Python

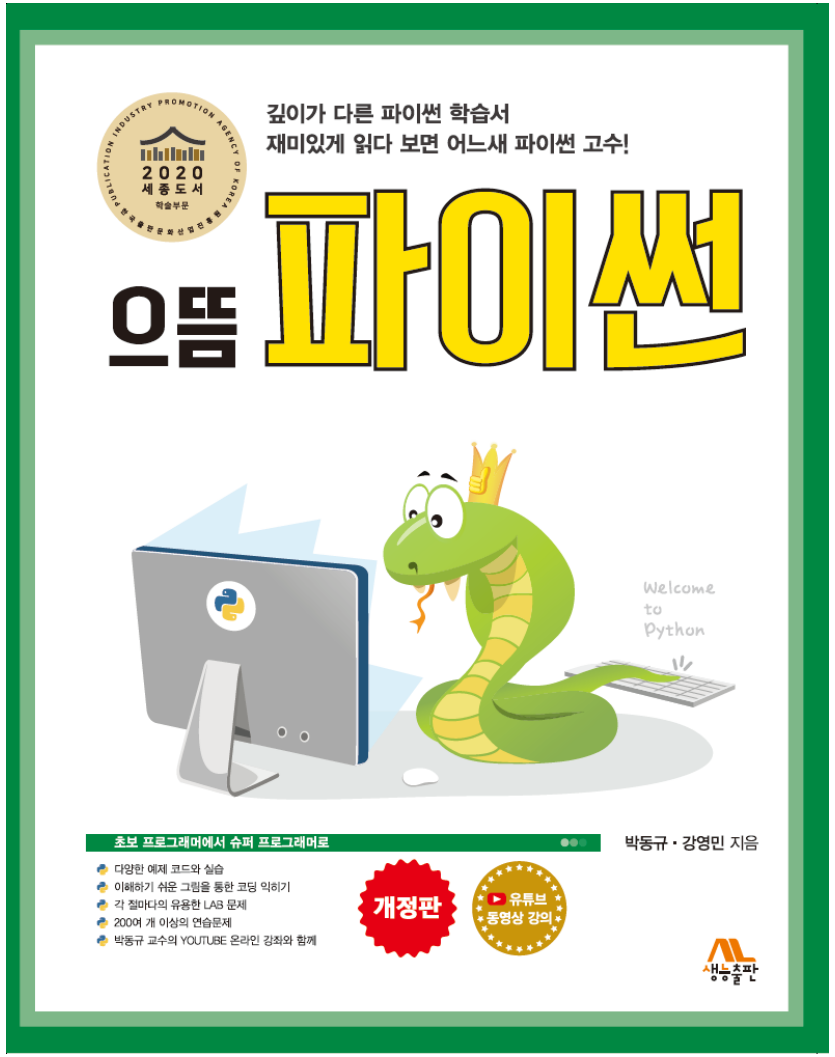
모듈과 활용

Spring 2025

AI융합학과
Seongbok Baik
sbbaik@dju.kr

00 Text Book

교재명	으뜸 파이썬
저자	박동규, 강영민
출판사	생능출판사
발행년	2024.06.14



학습목표

- 모듈의 정의와 종류에 대해서 알아본다.
- 파이썬에서 제공하는 모듈을 활용하는 방법을 이해한다.
- 파이썬에서 제공하는 기본 모듈을 이용하여 강력한 기능을 활용할 수 있다.
- from ... import 구문을 이용하여 특정 클래스나 메소드를 가지고 올 수 있다.
- Python Module Index 페이지를 이용하여 원하는 모듈에 대한 정보를 알아본다.
- 시간과 날짜를 다루는 모듈을 이해하고 활용할 수 있다.
- 강력한 수학 모듈을 통해 다양하고 복잡한 연산을 수행할 수 있다.
- random 모듈을 사용하여 임의의 수를 생성하는 기능을 활용할 수 있다.

