time.mktime(t2)
```

```
Out[6]: 1523749215.0
```

```
In [9]: import datetime
        now = datetime.datetime.now()
        print(now)
```

```
2018-04-15 18:30:11.943773
```

```
print(now.strftime("%Y:%m:%d %H:%M:%S %A"))
st1=str(now)
print(st1)
st2=datetime.datetime.strptime(st1, "%Y-%m-%d %H:%M:%S.%f")
print(st2)
st2
```

```
2018:04:15 18:30:11 Sunday
2018-04-15 18:30:11.943773
2018-04-15 18:30:11.943773

datetime.datetime(2018, 4, 15, 18, 30, 11, 943773)
```

```
t1= time.gmtime()
print(t1)
print("\n", time.strftime("%Y:%B:%d %H:%M:%S %A", t1))
```

```
time.struct_time(tm_year=2018, tm_mon=4, tm_mday=15, tm_hour=1
1, tm_min=53, tm_sec=13, tm_wday=6, tm_yday=105, tm_isdst=0)

 2018:April:15 11:53:13 Sunday
```

```
t3=time.localtime()
print(t3)
print("\n", time.strftime("%Y:%B:%d %H:%M:%S %a", t3))
```

```
time.struct_time(tm_year=2018, tm_mon=4, tm_mday=15, tm_hour=2
0, tm_min=55, tm_sec=4, tm_wday=6, tm_yday=105, tm_isdst=0)

 2018:April:15 20:55:04 Sun
```

strftime() - 날짜의 출력 형식 지정

```
from time import localtime, strftime
strftime("%B %dth %A %I:%M", localtime())
```

```
'May 07th Sunday 04:52'
```

```
strftime("%Y-%m-%d %A %I:%M", localtime())
```

```
'2017-05-07 Sunday 04:53'
```

```
strftime("%y-%m-%d %A %I:%M", localtime())
```

```
'17-05-07 Sunday 04:57'
```

```
strftime("%x %X", localtime())
```

```
'05/07/17 16:58:34'
```

- 현재시간을 2022/04/13 과 같이 /로 구분해서 그리고 아래와 같이 출력해보세요.

년도: 2022
월: 04
일: 13
요일: Wednesday

time.strptime(string [, format])

- 사용자가 정의한 형식 문자열을 struct_time 시퀀스 객체로 변환한다.

형식 지정자	내용
%y	연도를 축약하여 표시한다.
%Y	연도를 축약하지 않고 표시한다.
%b	월 이름을 축약하여 표시한다.
%B	월 이름을 축약하지 않고 표시한다.
%m	월을 숫자로 표시한다. (01 ~ 12)
%d	일을 표시한다. (01 ~ 31)

%H	24시를 기준으로 한 시간을 표시한다. (00 ~ 23)
%I	12시를 기준으로 한 시간을 표시한다. (01 ~ 12)
%M	분을 표시한다. (00 ~ 59)
%S	초를 표시한다. (00 ~ 61)
%p	오전(AM) / 오후 (PM)을 표시한다.
%a	축약된 요일 이름을 표시한다.
%A	축약되지 않은 요일 이름을 표시한다.
%w	요일을 숫자로 표시한다. (일요일은 0으로 표시한다.)
%j	1월 1일부터 누적된 날짜(001 ~ 366)를 표시한다.

```
# get_time.py

import time

if __name__ == ”__main__”:
    t1 = time.time( ) # 1
    print(t1)

    t2 = time.gmtime( ) # 2
    print(t2)

    t3 = time.localtime( ) # 3
    print(t3)
```

```
import time

if __name__ == "__main__":
    t1 = time.time()
    print(t1)

    t2=time.gmtime()
    print(t2)

    t3= time.localtime()
    print(t3)
```

```
1493542860.2602592
time.struct_time(tm_year=2017, tm_mon=4, tm_mday=30,
tm_hour=9, tm_min=1, tm_sec=0, tm_wday=6, tm_yday=12
0, tm_isdst=0)
time.struct_time(tm_year=2017, tm_mon=4, tm_mday=30,
tm_hour=18, tm_min=1, tm_sec=0, tm_wday=6, tm_yday=1
20, tm_isdst=0)
```

1. 1970년 1월 1일 자정 이후 현재까지 누적된 초
2. UTC기준의 현재시간
3. 로컬 표준시 기준의 현재시간

3초 sleep하는 예제

sleep_example.py

