

SW알고리즘 - 파이썬

□ 문자열+연산

- 홀따옴표 혹은 겹따옴표로 묶인 문자의 열
- 문자열 출력은 **print()**

* 연산 : 두 문자열을 붙여서 하나의 문자열을 만듦

```
- >>> a='my name is'
- >>> b='Jessica'
- >>> print(a+b)
(출력) my name is Jessica
```

* 문자열*연산

```
// 10에 'a'를 곱하면 //
>>> print(10*'a')
(출력) aaaaaaaaaa
```

* 특정 수의 공백을 가진 문자열들로 줄을 맞춰 메시지 출력하기

- 코드 입력 후 File>SaveAs 선택 - 파일명 myletter.py - Run>RunModule을 클릭하여 실행

○ 특정 수의 공백을 가진 문자열들로 줄을 맞춰 메시지 출력

```
// 첫 번째 줄은 빈 칸을 25번 곱한 spaces 변수를 생성 //
그 다음의 세 줄에서 텍스트를 쉼의 오른쪽 정렬하기 위해 그 변수를 사용
(File - New)
>>> spaces=' '*25 //스페이스를 25번 누름
>>> print('%s 12Butts Wynd' %spaces) //%s에 %spaces를 씀
>>> print('%s West Snoring' %spaces)
>>> print() //열을 띄워줌 \n과 같다.
>>> print(' Dear Sir ')
>>> print('-----')
(저장 - 실행)
```

```
// print 예제 //
>>> a='my name is '
>>> b='Jessica'
>>> print(a+b)
(출력) my name is Jessica
>>> print(5 * b)
(출력) Jessica Jessica Jessica Jessica Jessica
>>> b=5
>>> print(5*b)
(출력) 25
```

□ 배열 집합

○ 리스트(list)

- * 리스트는 조작이 가능
- * 리스트의 위치(인덱스 위치)를 대괄호([])안에 입력
- * wizard_list의 세 번째 항목 출력 가능 //리스트는 인덱스 위치가 0부터 시작
- * >>>print(wizard_list[2])

// 리스트 예제 //

```
>>> numbers= [1,2,3,4]
>>> strings= ['I', 'kicked', 'my', 'toe', 'and', 'it','is','sore']
>>> mylist = [numbers,strings]
>>> print(mylist)
(출력) [[1, 2, 3, 4], ['I', 'kicked', 'my', 'toe', 'and', 'it', 'is', 'sore']]
```

* 리스트에 항목추가

- append사용 / 리스트이름.append('항목이름') //먼저 리스트가 있어야함
- print('리스트[번호]')를 치면 해당 번째의 항목이 출력
- print('리스트[번호:번호]')를 치면 해당 번호와 다음 번호번째의 항목이 출력 (~아님!)
- 리스트이름[2]='다른 이름' 으로 치면 그 번호의 이름이 바뀐다.
- 항목삭제는 del 항목이름[번호]
- 리스트 알파벳순으로 정렬 : 리스트이름.sort() / 반대로 정렬: 리스트이름.sort(reverse=True)
- 리스트 곱하기(나누기빼기x) / >>>list1 = [1,2] / >>>print(list1*5) / (출력) [1,2,1,2,1,2,1,2,1,2]

○ 튜플(tuple)

- * 괄호를 사용하는 리스트 / 소괄호 사용

```
>>> fibs = (0, 1, 1, 2, 3)
>>> print(fibs[3]) //4번째 항목을 출력
(출력) 2
```

- * 튜플에서 인덱스 위치 3의 항목을 출력
- * 튜플과 리스트의 차이점 : 튜플은 한 번 생성하면 수정할 수 없다.>>> fibs[0] = 4 : 오류남)
- * 리스트가 변수라면 튜플은 상수라 보이면 될 듯.

○ 맵(map)

- 각각의 항목들은 키(key)와 그에 대응하는 값(value)를 가지는 집합
- ex) >>> favorite_sports = {'철수' : '축구', '영희' : '전투축구'}
- 키와 값을 구분하기 위해 콜론(:)을 사용 / 중괄호 사용
- 값 변경 : >>> favorite_sports['철수'] = '미식축구' //철수가 축구대신 미식축구가 됨.
- 값 삭제 : >>> del favorite_sports['철수'] // 철수가 아예 삭제됨

	list	tuple	map
사용 괄호	대괄호 ['a', 'b', 'c']	소괄호 (a, b, c)	중괄호 {'a' : '1'}
특징	기본적인 형태임	값이 고정됨	각각에 값을 가짐
단점	더하기, 곱하기 연산이 가능	값을 바꿀 수 없음	더하기 연산자(+)로 결합 불가능

