

으뜸 파이썬



5강 모듈의 활용

- 본 강의노트는 으뜸 파이썬(박동규, 강영민 著) 1판의 강의자료를 활용하여 교양수업에 맞게 편집되었습니다.

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

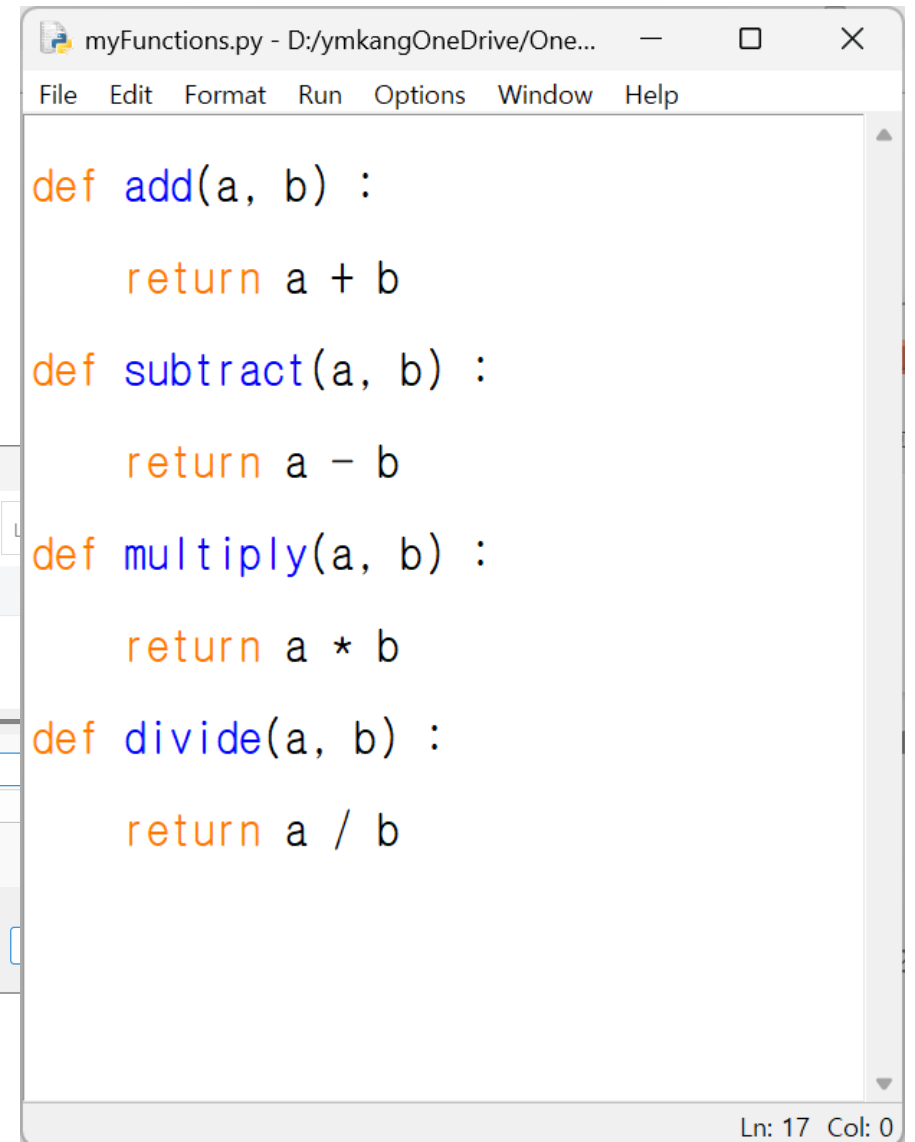
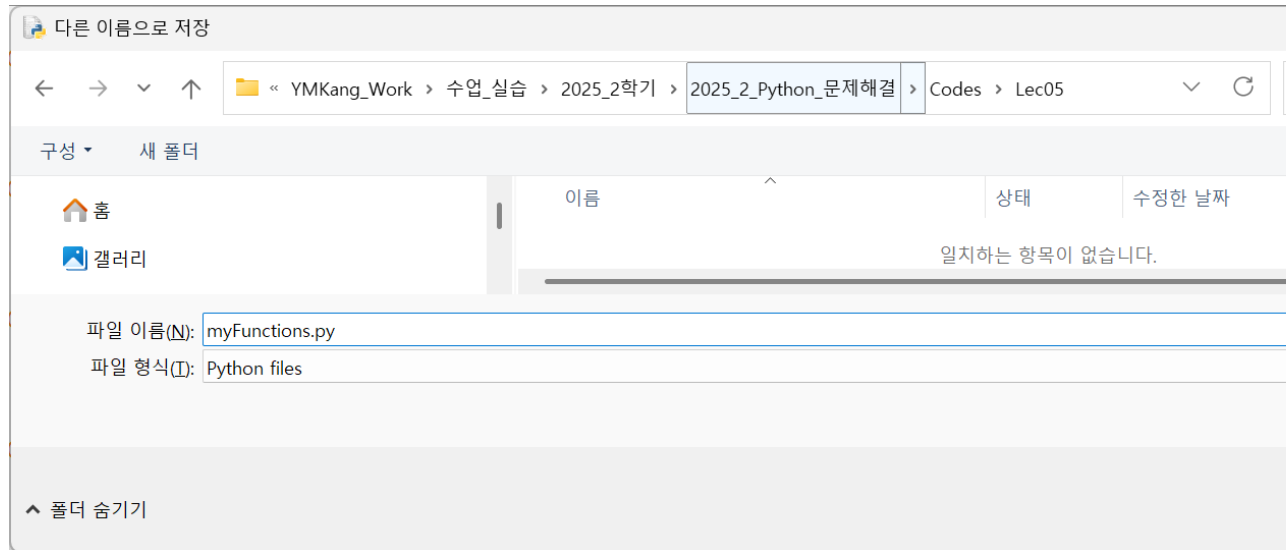
```
import 모듈이름1 [, 모듈이름2, ...]
```

파이썬 설치와 함께 제공되는 모듈을 파이썬 표준 라이브러리 `python standard library`라고 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

