

컴퓨팅 사고와 프로그래밍



storyset by Freepik

CH 07.

함수



학습목표



1. 함수란 무엇인지, 왜 필요한지를 설명할 수 있다.
2. 사용자 정의 함수를 직접 작성하고 호출하여 사용할 수 있다.
3. 함수 안으로 값을 넣는 방법과 함수 호출 후 만들어진 값을 반환하는 방법을 익힌다.

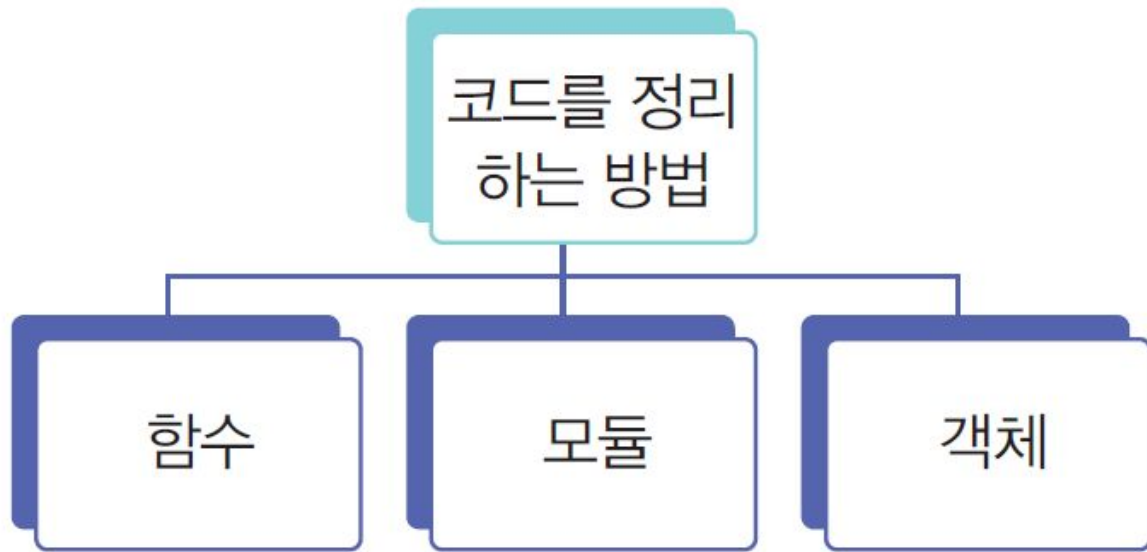




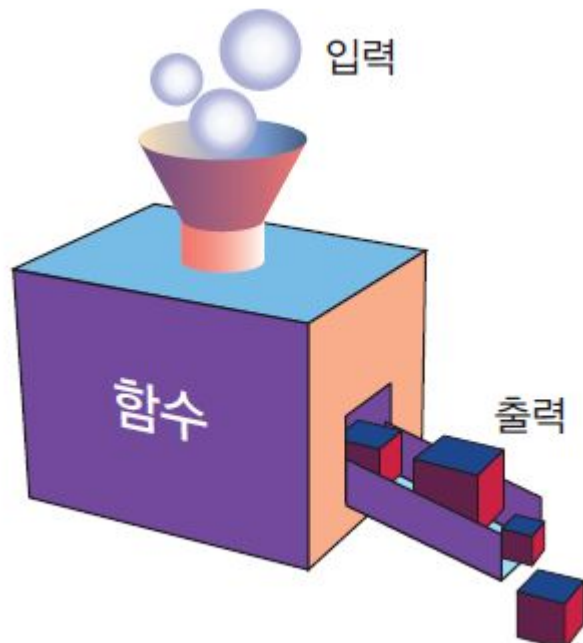
이론수업



- 함수(function) : 서로 관련된 코드를 묶은 것
- 모듈(module) : 여러 개의 함수들이 포함되어 있는 소스 파일
- 객체(object) : 서로 관련된 변수와 함수를 하나로 묶은 것



- 함수는 코드의 묶음에 이름을 붙인 것
- 함수는 입력을 받아서 출력을 내보내는 (속을 볼 필요가 없는) 박스



함수 정의

```
def print_address():  
    print("서울특별시 종로구 1번지")  
    print("파이썬 빌딩 7층")  
    print("홍길동")
```

함수 호출

```
print_address()
```

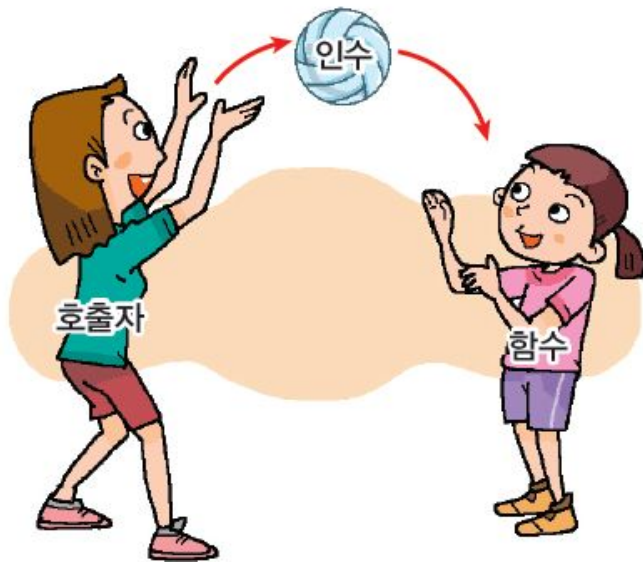
서울특별시 종로구 1번지
파이썬 빌딩 7층
홍길동

한 번만 함수를 정의하면
언제든지 필요할 때면
함수를 불러서 일을 시킬 수
있다.

```
print_address()  
print_address()  
print_address()
```

```
서울특별시 종로구 1번지  
파이썬 빌딩 7층  
홍길동  
서울특별시 종로구 1번지  
파이썬 빌딩 7층  
홍길동  
서울특별시 종로구 1번지  
파이썬 빌딩 7층  
홍길동
```


우리는 함수에 값(정보)을 전달할 수 있다. 이 값을 인수(argument)라고 한다.



매개변수

```
def print_address(name):  
    print("서울특별시 종로구 1번지")  
    print("파이썬 빌딩 7층")  
    print(name)
```

```
print_address("홍길동 ")
```

인수(인자)

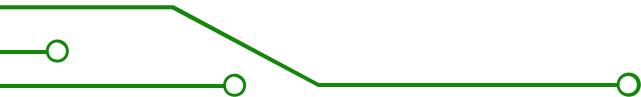


함수는 값을 반환할 수 있다.



```
def calculate_area (radius):  
    area = 3.14 * radius**2  
    return area
```

```
c_area = calculate_area(5.0)
```



```
def get_input():  
    return 2, 3
```

```
x, y = get_input()
```

```
def calculate (radius):  
    area = 3.14 * radius**2  
    perimeter = 2.0 * 3.14 * radius  
    return area, perimeter
```

```
x, y = calculate()
```

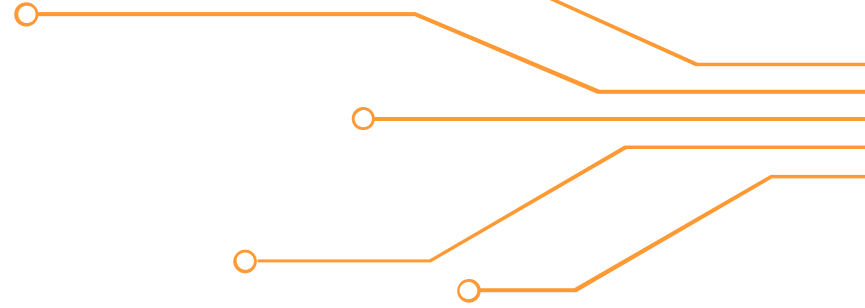


```
def get_sum(start, end):  
    sum = 0  
    for i in range(start, end+1):  
        sum += i  
    return sum
```

```
print(get_sum(1, 10))
```

get_sum(1 , 10)

```
def get_sum( start , end ):  
    sum = 0  
    for i in range(start, end+1):  
        sum += i  
    return sum
```





실습



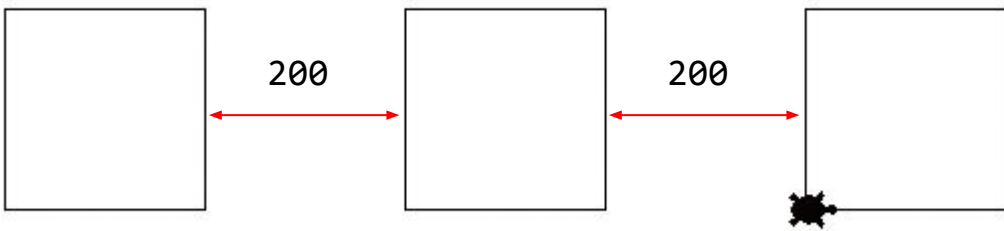
정사각형을 그리는 함수는 다음과 같다.

3개의 정사각형을 그리자.

정사각형들의 간격은 200 픽셀이다.

```
def square(x, y, length):  
    t.up()  
    t.goto(x, y)  
    t.down()  
  
    for i in range(4):  
        t.forward(length)  
        t.left(90)
```

length는 변의 길이
펜을 든다.
(x, y)으로 이동한다.
펜을 내린다.



```
import turtle
t = turtle.Turtle()
t.shape("turtle")
```

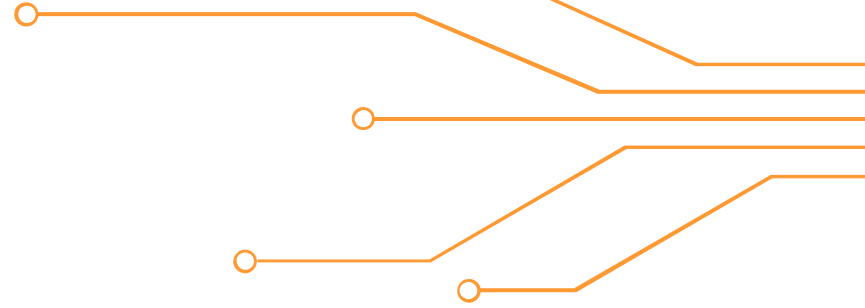
```
def square(x, y, length):
    t.up()
    t.goto(x, y)
    t.down()
```

length는 한번의 길이
펜을 든다.
(x, y)으로 이동한다.
펜을 내린다.

```
    for i in range(4):
        t.forward(length)
        t.left(90)
```

```
square(-200, 0, 100)
square(0, 0, 100)
square(200, 0, 100)
turtle.done()
```

square() 함수를 호출한다.

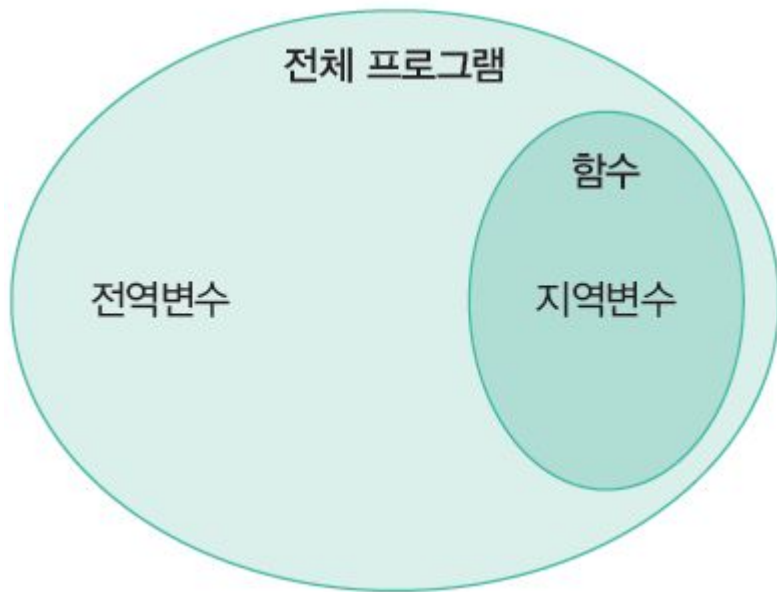




이론수업



- 지역 변수(local variable): 함수 안에서 선언되는 변수
- 전역 변수(global variable): 함수 외부에서 선언되는 변수



- 지역 변수는 함수 안에서만 사용이 가능하다.
 - 변수는 만들어진 공간(scope) 안에서만 유효
- 아래의 코드에서 지역 변수를 찾아보면..

```
def calculate_area (radius):  
    result = 3.14 * radius**2  
    return result  
  
r = float(input("원의 반지름: "))  
area = calculate_area(r)  
print(result)
```

```
def calculate_area (radius):  
    result = 3.14 * radius**2  
    return result  
  
r = float(input("원의 반지름: "))  
area = calculate_area(r)  
print(result)
```

```
원의 반지름: 10  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
NameError: name 'result' is not defined
```

변수는 자신이 만들어진 공간이
사라질때 함께 사라진다 .

함수 안에 만들어진 변수는
함수가 종료되면 사라진다 .



전역 변수 : 어디서나 사용할 수 있는 변수

아래의 코드에서 전역 변수를 찾아보면 ..

```
def calculate_area ():  
    result = 3.14 * r**2  
    return result  
  
r = float(input("원의 반지름: "))  
area = calculate_area()  
print(area)
```




```
def calculate_area (radius):  
    area = 3.14 * radius**2    # 전역변수 area에 계산값을 저장 (시도)  
    return  
  
area = 0  
r = float(input("원의 반지름: "))  
calculate_area(r)  
print(area)
```

원의 반지름: 10

0

>>>

왜 0 이 나올까??

```
def calculate_area (radius):  
    area = 3.14 * radius**2    # 전역변수 area에 계산값을 저장 (시도)  
    return  
  
area = 0  
r = float(input("원의 반지름: "))  
calculate_area(r)  
print(area)
```

```
원의 반지름: 10  
0  
>>>
```

새 지역변수 **area**

```
def calculate_area (radius):
```

```
    global area
```

```
    area = 3.14 * radius**2    # 전역변수 area에 계산값을 저장 (시도)
```

```
    return
```

```
area = 0
```

```
r = float(input("원의 반지름: "))
```

```
calculate_area(r)
```

```
print(area)
```

```
원의 반지름: 10
```

```
314.0
```

```
>>>
```

전역변수 **area** 를
사용한다고 알림



파이썬에서는 함수의 매개변수가 기본값을 가질 수 있다. 이것을 디폴트 인수(default argument)라고 한다.

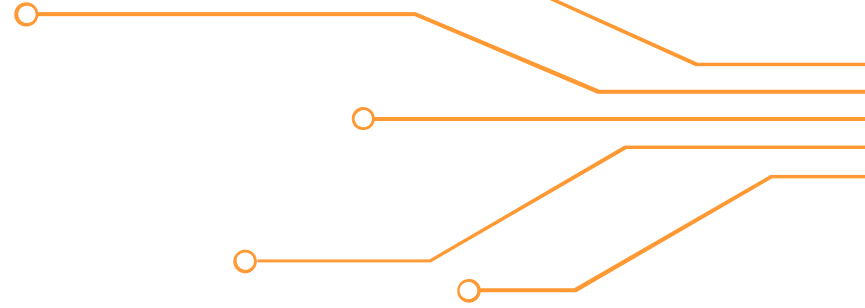
```
def greet(name, msg="별일없죠?"):
    print("안녕 " + name + ', ' + msg)

greet("영희")
greet("영희", "건강은 어때요?")
```

안녕 영희, 별일없죠?
>>>

안녕 영희, 건강은 어때요?
>>>





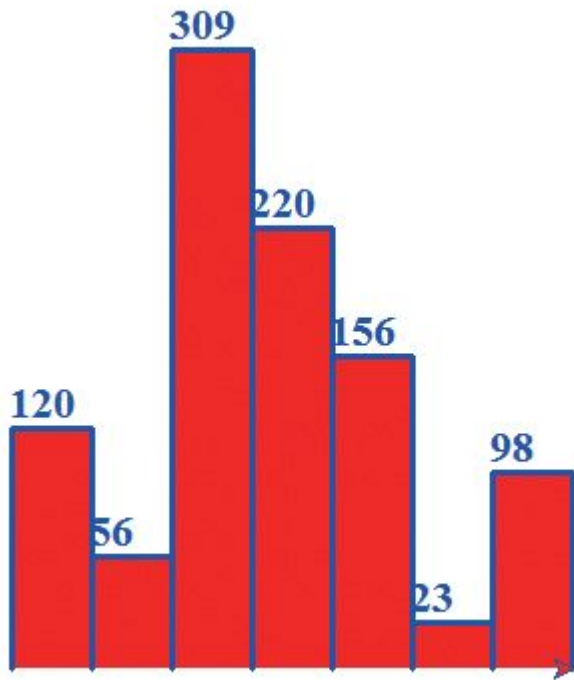


실습



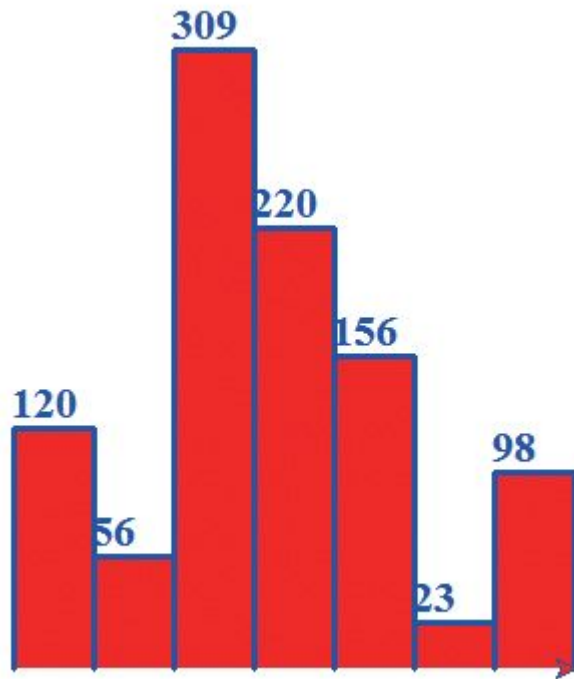
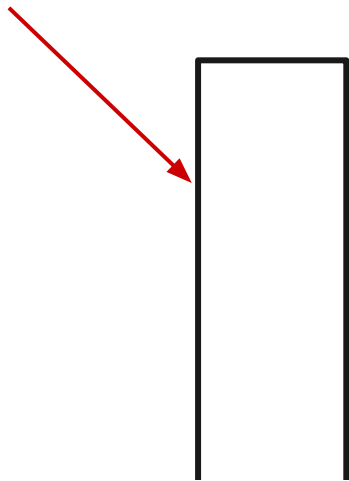
파이썬의 터틀 그래픽을 이용해서 막대 그래프를 그려봅시다.
다음 데이터로 막대 그래프를 그려보겠습니다.

```
data = [120, 56, 309, 220, 156, 23, 98]
```



```
data = [120, 56, 309, 220, 156, 23, 98]
```