8.1 모듈과 import 문법

- **모듈**module
  - 파이썬 함수나 변수 또는 클래스들을 모아놓은 스크립트 파일
  - 파이썬은 수많은 개발자들에 의해서 개발된 많은 모듈이 있음
  - 만들어진 모듈을 가져올 때에는 'import'와 함께 모듈 이름을 써 줌

```
import 모듈 이름1 [, 모듈 이름2, ...]
```

  - 사용할 때에는 모듈 이름에 점(.)을 찍은 후 모듈 안의 구성요소를 작성

8.1 모듈과 import 문법

- 파이썬 설치와 함께 제공되는 모듈을 **파이썬 표준 라이브러리**python standard library라고 함
- 문자열과 텍스트 처리를 위한 모듈, 이진 데이터 처리, 날짜, 시간, 배열 등의 자료형 처리를 위한 모듈, 수치 연산과 수학 함수 모듈, 파일과 디렉터리 접근, 유닉스 시스템의 데이터베이스 접근을 위한 모듈, 데이터 압축, 그래픽 모듈 등 100여 가지 이상의 표준 라이브러리들이 있다

8.2 날짜와 시간 모듈 datetime

- datetime 모듈
  - 날짜와 시간에 관한 기능을 제공하고 조작할 수 있는 모듈
- 아래의 datetime.datetime.now()에서 앞의 datetime은 모듈의 이름, 뒤의 datetime은 클래스의 이름
  - 이 메소드를 사용하면 현재의 년, 월, 일, 시, 분, 초, 밀리초 단위의 시간까지 나타남

대화창 실습: datetime 모듈의 임포트와 사용

```
>>> import datetime        # 이하 이 문장은 생략함
>>> datetime.datetime.now()
    datetime.datetime(2025, 1, 2, 6, 57, 27, 904565)
```

8.2 날짜와 시간 모듈 datetime

- date 클래스의 today() 메소드는 현재 날짜를 today에 반환
- today 값을 프롬프트에서 출력하면 datetime.date 클래스가 가진 년, 월, 일을 출력해줌

대화창 실습: datetime 모듈의 사용법

```
>>> today = datetime.date.today()
>>> print(today)
    2025-01-02
>>> today
    datetime.date(2025, 1, 2)
>>> today.year
    2025
>>> today.month
    1
>>> today.day
    2
```

8.2 날짜와 시간 모듈 datetime

- dir() 함수
  - 모듈이 가진 클래스의 목록을 출력

대화창 실습: datetime 모듈의 속성과 클래스를 알아보는 dir() 함수

```
>>> dir(datetime)
['MAXYEAR', 'MINYEAR', '__builtins__', '__cached__', '__doc__', '__
file__', '__loader__', '__name__', '__package__', '__spec__', '_divide_
and_round', 'date', 'datetime', 'datetime_CAPI', 'time', 'timedelta',
'timezone', 'tzinfo']
```

- dir() 함수는 datetime 오브젝트에서 어떤 클래스, 속성, 메소드를 반환
- MAXYEAR는 datetime 오브젝트가 표현 가능한 최대 년도로 9999 값을 가짐
- MINYEAR는 1 값을 가짐
- date, datetime, datetime_CAPI, time, timedelta, timezone, tzinfo와 같은 클래스들은 날짜, 시간, 시간대, 시간대 정보를 편리하게 이용할 수 있는 기능이 있음

실습

replace() 메소드

- datetime에서 날짜나 시간 값을 변경하고 싶을 때 사용

대화창 실습: 날짜 및 시간 변경

```
>>> start_time = datetime.datetime.now()
>>> start_time.replace(month = 12, day = 25)
    datetime.datetime(2024, 12, 25, 7, 1, 25, 880317)
```

8.2 날짜와 시간 모듈 datetime

실습

as 구문과 replace() 메소드

- 매번 "[모듈 이름].[클래스 이름].[메소드이름]()"을 점 연산자로 구분해서 적는 것은 매우 번거로움
- as 구문을 사용하여 모듈 이름 datetime을 별칭인 dt로 간단하게 줄일 수 있다.

```
import [모듈 이름] as [모듈의 별칭]
```

대화창 실습: 현재 날짜 및 시간 변경

```
>>> import datetime as dt    # dt라는 별칭을 사용함
>>> start_time = dt.datetime.now()
>>> start_time.replace(month = 12, day = 25)
    datetime.datetime(2024, 12, 25, 7, 1, 25, 880317)
```