□ 조건문

○ if문

```
// if문 예제 //  
>>> if x>3 : break // x가 3보다 클 경우 break  
>>> for x
```

* 10진(%d), 8진(%o), 16진(%x) 또는 16진 알파벳대문자(%X)

□ 반복문

○ for문

```
// for문 예제1 //  
for i in range(a, b): : 항목이 정해져 있지 않을 경우 사용  
// a,b는 숫자 (0,5) 입력 시 0에서5까지  
>>> for x in range(0, 5): // 시작 숫자인 0부터 5가 되기 전에 멈춘다.  
print('hello %s' %x)  
// hello를 다섯 번 출력, %s는 0부터 4까지 숫자 출력 %x는 for문에 입력한 변수이름  
>>> for x in range(10):  
>>> print(list(range(0, 10))) // [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]  
  
// for문 예제2 //  
for i in wizard_list: : 항목이 정해져 있는 리스트의 경우 사용  
>>> wizard_list = ['spider', 'toe', 'snail', 'bat', 'slug', 'bear']  
>>> for I in wizard_list:  
print(i) // spider / toe / snail / bat / slug / bear 출력(/는 다음 줄)  
print(i) // 이것 한번 더 쓰면 두 번씩 출력 spider / spider / toe / toe 이런 식.  
  
// for문 예제3 (1부터 10까지 더하기) //  
>>> a=0  
>>> for x in range(1, 11):  
a = a + x  
>>> print(a) // 55 출력
```

○ for 루프

```
//할아버지의 동전 복제 장치를 사용하여 일년 동안 얼마나 많은 금화를 얻을 수 있나? //  
>>> found_coins = 20 // 발견한 동전 20개  
>>> magic_coins = 70 // 매일 10개의 동전, 한 주는 7일  
>>> stolen_coins = 3 // 까마귀가 매주 3개씩 훔쳐감  
>>> coins = found_coins // coins의 초기 값을 설정  
>>> for week in range(1, 53):  
coins = coins + magic_coins - stolen_coins  
print('Week %s = %s' %(week,coins))
```

○ 이중for문 (for문 안의 for문)

```
// 구구단 출력문 만들기 //  
>>> for x in range(2, 10):  
for y in range(1, 10):  
print('%2d * %2d = %2d' %(x, y, x*y))  
//%2d는 최소 2자리 범위를 두고 정수 출력  
print() // %d만 쓰면 한자리수일 경우 그냥 한자리 수로 나옴(띄어쓰기)
```

○ while문

```
// while문 예제1 //  
x = 45  
y = 80  
while x < 50 and y < 100:  
//x가 50까지 1씩 증가, y도 100까지 1씩 증가 둘중하나가 만족시 끝  
x = x + 1  
y = y + 1  
print(x, y)  
  
// while문 예제2 //  
while True: // while문은 항상 참 (무한루프에 빠짐)  
코드1...  
if some_value == True:  
break // 무한루프를 끊어줄 수 있는 것  
  
// while문 예제3 (1부터 10까지 더하기) //  
>>> acc = 0  
>>> while a<10:  
>>> a = 0  
a=a+1  
acc=acc+a  
>>> print(acc) // 55 출력
```

□ 코드 재사용하기

○ def : 함수를 지정할 때 쓰임

```
// 함수 예제 //
>>> def testfunc(myname): // 함수 설정(생성)
>>>     print('hello %s' %myname)

>>> testfunc('Mary') // 함수 사용
      (출력) hello Mary
```

○ 전역 변수와 지역변수

- 전역 변수 : 함수 밖 필드에 선언되어 어떤 함수든 사용할 수 있는 변수
- 지역 변수 : 함수 내에 선언된 변수로서, 그 함수 내에서만 사용 가능한 변수

```
// 전역변수, 지역변수 예제 //
>>> a = 100 //전역변수
>>> def testfunc(): // 함수 설정(생성)
>>>     b=10 //지역변수
>>>     c=20 //지역변수
>>>     return a*b*c
>>> print(testfunc())
      (출력) 20000
```