1. 함수를 만듭니다. (매개변수 height)
 - a. 사각형을 그리는 함수 ($40 \times \text{height}$)
 - b. 왼쪽 90도 회전
 - c. 직진 height
 - d. 오른쪽 90도 회전
 - e. 직진 40
 - f. 오른쪽 90도 회전
 - g. 직진 height
 - h. 왼쪽 90도 회전




```
import turtle

def drawBar(height, width=40):
    t.begin_fill()
    t.left(90)
    t.forward(height)
    t.write(str(height),
            font = ('Times New Roman', 16, 'bold'))
    t.right(90)

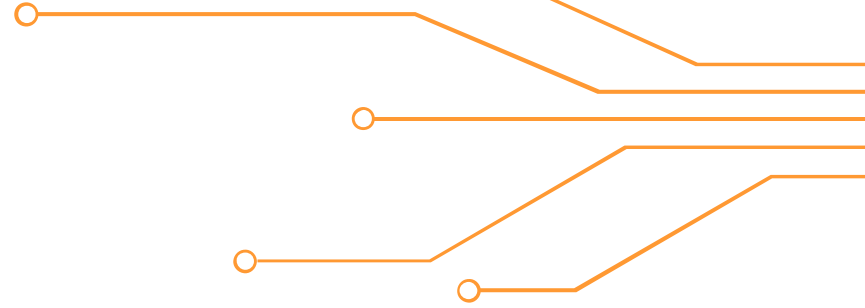
    t.forward(width)
    t.right(90)
    t.forward(height)
    t.left(90)
    t.end_fill()

data = [120, 56, 309, 220, 156, 23, 98]
```

```
t = turtle.Turtle()
t.color("blue")
t.fillcolor("red")
t.pensize(3)

for d in data:
    drawBar(d)

turtle.done()
```





퀴즈

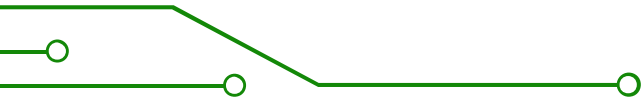


1. 다음 파이썬 코드의 출력 결과는?

```
def my_function(x, y):  
    return x + y  
  
print(my_function(5, 6))
```

- 1) 5
- 2) 6
- 3) 11
- 4) -1

30초



1. 다음 파이썬 코드의 출력 결과는? [3]

```
def my_function(x, y):  
    return x + y  
  
print(my_function(5, 6))
```

1) 5

2) 6

3) 11

4) -1



1. 다음 파이썬 코드의 출력 결과는? [3]

```
def my_function(x, y):  
    return x + y  
  
print(my_function(5, 6))
```

1) 5

2) 6

3) 11

4) -1

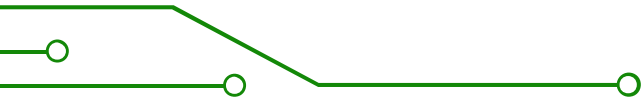
매개변수 x, y 를 통해 각각 5와 6이 들어가서 합이 return 되므로, 11이 정답



2. 파이썬 함수에 대한 다음 설명 중 잘못된 것을 고르시오.

- 1) 파이썬 함수는 파이썬 코드에서 자주 사용되는 코드를 한 곳에 모아 놓은 것이다.
- 2) 함수를 사용하면 코드를 더 간결하고 읽기 쉽게 만들 수 있고, 코드를 재사용할 수 있게 하여 코드의 유지 관리를 쉽게 만든다.
- 3) 파이썬 함수는 def 키워드를 사용하여 정의된다.
- 4) 함수 이름은 문자, 숫자, 언더스코어, 공백을 사용할 수 있다

30초



2. 파이썬 함수에 대한 다음 설명 중 잘못된 것을 고르시오. [4]

1) 파이썬 함수는 파이썬 코드에서 자주 사용되는 코드를 한 곳에 모아 놓은 것이다.

2) 함수를 사용하면 코드를 더 간결하고 읽기 쉽게 만들 수 있고, 코드를 재사용할 수 있게 하여 코드의 유지 관리를 쉽게 만든다.

3) 파이썬 함수는 def 키워드를 사용하여 정의된다.

4) 함수 이름은 문자, 숫자, 언더스코어, 공백을 사용할 수 있다



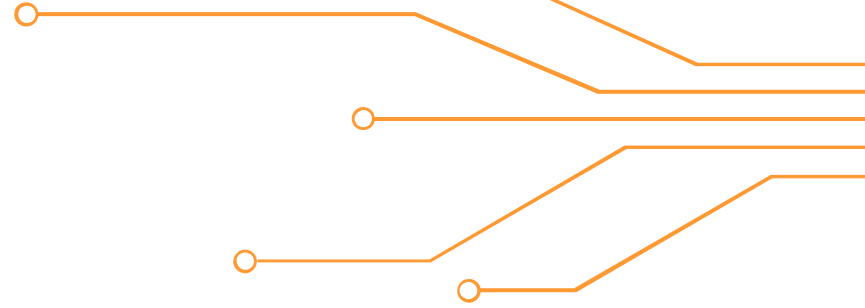
2. 파이썬 함수에 대한 다음 설명 중 잘못된 것을 고르시오. [4]

- 1) 파이썬 함수는 파이썬 코드에서 자주 사용되는 코드를 한 곳에 모아 놓은 것이다.
- 2) 함수를 사용하면 코드를 더 간결하고 읽기 쉽게 만들 수 있고, 코드를 재사용할 수 있게 하여 코드의 유지 관리를 쉽게 만든다.
- 3) 파이썬 함수는 def 키워드를 사용하여 정의된다.

4) 함수 이름은 문자, 숫자, 언더스코어, 공백을 사용할 수 있다

함수 이름에 공백은 들어갈 수 없습니다.







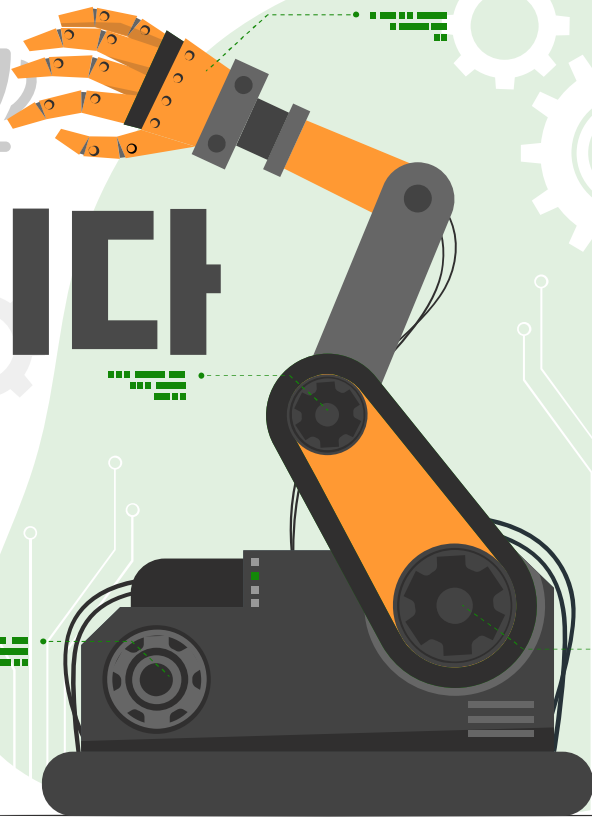
핵심정리



- ❑ 함수는 특별한 작업을 하는 코드를 하나로 모아서 이름을 붙인 것입니다.
- ❑ 파이썬에서 함수를 작성할 때는 `def` 키워드를 이용합니다.
- ❑ 작성된 함수를 호출하여 사용할 때는 함수의 이름과 함께 인수를 전달합니다. 예를 들어 `calculate(100)` 과 같습니다.
- ❑ 함수 호출시 함수 안으로 들어가는 데이터를 인수라고 하고 함수 호출이 끝난 다음 나오는 값을 반환값이라고 합니다.



수고하셨습니다



컴퓨팅 사고와 프로그래밍



CH 08.

프로젝트1



학습목표



1. 파이썬을 이용하여 게임을 만들어 봅니다.
2. 파이썬을 이용하여 애니메이션을 만들어 봅니다.

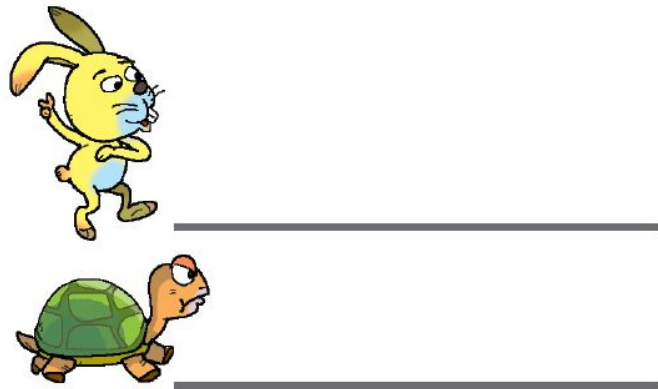




이론수업



https://github.com/MLBasic/CS_Programming 에서 /img 에 저장된 rabbit.gif 와 turtle.gif 를 다운로드



난수(random number)는 게임과 시뮬레이션에 필수적이다.

파이썬은 random 모듈을 통하여 난수 발생을 지원한다.

```
>>> import random
```

```
>>> random.random() # 0.0부터 1.0보다 작은 실수 난수  
0.8345121533431609
```

```
>>> random.randint(1, 100) # 1부터 100 사이의 정수 난수  
49
```



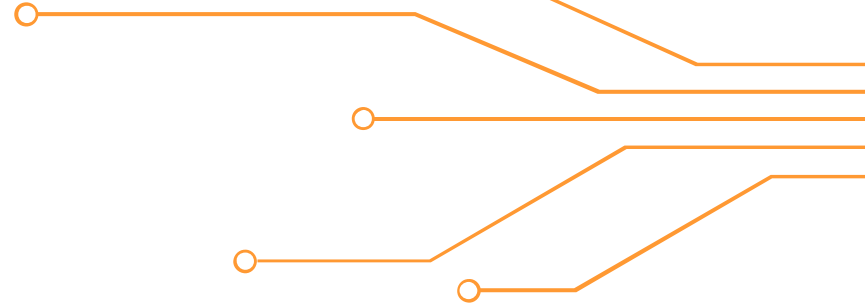
```
>>> import random  
>>> random.choice('abcdefghij') # 하나의 랜덤한 항목  
'c'
```

```
>>> items = [1, 2, 3, 4, 5, 6, 7]  
>>> random.shuffle(items)  
>>> items  
[7, 3, 2, 5, 6, 4, 1]
```



함수	인수	설명
forward()	픽셀값	거북이를 지정된 거리만큼 앞으로 이동한다.
backward()	픽셀값	거북이를 지정된 거리만큼 뒤로 이동한다.
right()	각도	거북이를 시계방향으로 회전시킨다.
left()	각도	거북이를 반시계방향으로 회전시킨다.
penup()	None	거북이의 펜을 올린다. 그림이 그려지지 않는다.
pendown()	None	거북이의 펜을 내린다. 그림이 그려진다.
up()	None	거북이의 펜을 올린다. 그림이 그려지지 않는다.
down()	None	거북이의 펜을 내린다. 그림이 그려진다.
color()	색상 이름	거북이 펜의 색상을 변경한다.

함수	인수	설명
fillcolor()	색상 이름	다각형을 채우는 색상을 변경한다.
heading()	None	현재의 방향을 반환한다.
position()	None	현재 위치를 반환한다.
goto()	x, y	거북이를 (x, y) 위치로 이동시킨다.
begin_fill()	None	채워진 다각형을 시작한다.
end_fill()	None	채워진 다각형을 닫는다.
dot()	None	현재 위치에 점을 남긴다.
stamp()	None	현재 위치에 거북이 모양을 남긴다.
shape()	모양 이름	거북이의 모양을 'arrow', 'classic', 'turtle', 'circle' 중의 하나로 변경한다.





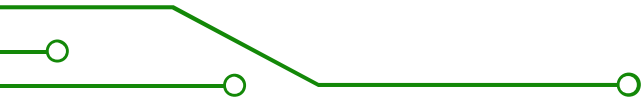
실습



거북이 2마리를 만들려면

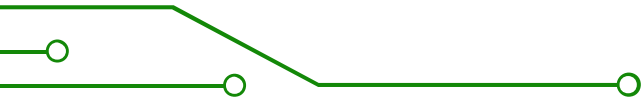
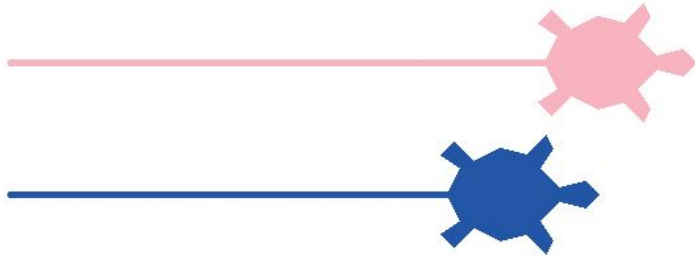
```
import turtle

t1 = turtle.Turtle() # 첫 번째 거북이
t2 = turtle.Turtle() # 두 번째 거북이
```



거북이들을 구별하기 위하여 색상을 다르게 하고 모양도 다르게

```
t1.color("pink")  
t1.shape("turtle")  
t1.shapesize(5)  
t1.pensize(5)  
  
t2.color("blue")  
t2.shape("turtle")  
t1.shapesize(5)  
t2.pensize(5)
```



출발점에 세우기

```
t1.penup()  
t1.goto(-300, 0)  
  
t2.penup()  
t2.goto(-300, -100)
```

$(-300,300)$



$(300,300)$



Y
axis

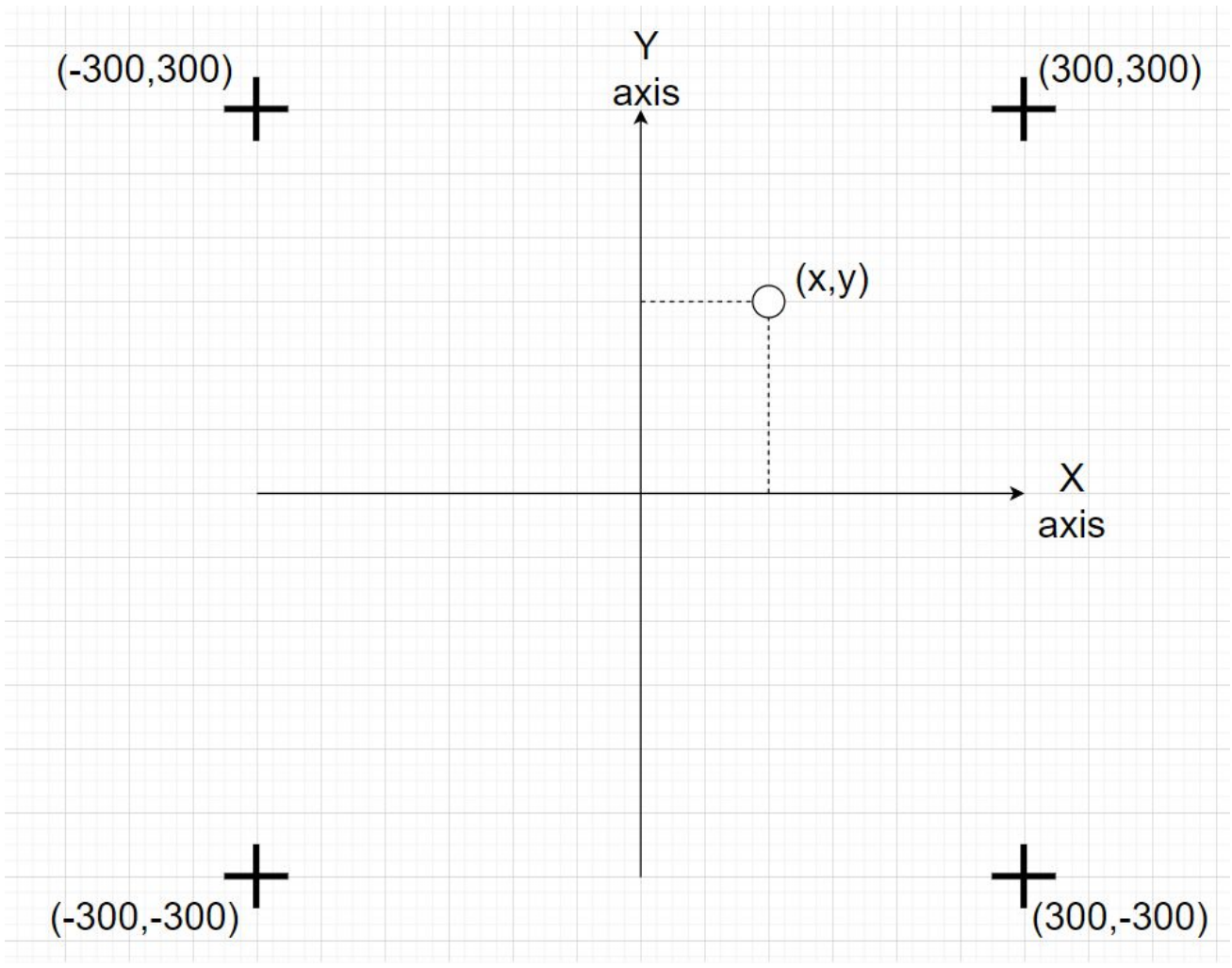
(x,y)