이 결과는 컴퓨터의 실행시간에 따라 매번 달라질 수 있음

8.2 날짜와 시간 모듈 datetime

import ~ as

- 긴 모듈 이름을 간략하게 부르는 별명을 붙여주는 방법

NOTE: import ~ as 문법

모듈 내에 있는 클래스나 메소드를 활용할 때 점으로 연결해주는데, 모듈 이름이 너무 긴 경우 계속해서 이름을 써주는 것이 번거로울 수 있다. 이때, as를 활용하여 모듈의 새 이름을 지정할 수 있다. 보통 math 모듈은 m, datetime은 dt, random은 rd, turtle은 t와 같은 짧은 이름을 널리 사용한다.

```
ex1) import datetime as dt
ex2) import random as rd
ex3) import math as m
ex4) import turtle as t
```

from ~ import ~ 구문

- "[모듈 이름].[클래스 이름].[메소드이름]()" 방식의 호출을 간단하게 하는 방법

대화창 실습: from import를 이용한 현재 날짜 및 시간 변경

```
>>> from datetime import datetime
>>> start_time = datetime.now()
>>> start_time.replace(month = 12, day = 25)
datetime.datetime(2024, 12, 25, 7, 1, 25, 880317)
```

datetime.datetime.을 datetime.으로 생략할 수 있다.

8.2 날짜와 시간 모듈 datetime

from ~ import ~ 구문

LAB 8-1: 오늘의 날짜와 현재 시간 출력하기

1. datetime 모듈을 사용하여 오늘의 날짜와 시간을 다음과 같이 출력하여라. 이때 hour 값이 10일 경우 오전 10시, 13이면 오후 1시와 같이 오전/오후 정보를 출력하여라.

```
오늘의 날짜 : 2025년 2월 19일
현재시간 : 오후 6시 50분 11초
```

8.2 날짜와 시간 모듈 datetime

8.2.1 남은 시간 구하기

코드 8-1: datetime 모듈을 사용하여 크리스마스까지 남은 시간 구하기

xmas_left_day.py

```
import datetime as dt

today = dt.date.today()
print(f'오늘은 {today.year}년 {today.month}월 {today.day}일입니다')

xMas = dt.datetime(2026, 12, 25)
gap = xMas - dt.datetime.now()

print(f'크리스마스까지는 {gap.days}일 {gap.seconds // 3600}시간 남았습니다.')
```

실행결과

```
오늘은 2023년 6월 17일입니다
크리스마스까지는 1286일 13시간 남았습니다.
```

8.2 날짜와 시간 모듈 datetime

8.2.1 남은 시간 구하기

LAB 8-2: 남은 날짜 계산하기

1. 앞서 배운 남은 날짜 계산 프로그램을 수정하여 오늘 날짜와 함께 2027년 크리스마스까지의 남은 날짜와 시간을 구해서 출력해 보자.

```
오늘은 2023년 6월 17일입니다.
2027년 크리스마스 까지는 1651일 13시간 남았습니다.
```

2. 앞서 배운 남은 날짜 계산 프로그램을 수정하여 2036년 1월 1일까지의 남은 날짜와 시간을 구해서 출력해 보자.

```
오늘은 2023년 6월 17일입니다
2036년 새해 까지는 4580일 15시간 남았습니다.
```

3. 자신의 생일을 비교 대상 날짜로 설정하여, 다가오는 자신의 생일까지 남은 날짜와 시간을 출력해 보자.

```
오늘은 2025년 8월 29일입니다
2026년 생일까지는 309일 15시간 남았습니다.
```

8.2 날짜와 시간 모듈 datetime

8.2.2 100일 뒤 날짜 구하기

- timedelta 클래스는 +, - 연산을 이용하여 시간의 연산이 가능
- timedelta를 이용하여 100일 후와 100일 전의 날짜를 구해보기

코드 8-2: 100일 후, 100일 전 날짜 구하기

day_delta.py

```
import datetime as dt

print('오늘 =', dt.datetime.now())          # 현재 시간을 구한다.
hundred = dt.timedelta(days = 100)          # 100일 경과시간
plus100day = dt.datetime.now() + hundred # 현재 시간에서 100일 경과시간을 더함
print('100일 후 =', plus100day)

minus100day = dt.datetime.now() - hundred # 현재 시간에서 100일 경과시간을 뺌
print('100일 전 =', minus100day)
```

실행결과

```
오늘 = 2023-06-17 10:32:49.474428
100일 후 = 2023-09-25 10:32:49.600513
100일 전 = 2023-03-09 10:32:49.667744
```

8.2.2 100일 뒤 날짜 구하기

- timedelta에 들어가는 인자

[표 8-1] timedelta 클래스의 키워드 인자와 코드

나타내는 날짜	코드
1주	datetime.timedelta(weeks = 1)
1일	datetime.timedelta(days = 1)
1시간	datetime.timedelta(hours = 1)
1분	datetime.timedelta(minutes = 1)
1초	datetime.timedelta(seconds = 1)
1밀리초	datetime.timedelta(milliseconds = 1)
1마이크로초	datetime.timedelta(microseconds = 1)

8.2.2 100일 뒤 날짜 구하기

실습

LAB 8-3: 날짜 계산하기

1. timedelta 클래스를 사용하여 오늘 날짜로부터 1,000일 후의 날짜를 구하시오.