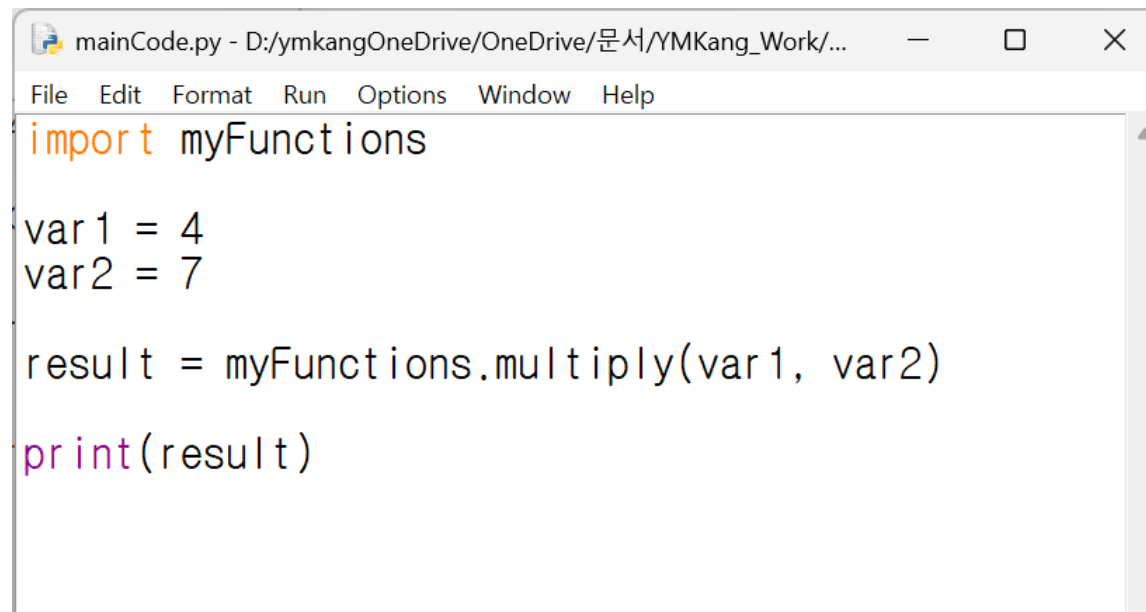
내가 만든 모듈 사용해 보기

- 파일을 하나 만들어서 함수를 담아 보자
 - myFunctions.py



이 모듈을 사용해 볼까

- mainCode.py

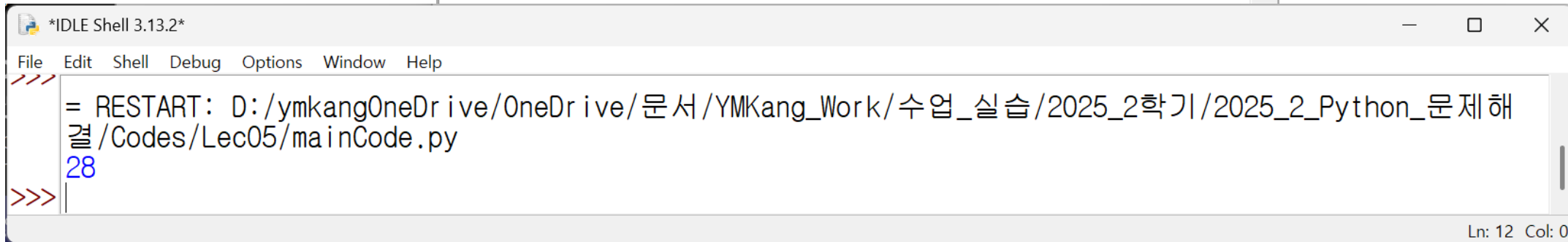


```
mainCode.py - D:/ymkangOneDrive/OneDrive/문서/YMKang_Work/...
File Edit Format Run Options Window Help
import myFunctions

var1 = 4
var2 = 7

result = myFunctions.multiply(var1, var2)

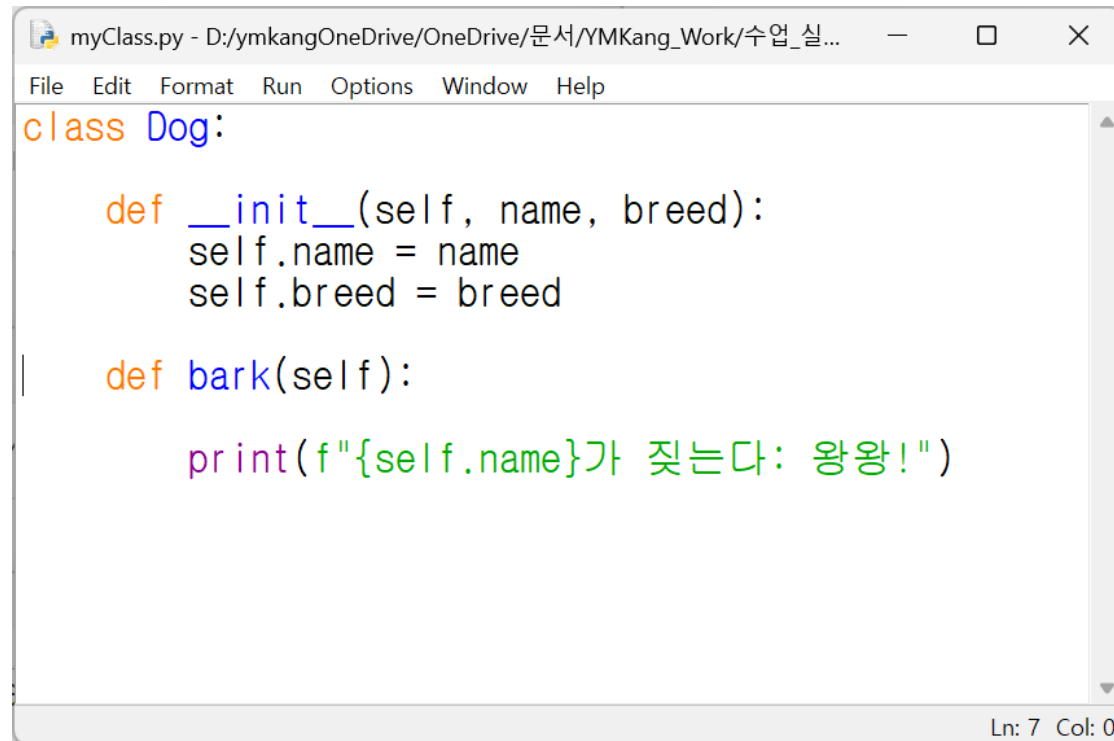
print(result)
```



```
*IDLE Shell 3.13.2*
File Edit Shell Debug Options Window Help
///
= RESTART: D:/ymkangOneDrive/OneDrive/문서/YMKang_Work/수업_실습/2025_2학기/2025_2_Python_문제해결/Codes/Lec05/mainCode.py
28
>>>
```

클래스도 답아 보자

- 파일을 하나 만들어서 함수를 답아 보자
 - myClass.py



The screenshot shows a window titled 'myClass.py - D:/ymkangOneDrive/OneDrive/문서/YMKang_Work/수업_실...'. The window contains a Python class definition for 'Dog'. The code is as follows:

```
class Dog:

    def __init__(self, name, breed):
        self.name = name
        self.breed = breed

    def bark(self):
        print(f"{self.name}가 짖는다: 왕왕!")
```

The status bar at the bottom right indicates 'Ln: 7 Col: 0'.

이 모듈을 사용해 볼까

- mainCode.py

```
mainCode.py - D:/ymkangOneDrive/OneDrive/문서/YMKang_Work/수...
File Edit Format Run Options Window Help

import myFunctions
import myClass

var1 = 4
var2 = 7

result = myFunctions.multiply(var1, var2)

myDog = myClass.Dog("해피", 4)
yourDog = myClass.Dog("메리", 3)

print(result)

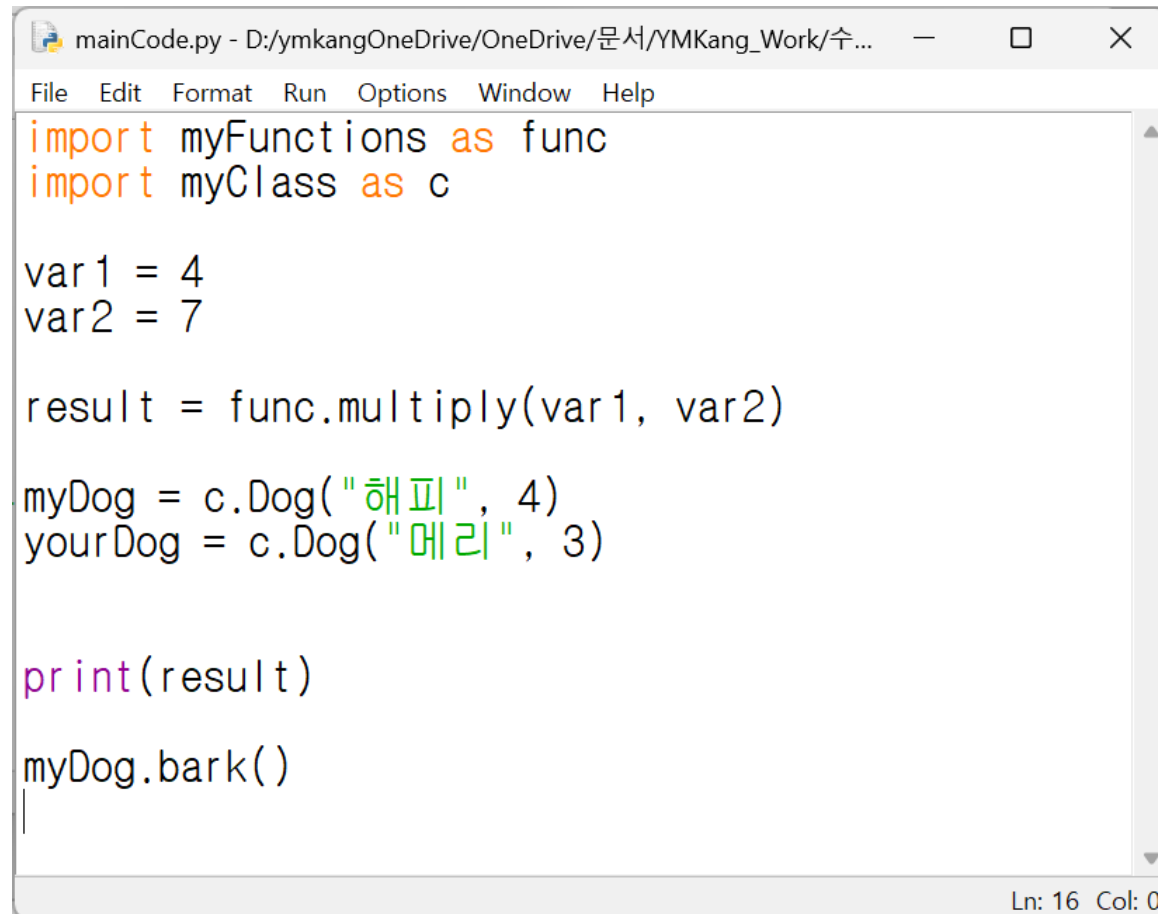
myDog.bark()
```

```
IDLE Shell 3.13.2
File Edit Shell Debug Options Window Help

>>>
= RESTART: D:/ymkangOneDrive/OneDrive/문서/YMKang_Work/수업_실습/2025_2학기/2025_2_Py
thon_문제해결/Codes/Lec05/mainCode.py
28
해피가 짖는다: 왕왕!
>>>
```

모듈 이름이 너무 길어요 – 별명으로 사용

- mainCode.py



```
mainCode.py - D:/ymkangOneDrive/OneDrive/문서/YMKang_Work/수...  
File Edit Format Run Options Window Help  
import myFunctions as func  
import myClass as c  
  
var1 = 4  
var2 = 7  
  
result = func.multiply(var1, var2)  
  
myDog = c.Dog("해피", 4)  
yourDog = c.Dog("메리", 3)  
  
print(result)  
  
myDog.bark()  
|
```

Ln: 16 Col: 0

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. 이 때, 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