X
axis

$(-300,-300)$

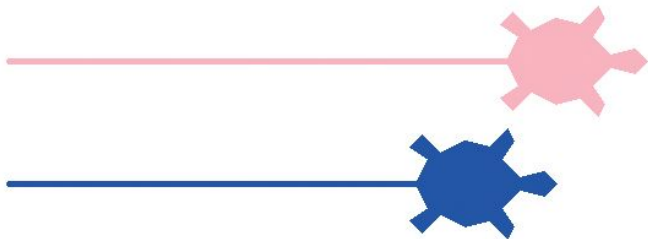


$(300,-300)$



100번 반복하면서 한 번 반복할 때마다 난수만큼 이동

```
for i in range(100):  
    d1 = random.randint(1, 60)    # 100번 반복한다.  
    t1.forward(d1)                # 1부터 60 사이의 난수를 발생한다.  
    d2 = random.randint(1, 60)    # 난수만큼 이동한다.  
    t2.forward(d2)                # 1부터 60 사이의 난수를 발생한다.  
                                # 난수만큼 이동한다.
```



```
image1 = "rabbit.gif"
image2 = "turtle.gif"

screen.addshape(image1)
screen.addshape(image2)

t1 = turtle.Turtle() # 첫 번째 거북이 생성
t1.shape(image1)

t2 = turtle.Turtle() # 두 번째 거북이 생성
t2.shape(image2)
```




```
import turtle          # 터틀 그래픽 모듈을 불러온다.
import random          # 난수 모듈을 불러온다.

screen = turtle.Screen()
image1 = "rabbit.gif"
image2 = "turtle.gif"
screen.addshape(image1)
screen.addshape(image2)

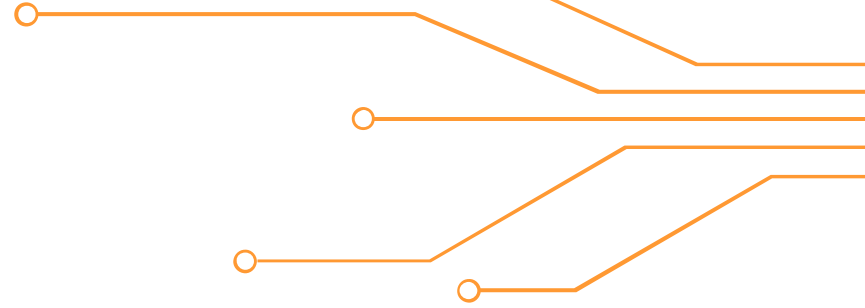
t1 = turtle.Turtle()   # 첫 번째 거북이를 생성한다.
t1.shape(image1)
t1.pensize(5)          # 팬의 두께를 5로 한다.
t1.penup()             # 펜을 든다.
t1.goto(-300, 0)       # (-300, 0) 위치로 간다.

t2 = turtle.Turtle()   # 두 번째 거북이를 생성한다.
t2.shape(image2)
t2.pensize(5)          # 팬의 두께를 5로 한다.
t2.penup()             # 펜을 든다.
t2.goto(-300, -200)    # (-300, -200) 위치로 간다.

t1.pendown()           # 첫 번째 거북이의 펜을 내린다.
t2.pendown()           # 첫 번째 거북이의 펜을 내린다.
t1.speed(1)
t2.speed(1)
```

```
for i in range(100):
    d1 = random.randint(1, 60)
    t1.forward(d1)
    d2 = random.randint(1, 60)
    t2.forward(d2)
```

race.py



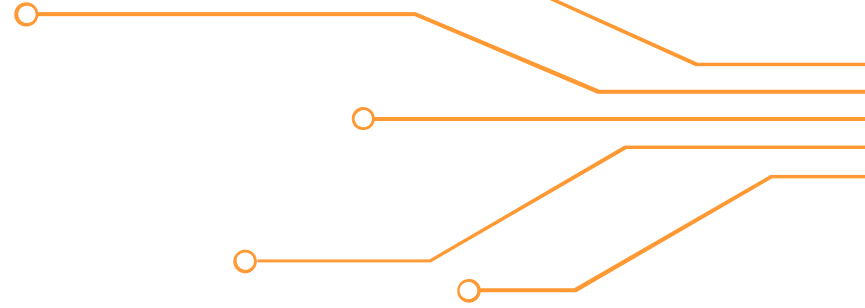


이론수업



애니메이션도 쉽게 제작할 수 있다. 다음과 같은 애니메이션을 작성해보자. 여러분은 이 애니메이션을 자신의 홈페이지에서 사용하거나 동영상 제작에도 사용할 수 있다.







실습

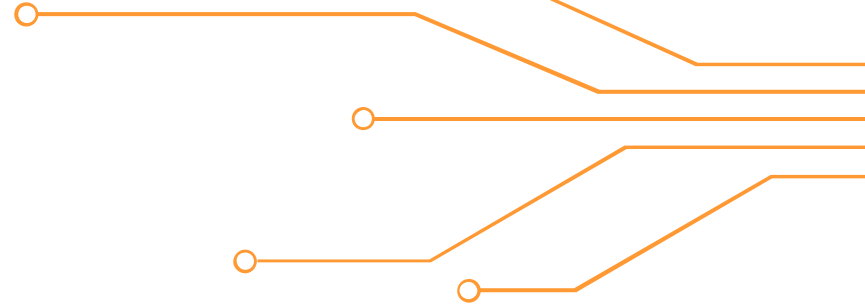


```
import turtle
import random
t = turtle.Turtle()

t.speed(0)
t.pensize(5)
t.goto(0,0)

while True:
    for i in range(30):
        t.circle(1+5*i)
        t.color((random.random(),random.random(),random.random()))
        t.goto(i*20, 0)
    t.clear()

turtle.done()
```

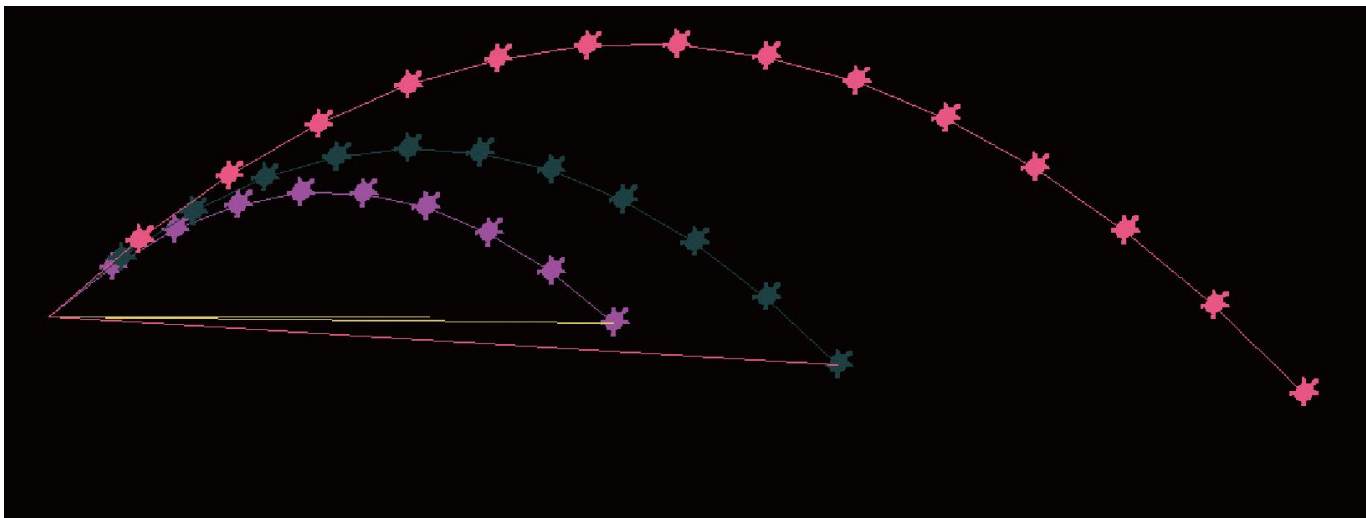




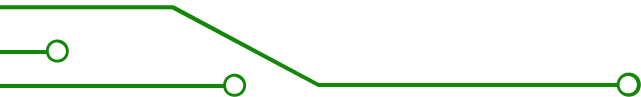
이론수업



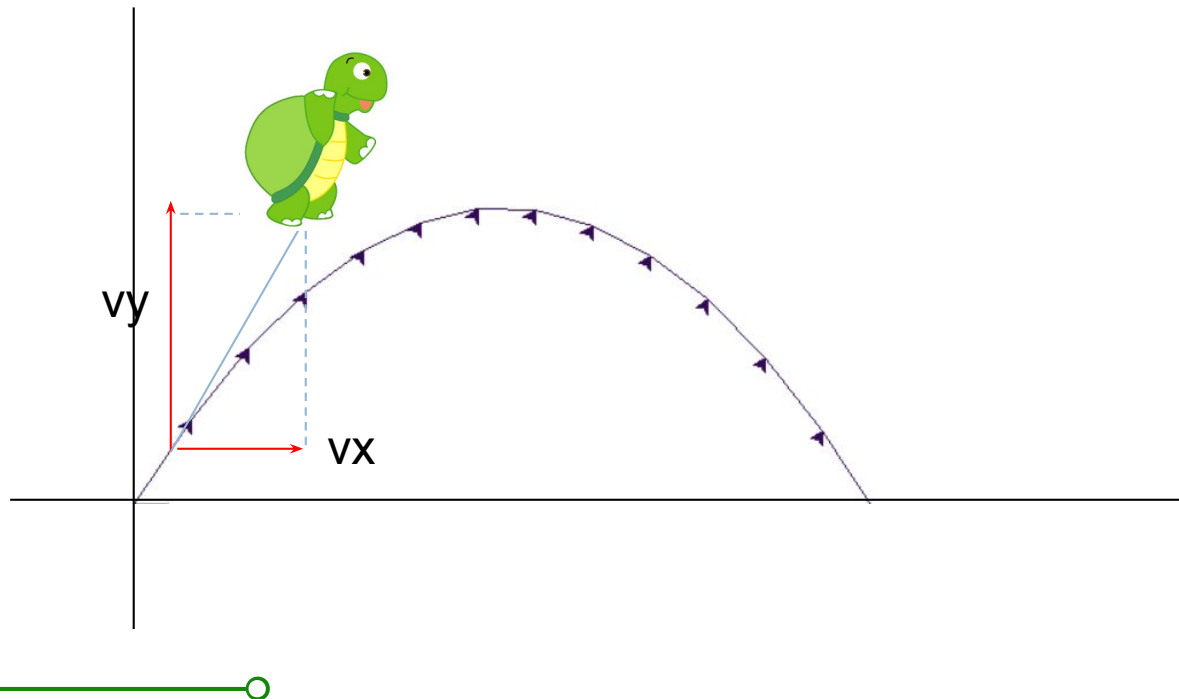
앵그리 버드와 유사한 게임 “앵그리 터틀”을 제작하여 보자. 공을 발사하면 초기 속도와 초기 각도에 의하여 비행하다가 목표물에 맞추는 게임이다. 우선 포물선을 그리며 날아가는 모습을 구현해 보자



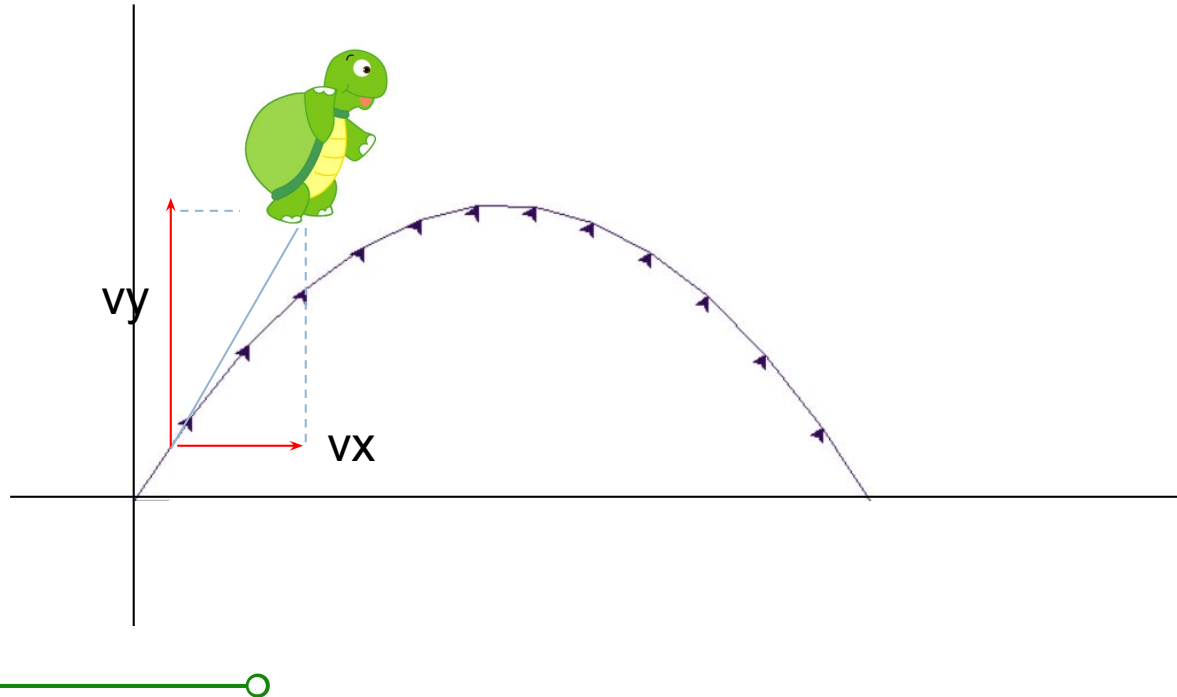
- `int v;` - 거북이의 속도이다.
- `int vx;` - 거북이의 x 방향 속도이다.
- `int vy;` - 거북이의 y 방향 속도이다.
- `int x;` - 거북이의 현재 x좌표이다.
- `int y;` - 거북이의 현재 y좌표이다.



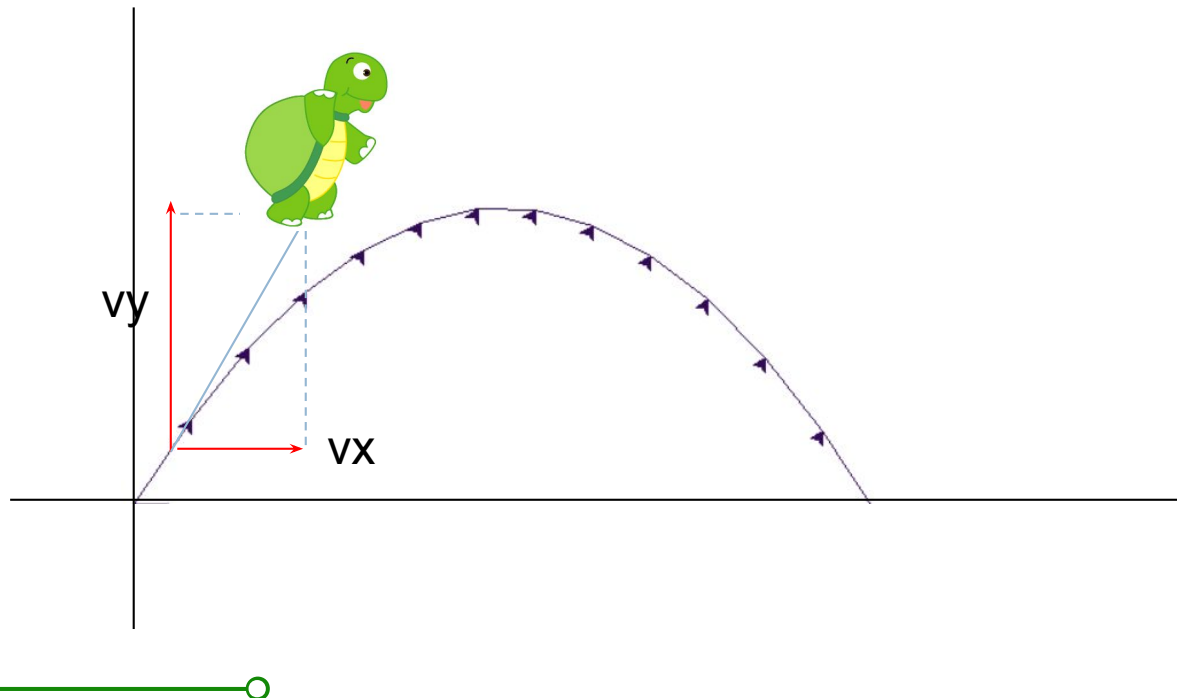
- v_x ; 초기 속도에서 변하지 않는다.
- v_y ; 초기 속도에서 중력 가속도 만큼 점점 느려진다.