2. timedelta 클래스를 이용하여 커플들을 위한 프로그램을 작성하자. 다음과 같이 처음으로 사귄 연도와 월, 일을 띄어쓰기로 입력하면 100일 기념일을 출력하는 기능이 있다. 프로그램의 이름은 couple_day.py라고 정하도록 하자.

```
처음으로 사귄 연도와 월, 일을 입력하시오 : 2019 3 30
100일 기념일은 : 2019년 7월 7일입니다.
```

8.3 time 모듈

- 시간에 관련된 정보를 제공하는 모듈
- 유닉스 시스템의 시작 시간인 1970년 1월 1일 0시 0분 0초 협정 세계시(UTC)를 **에폭** epoch이라고 함
- 유닉스 운영체제에서 표준으로 사용되는 시간 체계는 **에폭 시간** 혹은 **유닉스 시간**이라고도 한다.

대화창 실습: time 모듈을 이용한 에폭 이후의 시간 출력

```
>>> import time
>>> seconds = time.time()
>>> print('에폭 이후의 시간 = ', seconds)
    에폭 이후의 시간 = 1686965711.0023367
```

8.3 time 모듈

[표 8-2] 시간과 이 시간에 해당되는 초

시간	초
1분	60초
1시간	3,600초
1일	86,400초
1주	604,800초
1개월(30.44일)	2,629,743초
1년(356.24일)	31,556,926초

8.3 time 모듈

- strftime()
  - localtime() 함수가 반환한 struct_time이라는 형식의 튜플 값을 지정된 포맷의 문자열로 변환

코드 8-3: time.time()을 통한 현재 시간 출력

epoch_time_2_local_time.py

```
import time


unix_timestamp = time.time()
local_time = time.localtime(unix_timestamp)
print(time.strftime('%Y-%m-%d %H:%M:%S', local_time))
```

실행결과

```
2024-04-19 12:08:58
```

8.3 time 모듈

코드 8-4: time 모듈의 sleep() 함수 사용

sleep_time.py

```
import time

print("바로 출력되는 구문.")       # 이 문장은 바로 출력된다.
time.sleep(4.5)
print("4.5초 후 출력되는 구문.") # 이 문장은 4.5초 후에 출력된다.
```

실행결과

```
바로 출력되는 구문.
4.5초 후 출력되는 구문.
```

- 쓰레드thread
  - 한 프로그램 안에서 실행되는 작은 실행 단위를 지칭하는 용어
- sleep()
  - 일정한 시간동안 현재 실행 중인 쓰레드를 일시 중지

8.3 time 모듈

- 1에서 10까지의 합을 구하여 출력하는 데까지 걸리는 시간을 정확하게 알아보기

코드 8-5: time 모듈을 사용한 경과 시간의 출력

elapsed_time.py

```
import time
start_time = time.time()          # 시작 시각을 기록
print(1+2+3+4+5+6+7+8+9+10)
end_time = time.time()            # 종료 시각을 기록
gap = end_time - start_time
print('1에서 10까지의 합을 구하고 출력하는 시간:{:7.4f}초'.format(gap))
```

실행결과

```
55
1에서 10까지의 합을 구하고 출력하는 시간 : 0.0008초
```

8.4 수학 관련 모듈 math

- math 모듈
  - 수학과 관련된 함수들이 있는 모듈
  - 원주율 파이 값, 자연 상수 e값 등이 정의되어 있음
  - sin(), cos(), tan(), log(), pow(), ceil(), floor(), trunc(), fabs(), copysign(x,y) 등의 수학 관련 함수 포함

8.4 수학 관련 모듈 math

- dir()이라고 하는 내장 함수를 이용하여 math 모듈 내장함수를 볼 수 있음

대화창 실습: math 모듈의 내장함수 출력

```
>>> import math
>>> dir(math) # math 모듈의 내장함수 목록을 반환함
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos',
'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos',
'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial',
'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose',
'isfinite', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'log1p',
'log2', 'modf', 'nan', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan',
'tanh', 'tau', 'trunc']
```