○ 가변 인수 리스트 : 고정되지 않은 수의 인수를 함수에 전달하는 방법

- 함수를 정의할 때, 인수 리스트에 반드시 넘겨야 하는 고정 인수들을 우선 나열

```
// 가변 인수 리스트 예제 // 나머지를 튜플 형식으로 한꺼번
>>> def varg(a, *arg): // 함수 설정(생성)
>>>     print(a, arg)

>>> varg(1)
      1 ()
>>> varg(2,3)
      2 (3)
>>> varg(2,3,4,5,6)
      2 (3,4,5,6) // varg에 저장된 숫자
// 함수 varg를 호출할 때 넘겨지는 첫 인수는 a가 받으며, 나머지는 모두 튜플 형
// 식으로 arg가 받는다.
```

○ import : 모듈 호출하기 (모듈 : 코드들을 묶어 재사용 가능하게 만든 단위/함수들의 구성/ 클래스 개념)

* 사용자 모듈

```
// 사용자 모듈 예제 - mymath 모듈 생성(확장자는 .py) //
>>> #FILE : mymath.py
>>> mypi =3.14

>>> def add(a, b):
>>>     return a+b

>>> def area(r):
>>>     return mypi *r*r

// 사용자 모듈 예제 - mymath 모듈 호출하기 //
>>> import mymath //import로 mymath 함수 호출
>>> dir(mymath) // mymath에 정의된 각종 이름들을 보여줌/ __package__까지는 윈도우에서 자동설정
      ['__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__', '__package__', 'add', 'area', 'mypi']
>>> mymath.mypi //mymath의 mypi를 참조
      3.14
>>> mymath.area(5) // mymath의 area 호출
      78.5
```

* 내장 모듈 : 파이썬 내의 자체 모듈 // time(시간과 날짜를 계산)과 같은 모듈이 있음

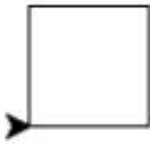
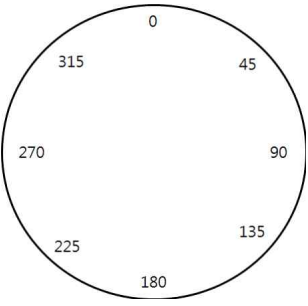
- 점(.)을 이용하여 모듈 내의 함수들을 호출 //ex) time.asctime()
- **sys.stdin.readline** : 엔터를 누르기 전까지 키보드로 입력한 텍스트를 읽는데 사용
- 람다 함수 : 이름이 없는 한 줄 짜리 함수 / 항상 참을 리턴

```
// 람다 함수 예제 //
>>> g = lambda x,y : x+y
>>> g(1,2) // 3을 리턴
```

□ turtle로 그림그리기

○ **turtle** : 검고 작은 화살표 / 선 그래픽을 그리기 위한 함수




* **turtle 화살표 각도**

<pre>// turtle 예제 // >>> import turtle >>> t = turtle.Pen() >>> t.forward(50) >>> t.left(90) >>> t.forward(50) >>> t.left(90) >>> t.forward(50) >>> t.left(90) >>> t.forward(50) >>> t.left(90)</pre>		<pre>// 화살표 각도 //</pre> 
---	---	---

* **turtle 캔버스 지우기**

* **t.reset()** : 거북이를 시작 위치에 둠


* **t.clear()** : 거북이의 방향은 그대로 있고, 그림만 지우는 것

<pre>// turtle 예제 // >>> import turtle >>> t = turtle.Pen() >>> t.forward(50)</pre>	
<pre>// t.clear() // 선만 없어짐</pre> 	<pre>// t.reset() // 원래 상태</pre> 

* **turtle 두 개의 선 그리기**

* **t.up()** : 거북이 펜을 들어줌 (선x)

* **t.down()** : 거북이 펜을 내림 (선o)

<pre>// turtle 예제 // >>> import turtle >>> t = turtle.Pen() >>> t.forward(50) >>> t.left(90) >>> t.up() // 펜을 들어줌 >>> t.forward(50) >>> t.left(90) >>> t.down() // 다시 선을 그릴 수 있음 >>> t.forward(50)</pre>	
---	---

□ 객체와 클래스

- Class : 객체를 만들어 내는 틀
- 클래스 특성 추가하기 : def을 사용

```
>>> Class Animals(Animate): // Animals 클래스에 함수 추가
    def brathe(self):
        pass
    def move(self):
        pass
    def eat_food(self):
        pass
```

- 클래스 불러오기 : >>> name = ClassName()
- ex) Jo = Human() // Jo라는 이름의 Human을 불러옴
- 클래스 함수 호출 : >>> name.move() // ClassName()의 move함수 호출