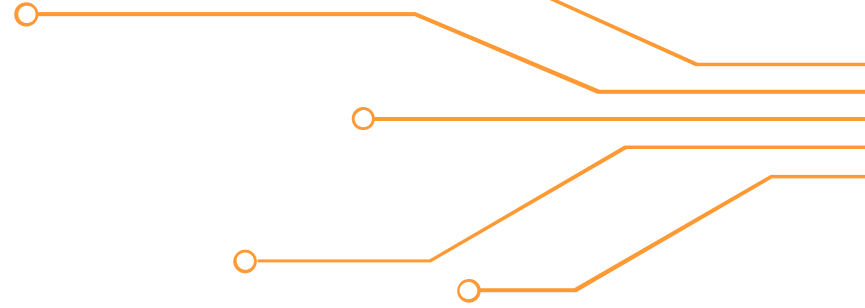


- $VX = VX$
- $vy = vy - 10$ (중력가속도를 10으로 가정)



- $v_x = \text{velocity} * \text{math.cos}(\text{angle} * 3.14 / 180.0)$
- $v_y = \text{velocity} * \text{math.sin}(\text{angle} * 3.14 / 180.0)$







실습




```
import turtle
import math
import random

player = turtle.Turtle()
player.shape("turtle")
screen = player.getscreen()
screen.bgcolor("black")      # 화면 배경을 검정색으로 한다.
screen.setup(800, 600)       # 화면의 크기를 800x600으로 한다.
player.color("yellow")       # 색상은 파랑색으로 하자.

player.goto(-300, 0)
velocity = 70                # 초기속도 70픽셀/sec
player.left(45)
```



```
def turnleft():  
    player.left(5)           # 왼쪽으로 5도 회전  
  
def turnright():  
    player.right(5)         # 오른쪽으로 5도 회전  
  
def turnup():  
    global velocity  
    velocity += 10  
  
def turndown():  
    global velocity  
    velocity -= 10
```

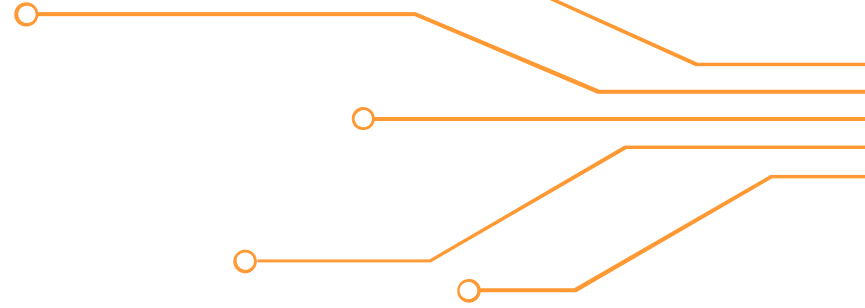


```
def fire():
    x = -300
    y = 0
    player.color(random.random(),random.random(),random.random())
    player.goto(x, y)
    angle = player.heading()          # 초기각도
    vx = velocity * math.cos(angle * 3.14 / 180.0) # 도 -> 라디안
    vy = velocity * math.sin(angle * 3.14 / 180.0) # 도 -> 라디안
    while player.ycor() >= 0 :          # y좌표가 음수가 될때까지
        vx = vx
        vy = vy - 10
        x = x + vx
        y = y + vy
        player.goto(x, y)
        player.stamp()
```

```
screen.onkeypress(turnleft, "Left")  
screen.onkeypress(turnright, "Right")  
screen.onkeypress(turnup, "Up")  
screen.onkeypress(turndown, "Down")  
screen.onkeypress(fire, "space")
```

```
screen.listen()  
turtle.mainloop()
```







퀴즈



1. 파이썬의 난수 발생에 대한 다음 설명 중 잘못된 것은?

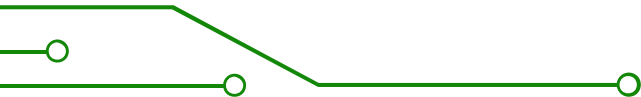
1) 파이썬의 `random.random()` 함수는 0과 1 사이의 임의의 실수값을 반환합니다.

2) `random.random()` 함수를 사용하려면 먼저 `random` 모듈을 import해야 합니다.

3) `random.random()` 함수를 사용하여 게임, 시뮬레이션, 기타 다양한 응용 프로그램에서 난수를 생성할 수 있습니다.

4) 1에서 100 사이의 정수 난수를 사용할 때 `random.random(1,100)` 을 사용합니다.

30초



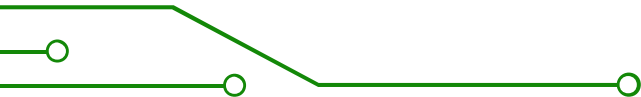
1. 파이썬의 난수 발생에 대한 다음 설명 중 잘못된 것은? [4]

1) 파이썬의 `random.random()` 함수는 0과 1 사이의 임의의 실수값을 반환합니다.

2) `random.random()` 함수를 사용하려면 먼저 `random` 모듈을 import해야 합니다.

3) `random.random()` 함수를 사용하여 게임, 시뮬레이션, 기타 다양한 응용 프로그램에서 난수를 생성할 수 있습니다.

4) 1에서 100 사이의 정수 난수를 사용할 때 `random.random(1,100)` 을 사용합니다.



1. 파이썬의 난수 발생에 대한 다음 설명 중 잘못된 것은? [4]

1) 파이썬의 `random.random()` 함수는 0과 1 사이의 임의의 실수값을 반환합니다.

2) `random.random()` 함수를 사용하려면 먼저 `random` 모듈을 import해야 합니다.

3) `random.random()` 함수를 사용하여 게임, 시뮬레이션, 기타 다양한 응용 프로그램에서 난수를 생성할 수 있습니다.

4) 1에서 100 사이의 정수 난수를 사용할 때 `random.random(1,100)` 을 사용합니다.

1에서 100사이의 정수 난수를 만들때는 `random.randint(1,100)` 을 사용합니다.



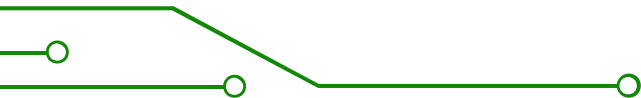
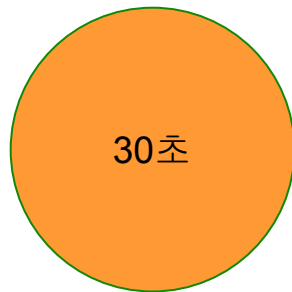
2. 다음 터틀 그래픽 함수에 대한 설명 중 옳은 것을 모두 고르세요.

1) `forward(100)` 함수는 거북이를 100 픽셀만큼 앞으로 이동시킵니다.

2) `left(45)` 함수는 거북이를 왼쪽으로 45도 만큼 회전시킵니다.

3) `shape()` 함수는 거북이의 모양을 바꾸어준다. 'arrow',
'classic','turtle','circle' 등을 인수로 넣어준다.

4) `penup()` 거북이의 펜을 내려 그림이 그려진다.



2. 다음 터틀 그래픽 함수에 대한 설명 중 옳은 것을 모두 고르세요. [1, 2, 3]

1) `forward(100)` 함수는 거북이를 100 픽셀만큼 앞으로 이동시킵니다.

2) `left(45)` 함수는 거북이를 왼쪽으로 45도 만큼 회전시킵니다.

3) `shape()` 함수는 거북이의 모양을 바꾸어준다. 'arrow', 'classic', 'turtle', 'circle' 등을 인수로 넣어준다.

4) `penup()` 거북이의 펜을 내려 그림이 그려진다.



2. 다음 터틀 그래픽 함수에 대한 설명 중 옳은 것을 모두 고르세요. [1, 2, 3]

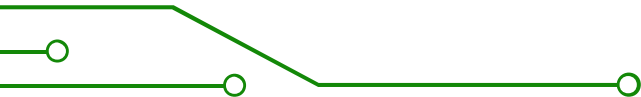
1) `forward(100)` 함수는 거북이를 100 픽셀만큼 앞으로 이동시킵니다.

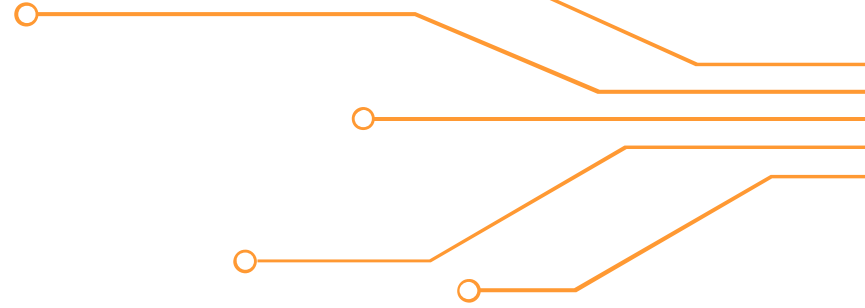
2) `left(45)` 함수는 거북이를 왼쪽으로 45도 만큼 회전시킵니다.

3) `shape()` 함수는 거북이의 모양을 바꾸어준다. 'arrow', 'classic', 'turtle', 'circle' 등을 인수로 넣어준다.

4) `penup()` 거북이의 펜을 내려 그림이 그려진다.

`penup()` 은 펜을 올려 그림이 그려지지 않게 하는 함수입니다.







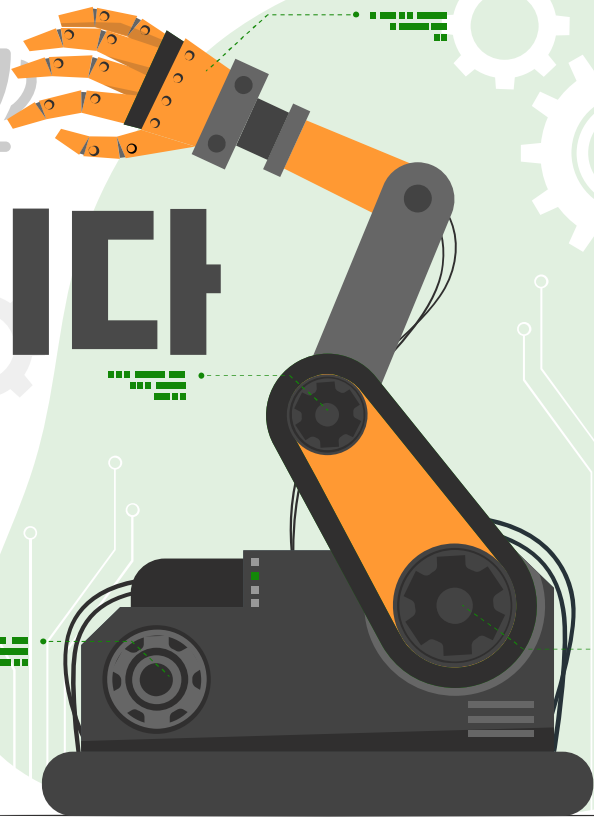
핵심정리



- ❑ 난수는 게임과 시뮬레이션에 필수적입니다. 파이썬은 random 모듈을 사용하여 난수를 만들 수 있습니다.
- ❑ 터틀그래픽에 있는 다양한 기능을 사용하면 간단한 게임을 만들 수 있습니다.
- ❑ 애니메이션을 만들기 위해 터틀그래픽을 사용할 수 있습니다.
- ❑ 애니메이션, 게임을 만들기 위해서는 단순한 프로그래밍 문법만이 아닌 수학과 물리학에 대한 지식이 필요합니다.



수고하셨습니다



컴퓨팅 사고와 프로그래밍



CH 09.

리스트와 딕셔너리1



학습목표



1. 리스트가 무엇인지 설명할 수 있다.
2. 리스트의 여러 연산 중 중요한 연산들을 이해하고 사용할 수 있다.



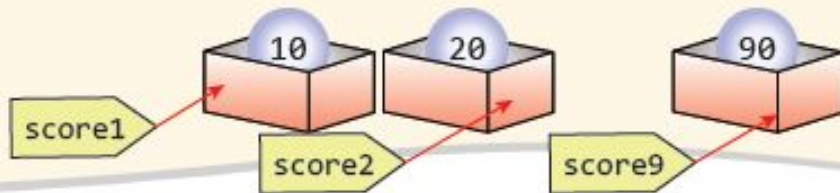


이론수업

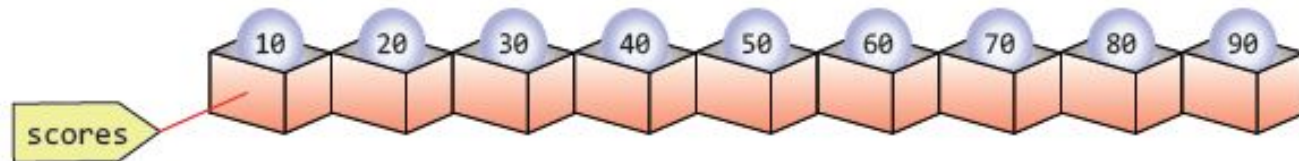


여러 개의 데이터를 하나로 묶어서 저장하는 것이 필요

```
score1 = 10  
score2 = 20  
...  
score9 = 90
```



```
scores = [ 10, 20, 30, 40, 50, 60, 70, 80, 90 ]
```





단독 주택은 서로 떨어져 있어서
아무래도 전달하기 불편하다.



아파트는 입주민들이 모여
있어서 전달하기 쉽다.

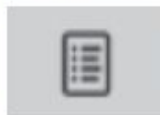
데이터 구조

리스트



```
myList = [ 10, 20, 30, 40, 50 ]
```

튜플



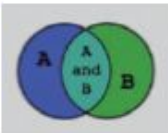
```
myTuple = ( 10, 20, 30, 40, 50 )
```

딕셔너리



```
myDict = { 1:"one", 2:"two", 3:"three" }
```

세트



```
mySet = { "one", "two", "three" }
```

fruits = ["사과", "수박", "참외", "바나나"]

리스트를 변수에 저장한다.

리스트의 참조값을 변수에 저장한다.

대괄호로 데이터를 묶는다.

```
>>> friends = [ "게이츠", "잡스", "브린", "베이조스" ]  
>>> friends  
['게이츠', '잡스', '브린', '베이조스']
```



```
>>> letters = ['A', 'B', 'C', 'D', 'E', 'F']
```

```
>>> letters[0]
```

```
A
```

```
>>> letters[1]
```

```
B
```

```
>>> letters[2]
```

```
C
```

리스트
인덱스

'A'	'B'	'C'	'D'	'E'	'F'
-----	-----	-----	-----	-----	-----

0

1

2

3

4

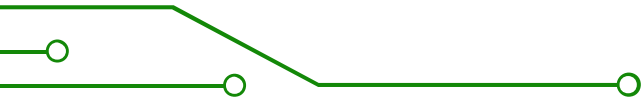
5



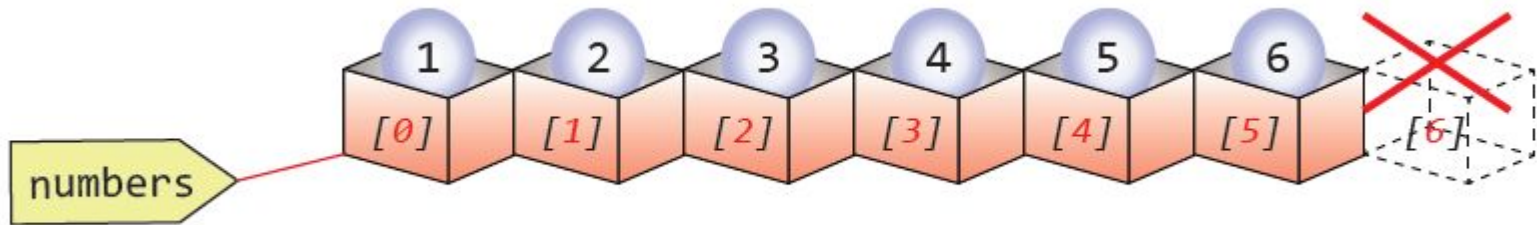
len() 함수를 이용

```
>>> numbers = [ 1, 2, 3, 4, 5, 6 ]  
>>> len(numbers)  
6
```

```
>>> s = "Hello World!"  
>>> len(s)  
12
```



리스트의 길이보다 더 큰 인덱스를 사용하면 오류가 발생한다



```
>>> numbers = [ 1, 2, 3, 4, 5, 6 ]  
>>> numbers[6]  
Traceback (most recent call last):  
File "<pyshell#14>", line 1, in <module>  
numbers[6]  
IndexError: list index out of range
```

어떤 값이 리스트 내부에 있는 지를 확인

```
>>> numbers = [ 1, 2, 3, 4, 5, 6 ]
>>> 6 in numbers
True
>>> 10 in numbers
False
>>> 10 not in numbers
True
>>> 'e' in 'Hello'
True
>>> 'z' in 'Hello'
False
```



리스트를 이용한 가장 일반적인 작업은 리스트에 저장된 데이터를 하나씩 꺼내서 어떤 처리를 하는 것

```
myList = [ "우유", "사과", "두부", "소고기"]  
  
for item in myList :  
    print(item)
```

우유
사과
두부
소고기

인덱스를 이용하여 항목을 꺼낼 수도 있다.

```
myList = [ "우유", "사과", "두부", "소고기"]  
  
for i in range(len(myList)) :  
    print(myList[i])
```



인덱스를 사용하여 항목들을 수정, 교체할 수 있다.

```
myList = [ "우유", "사과", "두부", "소고기"]  
myList[1] = '커피'  
  
print(myList)
```

```
['우유', '커피', '두부', '소고기']
```




```
myList = [ ]  
myList.append("우유")  
myList.append("사과")  
myList.append("두부")  
myList.append("소고기")  
print(myList)
```

```
['우유', '사과', '두부', '소고기']
```



```
myList = [ "우유", "사과", "두부", "소고기"]  
myList.insert(1, '커피')  
print(myList)
```

```
['우유', '커피', '사과', '두부', '소고기']
```



파이썬에서 모든 것은 객체(object)이다. 객체는 관련되는 변수와 함수를 묶은 것이다.