8.4 수학 관련 모듈 math

대화창 실습: math 모듈의 함수 사용

```
>>> import math as m
>>> m.pow(3, 3)              # 3의 3 제곱
27.0
>>> m.fabs(-99)              # -99의 실수 절대값 # 2.1의 올림값
99.0
>>> m.ceil(2.1)              # -2.1의 올림값
3
>>> m.ceil(-2.1)             # 2.1의 내림값
-2
>>> m.floor(2.1)
2
>>> m.log(2.71828)
0.999999327347282
>>> m.log(100, 10)           # 로그 10을 밑으로 하는 100값
2.0
```

8.4 수학 관련 모듈 math

sin(90) 값이 1이 아닌 이유

- 파이썬 삼각함수는 라디안 각도를 인자 값으로 사용하기 때문
- math.pi/2.0과 같은 라디안 표기로 변환해서 사용해야 함
  - π/2의 근사값이기 때문에 정확히 1은 나오지 않음

대화창 실습: math 모듈의 sin() 함수 사용(일반 각 사용)

```
>>> import math as m
>>> m.sin(0.0)
    0.0
>>> m.sin(90.0)
    0.8939966636005579
```

대화창 실습: math 모듈의 sin() 함수 사용(라디안 사용)

```
>>> import math as m
>>> m.sin(3.14159/2.0) # sin() 함수의 인자로 PI/2.0 근사값을 넣어보자.
    0.9999999999991198
```

8.4 수학 관련 모듈 math

sin(90) 값이 1이 아닌 이유

- math 모듈은 자연 상수 e 값을 제공함
- sin(π/2)는 m.sin(m.pi/2.0)과 같은 방법으로 구함
- $\sin^{-1}(1.0)$ 값은 m.asin(1.0)과 같은 방법으로 구함

대화창 실습: math 모듈의 r 값과 라디안 변환 함수

```
>>> m.pi                # 원주율
3.141592653589793
>>> m.sin(m.pi/2.0)    # sin() 함수의 인자로 PI/2.0를 넣어보자.
1.0
>>> m.e
2.718281828459045
>>> m.radians(90)
1.5707963267948966
```

대화창 실습: math 모듈의 π값과 함수

```
>>> m.sin(m.pi/2.0)
1.0
>>> m.asin(1.0)
1.5707963267948966
>>> m.degrees(m.asin(1.0))
90.0
>>> m.tan(2*m.pi)
-2.4492935982947064e-16
>>> m.tan(0.0)
0.0
```

8.4 수학 관련 모듈 math

LAB 8-4: math 모듈을 이용한 계산

1. math 모듈과 for - in range()를 사용하여 4의 2승부터 10승까지를 화면에 출력하여라.

```
4**2=  16.0
4**3=  64.0
... 중간 생략 ...
4** 9 =  262144.0
4**10 = 1048576.0
```

2. 일반각도 0도에서 180도를 10도 단위로 출력하라. 이때 이 일반각도에 대응하는 라디안 각도 값을 함께 출력하라.

```
0 degree = 0.000 radian
10 degree = 0.175 radian
20 degree = 0.349 radian
... 중간 생략 ...
170 degree = 2.967 radian
180 degree = 3.142 radian
```

3. math 모듈과 for 문을 사용하여 일반각도 sin 0도에서부터 180도까지의 값을 10도 간격으로 출력하여라.

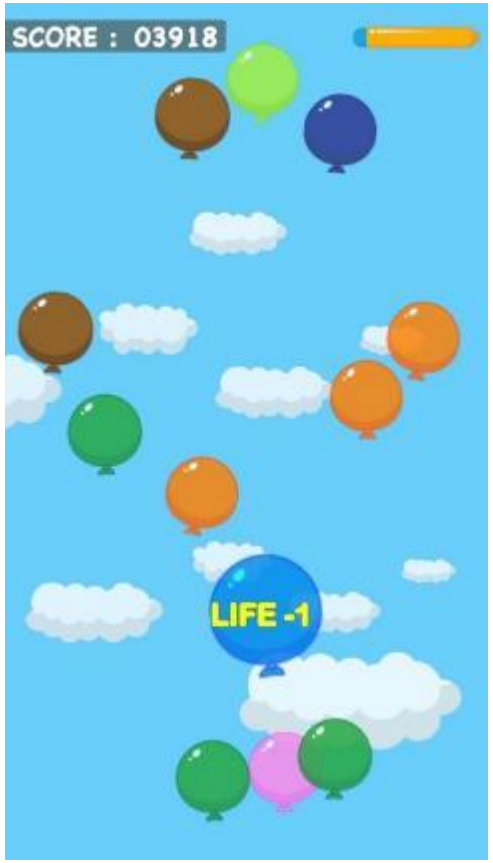
```
sin( 0)=0.00
sin( 10) = 0.17
(... 중간 생략 ...)
sin( 170) = 0.17
sin( 180) = 0.00
```