```
import time

if __name__ == "__main__":
    input("Enter a number:")
    t1 = time.time( )

    time.sleep(3)
    input("Enter a number again:")

    t2 = time.time( )
    time_gap = t2-t1
    print("Time gap:", time_gap)
```

```
Enter a number:1
Enter a number again:1
Time gap: 3.515467882156372
```

```
import time

if __name__ == "__main__":
    input("Enter a number:")
    t1 = time.time( )

    #time.sleep(3)
    input("Enter a number again:")

    t2 = time.time( )
    time_gap = t2-t1
    print("Time gap:", time_gap)
```

```
Enter a number:1
Enter a number again:1
Time gap: 0.4363210201263428
```

strptime() - 지정된 형식 문자열 형태로 표현된 시간을 struct_time 객체로 변환

```
import time
tstr= time.ctime(1523781615.1958458)
print(tstr)
```

```
Sun Apr 15 17:40:15 2018
```

```
time.strptime(tstr)        #생략되면 "%a %b %d %H:%M:%S %Y" 포맷으로 설정
```

```
time.struct_time(tm_year=2018, tm_mon=4, tm_mday=15, tm_hour=17, tm_min
=40, tm_sec=15, tm_wday=6, tm_yday=105, tm_isdst=-1)
```

```
time.strptime(tstr, "%a %b %d %H:%M:%S %Y")
```

```
time.struct_time(tm_year=2018, tm_mon=4, tm_mday=15, tm_hour=17, tm_min
=40, tm_sec=15, tm_wday=6, tm_yday=105, tm_isdst=-1)
```

```
tstr2= time.strftime("%B %dth %A %H:%M:%S", time.localtime())
print(tstr2)
```

```
April 15th Sunday 21:29:15
```

```
time.strptime(tstr2)
```

⇦ 어떻게 될까요?

```
time.strptime(tstr2)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-32-dccba7e2442f> in <module>()
----> 1 time.strptime(tstr2)

C:\ProgramData\Anaconda3\lib\_strptime.py in _strptime_time(data_string, format)
    557     """Return a time struct based on the input string and the
    558     format string."""
--> 559     tt = _strptime(data_string, format)[0]
    560     return time.struct_time(tt[:time._STRUCT_TM_ITEMS])
    561

C:\ProgramData\Anaconda3\lib\_strptime.py in _strptime(data_string, format)
    360     if not found:
    361         raise ValueError("time data %r does not match format %r" %
--> 362                          (data_string, format))
    363     if len(data_string) != found.end():
    364         raise ValueError("unconverted data remains: %s" %

ValueError: time data 'April 15th Sunday 21:29:15' does not match format '%a %b %d %H:%M:%S %Y'
```

5 datetime

- datetime 모듈은 날짜와 시간에 관련된 클래스를 제공한다.
- datetime객체는 date객체와 time객체의 모든 정보를 포함하는 단일 객체

class datetime.**datetime**(*year, month, day, hour=0, minute=0, second=0, microsecond=0, tzinfo=None, *, fold=0*)

메서드	내용
datetime.date(year, month, day)	일반적으로 사용하는 년,월,일로 표시되는 날짜를 표현한다.
datetime.date.fromtimestamp(timestamp)	타임스탬프 값을 인자로 받아서 date 객체를 반환한다.
datetime.date.fromordinal(ordinal)	1년 1월 1일 이후로 누적된 날짜로부터 date 객체를 반환한다.
datetime.date.today()	현재 시스템의 오늘 날짜 date 객체를 반환한다.

```
>>> import datetime
>>> import time
>>> datetime.date(2014, 8, 11)
datetime.date(2014, 8, 11)
>>> datetime.date.fromtimestamp(time.time())
datetime.date(2014, 8, 11)
>>> datetime.date.today( )
datetime.date(2014, 8, 11)
>>> datetime.date(2014, 9, 31)
Traceback (most recent call last):
    ...
    ...
ValueError: day is out of range for month
```