파이썬에서 리스트도 객체이다. 객체 안에 있는 무엇인가를 사용할 때는 객체의 이름을 쓰고 점(.)을 붙인 후에 함수의 이름을 적는다.

`myList.append("커피")`

객체

안에 있는

메소드

remove()와 pop()을 사용하여 삭제

```
myList = [ "우유", "사과", "두부", "소고기"]  
myList.remove("소고기")  
print(myList)
```

```
['우유', '사과', '두부']
```



지우고자 하는 항목이 없으면 오류 발생

```
myList = [ "우유", "사과", "두부", "소고기"]  
if "소고기" in myList:  
    myList.remove("소고기")  
print(myList)
```

```
['우유', '사과', '두부']
```



pop(index)는 특정 인덱스에 있는 항목을 삭제하고 항목을 반환

```
myList = [ "우유", "사과", "두부", "소고기"]  
item = myList.pop(0)  
print(myList)
```

```
['사과', '두부', '소고기']
```

```
item = myList.pop()  
print(myList)
```

```
['사과', '두부']
```

리스트에서 특정한 항목 검색

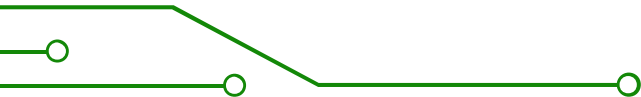
```
myList = [ "우유", "사과", "두부", "소고기"]  
if "소고기" in myList:  
    i = myList.index("소고기")          # i는 3
```



```
>>> myList = [ "우유", "사과"]  
>>> yourList = [ "두부", "소고기"]  
>>> myList+yourList  
['우유', '사과', '두부', '소고기']
```

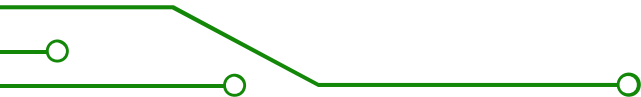



```
>>> myList = [ "우유", "사과" ]  
>>> myList*2  
['우유', '사과', '우유', '사과']
```

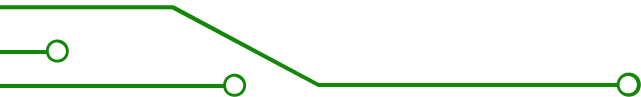


슬라이싱(slicing)은 리스트에서 한 번에 여러 개의 항목을 추출하는 기법

```
>>> letters = ['A', 'B', 'C', 'D', 'E', 'F']  
>>> print(letters[0:3])  
['A', 'B', 'C']
```

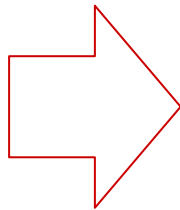


```
>>> letters = ['A', 'B', 'C', 'D', 'E', 'F']
>>> print(letters[:3])
['A', 'B', 'C']
>>> print(letters[3:])
['D', 'E', 'F']
>>> print(letters[:])    # 리스트 복사시 사용
['A', 'B', 'C', 'D', 'E', 'F']
```



함수	설명
min()	리스트에서 최소값을 찾는다.
max()	리스트에서 최대값을 찾는다.
sum()	리스트 안의 값들의 합계를 반환한다.

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]
print(min(numbers))
print(max(numbers))
print(sum(numbers))
```



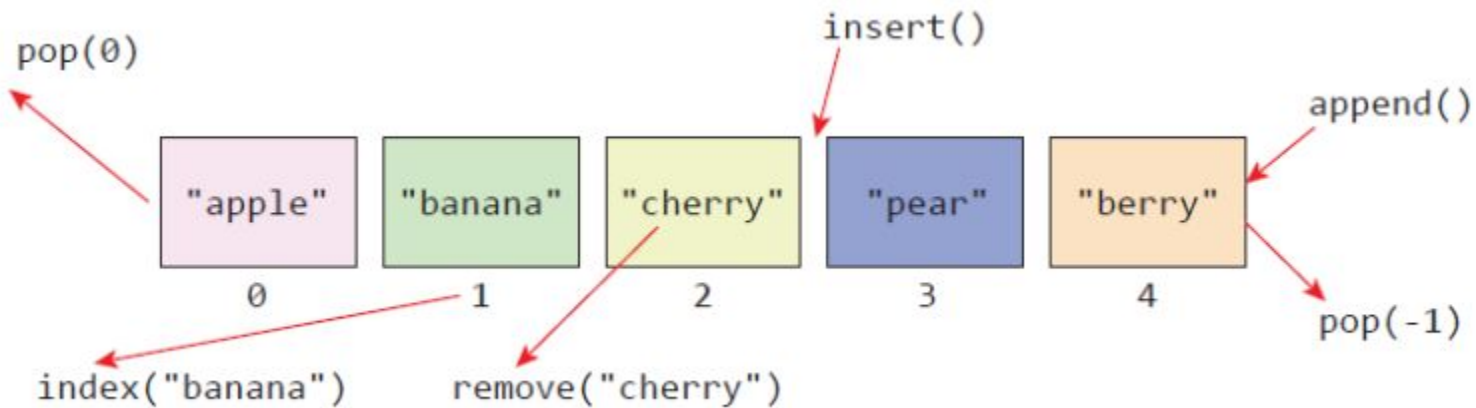
1
9
45



```
>>> min("abcdefghijklmnopqrstuvwxyz")  
'a'
```

```
>>> min(["dog", "cat", "tiger"])  
'cat'
```

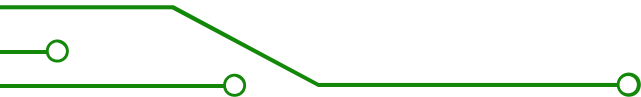




연산의 예	설명
<code>myList[2]</code>	인덱스 2에 있는 요소
<code>myList[2] = 3</code>	인덱스 2에 있는 요소를 3으로 설정한다.
<code>myList.pop(2)</code>	인덱스 2에 있는 요소를 삭제한다.
<code>len(myList)</code>	<code>myList</code> 의 길이를 반환한다.
<code>"value" in myList</code>	<code>"value"</code> 가 <code>myList</code> 에 있으면 <code>True</code>
<code>"value" not in myList</code>	<code>"value"</code> <code>myList</code> 에 없으면 <code>True</code>
<code>myList.sort()</code>	<code>myList</code> 를 정렬한다.
<code>myList.index("value")</code>	<code>"value"</code> 가 발견된 위치를 반환한다.
<code>myList.append("value")</code>	리스트의 끝에 <code>"value"</code> 요소를 추가한다.
<code>myList.remove("value")</code>	<code>myList</code> 에서 <code>"value"</code> 가 나타나는 위치를 찾아서 삭제한다.

sort()

```
numbers = [ 9, 6, 7, 1, 8, 4, 5, 3, 2 ]  
numbers.sort()  
print(numbers)
```



sorted()

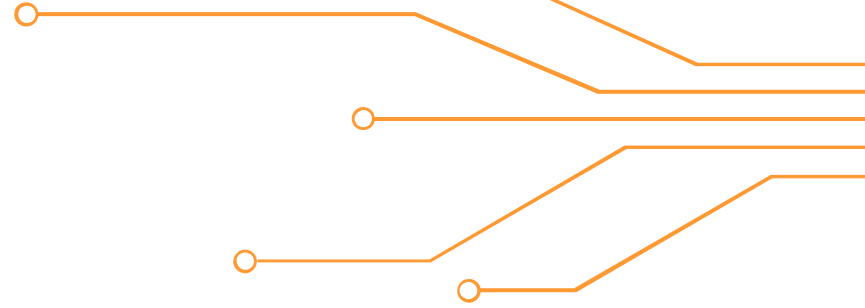
```
numbers = [ 9, 6, 7, 1, 8, 4, 5, 3, 2 ]  
new_list = sorted(numbers)  
print(new_list)
```



sorted()

```
numbers = [ 9, 6, 7, 1, 8, 4, 5, 3, 2 ]  
new_list = sorted(numbers, reverse=True)  
print(new_list)
```





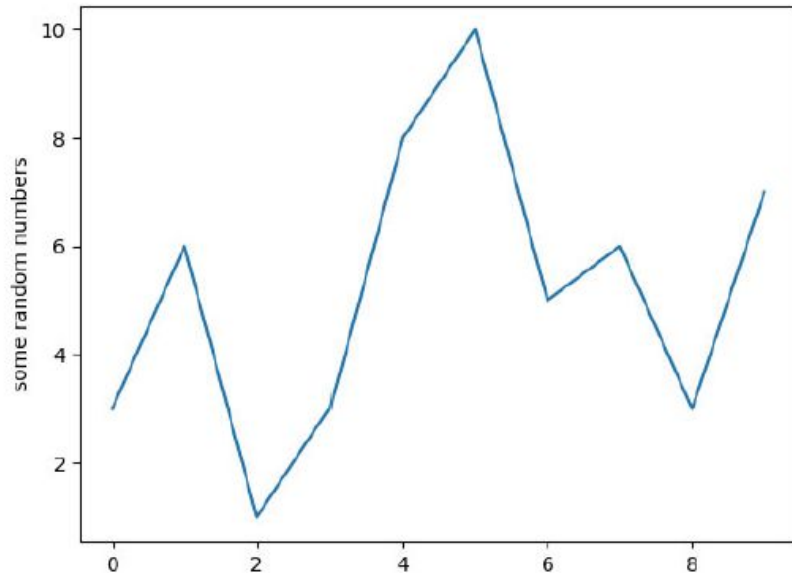


실습



데이터 과학에서는 데이터를 시각화하는 것도 매우 중요하다. 시각화는 단순히 원시 데이터를 보는 것 이상의 데이터를 이해하는 강력한 방법을 제공한다. 우리는 리스트를 10개의 난수로 채우고 이 데이터를 그래프로 시각화하여 보자.

```
C> pip install matplotlib
```



```
import matplotlib.pyplot as plt  
  
plt.plot(numbers)  
  
plt.show()
```

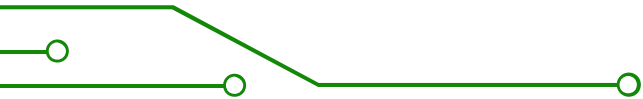


```
import matplotlib.pyplot as plt
import random

numbers = []

for i in range(10):      # 난수로 리스트를 채운다.
    numbers.append(random.randint(1, 10))

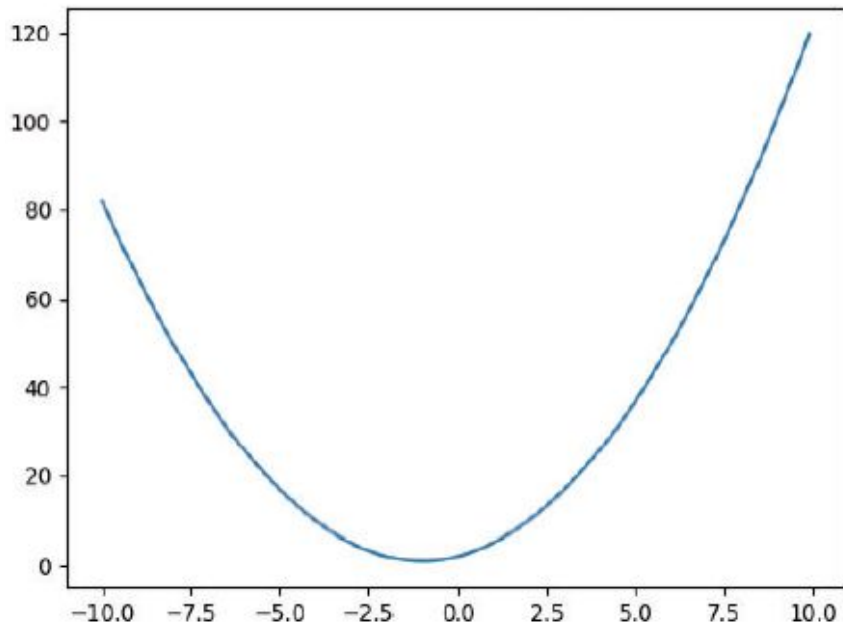
plt.plot(numbers)        # 리스트를 선그래프로 그린다.
plt.ylabel('some random numbers')  # 레이블을 붙인다.
plt.show()
```



사용자로부터 2차 함수의 계수를 입력받아서 2차 함수를 그려보자

$$y = ax^2 + bx + c$$

a: 1
b: 2
c: 2



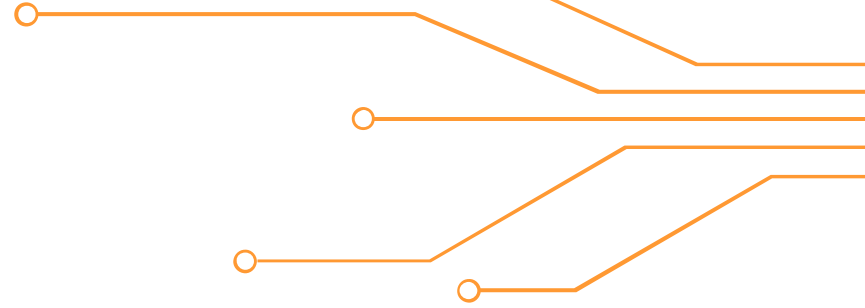