8.5 난수 모듈 random

- 임의의 수를 생성하거나, 리스트 내의 원소를 무작위로 섞거나 선택하는 함수를 포함

[표 8-3] random 모듈에 포함된 함수들

함수	하는 일
random()	0에서 1 사이의 실수를 생성한다(1은 포함하지 않음).
randrange()	지정된 범위 내의 정수를 반환한다.
randint(a, b)	a <= N <= b 사이의 랜덤 정수 N을 반환한다.
shuffle(seq)	주어진 seq 리스트의 요소를 랜덤하게 섞는다.
choice(seq)	seq 시퀀스 내의 임의의 요소를 선택한다.
sample()	지정된 개수의 요소를 임의로 선택한다.



랜덤하게 나타나는 풍선 게임 예시

8.5 난수 모듈 random

- random() 함수는 0 이상 1 미만의 임의의 실수를 반환함
- randrange(n, m)은 n 이상 m 미만의 임의의 정수를 반환

대화창 실습: random 모듈을 활용한 값 생성

```
>>> import random as rd
>>> rd.random() # 0 이상 1 미만의 실수를 반환함
0.19452357419514088
>>> rd.random() # 매번 다른 실수를 반환함
0.6947454047320903
>>> rd.randrange(1, 7) # 1 이상 7 미만의 정수를 반환함
6
>>> rd.randrange(0, 10, 2) # 1 이상 10 미만 정수 중 2의 배수를 반환함
2
>>> rd.randint(1, 10) # 1 이상 10 이하의 임의의 정수를 반환함
3
```

8.5 난수 모듈 random

대화창 실습: random 모듈을 활용한 섞기와 고르기

```
>>> numlist = [10, 20, 30, 40, 50]
>>> rd.shuffle(numlist)     # 리스트의 원소를 랜덤하게 섞는다.
>>> numlist
[20, 30, 10, 40, 50]
>>> rd.choice(numlist)      # 리스트의 원소들 중에서 랜덤하게 하나를 고른다.
20
>>> rd.sample(numlist, 3) # 리스트의 원소들 중에서 랜덤하게 세 개를 고른다.
[40, 20, 30]
```

- shuffle()
  - 시퀀스의 원소를 랜덤하게 섞어 반환
- choice()
  - 인자로 들어온 시퀀스로부터 임의의 원소를 추출
- sample()
  - 원소를 랜덤하게 반환
  - 반환할 원소의 개수를 인자로 넣어줄 수 있음

8.5 난수 모듈 random

- shuffle()은 매번 다른 순서로 시퀀스를 섞어서 반환함

대화창 실습: random 모듈을 활용한 섞기

```
>>> a = list(range(1, 11))    # 1에서 10까지의 연속적인 정수를 생성
>>> rd.shuffle(a)             # 리스트의 a의 원소를 랜덤하게 섞는다.
>>> print(a)
    [2, 1, 4, 3, 10, 6, 9, 8, 5, 7]
```

8.5 난수 모듈 random

LAB 8-5: 랜덤 번호 생성기

1. random 모듈의 randrange() 함수를 이용하여 0에서 100 이하의 정수 중에서 5의 배수 값을 임의로 3개 선택하여 리스트 형식으로 출력해 보시오.

```
0에서 100 이하의 정수 중에서 5의 배수 [15, 45, 60]
```

2. 1에서 10 사이의 정수 중 임의의 정수 3개를 sample() 함수를 이용하여 출력하시오.

```
1에서 10 사이의 임의의 정수 : [3, 2, 5]
```

8.5 난수 모듈 random

8.5.1 의사 랜덤과 시드번호

- 많은 프로그래밍 언어에서 사용하는 랜덤 함수는 **의사 랜덤**pseudo random 생성기로 만들어 낸다.
- X는 의사 랜덤 값의 열(sequence)이 되며, X_n 값은 X_{n+1} 값을 만드는데 사용될 수 있다.