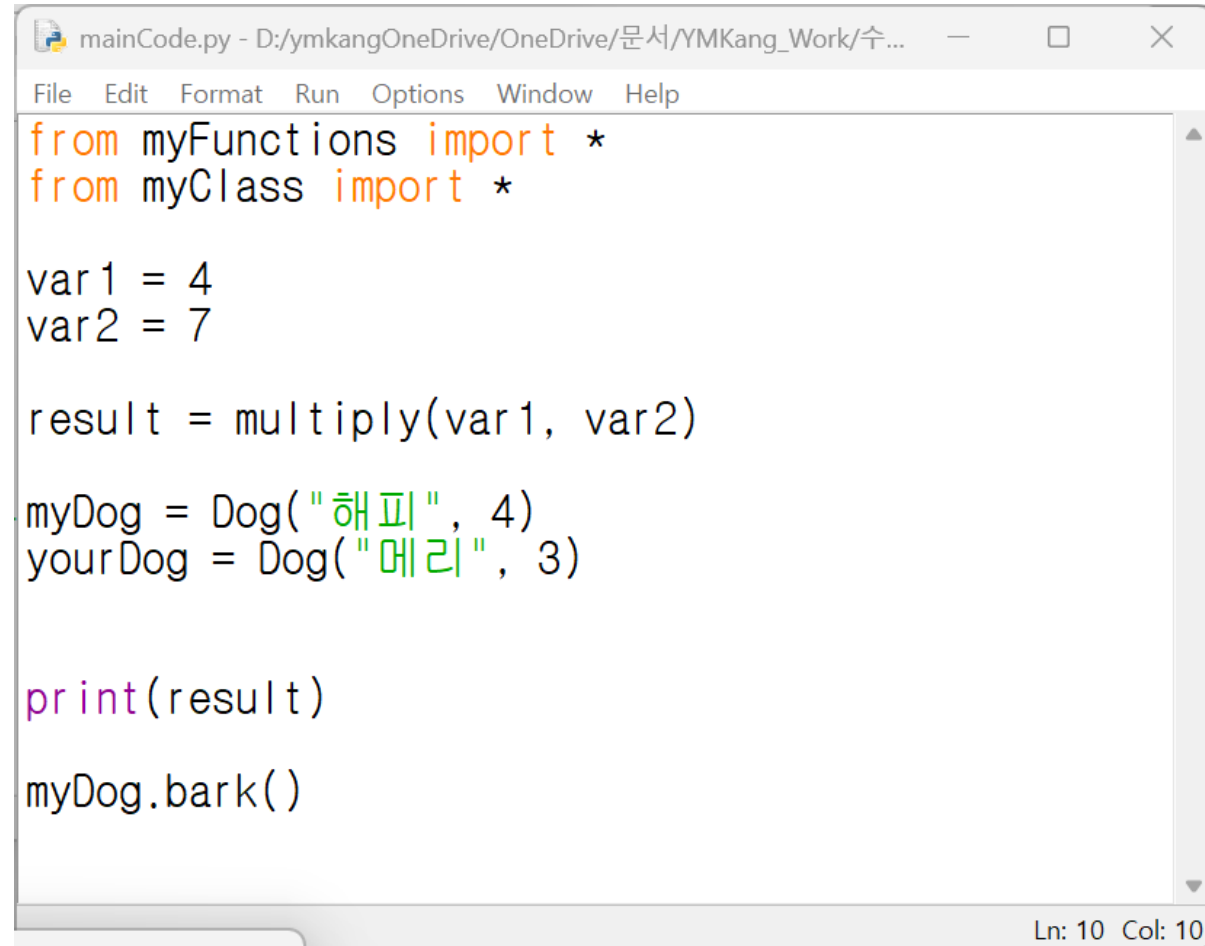
ex2) import random as rd

ex3) import math as m

ex3) import turtle as t

모듈을 지정하지 않고 내 것처럼 쓰래요

- mainCode.py



```
mainCode.py - D:/ymkangOneDrive/OneDrive/문서/YMKang_Work/수...
File Edit Format Run Options Window Help
from myFunctions import *
from myClass import *

var1 = 4
var2 = 7

result = multiply(var1, var2)

myDog = Dog("해피", 4)
yourDog = Dog("메리", 3)

print(result)

myDog.bark()

Ln: 10 Col: 10
```

유용한 모듈 - 날짜와 시간 모듈 `datetime`

- `datetime` 모듈
 - 날짜와 시간에 관한 기능을 제공하고 조작할 수 있는 모듈
- 아래의 `datetime.datetime.now()`에서 앞의 `datetime`은 모듈의 이름, 뒤의 `datetime`은 클래스의 이름

대화창 실습 : `datetime` 모듈의 임포트와 사용

```
>>> import datetime      # 이하 이 문장은 생략함
```

```
>>> datetime.datetime.now()
```

```
datetime.datetime(2019, 1, 2, 6, 57, 27, 904565)
```

- 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)
```

```
2019-01-02
```

```
>>> today
```

```
datetime.date(2019, 1, 2)
```

```
>>> today.year
```

```
2019
```

```
>>> today.month
```

```
1
```

```
>>> today.day
```

```
2
```

- **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 사용가능한 속성을 반환**
- **MAXYEAR는 datetime 오브젝트가 표현 가능한 최대 년도로 9999 값을 가짐**
- **MINYEAR는 1 값을 가짐**
- **date, datetime, datetime_CAPI, time, timedelta, timezone, tzinfo 와 같은 클래스들은 날짜, 시간, 시간대, 시간대 정보를 편리하게 이용할 수 있는 기능이 있음**

- 별칭 사용법 다시 확인
 - 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
```

```
>>> start_time = dt.datetime.now()
```

```
>>> start_time.replace(month = 12, day = 25)
```

```
datetime.datetime(2019, 12, 25, 7, 1, 25, 880317)
```

이 결과는 컴퓨터의 실행시간에 따라
매번 달라질 수 있음

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(2019, 12, 25, 7, 1, 25, 880317)
```

datetime. 혹은 dt. 을 생략할 수 있다.



LAB 7-1 : 오늘의 날짜와 현재시간 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 : 2019년 2월 19일

현재시간 : 오후 6시 50분 11초

D-Day Counter

코드 7-1 : datetime 모듈을 사용하여 크리스마스까지 남은 시간 구하기

xmas_left_day.py

```
import datetime as dt
today = dt.date.today()
print('오늘은 {}년 {}월 {}일입니다'.format(today.year, today.month, today.day))
xMas = dt.datetime(2019, 12, 25)
time_gap = xMas - dt.datetime.now()
print('다음 크리스마스 까지는 {}일 {}시간 남았습니다.'.format( \
    time_gap.days,time_gap.seconds // 3600))
```

실행결과

오늘은 2019년 1월 2일입니다

다음 크리스마스 까지는 356일 16시간 남았습니다.



LAB 7-2 : 남은 날짜 계산하기

1. 앞서 배운 남은 날짜 계산 프로그램을 수정하여 오늘 날짜와 함께 2025년 크리스마스까지의 남은 날짜와 시간을 구해서 출력해 보자.

오늘은 2019년 8월 29일입니다
2025년 크리스마스 까지는 2309일 16시간 남았습니다.

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

오늘은 2019년 8월 29일입니다
2036년 새해 까지는 5988일 15시간 남았습니다.

3. 다가오는 자신의 생일까지 남은 날짜와 시간을 출력해 보자.

오늘은 2019년 8월 29일입니다
2020년 생일까지는 309일 15시간 남았습니다.

100일 뒤 날짜 구하기

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

코드 7-2 : 100일 후, 100일 전 날짜 구하기

day_delta.py

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

실행결과

오늘 = 2020-04-23 18:01:31.657261

100일 후 = 2020-08-01 18:01:31.657261

- timedelta에 들어가는 인자

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



LAB 7-3 : 날짜 계산하기

1. `timedelta` 클래스를 사용하여 오늘 날짜로부터 1,000일 후의 날짜를 구하시오.
2. `timedelta` 클래스를 이용하여 커플들을 위한 프로그램을 작성하자. 다음과 같이 처음으로 사권 연도와 월, 일을 띄어쓰기로 입력하면 100일 기념일을 출력하는 기능이 있다. 프로그램의 이름은 `couple_day.py`라고 정하도록 하자.

처음으로 사권 연도와 월, 일을 입력하시오 : 2019 3 30

100일 기념일은 : 2019년 7월 7일입니다.

time 모듈

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

대화창 실습 : time 모듈을 이용한 에폭 이후의 시간 출력

```
>>> import time
>>> seconds = time.time()
>>> print('에폭 이후의 시간 = ', seconds)
에폭 이후의 시간 = 1555674634.0023367
```

코드 7-4 : 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 실행단위

• sleep()

- 일정한 시간동안 현재 실행 중인 쓰레드를 일시 중지

1에서 10까지의 합을 구하여 출력하는데까지 걸리는 시간을 알아보기

코드 7-5 : time 모듈을 사용한 경과시간의 출력

elapsed_time.py

```
import time
```

```
start_time = time.time()  # 시작시간을 기록
```

```
print(1+2+3+4+5+6+7+8+9+10)
```

```
end_time = time.time()  # 종료시간을 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.

1에서 10까지의 합을 구하는 데에 걸리는 시간을 출력해 보라.

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

- `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']
```

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
```

```
99.0
```

```
>>> m.ceil(2.1) # 2.1의 올림값
```

```
3
```

```
>>> m.ceil(-2.1) # -2.1의 올림값
```

```
-2
```

```
>>> m.floor(2.1) # 2.1의 내림값
```

```
2
```

```
>>> m.log(2.71828)
```

```
0.999999327347282
```

```
>>> m.log(100, 10) # 로그 10을 밑으로 하는 100값
```

```
2.0
```

sin(90) 값이 1이 아닌 이유

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

대화창 실습 : math 모듈의 sin() 함수 사용(라디안 사용)

```
>>> import math as m
>>> # sin() 함수의 인자로 PI/2.0 근사값을 넣어보자
>>> m.sin(3.14159/2.0) # 라디안 사용
0.9999999999999991198
```

대화창 실습 : math 모듈의 sin() 함수 사용(일반 각 사용)

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

- math 모듈은 자연 상수 e 값을 제공함
- $\sin(\pi/2)$ 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) # 일반각 90도에 해당하는 라디안 값
1.5707963267948966
```

대화창 실습 : math 모듈의 π 값과 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) # tan(360)인 0의 근사치를 반환
-2.4492935982947064e-16
>>> m.tan(0.0)
0.0
```



LAB 7-4 : 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
4** 4 =     256.0
4** 5 =    1024.0
4** 6 =    4096.0
4** 7 =   16384.0
4** 8 =   65536.0
4** 9 =  262144.0
4**10 = 1048576.0
```

2. 일반각도 0도에서 180도를 10도 단위로 출력하라. 이때 이 일반각도에 대응되는 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 0도에서 부터 180도까지의 값을 10도 간격으로 출력하여라.

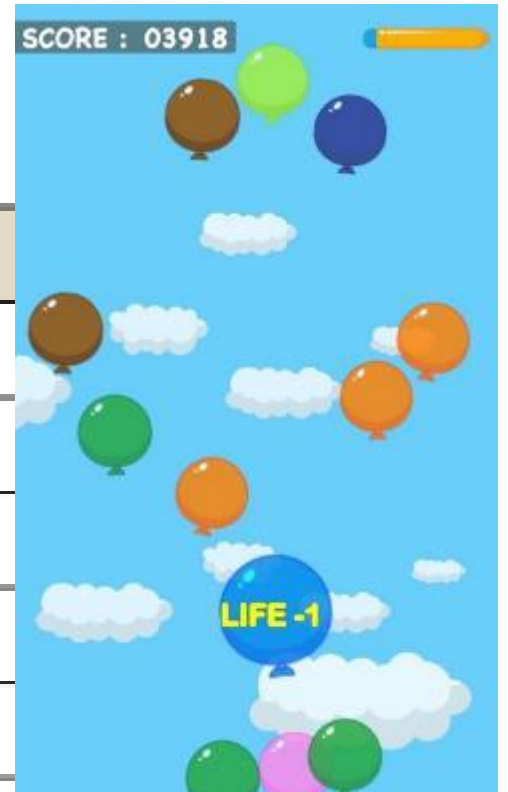
```
sin( 0) = 0.00
sin( 10) = 0.17
(... 중간 생략 ...)
sin( 170) = 0.17
sin( 180) = 0.00
```

랜덤 모듈 random

- 임의의 수를 생성하거나, 리스트 내의 원소를 무작위로 섞거나 선택하는 함수를 포함 – 예측할 수 없는 자연의 현상을 흉내내기 좋음

[표 7-3] random 모듈에 포함된 함수들

함수	하는 일
random()	0에서 1사이의 실수를 생성한다. (1은 포함하지 않음)
randrange()	지정된 범위 내의 정수를 반환한다.
randint(a, b)	$a \leq N \leq b$ 사이의 랜덤 정수 N 을 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 풍선게임

- random() 함수는 0 이상 1 미만의 임의의 실수를 반환함
- randrange(n, m)은 n이상 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(1, 10이 포함) 임의의 정수를 반환함
```

```
3
```

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 세개를 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

- **shuffle()**은 매번 다른 순서로 시퀀스를 섞어서 반환함
 - 카드 게임 같은 곳에 유용

대화창 실습 : random 모듈을 활용한 섞기

```
>>> a = list(range(1, 11)) # 1에서 10까지의 연속적인 정수를 생성
```

```
>>> rd.shuffle(a) # 리스트의 원소를 랜덤하게 섞는다
```

```
>>> print(a)
```

```
[2, 1, 4, 3, 10, 6, 9, 8, 5, 7]
```



LAB 7-5 : 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]

로또 번호 만들기

- 로또는 1에서 45 사이의 임의의 정수를 6개 맞추는 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.