```
import matplotlib.pyplot as plt
xlist = []

for i in range(-100, 100): # -10.0에서 10.0까지의 실수 200개를 만든다.
    xlist.append(i/10.0)

a = int(input("a : "))
b = int(input("b : "))
c = int(input("c : "))

ylist = []
for i in xlist:
    ylist.append(a*i**2 + b*i + c)

plt.plot(xlist, ylist)
plt.show()
```





퀴즈



1. 파이썬 리스트를 생성하는 방법은?

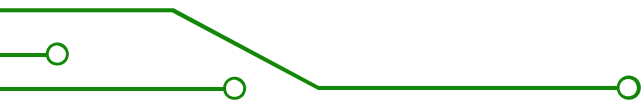
1) {1, 2, 3}

2) (1, 2, 3)

3) [1, 2, 3]

4) {1:'one', 2:'two', 3:'three'}

30초



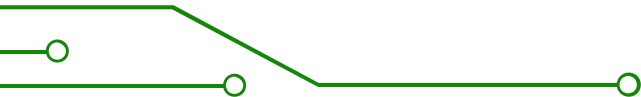
1. 파이썬 리스트를 생성하는 방법은? [3]

1) {1, 2, 3}

2) (1, 2, 3)

3) [1, 2, 3]

4) {1:'one', 2:'two', 3:'three'}



1. 파이썬 리스트를 생성하는 방법은? [3]

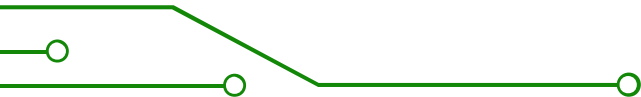
1) {1, 2, 3}

2) (1, 2, 3)

3) [1, 2, 3]

리스트는 대괄호 [] 로 묶입니다. 중괄호는 세트와 딕셔너리, 소괄호는 튜플에 사용됩니다.

4) {1:'one', 2:'two', 3:'three'}

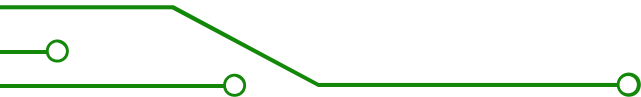


2. 파이썬 리스트 `a` 의 끝에 요소 `40` 을 추가하는 바른 방법을 고르시오.

`a = [10, 20, 30]` `=== ? ==> [10, 20, 30, 40]`

- 1) `a.list(40)`
- 2) `a.list + [40]`
- 3) `a.insert(0,40)`
- 4) `a.append(40)`

30초



2. 파이썬 리스트 a 의 끝에 요소 40 을 추가하는 바른 방법을 고르시오. [4]

a = [10, 20, 30] === ? ==> [10, 20, 30, 40]

- 1) a.list(40)
- 2) a.list + [40]
- 3) a.insert(0,40)
- 4) **a.append(40)**



2. 파이썬 리스트 `a` 의 끝에 요소 `40` 을 추가하는 바른 방법을 고르시오. [4]

`a = [10, 20, 30]` `=== ? ==> [10, 20, 30, 40]`

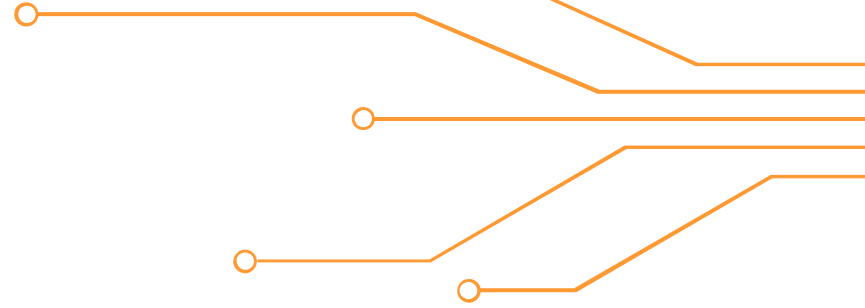
- 1) `a.list(40)`
- 2) `a.list + [40]`
- 3) `a.insert(0,40)`
- 4) **`a.append(40)`**

1), 2) 는 문법오류가 나는 잘못된 문장이다.

3) 은 인덱스 `0` 위치에 `40` 이 삽입되는 명령으로

`a = [40, 10, 20, 30]` 이 된다.







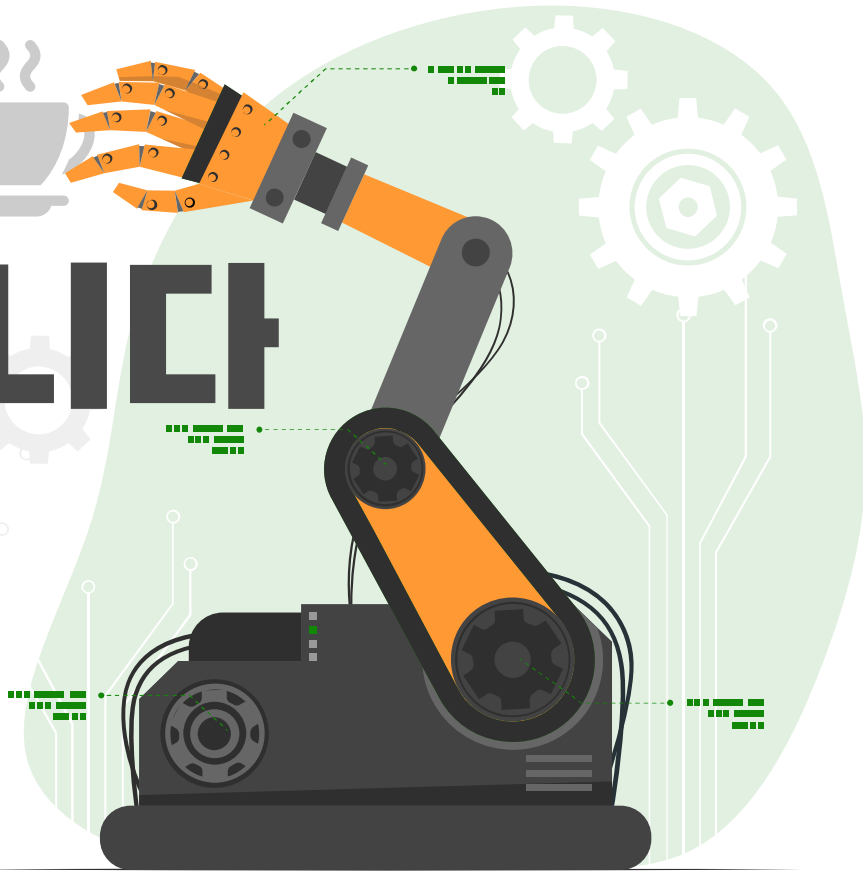
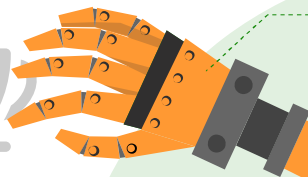
핵심정리



- ❑ 리스트는 여러 가지 값을 저장하기 위하여 만들어진 자료형입니다. 수치값이나 문자열, 객체 등을 저장할 수 있다.
- ❑ 리스트에서 항목을 꺼낼 때는 인덱스를 사용합니다.
- ❑ 리스트는 `append()`를 사용하여 리스트의 끝에 요소를 추가하고 `pop()`을 이용하여 항목을 삭제할 수 있습니다.



수고하셨습니다



컴퓨팅 사고와 프로그래밍



CH 10.

리스트와 딕셔너리2



학습목표



1. 딕셔너리가 무엇인지 설명할 수 있다.
2. 딕셔너리의 여러 연산 중 중요한 연산들을 이해하고 사용할 수 있다.
3. 튜플과 세트가 무엇인지 설명할 수 있다.





이론수업



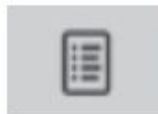
데이터 구조

리스트



```
myList = [ 10, 20, 30, 40, 50 ]
```

튜플



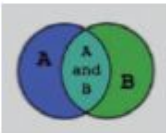
```
myTuple = ( 10, 20, 30, 40, 50 )
```

딕셔너리



```
myDict = { 1:"one", 2:"two", 3:"three" }
```

셋



```
mySet = { "one", "two", "three" }
```

`myTuple = (10, 20, 30, 40, 50)`

튜플(tuple)은 리스트와 비슷한 자료형이지만, 요소가 변경될 수 없음

튜플은 쉼표(,)로 구분된 값들의 모음으로 소괄호로 묶어서 표현

튜플은 리스트보다 효율적(더 적은 메모리 사용)

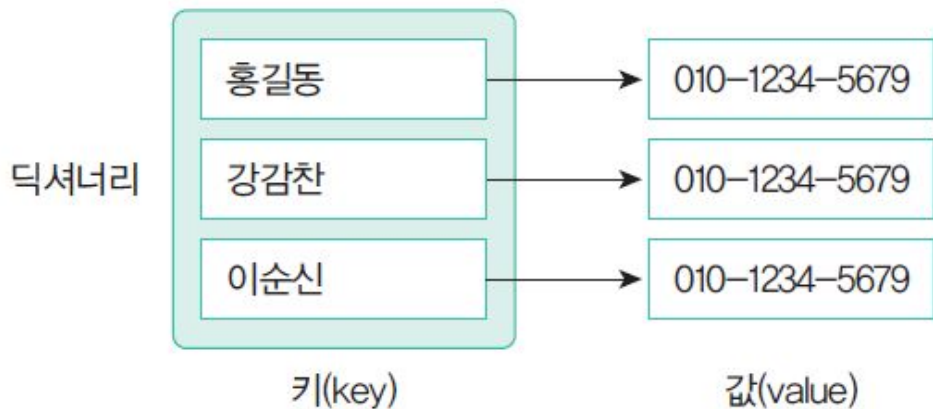
- 순서가 있다. 인덱스 사용
- 중복된 값을 가질 수 있다.
- 변경할 수 없다. 튜플의 요소는 한번 만들어지면 삭제, 수정이 불가
- 소괄호로 묶는다.

`mySet = {10, 20, 30, 40, 50}`

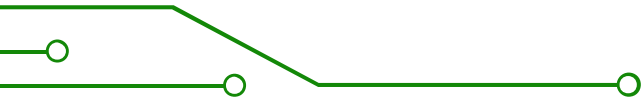
셋(set)은 중복이 없고 순서가 없는 자료형으로 수학에서의 집합과 같은 개념

- 순서가 없다. 검색이 빠르다.
- 중복된 값을 가질 수 없다. 리스트보다 메모리 관리가 효율적
 - 중복된 원소를 제거하는 용도로 많이 사용
- 중괄호로 묶는다.
- 합집합, 교집합, 차집합, 여집합의 연산을 지원한다.

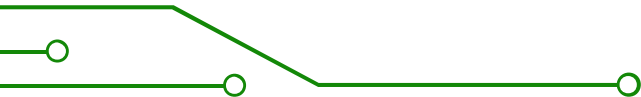
딕셔너리(dictionary)도 리스트와 같이 값을 저장하는 방법이다. 하지만 딕셔너리에는 값(value)과 관련된 키(key)가 있다.



```
>>> phone_book = {'홍길동': '1234', '이순신': '1235', '강감찬': '1236'}  
>>> phone_book["강감찬"]  
'1236'
```




```
>>> phone_book["강감찬"] = '9999'  
>>> phone_book["강감찬"]  
'9999'
```



```
>>> phone_book = { }
```

```
>>> phone_book["홍길동"] = '1234'
```

```
>>> phone_book["이순신"] = '1235'
```

```
>>> phone_book["강감찬"] = '1236'
```

```
>>> phone_book
```

```
{'홍길동': '1234', '이순신': '1235', '강감찬': '1236'}
```



```
>>> phone_book.keys()  
dict_keys(['홍길동', '이순신', '강감찬'])
```

```
>>> phone_book.values()  
dict_values(['1234', '1235', '1236'])
```

items	
1	"Kim"
2	"Park"
3	"Lee"
4	"Hong"
5	"Han"
6	"Choi"
7	"Nam"

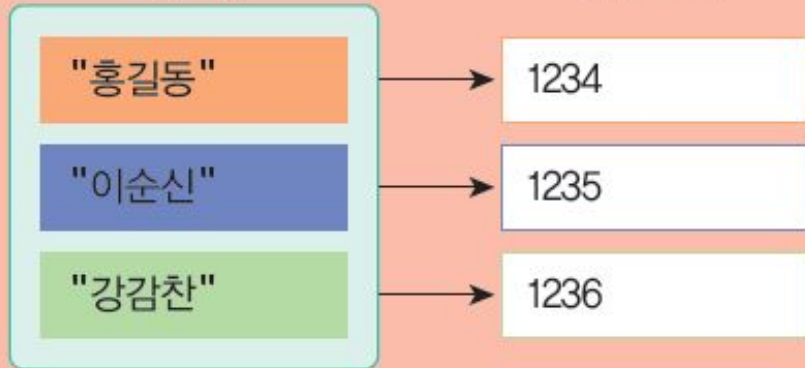
keys() { 4, 5, 6, 7 }

values() { "Hong", "Han", "Choi", "Nam" }

딕셔너리

키(key)

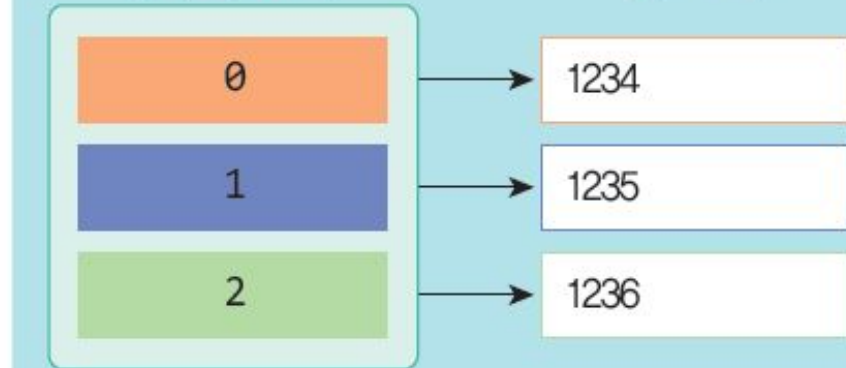
값(value)



리스트

인덱스(index)

값(value)



```
for key in sorted(phone_book.keys()):  
    print(key, phone_book[key])
```

```
강감찬 1235  
이순신 1236  
홍길동 1234
```



```
>>> phone_book.pop("홍길동")  
'1234'
```

```
>>> phone_book  
{'강감찬': '1235', '이순신': '1236'}
```

```
>>> del phone_book['이순신']
```

```
>>> phone_book  
{'강감찬': '1235'}
```

```
>>> phone_book.clear()
```

```
>>> print(phone_book)  
{}
```

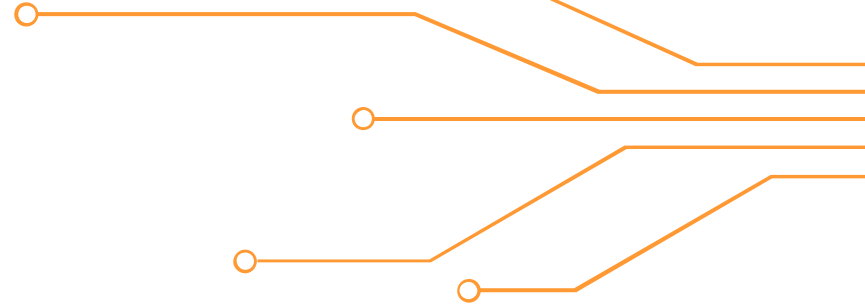
pop()

del

clear()



연산	설명
<code>d = { }</code>	공백 딕셔너리를 생성한다.
<code>d = {k₁ : v₁, k₂ : v₂, ..., k_n : v_n}</code>	초기값으로 딕셔너리를 생성한다.
<code>len(d)</code>	딕셔너리에 저장된 항목의 개수를 반환한다.
<code>k in d</code>	k가 딕셔너리 d 안에 있는지 여부를 반환한다.
<code>k not in d</code>	k가 딕셔너리 d 안에 없으면 True를 반환한다.
<code>d[key] = value</code>	d에 키와 값을 저장한다.
<code>v = d[key]</code>	딕셔너리에서 key에 해당되는 값을 반환한다.
<code>d.get(key, default)</code>	주어진 키를 가지고 값을 찾는다. 만약 없으면 default 값이 반환된다.
<code>d.pop(key)</code>	항목을 삭제한다.
<code>d.values()</code>	딕셔너리 안의 모든 값의 시퀀스를 반환한다.
<code>d.keys()</code>	딕셔너리 안의 모든 키의 시퀀스를 반환한다.
<code>d.items()</code>	딕셔너리 안의 모든 (키, 값)을 반환한다.





실습



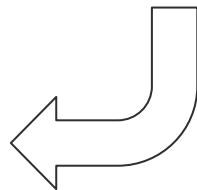
편의점에서 재고 관리를 수행하는 프로그램을 작성해보자.

편의점에서 판매하는 물건의 재고를 딕셔너리에 저장한다.

```
# {"커피음료": 7, "펜": 3, "종이컵": 2, "우유": 1, "콜라": 4, "책": 5 }  
# 을 가지고 있는 편의점에서 물건의 이름을 입력하면  
# 재고 갯수를 알려주는 코드를 만들어보세요
```

물건의 이름을 입력하시오: 콜라

4



편의점에서 재고 관리를 수행하는 프로그램을 작성해보자.

편의점에서 판매하는 물건의 재고를 딕셔너리에 저장한다.

```
items = { "커피음료": 7, "펜": 3, "종이컵": 2, "우유": 1, "콜라": 4, "책": 5 }
```

```
item = input("물건의 이름을 입력하시오: ");  
print (items[item])
```



도전문제

위의 프로그램을 편의점의 재고를 관리하는 프로그램으로 업그레이드해보자. 즉 재고를 증가, 또는 감소시킬 수도 있도록 코드를 추가해보자. 간단한 메뉴도 만들어보자.

앞에서 만든 프로그램을 업그레이드 해보자.

1. 반복문을 사용해서 재고를 확인, 증가, 감소할 수 있게 하자.



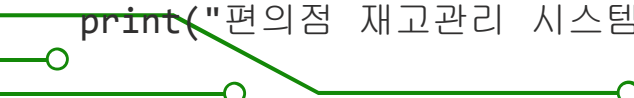
```
items = { "커피음료": 7, "펜": 3, "종이컵": 2, "우유": 1, "콜라": 4, "책": 5 }
```

```
while True:
    item = ''
    while item not in items.keys():
        item = input("물건의 이름을 입력하세요 (모르면 '0' : ")
    if item == '0':
        print(items.keys())
```



```
menu = input("재고확인 '1', 재고증가 '2', 재고감소 '3' 을 입력하세요 (종료는 'q') : ")
if menu == '1':
    print(item, '의 재고는', items[item], '개 입니다.')
elif menu == '2':
    cnt = int(input("몇개를 추가 입고하시겠습니까? : "))
    items[item] += cnt
    print(item, '의 재고는', items[item], '개 입니다.')
elif menu == '3':
    cnt = int(input("몇개를 출고하시겠습니까? : "))
    items[item] -= cnt
    print(item, '의 재고는', items[item], '개 입니다.')
elif menu == 'q':
    break
else :
    print('1,2,3,q 중 하나의 문자로 다시 입력해주세요.')
    continue

print("편의점 재고관리 시스템을 종료합니다.")
```



실습



날짜를 입력하시오 : 2022.11.1

일정을 입력하시오 : 과제 #1 제출

날짜를 입력하시오 : 2022.11.2

일정을 입력하시오 : 과제 #2 제출

날짜를 입력하시오 : q

```
{'2022.11.1': '과제 #1 제출', '2022.11.2': '과제 #2 제출'}
```




```
mydict = {}

while True:
    date = input("날짜를 입력하시오: ")
    if date == "q" : break
    job = input("일정을 입력하시오: ")
    if date not in mydict:
        mydict[date]=job
    else:
        print("오류입니다.")
print(mydict)
```



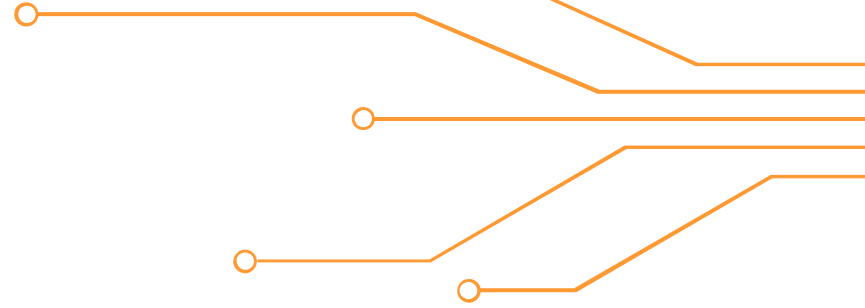
도전문제

어떤 특정한 날에는 일정이 두 개 이상일 수 있다. 이때는 어떻게 해야 할까? 일정을 리스트로 저장하면 어떨까? 딕셔너리 안에 값으로 리스트도 저장할 수 있다. 구현해보자.