□ 클래스의 기능 - 상속, self매개변수, 객체의 초기화

- 상속 : 클래스의 부모-자식 관계를 나타냄 (부모의 기능도 사용 가능)
- ex) class Child(Parent): // Parent는 Child의 부모 클래스
- self 매개변수 : 클래스에 있는 함수가 그 클래스에 있는 다른 함수를 호출하는 방법

```
>>> class Giraffes(Mammals):
    def find_food(self): // self를 씌으로써 이 클래스 내의
        self.move() // 다른함수를 호출하였다.
        print("I've found food!")
        self.eat_food()
    def eat_leaves_from_trees(self):
        self.eat_food()
    def dance_a_jig(self):
        self.move()
        self.move()
        self.move()
```

- init 함수 : 객체의 초기 값을 설정해줌

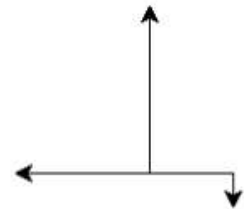
```
>>> class Giraffes:
    def _init_(self, spots):
        self.giraffe_spots = spots // spots를 설정
```

□ 그래픽 객체와 클래스 및 내장 함수

○ turtle.Pen() // Pen클래스 객체 생성 가능

```
import turtle // turtle을 포함시킴
>>> avery = turtle.Pen() // avery라는 이름의 터틀 펜을 만듦
>>> kate = turtle.Pen() // kate라는 이름의 터틀 펜을 만듦
//avery의 이동
>>> avery.forward(50) // 직진(오른쪽)으로 50픽셀 앞으로 이동
>>> avery.right(90) // 오른쪽으로 90도 회전
>>> avery.forward(20) // 직진으로 20픽셀 앞으로 이동
>>> avery.forward(50)
//avery의 이동
>>> kate.left(90) // 왼쪽으로 90도 회전
>>> kate.forward(100) // 100픽셀만큼 이동
// jacob이라는 이름의 터틀펜을 추가 및 이동
>>> jacob = turtle.Pen()
>>> jacob.left(180)
>>> jacob.forward(80)
```

결과



* 내장함수들

○ abs 함수 : 숫자의 절대 값을 반환

- >>> print(abs(-10)) // 출력 : 10

○ bool 함수 : True와 False만을 반환하는 함수

- >>> print(bool(0)) // 출력 : False

- >>> print(bool(1123.23)) // 출력 : True

○ eval 함수 : 매개변수로 문자열을 받음

- >>> your_calculation = input('Enter a calculation : ')

// Enter a calculation : 12*52 (입력)

- >>> eval(your_calculation) // 출력 : 624

○ exec 함수 : 파일로 읽어 들인 작은 프로그램을 실행할 때 사용

- >>> my_small_program = '''print('ham')

// print('sandwich')''' (입력)

- >>> exec(my_small_program)

// 출력 : ham sandwich

○ float 함수 : 문자열이나 숫자를 실수로 변환

○ int 함수 : 문자열이나 숫자를 정수로 변환

○ len 함수 : 객체(문자열)의 길이(글자 개수)를 반환

- >>> alist = ['a','b','c','d','e']

- >>> print(len(alist)) // 출력 : 5

○ min, max 함수 : 현재 리스트, 튜플, 문자열에 있는 가장 작은(큰) 항목 반환

○ range 함수 : for 루프문에 사용되며, 시작과 끝, 증가 크기 기능이 있음

- >>> num = list(range(0,10,2))

- >>> print(num) // 출력 : [0, 2, 4, 6, 8]

○ sum 함수 : 리스트에 있는 항목을 더해줌

□ 프로시저(모듈) 사용하기

- **copy 모듈** : 함수와 클래스를 더 쉽게 사용하기 위해 사용하는 모듈
 - 객체의 복사본을 생성하는 함수들을 가지고 있다.
 - 객체 **copy** 모듈 사용 후 내용을 변경해도 본래의 내용은 변하지 않음

```
>>> my_animals[0].species = 'ghoul'
>>> print(my_animals[0].species) // 출력 : ghou(복사본은 바뀌었다.)
>>> print(more_animals[0].species) // 출력 : hippogriff(원본은 바뀌지 않았다.)
```

- 객체를 복사 후 원본을 바꾸면 복사본의 내용은 바뀌지 않는다.