코드 7-8 : 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 로또번호 :', lotto_list)
```

실행결과

이번 주의 추천 로또번호 : [2, 5, 7, 9, 25, 39].

파이썬의 만들어 주는 행운의 수를
테스트 해 보세요
로또 사러 고고씹~~

* 일확천금을 노리지 맙시다

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

코드 7-9 : random 모듈을 이용한 로또 번호 만들기 2

lotto_gen2.py

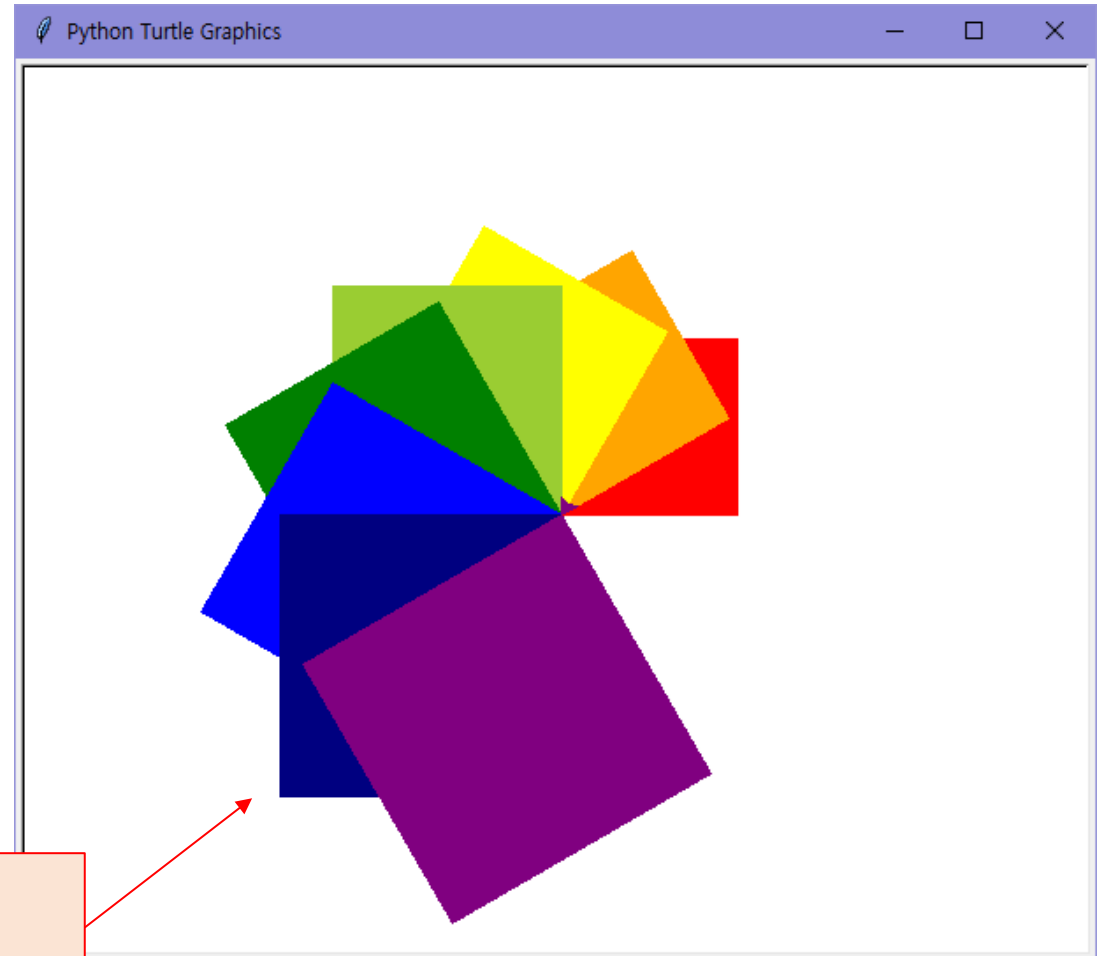
```
import random as rd
lotto_list = list(range(1, 46))      # 1부터 45까지 생성
lotto_list = rd.sample(lotto_list, 6) # 임의의 값 6개를 추출(샘플링)
lotto_list.sort()                   # 선택된 번호를 정렬
print('이번 주의 추천 로또번호 :', lotto_list)
```

실행결과

이번 주의 추천 로또번호 : [14, 17, 24, 28, 34, 43].

그림 그리기 모듈 turtle

- 터틀 그래픽 `turtle graphic`
 - 파이썬에서 그림을 그릴 수 있도록 지원하는 방식
 - 기본적으로 내장되어 있음
 - 아주 많은 메소드를 가지고 있음
 - 풍부한 기능이 있으며
 - 입문자의 흥미를 유발할 수 있음



아름다운 그래픽을 컴퓨터
화면에서 볼 수 있음

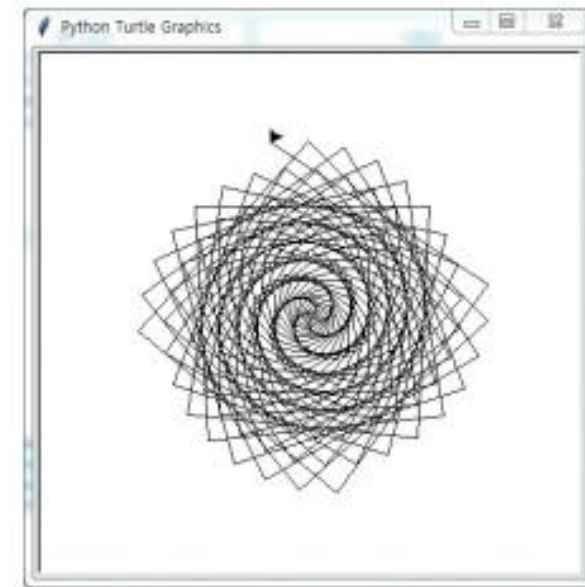
터틀 초기화와 모양 바꾸기

- 별도의 윈도우가 화면에 나타나고, 커서가 그림을 그린다

코드 7-10 : 터틀 그래픽의 setup(), forward(), left(), done() 메소드

turtle_example.py

```
import turtle as t
t.setup(width = 400, height = 400)
for i in range(200):
    t.forward(i)
    t.left(93)
t.done()
```

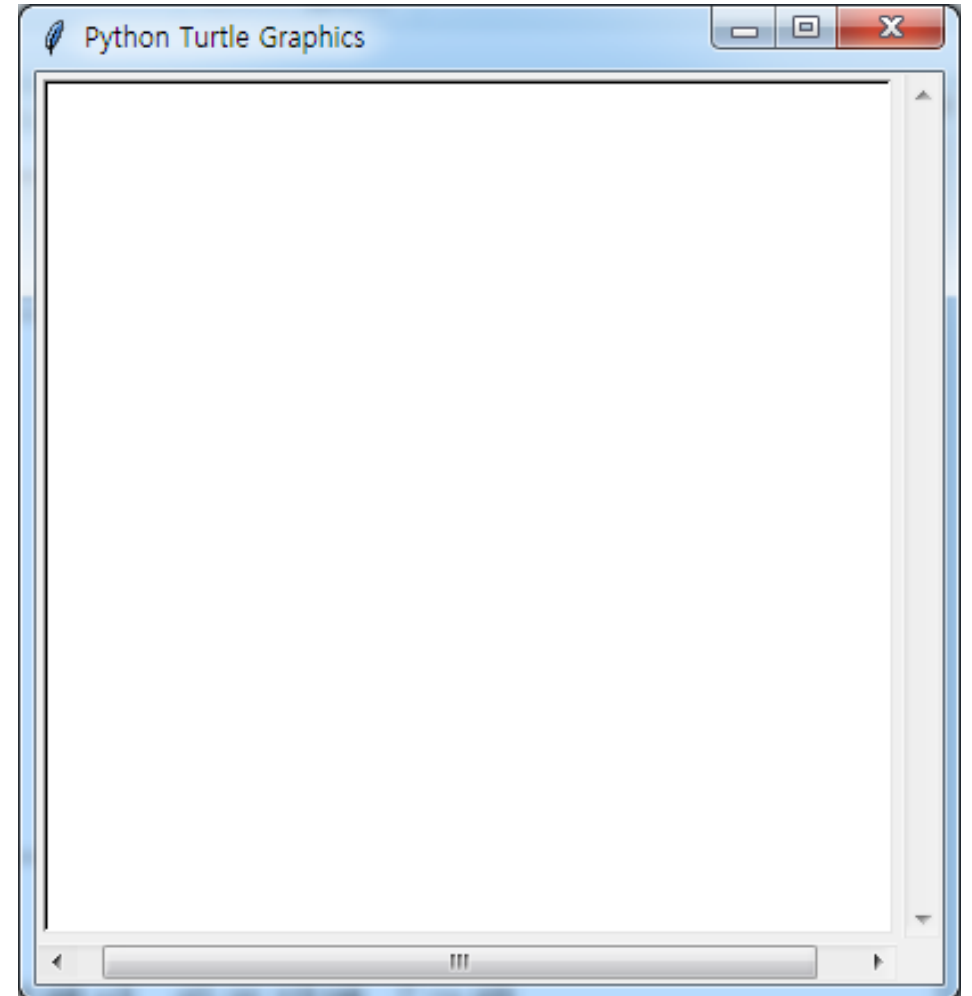


[그림 7-3] 터틀 그래픽 예제 turtle_example.py의 실행화면

- 파이썬 대화창에서 다음과 같은 코드를 입력하면 오른쪽의 창이 뜬다.
- `t.setup()` 메소드는 대화창의 크기와 제목, 여러 속성을 지정

대화창 실습 : 터틀 모듈의 활용