```
# sch_man2.py

mydict = {}

while True:
    date = input("날짜를 입력하시오: ")
    if date == "q" : break
    job = input("일정을 입력하시오: ")
    if date not in mydict:
        mydict[date]=[] # 빈 리스트
        mydict[date].append(job)
    else:
        mydict[date].append(job)
print(mydict)
```



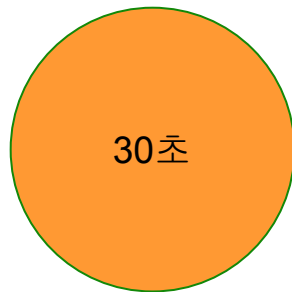


퀴즈



1. 파이썬 딕셔너리는 다음 중 어떤 특징을 가지고 있습니까? 맞는 것을 모두 고르세요.

- 1) 순서가 없다.
- 2) 중복된 키를 허용하지 않는다.
- 3) 키와 값으로 이루어져 있다.
- 4) 중복된 값을 허용하지 않는다.



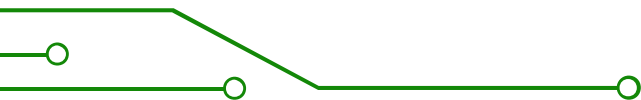
1. 파이썬 딕셔너리는 다음 중 어떤 특징을 가지고 있습니까? 맞는 것을 모두 고르세요. [1, 2, 3]

1) 순서가 없다.

2) 중복된 키를 허용하지 않는다.

3) 키와 값으로 이루어져 있다.

4) 중복된 값을 허용하지 않는다.



1. 파이썬 딕셔너리는 다음 중 어떤 특징을 가지고 있습니까? 맞는 것을 모두 고르세요. [1, 2, 3]

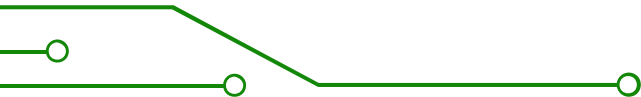
1) 순서가 없다.

2) 중복된 키를 허용하지 않는다.

3) 키와 값으로 이루어져 있다.

4) 중복된 값을 허용하지 않는다.

딕셔너리는 검색을 위해 키의 중복은 허용하지 않지만 값은 상관이 없습니다.



2. 파이썬 3 에서 딕셔너리의 키와 값을 모두 출력하기 위해 for 를 사용하는 바른 방법은 다음 중 어떤 것입니까?

```
data = {'name':'Hone', 'phone':'111-1234'}
```

- 1) for key, val in data.items():
- 2) for key in data:
- 3) for key in data:
- 4) for key, val in data.iteritems():



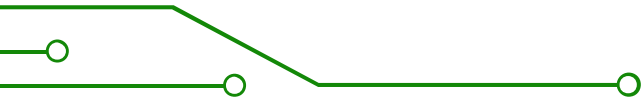
30초



2. 파이썬 3 에서 딕셔너리의 키와 값을 모두 출력하기 위해 for 를 사용하는 바른 방법은 다음 중 어떤 것입니까? [1]

```
data = {'name':'Hone', 'phone':'111-1234'}
```

- 1) **for key, val in data.items():**
- 2) for key in data:
- 3) for key in data:
- 4) for key, val in data.iteritems():



2. 파이썬 3 에서 딕셔너리의 키와 값을 모두 출력하기 위해 for 를 사용하는 바른 방법은 다음 중 어떤 것입니까? [1]

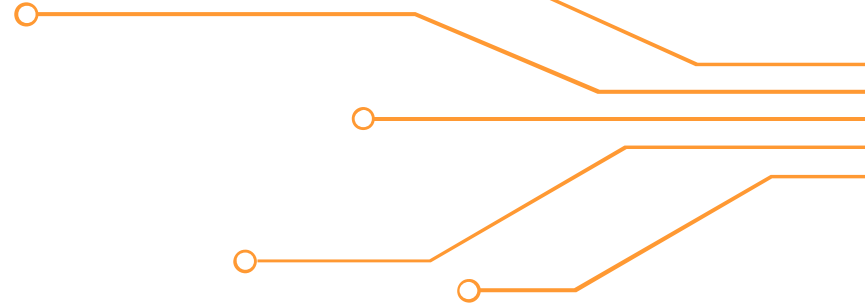
```
data = {'name':'Hone', 'phone':'111-1234'}
```

- 1) **for key, val in data.items():**
- 2) for key in data:
- 3) for key in data:
- 4) for key, val in data.iteritems():

items() 함수를 사용하면 키와 값을 모두 가져옵니다.

키와 값을 모두 가져오기 위해서는 2개의 변수(**key, val**)가 사용되어야 합니다.

iteritems() 함수는 파이썬 2.X 에서 사용되던 것으로 파이썬 3 에서는 더이상 사용되지 않습니다.

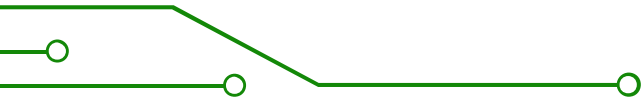




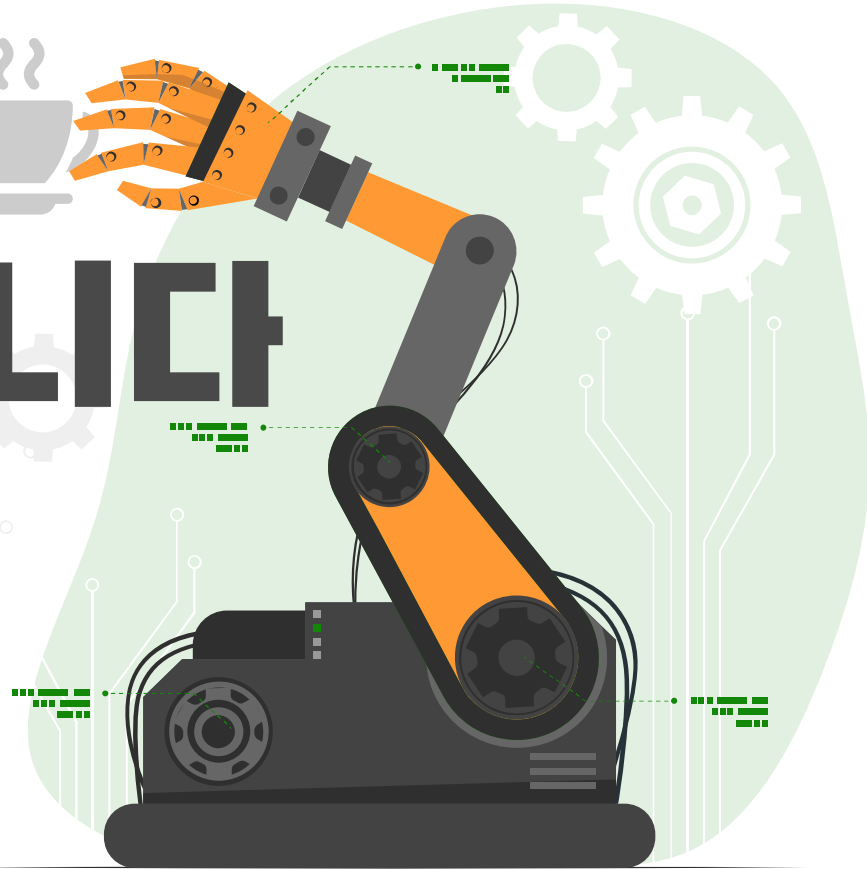
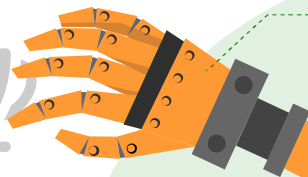
핵심정리



- ❑ 딕셔너리는 리스트와 유사하지만 키와 값을 쌍으로 저장하는 자료형입니다.
- ❑ 딕셔너리에서는 인덱스 대신 키를 이용해 값을 찾습니다.
- ❑ 딕셔너리는 `value = d[key]`와 같은 형식을 사용합니다.
- ❑ 딕셔너리에 새로운 요소(key + value)를 추가하려면 `d[key] = value` 형식을 사용합니다. `pop()`을 이용하여 요소를 삭제할 수 있습니다.



수고하셨습니다



컴퓨팅 사고와 프로그래밍



CH 11.

GUI

프로그래밍1



학습목표



1. GUI 에 대해 이해한다.
2. tkinter 모듈의 기본 사용법을 익힌다.
3. tkinter 를 이용하여 온도변환기 프로그램을 만들수 있다.

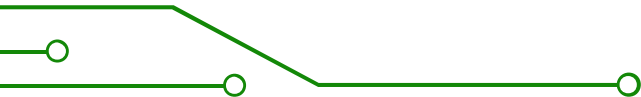




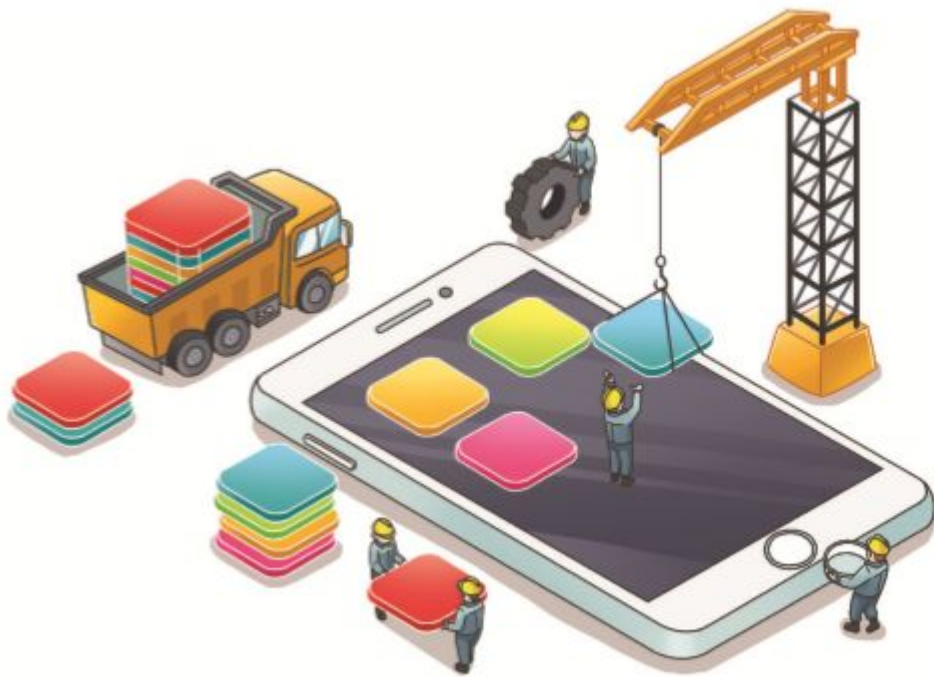
이론수업



- GUI는 그래픽 사용자 인터페이스 (Graphical User Interface)의 약자
- 컴퓨터와 사용자가 그래픽 요소를 사용하여 상호 작용하는 인터페이스
- GUI는 텍스트 기반 인터페이스보다 사용하기 쉽고 직관적
- 마우스, 터치스크린과 같은 포인팅 장치를 사용하여 컴퓨터와 상호 작용

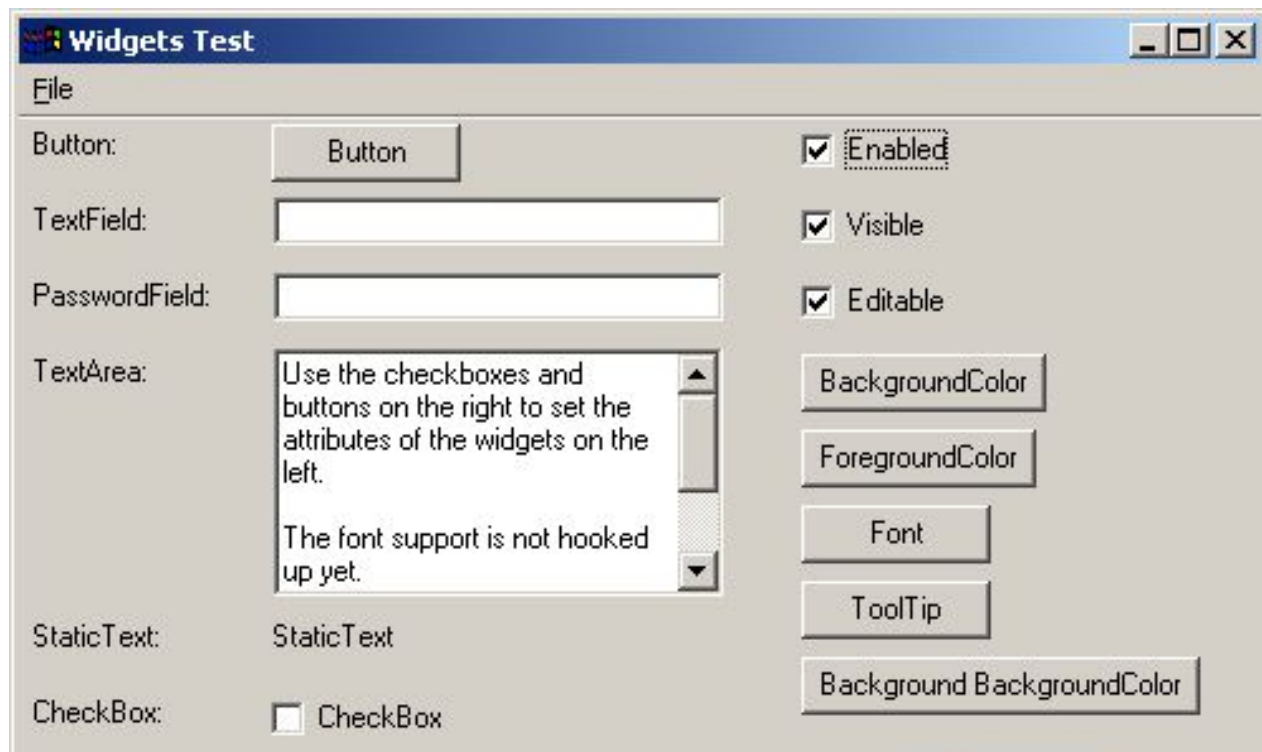


tkinter는 파이썬에서 그래픽 사용자 인터페이스(GUI: graphical user interface)를 개발할 때 필요한 모듈
기본설치

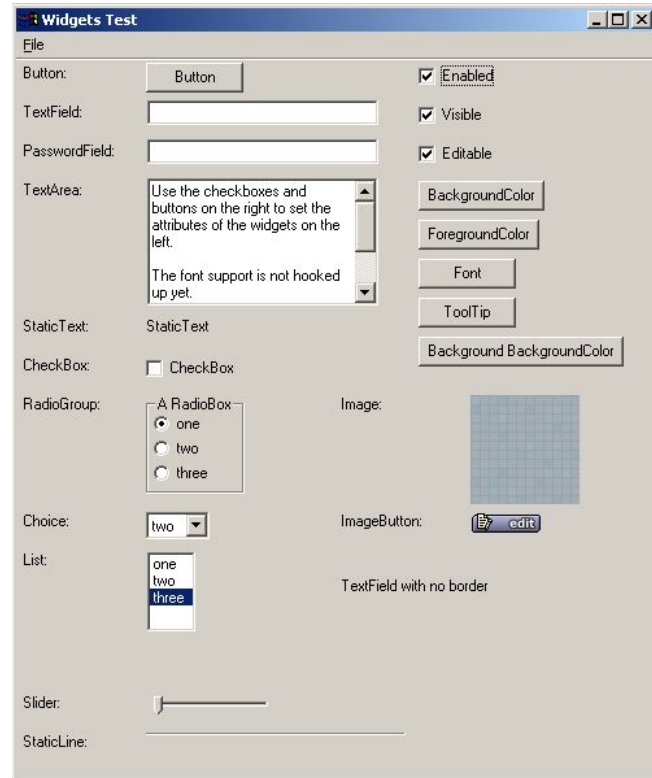


위젯 : 정보를 표시하거나
응용프로그램과 상호 작용할 수
있는 방법을 제공하는 GUI 의
요소

버튼, 체크 박스등



위젯	설명
Button	간단한 버튼으로 명령을 수행할 때 사용된다.
Canvas	화면에 무언가를 그릴 때 사용한다.
Checkbox	2가지의 구별되는 값을 가지는 변수를 표현한다.
Entry	한 줄의 텍스트를 입력받는 필드이다.
Frame	컨테이너 클래스이다. 프레임은 경계선과 배경을 가지고 있다. 다른 위젯들을 그룹핑하는데 사용된다.
Label	텍스트나 이미지를 표시한다.
Listbox	선택 사항을 표시한다.
Menu	메뉴를 표시한다. 풀다운 메뉴나 팝업 메뉴가 가능하다.
Menubutton	메뉴 버튼이다. 풀다운 메뉴가 가능하다.
Message	텍스트를 표시한다. 레이블 위젯과 비슷하다. 하지만 자동적으로 주어진 크기로 텍스트를 축소할 수 있다.
Radiobutton	여러 값을 가질 수 있는 변수를 표시한다.
Scale	슬라이더를 끌어서 수치를 입력하는데 사용된다.
Scrollbar	캔버스, 엔트리, 리스트 박스, 텍스트 위젯을 위한 스크롤 바를 제공한다.
Text	형식을 가지는 텍스트를 표시한다. 여러 가지 스타일과 속성으로 텍스트를 표시할 수 있다.
Toplevel	최상위 윈도우로 표시되는 독립적인 컨테이너 위젯이다.
LabelFrame	경계선과 제목을 가지는 프레임 위젯의 변형이다.
PanedWindow	자식 위젯들을 크기조절이 가능한 패널로 관리하는 컨테이너 위젯이다.
Spinbox	특정한 범위에서 값을 선택하는 엔트리 위젯의 변형



- 단순 위젯: Button, Canvas, Checkbutton, Entry, Label, Message 등이 여기에 속한다.
- 컨테이너 컴포넌트: 다른 컴포넌트를 안에 포함할 수 있는 컴포넌트로서 Frame, Toplevel, LabelFrame, PanedWindow 등이 여기에 속한다.

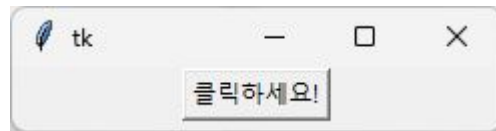
단순 위젯



컨테이너

버튼이 하나 있는 윈도우를 생성