- **keyword 모듈** : 모든 파이썬 키워드들의 목록을 반환하는 모듈
 - keyword와 iskeyword는 **kwlist**라는 변수를 가지고 있다.
 - iskeyword : 어떤 문자열이 파이썬 키워드일 경우 참을 반환

- **random 모듈** : 난수를 생성할 때 쓰이는 모듈

```
>>> import random
>>> print(random.randint(1, 100)) // 출력 : 1~100사이 아무 숫자를 하나 출력
>>> print(random.randint(1000, 5000)) // 출력 : 1000~5000 사이 숫자 하나 출력
```

- **choice** : 리스트에서 무작위로 항목을 뽑을 경우 사용

```
>>> import random
>>> arand = ['a','b','c','d','e','f']
>>> print(random.choice(arand)) // 출력 : a~f 중 하나 출력
```

- **shuffle** : 리스트를 섞을 때 사용하는 모듈(카드 섞기)

```
>>> import random
>>> arand = ['a','b','c','d','e','f']
>>> random.shuffle(arand)
>>> print(arand) // 출력 : ['e','c','a','f','b'] 와 같이 섞여서 출력
```

- **sys 모듈** : 파이썬 셸 자체를 컨트롤할 때 사용하는 시스템 함수 모듈
 - **stdin 객체** : 키보드로 입력한 텍스트 라인을 읽는 **readline** 함수를 가지고 있음

```
>>> import sys
>>> v = sys.stdin.readline() // ()안에 숫자를 넣으면 그 숫자만큼 문자를 읽는다
I Love You
>>> print(v) // 출력 : I Love You
```

- **stdout 객체** : **write**와 같은 함수들을 가지고 있는 파일 객체이다.

```
>>> import sys
>>> v = sys.stdout.write("I Love You") // 출력 : I Love You
```


- version 변수 : 현재 쓰고있는 파이썬 버전을 확인할 때 사용한다.

○ **time 모듈** : 시간을 표시하는 함수들을 가지고 있다.

- asctime : 날짜를 변환할 수 있는 기능

```
>>> import time
>>> print(time.asctime()) // 출력 : Thu Jan 9 21:24:38 2014
>>> t = (2016, 6, 15, 10, 30, 48, 6, 0, 0)
>>> print(time.asctime(t)) // 출력 : Wed May 15 10:30:48 2016
```

- localtime : 날짜와 시간을 객체로 반환

```
>>> import time
>>> t = time.localtime()
>>> year = t[0]
>>> month = t[1]
>>> print(year) // 출력 : 2016
>>> print(month) // 출력 : 6
```

- sleep : 프로그램에서 딜레이를 줄 때 사용

```
>>> for x in range(1, 61):
    print(x)
    time.sleep(1) // 출력시 마다 1초씩 딜레이를 준다.
```

○ **pickle 모듈** : 게임의 세이브 파일을 만들 때 사용

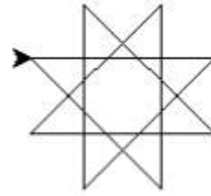
```
>>> import pickle
>>> game_data = {
    'player-position': 'N23 E45',
    'pockets': ['keys', 'pocket knife'],
    'backpack': ['rope', 'hammer'],
    'money': 642.42
}
>>> save_file = open('save.dat', 'wb')
>>> pickle.dump(game_data, save_file)
>>> save_file.close()
```

□ 도형그리기

○ for문 사용

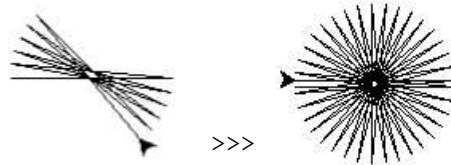
//8개 포인트 별 그리기//

```
>>> import turtle
>>> t = turtle.Pen()
>>> for x in range(1,9) :
>>>     t.forward(100)
>>>     t.left(225)
```



//별 그리기//

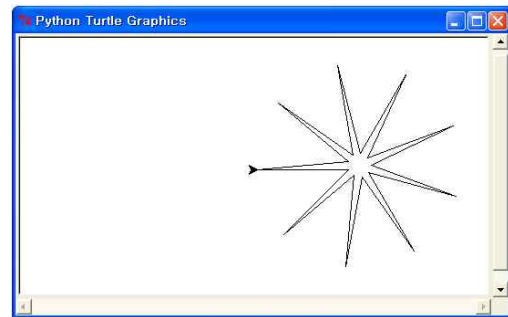
```
>>> import turtle
>>> t = turtle.Pen()
>>> for x in range(1,38) :
>>>     t.forward(100)
>>>     t.left(175)
```



○ for문 + if문 사용

//포인트 별 그리기//

```
>>> import turtle
>>> t = turtle.Pen()
>>> for x in range(1,19) :
>>>     t.forward(100)
>>>     if x%2==0:
>>>         t.left(175)
>>>     else :
>>>         t.left(225)
```