```
>>> import turtle as t  
>>> t.setup(width = 400, height = 400)
```



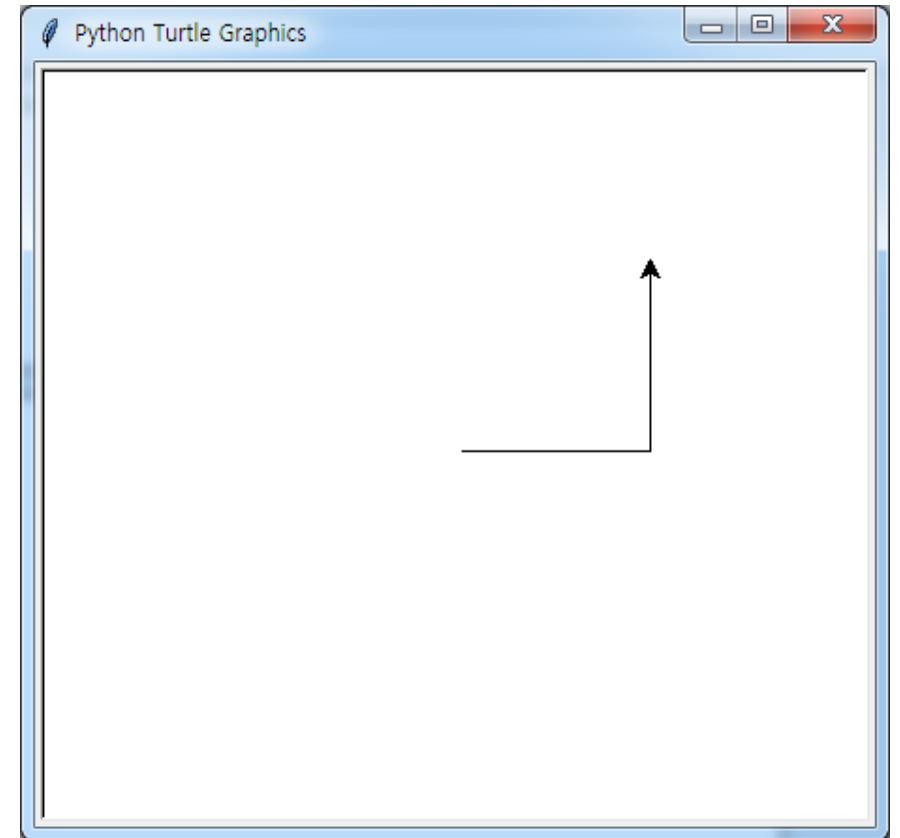
- t.forward(100)입력 시 **커서**가 100 픽셀 왼쪽으로 이동하면서 동시에 검은색 실선을 그린다.
- t.left(90)을 입력시 커서가 왼쪽으로 90도 회전
- T.forward(100)을 입력시 커서가 100 픽셀 이동하며 실선을 그린 후 정지

대화창 실습 : 터틀 모듈의 forward()와 left() 명령입력

```
>>> t.forward(100)
```







```
>>> t.left(90)
```

```
>>> t.forward(100)
```



- 터틀 그래픽의 디폴트 커서 모양은 화살표 모양
- shape() 메소드를 이용하여 커서의 모양을 바꿀 수 있음

[표 7-4] 터틀 그래픽의 커서 이름과 모양

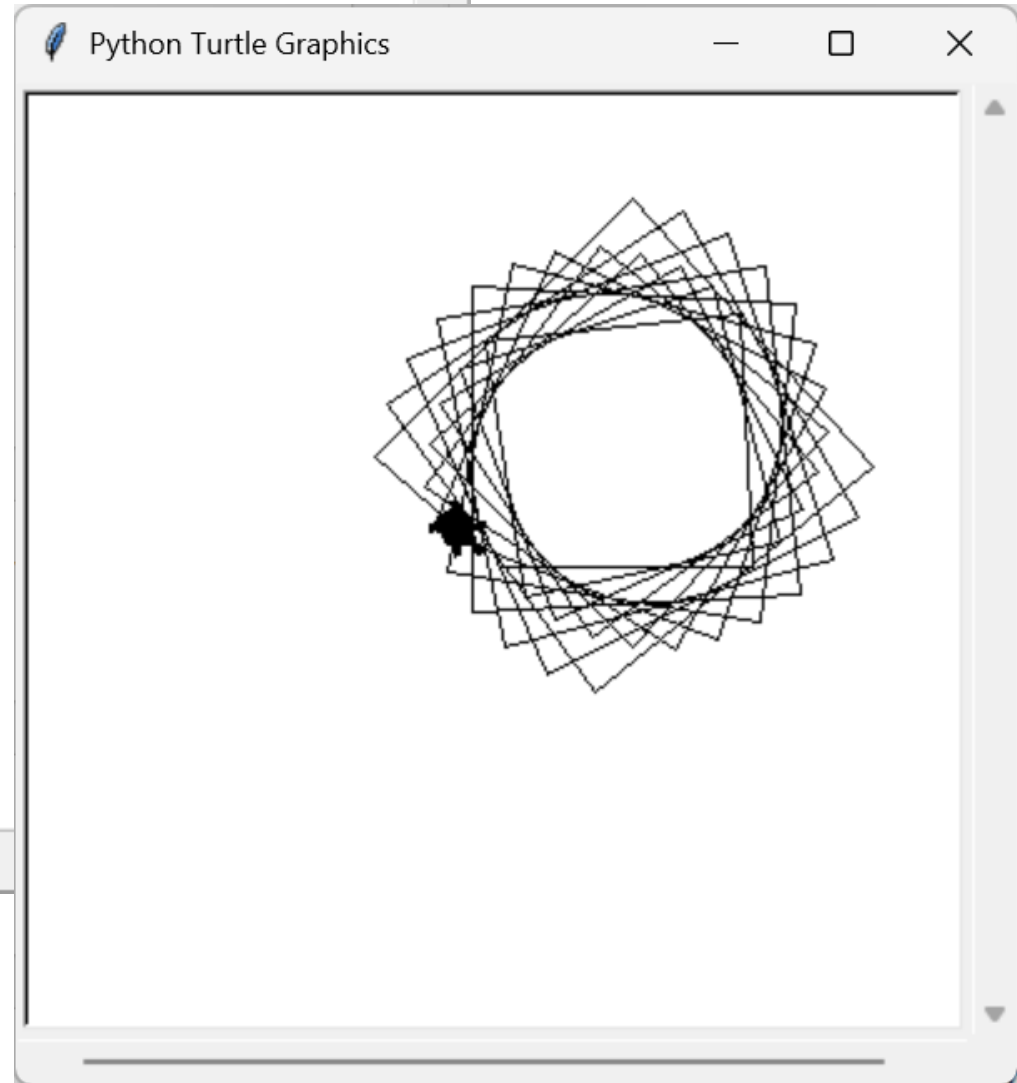
arrow	turtle	circle	square	triangle	classic
					

```
tgraphic.py - D:/ymkangOneDrive/OneDrive/문서/YMKang_Work/수...
File Edit Format Run Options Window Help

import turtle as t
t.setup(width = 400, height = 400)
t.shape('turtle')

t.speed(1)

for i in range(200):
    t.forward(i+100)
    t.left(93)
t.done()
```



그림을 그리기 위해 필요한 메소드

- 움직임과 좌표 변경, 그리기 메소드

메소드	하는 일
forward(d) fd(d)	터틀을 거리 d만큼 앞으로 이동한다.
backward(d) bk(d) back(d)	터틀을 거리 d만큼 뒤로 이동한다.
left(x) lt(x)	터틀을 각도만큼 왼쪽으로 회전한다.
right(x) rt(x)	터틀을 각도만큼 오른쪽으로 회전한다.
circle(r)	현재 위치에서 지정된 반지름 크기의 원을 그린다.
goto(x, y) setpos(x,y) setposition(x,y)	커서를 특정 위치(좌표)로 보낸다.
setx(x)	커서의 x 좌표를 지정한 위치로 이동한다.
sety(y)	커서의 y 좌표를 지정한 위치로 이동한다.
setheading(x) seth(x)	터틀이 바라보는 방향을 바꾼다.
home()	터틀의 위치와 방향을 초기화한다.
speed(sp)	터틀의 속도를 바꾼다. (0: 최고 속도, 1: 느린 속도, 10: 빠른 속도)

• 색상과 채우기, 상태 변경에 관련된 메소드

메소드	하는 일
begin_fill() ... end_fill()	begin_fill()과 end_fill() 사이의 코드로 그린 그림을 색칠한다.
color(c)	터틀의 색깔을 변경한다.
shape(s)	터틀의 모양을 변경한다.
shapesize(s) shapesize(w, h)	터틀의 크기를 변경한다.
pos() position()	터틀의 현재 위치를 구한다.
towards(x, y)	현재 터틀이 있는 위치에서 특정 위치까지 바라보는 각도를 구한다.
xcor()	터틀의 x좌표를 구한다.
ycor()	터틀의 y좌표를 구한다.
heading()	터틀이 현재 바라보는 각도를 구한다.
distance(x, y)	현재 터틀이 있는 위치에서 특정 위치까지의 거리를 구한다.
pendown() pd() down()	펜을 내린다. (그릴 수 있는 상태)
penup() pu() up()	펜을 올린다.
pensize(w) width(w)	펜 굵기 변경
isdown()	펜이 내려져 있는지 여부 확인

터틀 그래픽을 이용한 간단한 그림 그리기

- 터틀의 크기와 모양을 수정

코드 7-14 : 터틀 그래픽의 `shapsize()`, `shape()`, `done()` 메소드

turtle_shape1.py

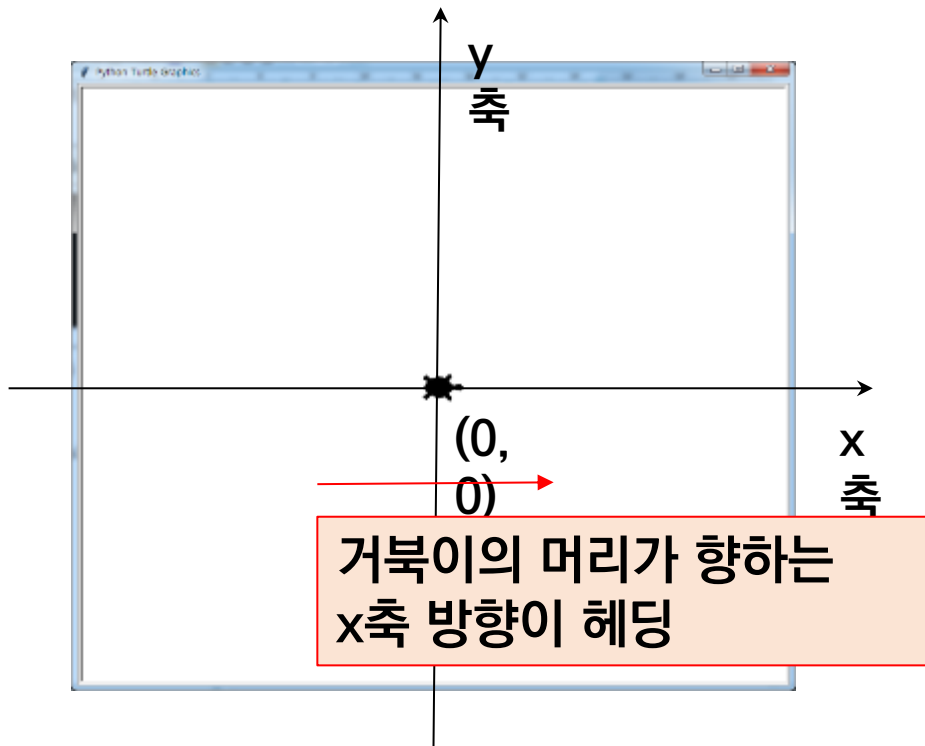
```
import turtle as t
```