X_{n+1}은 X_n 다음의 랜덤 값

$$X_{n+1} = (a\ X_n\ +\ b)\ \bmod\ m$$

8.5 난수 모듈 random

8.5.1 의사 랜덤과 시드번호

코드 8-6: 의사 랜덤 함수를 이용한 난수 만들기
pseudo_rand_ex.py

```
def pseudo_rand(x):
    a = 1103515245
    b = 12345
    m = 2 ** 31
    new_x = (a * x + b) % m
    return new_x

seed = 100              # 초기값을 임의로 설정하자.
x = pseudo_rand(seed)   # seed를 입력으로 하여 새로운 x를 만들자.
print(x)
x = pseudo_rand(x)      # x를 입력으로 하여 새로운 x를 만들자.
print(x)
```

실행결과

```
829870797
1933386042
```

- pseudo_rand()의 입력으로 100을 넣어주면 829870797와 같은 난수를 얻을 수 있음
- 입력 값을 101로 변경하면 1933386042와 같은 전혀 다른 난수 값이 나옴

8.5 난수 모듈 random

8.5.1 의사 랜덤과 시드번호

코드 8-7: 의사 랜덤 함수를 이용한 연속적인 난수 만들기

pseudo_rand_for.py

```
def pseudo_rand(x):
    new_x = (1103515245 * x + 12345) % ( 2 ** 31 )
    return new_x

x = 12234       # 씨앗(seed)이 되는 초기 번호
for _ in range(5):
    x = pseudo_rand(x)
    print(x)
```

실행결과

```
1323308347
1115894872
534792625
1490648726
389073431
```

8.5 난수 모듈 random

8.5.1 의사 랜덤과 시드번호

- 변수 a, b, m을 사용하지 않고 리터럴 상수를 이용하여 함수내부를 간단하게 수정
- 새롭게 얻어진 x 값을 다음 난수를 생성하는데 사용함
- 위 코드의 x와 같이 초기에 난수를 생성하기 위한 값을 **씨앗 값**seed number 혹은 **씨드 값**이라고 한다
- 씨드 값으로는 현재의 시간을 사용하면 규칙성이 덜하다

8.5 난수 모듈 random

8.5.2 로또 번호 만들기

- 로또는 1에서 45 사이의 임의의 정수를 6개 맞히는 규칙이 있음
- 파이썬의 random 모듈을 활용하여 로또 번호를 자동으로 생성하여 출력하는 프로그램을 만들어보기

1) 1에서 45까지의 번호를 리스트에 넣도록 하자.
2) 이 리스트를 임의의 순서대로 섞도록 하자.
3) 임의의 순서대로 섞은 리스트에서 최초 6개의 항목만 가져오자.
4) 선택된 6개 항목들을 오름차순으로 정렬하자.
5) 이 리스트를 출력하자.

8.5 난수 모듈 random

8.5.2 로또 번호 만들기

코드 8-8: random 모듈을 이용한 로또 번호 만들기 1

lotto_gen1.py

```
import random as rd

lotto_list = list(range(1, 46)) # 1부터 45까지 생성
rd.shuffle(lotto_list)          # 임의의 순서로 섞기
lotto_list = lotto_list[:6]     # 앞 부분 6개만 선택
lotto_list.sort()               # 선택된 번호를 정렬
print('이번 주의 추천 로또 번호 :', lotto_list)
```

실행결과

```
이번 주의 추천 로또 번호 : [2, 5, 7, 9, 25, 39]
```

파이썬의 만들어 주는 행운의 수를
테스트해 보세요
로또 사러 고고씽~~

8.5.2 로또 번호 만들기

- random 모듈의 추출(샘플링) 기능을 수행하는 sample() 함수를 사용하여 위의 코드를 간략하게 다시 만들 수 있음

코드 8-9: random 모듈을 이용한 로또 번호 만들기 2

lotto_gen2.py

```
import random as rd

lotto_list = list(range(1, 46))          # 1부터 45까지 생성
lotto_list = rd.sample(lotto_list, 6)    # 임의의 값 6개를 추출(샘플링)
lotto_list.sort()                        # 선택된 번호를 정렬
print('이번 주의 추천 로또 번호 :', lotto_list)
```