○ 자동차 그리기

- **begin_fill, end_fill** : 캔버스 영역을 색을 채우는데 사용
- **setheading** : 거북이가 특정 방향을 향하도록 함

// 자동차 몸체 //

```
>>> import turtle
>>> t = turtle.Pen()
>>> t.color(1,0,0)
>>> t.begin_fill()
>>> t.forward(100)
>>> t.left(90)
>>> t.forward(20)
>>> t.right(90)
>>> t.forward(20)
>>> t.left(90)
>>> t.forward(60)
>>> t.left(90)
>>> t.forward(20)
>>> t.right(90)
>>> t.forward(20)
>>> t.left(90)
>>> t.forward(20)
>>> t.end_fill()
```

// 자동차 첫 번째 바퀴 //

```
>>> t.color(0,0,0)
>>> t.up()
>>> t.forward(10)
>>> t.down()
>>> t.begin_fill()
>>> t.circle(10)
>>> t.end_fill()
```

// 자동차 두 번째 바퀴 //

```
>>> t.setheading(0)
>>> t.up()
>>> t.forward(90)
>>> t.right(90)
>>> t.forward(10)
>>> t.setheading(0)
>>> t.begin_fill()
>>> t.circle(10)
>>> t.end_fill()
```

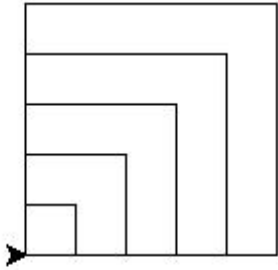
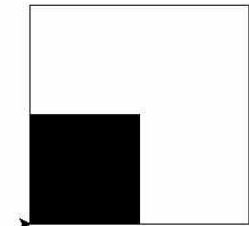


□ 그리기 코드 다시 사용하기 - 색으로 채워진 도형그리기

○ color 함수 : t.color(red,green,blue) 형태로 사용

- t.color(0,0,0) : 검은색
- t.color(1,1,1) : 흰 색

○ 사각형의 크기 조절 및 색 채우기

<pre>// 사각형 함수 // >>> t.begin_fill() >>> def mysquare(size,filled) : if filled==True : t.begin_fill() for x in range(1, 5) : t.forward(size) t.left(90) if filled==True: t.end_fill()</pre> <pre>// 크기에 맞게 생성 // >>> import turtle >>> t = turtle.Pen() >>> mysquare(25) >>> mysquare(50) >>> mysquare(75) >>> mysquare(100) >>> mysquare(125)</pre>	
<pre>// 색 채우기 // >>> t.reset() >>> mysquare(100,True) >>> mysquare(200,False)</pre>	


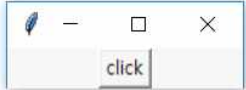
○ 도형 색 채우기, 검은색 외곽선 : 채워진 도형 + 선만 있는 도형

<pre>// turtle 과제 팔각형 그리기 // >>> import turtle >>> octagon = turtle.Pen() >>> def MyOctagon(size, filled): // 팔각형 함수 >>> if filled == True: >>> octagon.begin_fill() // 그려줌 >>> for i in range(0,8): >>> octagon.forward(size) >>> octagon.left(45) >>> if filled == True: >>> octagon.end_fill() // 그려줌(도형색) >>> >>> octagon.color(0.75, 0.9, 0) //색지정 >>> MyOctagon(50, True) // 도형색넣음 >>> octagon.color(0, 0, 0) // 색지정 >>> MyOctagon(50, False) // 도형색 넣지않음</pre>	
--	--

□ 캔버스에 그림그리기 - 좌표를 이용한 도형 그리기

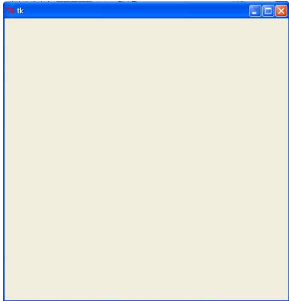
○ tkinter 모듈 : 그래픽을 생성하는 모듈 (자바 Swing 개념)

* from 모듈명 : 그 모듈의 기능을 전부 사용

<pre>// 버튼 만들기 // >>> from tkinter import * >>> tk = Tk() >>> btn = Button(tk, text="click") >>> btn.pack()</pre>	
<pre>// 버튼 클릭시 이벤트 설정 // >>> def hello() : print('Hi!') >>> from tkinter import * >>> tk = Tk() >>> btn = Button(tk, text="click", command=hello) >>> btn.pack() >>> btn.click() Hi! Hi! Hi! Hi!</pre>	<pre>>>> from tkinter import * >>> tk = Tk() >>> btn=Button(tk, text="click", command=hello) >>> btn.pack() >>> btn.click() Hi! Hi! Hi! Hi!</pre> 