```
t.shapesize(2,2)
```

```
t.shape("turtle")
```

```
t.done()
```


- 커서의 좌표는 기본적으로 원점 좌표인 (0, 0)
- 그래픽 공간의 x 축과 y 축은 그림의 가로축 세로축이 된다
- 머리가 향하는 진행방향, 헤딩heading
 - 초기 상태의 커서가 향하는 양의 x축 방향
 - forward() 메소드를 만나면 머리가 향하는 방향으로 나아간다.



- forward(d)를 사용하여 터틀을 앞으로 보내기

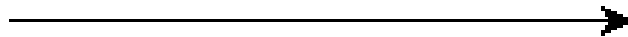
코드 7-15 : forward() 명령을 사용한 직선 그리기

turtle_forward.py

```
import turtle as t
```

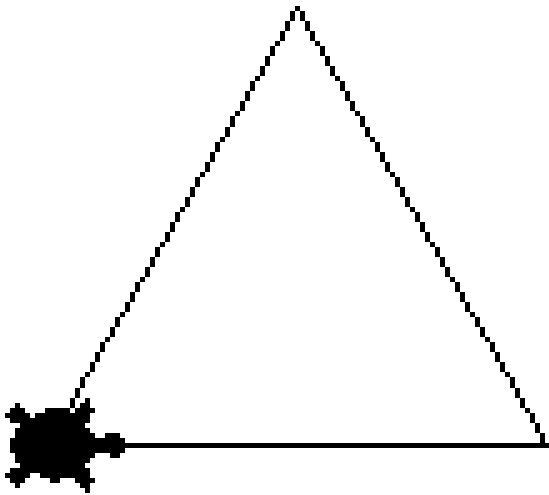
```
t.forward(200)
```

```
t.done()
```



실행화면

- 회전과 움직이기 메소드를 이용하여 다각형 그리기
- 왼쪽으로 회전할 때에는 left() 혹은 lt(), 오른쪽으로 회전할 때에는 right() 혹은 rt() 메소드를 사용
- 각도는 360도를 기준으로 입력함



as 구문 이용하여 turtle -> t 라는 별칭
사용 forward(100), left(120)을 세번
반복하여 삼각형 그리기

코드 7-16 : forward() 명령과 left() 명령을 사용한 삼각형 그리기

turtle_triangle.py

```
import turtle as t
```

```
t.shape("turtle")
```

```
t.forward(100) # 터틀을 헤딩 방향으로 100 픽셀 이동
```

```
t.left(120) # 터틀의 헤딩 방향을 왼쪽으로 120도 회전
```

```
t.forward(100) # 터틀을 헤딩 방향으로 100 픽셀 이동
```

```
t.left(120) # 터틀의 헤딩 방향을 왼쪽으로 120도 회전
```

```
t.forward(100) # 터틀을 헤딩 방향으로 100 픽셀 이동
```

```
t.left(120) # 터틀의 헤딩 방향을 왼쪽으로 120도 회전
```

```
t.done()
```

- for 문을 사용한 동일한 코드

코드 7-17 : for 문을 사용한 삼각형 그리기

turtle_triangle_with_for.py

```
import turtle as t
```

```
for _ in range(3):    # 아래의 기능을 세 번 반복
```

```
    t.forward(100)    #터틀을 헤딩 방향으로 100 픽셀 이동
```

```
    t.left(120)       #터틀의 헤딩 방향을 왼쪽으로 120도 회전
```

```
t.done()
```

- 사각형 그리기 코드

- t.forward(100), t.left(90)을 4번 반복

코드 7-18 : for 반복문과 forward(), left() 메소드를 사용한 사각형 그리기

turtle_rect.py

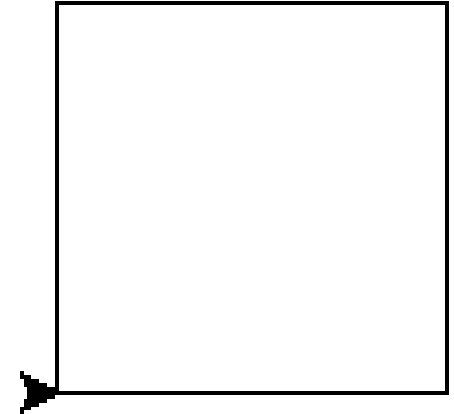
```
import turtle as t
```

```
for _ in range(4):
```

```
    t.forward(100)
```

```
    t.left(90)
```

```
t.done()
```



- 원 그리기

- n 을 100으로 초기화해서 for 루프의 반복 횟수를 100번으로 설정, $360/n$ 을 이용하여 3.6도씩 회전하고 전진

코드 7-19 : forward(), left() 명령을 사용한 원 그리기

turtle_circle_with_for.py

```
import turtle as t
```

```
n = 100
```

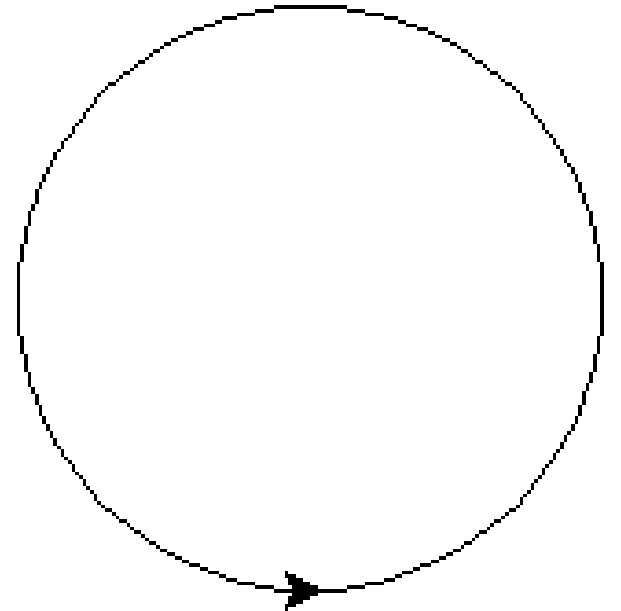
```
length = 5
```

```
for i in range(n):
```

```
    t.left(360/n) # 일반각 사용
```

```
    t.forward(length)
```

```
t.done()
```

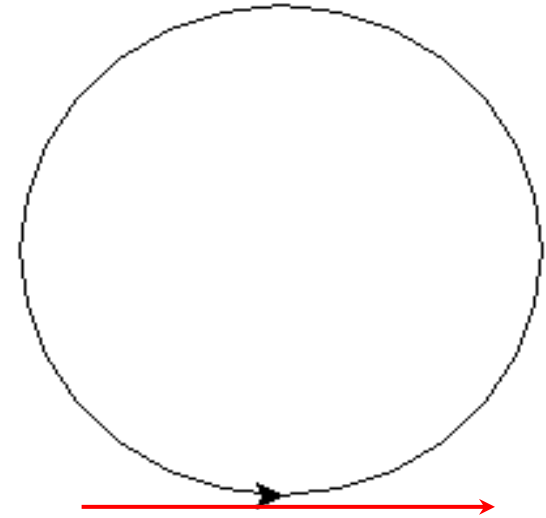


- circle() 메소드를 이용한 원 그리기
 - 인자로 원의 반지름(radius) 입력
 - 디폴트 헤딩 방향은 오른쪽

대화창 실습 : 원 그리기 메소드

```
>>> import turtle as t
```

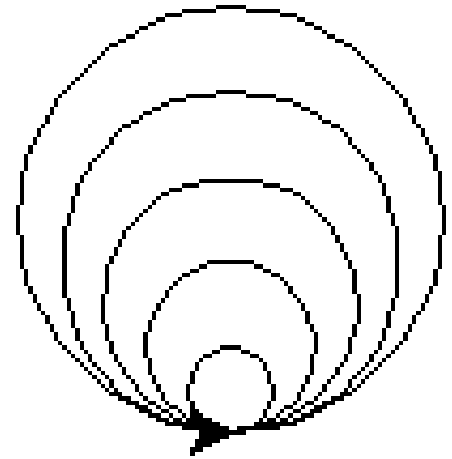
```
>>> t.circle(100) # 반지름을 100으로 하는 원
```



디폴트 헤딩 방향

대화창 실습 : 크기가 다른 5개의 원 그리기

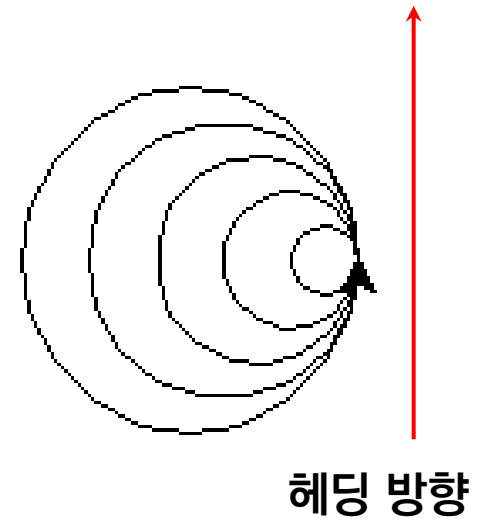
```
>>> import turtle as t  
>>> t.circle(10)   # 원 그리기는 디폴트 헤딩 방향을 기준으로 그린다  
>>> t.circle(20)  
>>> t.circle(30)  
>>> t.circle(40)  
>>> t.circle(50)
```



- setheading(90)을 추가
 - 헤딩 방향이 위쪽으로 변경됨

대화창 실습 : 크기가 다른 5개의 원 그리기