실행결과

```
이번 주의 추천 로또 번호 : [14, 17, 24, 28, 34, 43]
```

8.5 난수 모듈 random

8.5.2 로또 번호 만들기

LAB 8-6: 로또 번호 얻기

1. 파이썬의 random 모듈을 이용하여 다음과 같이 매주 5개의 랜덤하게 생성된 로또 번호를 얻고자 한다. **[코드 8-8]**을 수정하여 이와 같이 5개의 로또 번호를 생성하는 프로그램을 작성해 보자.

```
이번 주의 추천 로또 번호
[15, 22, 28, 30, 31, 34]
[11, 12, 20, 22, 27, 43]
[2, 28, 34, 35, 42, 45]
[1, 8, 29, 37, 41, 44]
[3, 8, 26, 30, 39, 42]
```

2. 이제는 매주 로또 시행사에서 발표하는 당첨 번호를 출력해 보자. 임의의 로또 당첨 번호와 함께 다음과 같이 보너스 번호를 큰 괄호로 묶어서 출력해 보자.

```
이번 주의 당첨 로또 번호 + 보너스
[11, 12, 20, 32, 37, 42] + [9]
```

8.6 sys 모듈

- 파이썬 인터프리터가 제공하는 변수들과 함수들을 직접 제어할 수 있게 해 주는 모듈
- prefix 속성은 파이썬이 설치된 경로를 알려줌
- version 속성은 파이썬 인터프리터의 버전을 알려줌
- copyright 속성은 파이썬의 저작권에 관련된 내용을 포함
- path는 모듈을 찾을 때 참조하는 경로를 나타낸다. 파이썬이 설치되면 내장 모듈도
- 함께 설치되는데, 이 모듈이 시스템의 경로상 어디에 있는지 이 path 정보를 통해 알 수 있다.

8.6 sys 모듈

대화창 실습: sys 모듈을 이용한 파이썬 버전과 경로, 저작권

```
>>> import sys
>>> sys.prefix
    'C:\\Users\\USER\\AppData\\Local\\Programs\\Python\\Python311'
>>> sys.version
     '3.11.3 (v3.11.3:ef4ec6ed12, Mar 25 2023, 22:22:05) [MSC v.1916 64 bit
(AMD64)]'
>>> sys.copyright
     'Copyright (c) 2001-2019 Python Software Foundation.\nAll Rights
Reserved.\n\n ... 중간 생략 ... All Rights Reserved.'
```

8.6 sys 모듈

대화창 실습: sys 모듈을 이용한 경로 설정 확인

```
>>> sys.path
['',  'C:\\Users\\USER\\AppData\\Local\\Programs\\Python\\Python311\\Lib\\
idlelib','C:\\Users\\USER\\AppData\\Local\\Programs\\Python\\Python311\\
python311.zip','C:\\Users\\USER\\AppData\\Local\\Programs\\Python\\
Python311\\DLLs','C:\\Users\\USER\\AppData\\Local\\Programs\\Python\\
Python311\\lib\\site-packages']
```

모듈을 가져다 쓰는 다양한 방법

1. import *module*

이 방법은 해당 모듈을 전체적으로 가져오는 가장 기본적인 방법이다. 모듈 이름을 사용하여 모듈 내의 클래스, 함수 및 변수를 접근할 수 있고 사용할 때에는 `module.name` 형식으로 사용한다.

```
import math
result = math.sqrt(25)
```

2. import *module* as *m*

이 방법은 모듈을 가져와서 별칭을 지정하는 방식으로, 모듈 이름이 길거나 충돌을 피하고자 할 때 유용하다. 모듈의 클래스, 함수 및 변수는 `m.name` 형식으로 접근한다.

```
import math as m
result = m.sqrt(25)
```

모듈을 가져다 쓰는 다양한 방법

3. from *module* import *function_name, class_name, ···*

이 방법은 모듈 내에서 특정 클래스나 함수 등을 선택적으로 가져온다. 이렇게 하면 해당 이름만을 사용하여 직접적으로 클래스나 함수를 사용할 수 있다.

```
from math import sqrt, exp
result = sqrt(25)
res_exp = exp(result)
```

4. from *module* import *

이 방법은 모듈 내의 모든 클래스, 함수 및 변수를 현재 네임스페이스로 가져온다. 이런 방식은 간단하지만 가독성 문제와 충돌 가능성 때문에 권장되지 않는다.

```
from math import *
result = sqrt(25)
tanVal = tan(result)
maxVal = max(result, tanVal)
```

Leistung ist nicht alles / Keinen Studierenden zurücklassen

DAEJEON UNIVERSITY
SINCE 1980
대전대학교