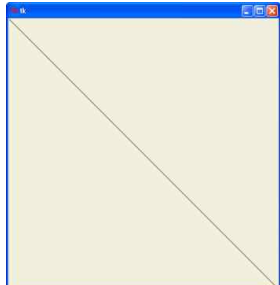
○ 캔버스 생성

- 캔버스의 폭과 높이의 픽셀을 입력

<pre>// 캔버스 크기 설정 (500*500) // >>> from tkinter import * >>> tk = Tk() >>> canvas = Canvas(tk,width=500,height=500) >>> canvas.pack()</pre>	
---	---

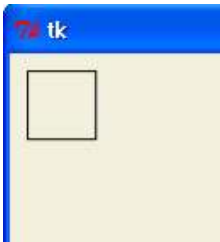
○ 선, 상자 그리기

- ex) canvas.create_line(0,0,500,500)

<pre>// 캔버스에 선 그리기 // >>> from tkinter import * >>> tk = Tk() >>> canvas = Canvas(tk,width=500,height=500) >>> canvas.pack() >>> canvas.create_line(0, 0, 500, 500)</pre>	
--	--

○ 상자 그리기

- ex) canvas.create_rectangle(10,10,50,50)

<pre>// 캔버스에 상자 그리기 // >>> from tkinter import * >>> tk = Tk() >>> canvas = Canvas(tk,width=500,height=500) >>> canvas.pack() >>> canvas.create_rectangle(10,10,50,50)</pre>	
---	---

□ 애니메이션 만들기

○ 화면 위에서 움직이기

```
// 삼각형 애니메이션 만들기 1 //
```

```
>>> import time
>>> from tkinter import *
>>> tk = Tk()
>>> canvas = Canvas(tk, width=400,height=200)
>>> canvas.pack()
>>> canvas.create_polygon(10,10,10,60,50,35) // 삼각형 폴리곤을 만듦
1 // True가 됨
>>> for x in range(0, 60):
    canvas.move(1,5,0) // ID가 1인 객체를 가로5픽셀, 세로0픽셀 만큼 움직임
    tk.update() // tkinter 화면을 업데이트함
    time.sleep(0.05) // 0.05초 딜레이
```



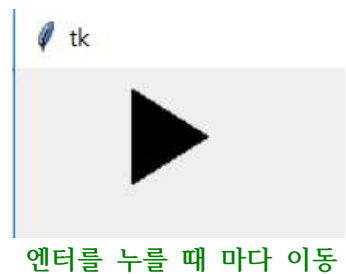
○ 이벤트 바인딩 : 플레이어가 특정한 무엇을 했을 때 특정 이벤트 발생

* <KeyPress-Return> : Enter 또는 Return을 누르는 것

```
// 키보드를 눌렀을 때 이벤트 발생 //
```

```
>>> from tkinter import *
>>> tk = Tk()
>>> canvas = Canvas(tk, width=400, height=400)
>>> canvas.pack()
>>> canvas.create_polygon(10, 10, 10, 60, 50, 35)
1
>>> def movetriangle(event):
    canvas.move(1, 5, 0)

>>> canvas.bind_all('<KeyPress-Return>',
movetriangle)
'52502040movetriangle'
```



엔터를 누를 때 마다 이동

```
// 키보드 방향키를 눌렀을 때 이벤트 발생 //
```

```
>>> from tkinter import *
>>> tk = Tk()
>>> canvas = Canvas(tk, width=400, height=400)
>>> canvas.pack()
>>> canvas.create_polygon(10, 10, 10, 60, 50, 35)
1
>>> def movetriangle(event):
    if event.keysym == 'Up':
        canvas.move(1, 0, -3)
    elif event.keysym == 'Down':
        canvas.move(1, 0, 3)
    elif event.keysym == 'Left':
        canvas.move(1, -3, 0)
    else:
        canvas.move(1, 3, 0)

>>> canvas.bind_all('<KeyPress-Up>', movetriangle)
'55512640movetriangle'
>>> canvas.bind_all('<KeyPress-Down>', movetriangle)
'55639856movetriangle'
>>> canvas.bind_all('<KeyPress-Left>', movetriangle)
'56053472movetriangle'
>>> canvas.bind_all('<KeyPress-Right>', movetriangle)
'59349312movetriangle'
```