```
>>> import turtle as t
>>> t.setheading(90) # 헤딩 방향을 이동시킴
>>> t.circle(10)      # 왼쪽 원 그리기를 수행
>>> t.circle(20)
>>> t.circle(30)
>>> t.circle(40)
>>> t.circle(50)
```



터틀 그래픽을 이용한 색칠

- 한 변의 길이가 100 픽셀이고 내부가 칠해진 정사각형 그리기
- `begin_fill()`, `end_fill()` 메소드 이용



- color() : 색상을 지정한다

코드 7-21 : color(), begin_fill(), end_fill()을 이용한 색상 사각형 그리기

turtle_fill_rect1.py

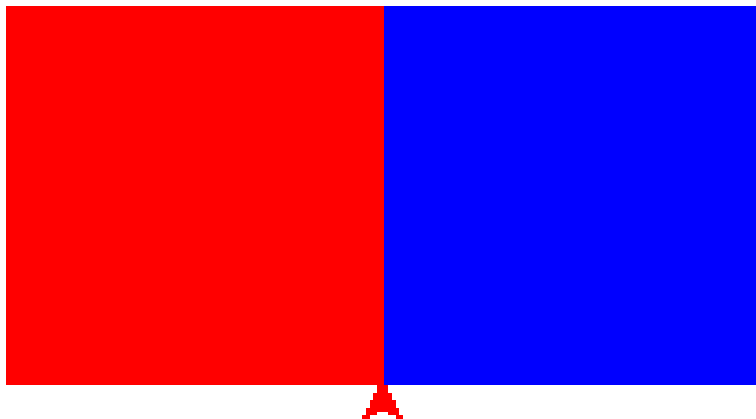
```
import turtle as t

t.color('blue') # 파란색을 선택
t.begin_fill() # 내부를 채움
for _ in range(4):
    t.forward(100)
    t.left(90)
t.end_fill() # 사각형 내부를 파란색으로 채워서 그리기

t.done()
```

- 여러 개의 다른색 사각형 그리기

실행 결과



turtle_fill_rect2.py

```
import turtle as t
```

파란색 네모 그리기

```
t.color('blue')
```

```
t.begin_fill()
```

```
for _ in range(4):
```

```
    t.forward(100)
```

```
    t.left(90)
```

```
t.end_fill() # 사각형 내부를 파란색으로 채워서 그리기
```

```
t.setheading(90)
```

빨간색 네모 그리기

```
t.color('red')
```

```
t.begin_fill()
```

```
for _ in range(4):
```

```
    t.forward(100)
```

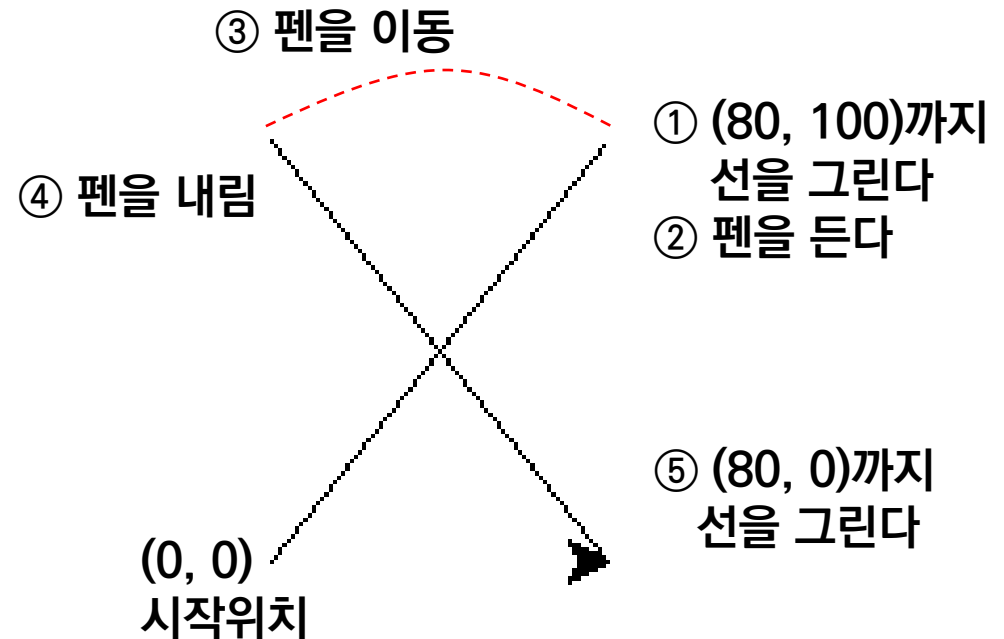
```
    t.left(90)
```

```
t.end_fill() # 사각형 내부를 빨간색으로 채워서 그리기
```

```
t.done()
```

연결되지 않은 2개의 선분 그리기

- penup()
 - 펜을 들어서 이동
- pendown()
 - 펜을 다운
- goto()
 - 지정된 위치로 선을 그리며 커서를 이동



코드 7-24 : goto(), penup(), pendown() 메소드

turtle_penup_pendown.py

```
import turtle as t
```

```
t.goto(80, 100)  # 1번
```

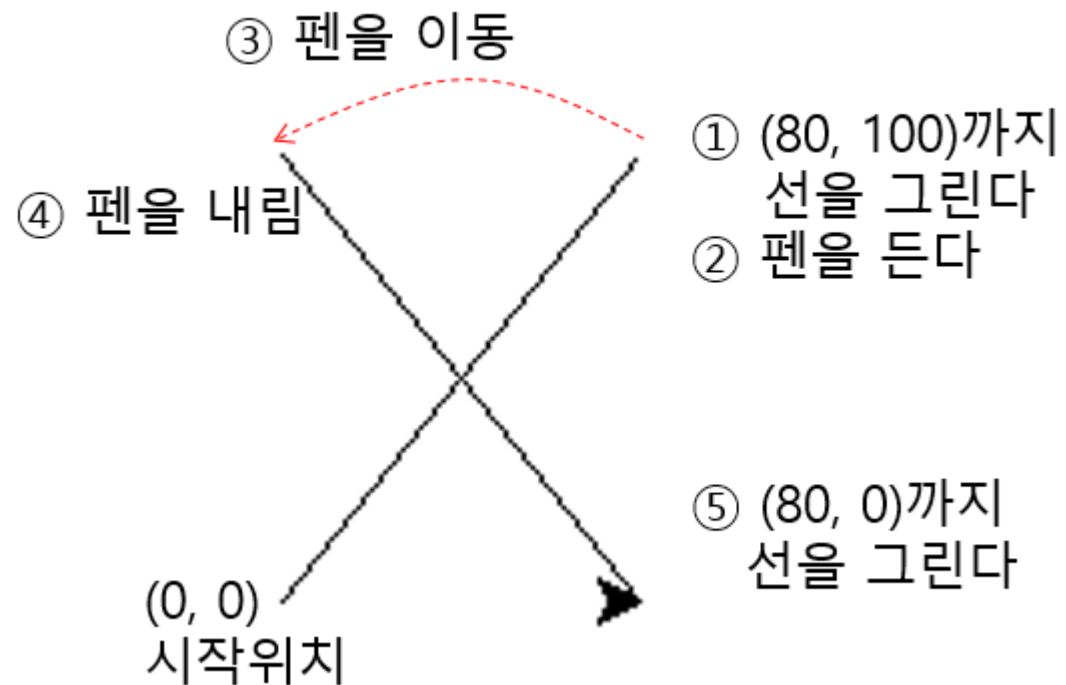
```
t.penup()        # 2번
```

```
t.goto(0, 100)   # 3번
```

```
t.pendown()      # 4번
```

```
t.goto(80, 0)    # 5번
```

```
t.done()
```



- random 모듈을 이용하여 임의의 선분을 마음대로 그리기
- 스탬프 기능을 이용하여 터틀 커서가 지나간 곳에 자취 남기기

코드 7-25 : 터틀 그래픽의 랜덤 플로팅

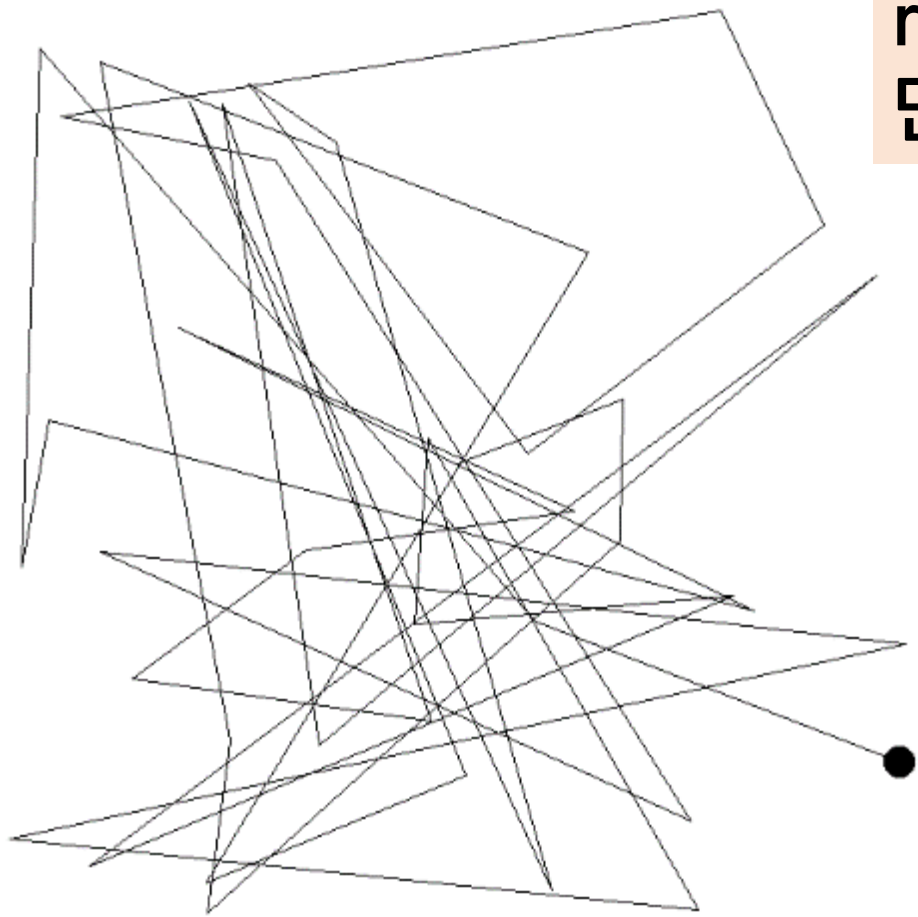
turtle_random.py