```
import datetime as dt
import time as tm
dt.date(2018,4, 30)
```

```
datetime.date(2018, 4, 30)
```

```
dt.date.fromtimestamp(tm.time())
```

```
datetime.date(2018, 4, 29)
```

```
dt.date.today()
```

```
datetime.date(2018, 4, 29)
```

```
dt.date(2018,4,31)
```

```
---------------------------------------------
ValueError                                Tra
<ipython-input-8-fad583cb2358> in <module>()
----> 1 dt.date(2018,4,31)

ValueError: day is out of range for month
```

실습 1

- datetime을 이용하여 요일을 구하는 프로그램을 작성하시오.
- 실행 예

```
Writing W_2.py
```

```
%run W_2.py 2019 10 1
화요일
```

```
%run W_2.py 2019 10 14
월요일
```

6 random

- random 모듈은 의사 난수 생성에 관한 함수 및 클래스를 제공한다.
- random 모듈은 확률 기반의 응용 프로그램을 만들거나 게임을 만들 때 유용하게 사용된다.

- 기본 메서드들

메서드	내용
random.seed(a=None, version=2)	랜덤 숫자 생성자를 초기화한다.
random.getstate()	현재 랜덤 생성자의 상태를 반환한다.
random.setstate(state)	getstate()에서 구한 랜덤 생성자의 상태로 설정한다.
random.getrandbits(k)	k개의 랜덤한 bit들을 가진 정수를 구한다.

- 정수 관련 메서드들

메서드	내용
random.randrange(stop)	range(stop)으로부터 랜덤하게 숫자를 고른다.
random.randrange(start, stop, [step])	range(start, stop, [step])으로부터 랜덤하게 숫자를 고른다.
random.randint(a, b)	a <= N <=b를 만족하는 정수 N을 구한다.

- 실수 관련 메서드들

메서드	내용
random.random()	[0.0, 1.0)사이의 부동소수를 랜덤하게 하나 구한다. 1.0은 포함되지 않는다.
random.uniform(a, b)	[a, b]사이의 부동소수를 랜덤하게 하나 구한다. b는 포함되지 않는다.
random.gauss(mu, sigma)	가우시안 분포를 따르게 랜덤하게 하나의 숫자를 구한다.

7 표준 모듈을 이용한 간단한 프로그래밍

```
# lotto_number_generator.py

import random

def get_lotto_numbers():
    lotto_numbers = []

    while True:
        if len(lotto_numbers) == 6:
            break

        number = random.randint(1, 45)
        if number in lotto_numbers:
            continue
        else:
            lotto_numbers.append(number)

    return sorted(lotto_numbers)

if __name__ == "__main__":
    lotto_numbers = get_lotto_numbers()
    print(lotto_numbers)
```

```
import random

def get_lotto_numbers():
    lotto_numbers = []

    while True:
        if len(lotto_numbers) == 6:
            break

        number = random.randrange(1,20, 3)
        #number = random.randint(1, 45)
        if number in lotto_numbers:
            continue
        else:
            lotto_numbers.append(number)

    return sorted(lotto_numbers)

if __name__ == "__main__":
    lotto_numbers = get_lotto_numbers()
    print(lotto_numbers)
```

```
[1, 4, 10, 13, 16, 19]
```

실습 2

1. 파이썬을 이용하여 자신의 운영체제의 종류를 구하시오.

2. 윤년 요일 구하기 프로그램을 수정하여 아래와 같이 연,월,일을 명령행 인자(command line arguments)의 형태로 받아들이도록 재작성하시오.

```
Overwriting leapArg.py
```

```
%run leapArg.py 2018 4 15
일요일
```

3. 다음의 수식을 math 모듈을 이용해서 작성하시오.

$\frac{2000}{\log(20)} * \sin(2\pi)$, $3^{100} * \log(300)$, $\sin(45^o) * \sqrt{2}$

4. random 모듈을 이용하여 가위, 바위, 보 게임을 만드시오

random 모듈을 이용하여 가위, 바위, 보 게임을 만들어 보시오. (실행예)

```
가위, 바위, 보 중 하나를 선택하시오: 가위
플레이어: 가위
컴퓨터: 가위
비겼습니다.
```

```
가위, 바위, 보 중 하나를 선택하시오: 보
플레이어: 보
컴퓨터: 바위
당신이 이겼습니다.
```

```
가위, 바위, 보 중 하나를 선택하시오: 바위
플레이어: 바위
컴퓨터: 보
컴퓨터가 이겼습니다.
```