상,하,좌,우 방향키를
누를 때 마다 해당 방향
으로 이동

```

// 공튀기기 //
from tkinter import*
import random
import time

tk = Tk()
tk.title("Game")
tk.resizable(0, 0)
tk.wm_attributes("-topmost", 1)
canvas = Canvas(tk, width=500, height=400, bd=0, highlightthickness=0)
canvas.pack()
tk.update()

class Ball:
    def __init__(self, canvas, color):
        self.canvas = canvas
        self.id = canvas.create_oval(10, 10, 25, 25, fill=color)
        self.canvas.move(self.id, 245, 100)
        self.x = 0
        self.y = -1
        self.canvas_height = self.canvas.winfo_height()

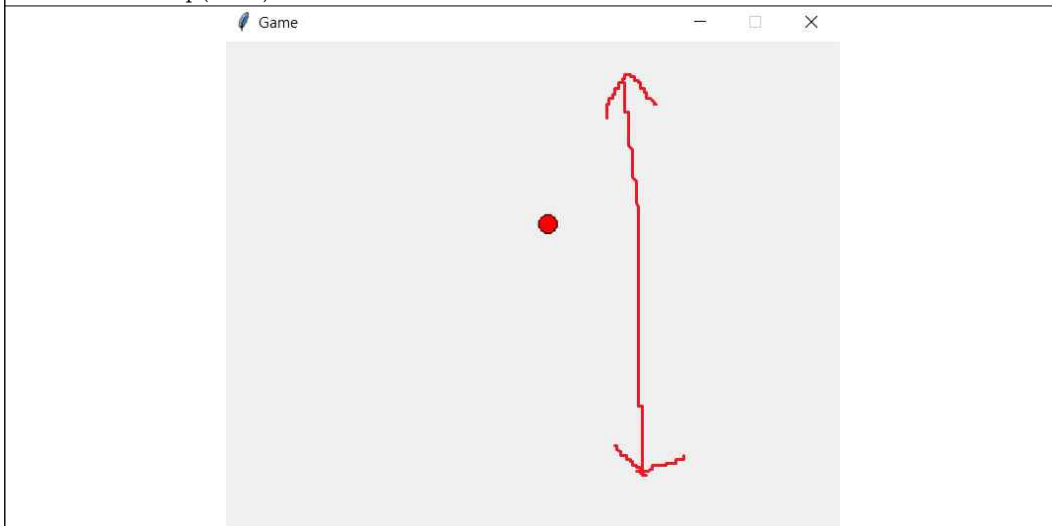
    def draw(self):
        self.canvas.move(self.id, self.x, self.y)
        pos = self.canvas.coords(self.id)
        if pos[1] <= 0:
            self.y = 1
        if pos[3] >= self.canvas_height:
            self.y = -1

    def movetriangle(event):
        canvas.move(1, 5, 0)

ball = Ball(canvas, 'red')

while 1:
    ball.draw()
    tk.update_idletasks()
    tk.update()
    time.sleep(0.01)

```



■ 첫 번째 과제(리스트 만들기)

- 여러분이 좋아하는 취미에 대한 리스트를 만들고, 그 리스트에 games라는 변수명을 붙여준다.
games 리스트를 알파벳순으로 정렬한다.
- 또 여러분이 좋아하는 음식에 대한 리스트를 만들고 foods라는 이름의 변수명을 준다.
foods 리스트를 알파벳순으로 정렬한다.
- 두 개의 리스트를 결합하여 하나의 리스트를 만들어 그 결과를 favorites라고 한다.
마지막으로 변수 favorites를 출력한다.
- >>> games = ['C프로그래밍','Java','Unity','Linux']
- >>> games.sort()
- >>> foods = ['치킨','돈까스','햄버거']
- >>> foods.sort()
- >>> favorites = [games,foods]
- >>> print(favorites)
- (출력) [['C프로그래밍','Java','Unity','Linux'],['치킨','돈까스','햄버거']]

■ 두 번째 과제(도형 그리기)

```
>>> import turtle
>>> octagon = turtle.Pen()
>>> def MyOctagon(size, filled): // 팔각형 함수
>>>     if filled == True:
>>>         octagon.begin_fill() // 그려줌
>>>         for i in range(0,8):
>>>             octagon.forward(size)
>>>             octagon.left(45)
>>>         if filled == True:
>>>             octagon.end_fill() // 그려줌(도형색)
>>>
>>> octagon.color(0.75, 0.9, 0) //색지정
>>> MyOctagon(50, True) // 도형색넣음
>>> octagon.color(0, 0, 0) // 색지정
>>> MyOctagon(50, False) // 도형색 넣지않음
```