```
import turtle as t
import random as rd

t.shape("circle")
d = 300
for _ in range(40):
    x = rd.randint(-d, d)
    y = rd.randint(-d, d)
    t.goto(x, y)

t.done()
```

- random 모듈 이용한 임의의 직선



random 모듈을 사용하면 재미있는 패턴을 만들어 볼 수 있어요.

• 재미있는 재귀함수와 프랙탈

```
import turtle as t
import math # 각도 변환용 (필요 시)
```

```
def draw_branch(length, reduce_rate, angle, depth=1):
```

```
    if depth <= 0: # 종료 조건
        return
```

```
    # 현재 가지 그리기
    t.pendown()
    t.forward(length)
    t.penup()
```

```
    # 재귀 호출: 왼쪽 가지
    t.left(angle) # angle만큼 왼쪽
    draw_branch(length * (1-reduce_rate), reduce_rate, angle, depth - 1)
    # 길이가 reduce_rate 비율로 줄어듦.
```

```
    # 오른쪽 가지
    t.right(angle+angle) # angle * 2 만큼 오른쪽 (총 angle 만큼 좌우 대칭)
    draw_branch(length * (1-reduce_rate), reduce_rate, angle, depth - 1)
```

```
    # 되돌리기 (백트래킹)
    t.left(angle) # 원래 방향 복구
    t.backward(length) # 뒤로 이동
```



```
# 창 설정
```

```
t.setup(width=800, height=600)
t.speed(0) # 최대 속도
t.tracer(0, 0) # 애니메이션 비활성화 (속도 대폭 향상!)
t.bgcolor("black")
t.pencolor("green") # 녹색 가지
```

```
# 메인 실행: 트리 그리기 시작
```

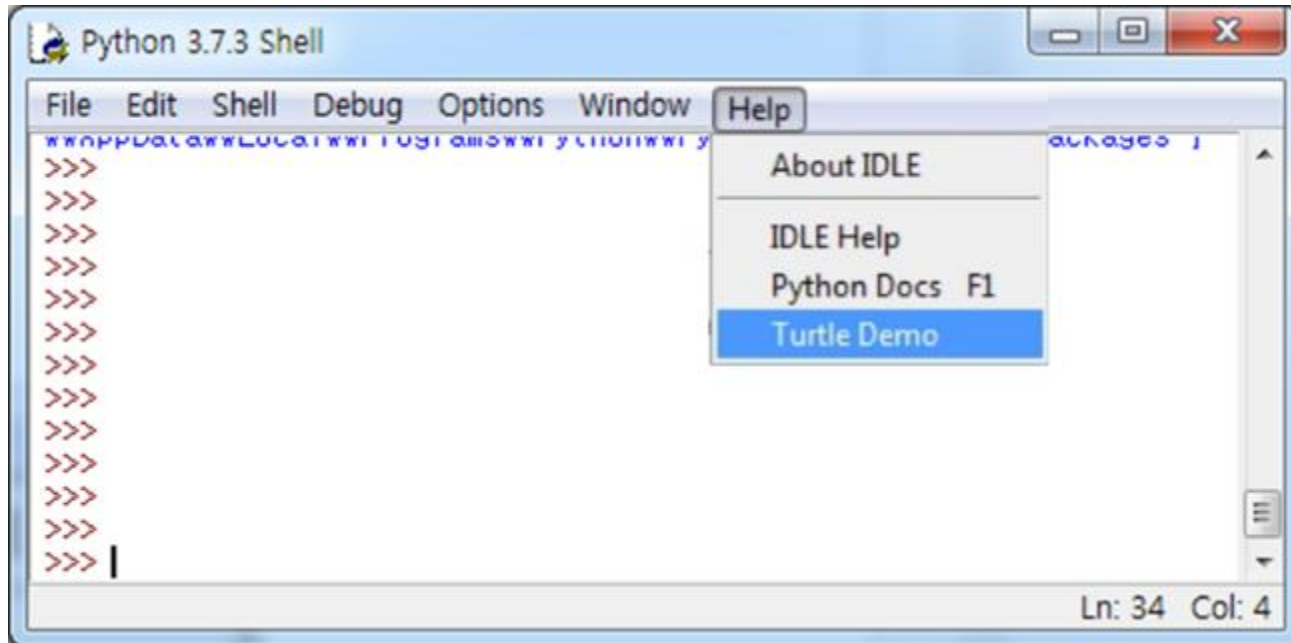
```
t.penup()
t.goto(0, -200) # 시작 위치 (아래쪽)
t.setheading(90) # 위쪽 향함
```

```
draw_branch(50, 0.25, 60, depth=10) # 초기 길이 100
```

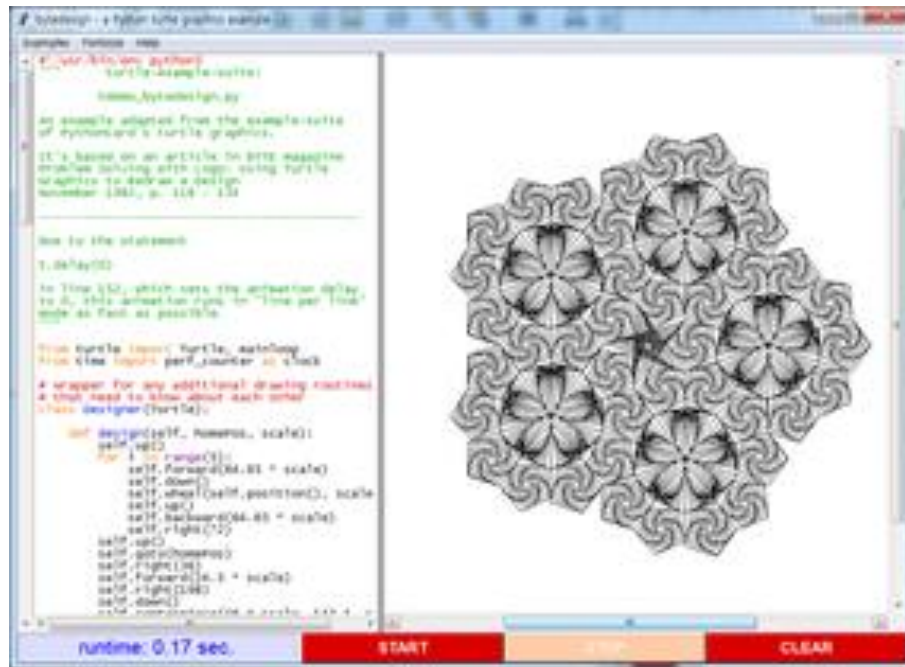
```
t.done() # 창 유지
```

파이썬 터틀 그래픽 데모

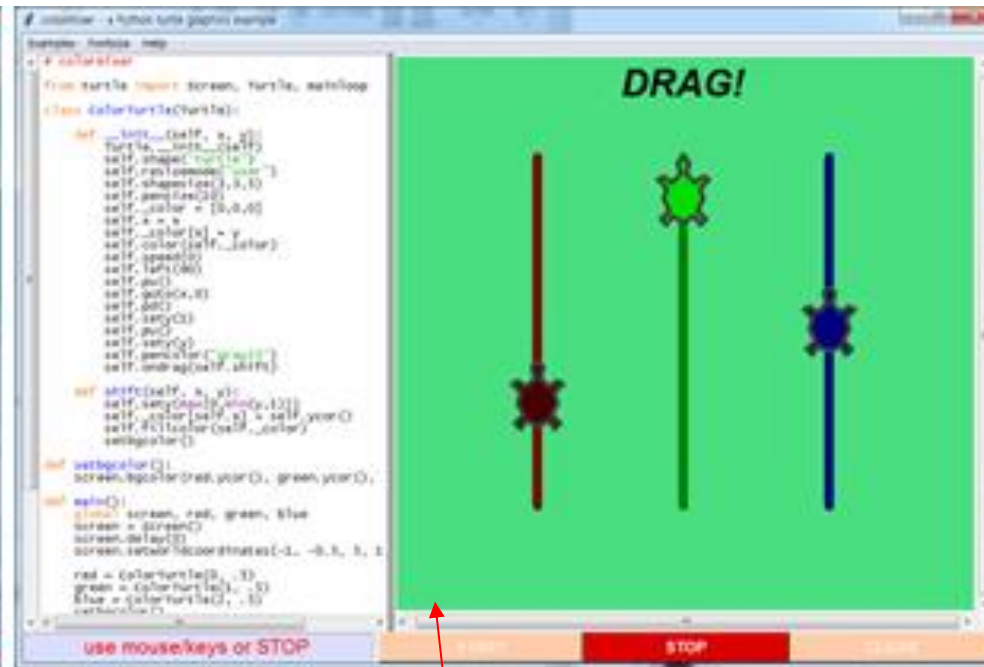
- 여러 가지 패턴이나 그래픽 사용자 인터페이스 프로그래밍을 익힐 수 있는 데모용 코드가 제공됨
- 파이썬 Shell의 메인 메뉴 중에서 Help 메뉴에 있는 Turtle Demo를 선택



- (a) bytedesign , BYTE라는 잡지책의 Logo를 이용한 문제 해결
- (b) colormixer , 세 마리의 거북이 그림을 위 아래로 움직여서 바탕화면의 빨강, 초록, 파랑 색상의 배합을 조합할 수 있는 그래픽 기반 사용자 인터페이스



(a)



(b)

turtle 모듈의 고급 예제를 볼 수 있음



Questions?