```
from tkinter import *  
  
window = Tk()  
button = Button(window, text="클릭하세요!")  
button.pack()  
  
window.mainloop()
```



엔트리 위젯 : 사용자 입력을 위해 필요

레이블 위젯 : 화면에 텍스트 표시



```
from tkinter import *
```

```
window = Tk()
```

```
l1 = Label(window, text="화씨")
```

```
l2 = Label(window, text="섭씨")
```

```
l1.pack()
```

```
l2.pack()
```

```
e1 = Entry(window)
```

```
e2 = Entry(window)
```

```
e1.pack()
```

```
e2.pack()
```

```
b1 = Button(window, text="화씨->섭씨")
```

```
b2 = Button(window, text="섭씨->화씨")
```

```
b1.pack()
```

```
b2.pack()
```

```
window.mainloop( )
```

- 적층(pack) 배치 관리자: 위젯들을 수직이나 수평으로 쌓아서 배치한다.
- 격자(grid) 배치 관리자: 위젯들을 격자(그리드) 모양으로 배치한다.
- (절대)위치(place) 배치 관리자: 절대 좌표를 사용하여 위젯을 배치한다.



위젯을 테이블 형태로 배치



```
from tkinter import *
```

```
window = Tk()
```

```
l1 = Label(window , text="화씨")
```

```
l2 = Label(window, text="섭씨")
```

```
l1.grid(row=0, column=0)
```

```
l2.grid(row=1, column=0)
```

```
e1 = Entry(window)
```

```
e2 = Entry(window)
```

```
e1.grid(row=0, column=1)
```

```
e2.grid(row=1, column=1)
```

```
b1 = Button(window, text="화씨->섭씨")
```

```
b2 = Button(window, text="섭씨->화씨")
```

```
b1.grid(row=2, column=0)
```

```
b2.grid(row=2, column=1)
```

```
window.mainloop()
```

버튼 클릭시 실행할 이벤트 처리

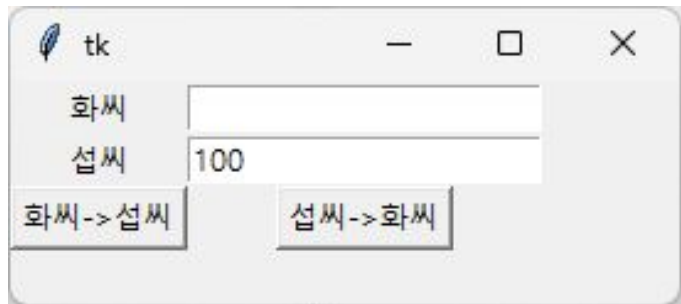
```
from tkinter import *  
  
def process():  
    print("안녕하세요?")  
  
window = Tk()  
button = Button(window, text="클릭하세요!", command=process)  
button.pack()  
window.mainloop()
```



```
from tkinter import *  
  
def process():  
    e2.insert(0, "100")  
  
window = Tk()
```

```
| l1 = Label(window , text="화씨")  
| l2 = Label(window, text="섭씨")  
| l1.grid(row=0, column=0)  
| l2.grid(row=1, column=0)  
|  
| e1 = Entry(window)  
| e2 = Entry(window)  
| e1.grid(row=0, column=1)  
| e2.grid(row=1, column=1)
```

```
b1 = Button(window, text="화씨 -> 섭씨", command=process)  
b2 = Button(window, text="섭씨 -> 화씨")  
b1.grid(row=2, column=0)  
b2.grid(row=2, column=1)  
  
window.mainloop()
```



```
from tkinter import *

def process():
    temperature = float(e1.get())
    mytemp = (temperature-32)*5/9
    e2.insert(0, str(mytemp))

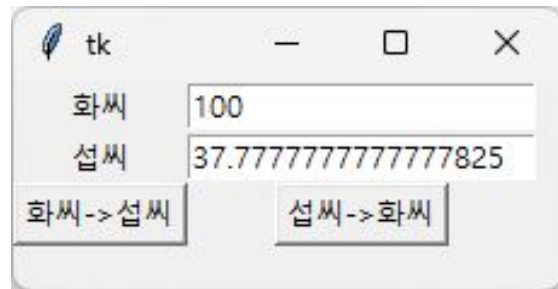
window = Tk()

l1 = Label(window , text="화씨")
l2 = Label(window, text="섭씨")
l1.grid(row=0, column=0)
l2.grid(row=1, column=0)
```

```
e1 = Entry(window)
e2 = Entry(window)
e1.grid(row=0, column=1)
e2.grid(row=1, column=1)

b1 = Button(window, text="화씨 -> 섭씨",
             command=process)
b2 = Button(window, text="섭씨 -> 화씨")
b1.grid(row=2, column=0)
b2.grid(row=2, column=1)

window.mainloop()
```




```
from tkinter import *

def process():
    temperature = float(e1.get())
    mytemp = (temperature-32)*5/9
    e2.insert(0, str(mytemp))

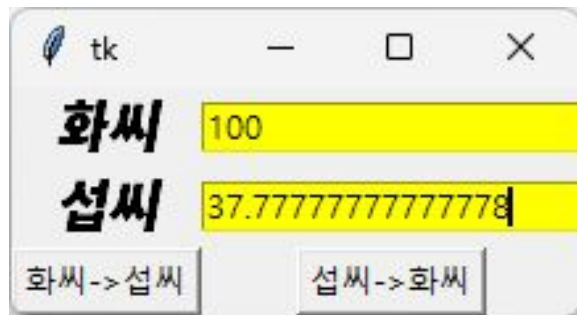
window = Tk()

l1 = Label(window , text="화씨",
            font='helvetica 16 italic')
l2 = Label(window, text="섭씨",
            font='helvetica 16 italic')
l1.grid(row=0, column=0)
l2.grid(row=1, column=0)
```

```
e1 = Entry(window, bg="yellow", fg="black")
e2 = Entry(window, bg="yellow", fg="black")
e1.grid(row=0, column=1)
e2.grid(row=1, column=1)

b1 = Button(window, text="화씨 -> 섭씨",
             command=process)
b2 = Button(window, text="섭씨 -> 화씨")
b1.grid(row=2, column=0)
b2.grid(row=2, column=1)

window.mainloop()
```



절대위치를 이용한 위젯 배치



```
from tkinter import *
```

```
window = Tk()
```

```
w = Label(window, text="박스 #1", bg="red", fg="white")
```

```
w.place(x=0, y=0)
```

```
w = Label(window, text="박스 #2", bg="green", fg="black")
```

```
w.place(x=20, y=20)
```

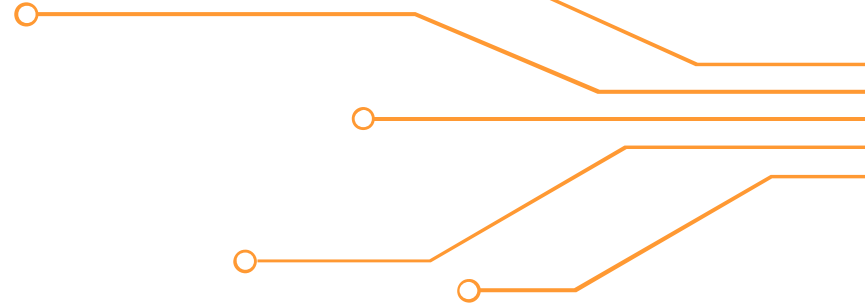
```
w = Label(window, text="박스 #3", bg="blue", fg="white")
```

```
w.place(x=40, y=40)
```

```
window.mainloop()
```

- `window.geometry('600x300')` # 윈도우의 가로 세로 크기 설정
- `font = '맑은고딕 12'`
- `width = 30`
- `height = 4`







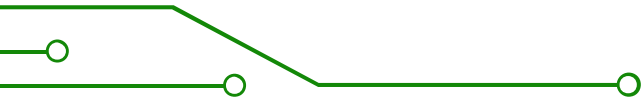
퀴즈



1. tkinter 는 어떤 용도로 사용되니까?

- 1) 게임개발
- 2) 웹개발
- 3) GUI 개발
- 4) 데이터분석

30초



1. tkinter 는 어떤 용도로 사용되니까? [3]

1) 게임개발

2) 웹개발

3) GUI 개발

4) 데이터분석



1. tkinter 는 어떤 용도로 사용되니까? [3]

1) 게임개발

2) 웹개발

3) GUI 개발

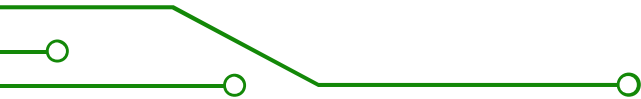
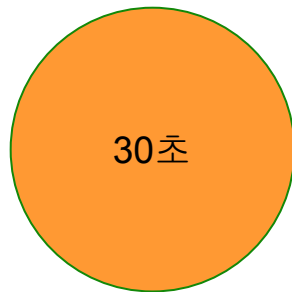
4) 데이터분석

tkinter 는 파이썬 처음 설치시 기본 설치되는 GUI 개발을 위한 패키지 입니다.



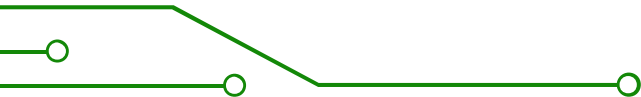
2. Tkinter를 사용해서 만들 수 있는 UI 요소는 무엇인가? 모두 고르시오.

- 1) 버튼
- 2) 라벨
- 3) 텍스트 입력창
- 4) 콤보박스
- 5) 체크박스
- 6) 리스트박스
- 7) 스피너박스
- 8) 스크롤바
- 9) 프레임



2. Tkinter를 사용해서 만들 수 있는 UI 요소는 무엇인가? 모두 고르시오.

- 1) 버튼
- 2) 라벨
- 3) 텍스트 입력창
- 4) 콤보박스
- 5) 체크박스
- 6) 리스트박스
- 7) 스펀박스
- 8) 스크롤바
- 9) 프레임

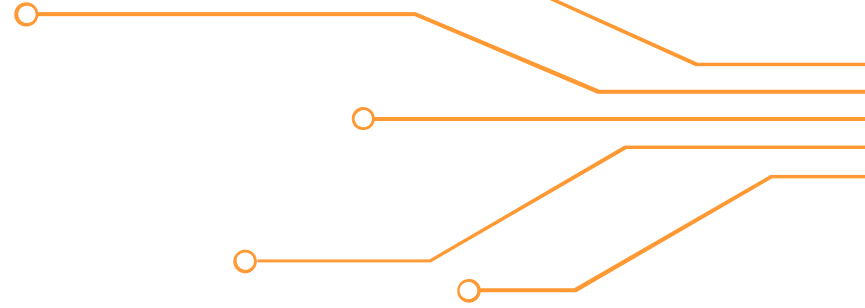


2. Tkinter를 사용해서 만들 수 있는 UI 요소는 무엇인가? 모두 고르시오.

- 1) 버튼
- 2) 라벨
- 3) 텍스트 입력창
- 4) 콤보박스
- 5) 체크박스
- 6) 리스트박스
- 7) 스펀박스
- 8) 스크롤바
- 9) 프레임

버튼, 라벨, 텍스트 입력창, 콤보박스, 체크박스, 리스트박스, 스펀박스, 스크롤바, 프레임 모두 tkinter 를 사용하여 만들수 있는 UI 요소들 입니다.





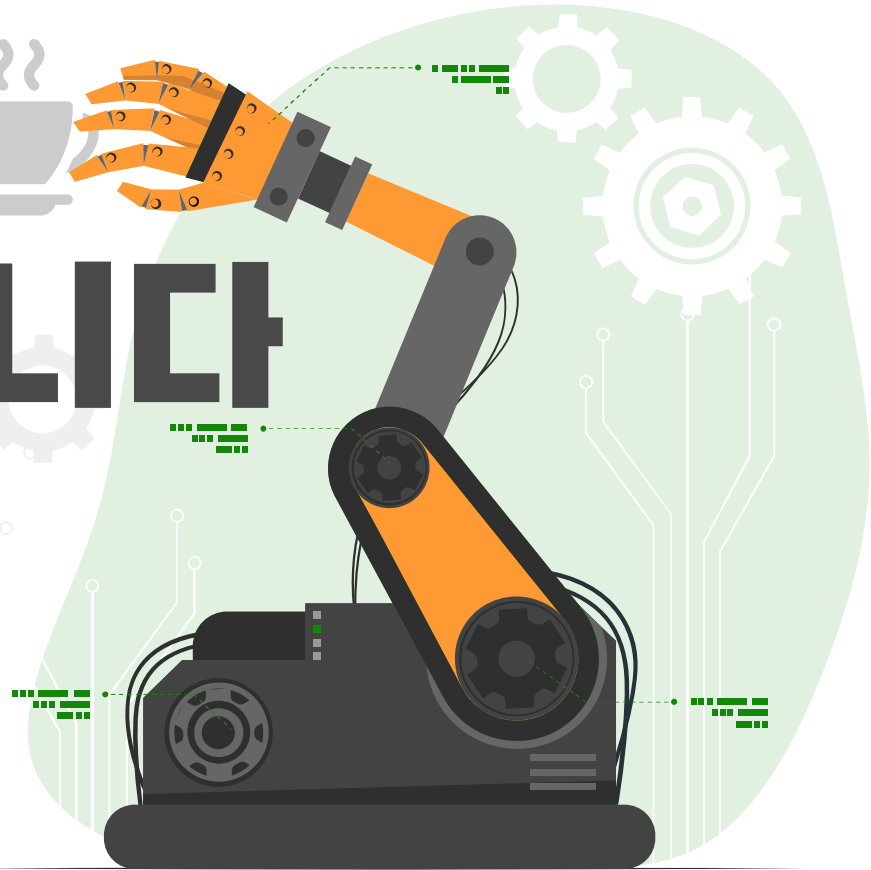
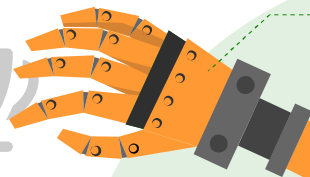


핵심정리



- ❑ tkinter는 파이썬에서 그래픽 사용자 인터페이스(GUI: graphical user interface)를 개발할 때 사용가능한 기본설치되어 있는 모듈입니다.
- ❑ tkinter 로 온도 변환기 GUI 프로그래밍을 한다면 다음 과정을 따릅니다.
 - 우선 최상위 윈도우를 생성하고 버튼과 엔트리를 최상위 윈도우에 추가합니다.
 - 버튼이 눌리면 이벤트가 발생하게 합니다.
 - 이벤트는 버튼의 command 매개 변수에 이벤트를 처리할 함수의 이름을 넣어서 처리합니다.
- ❑ 파이썬은 3종류의 배치 관리자를 제공합니다. 적층(pack) 배치 관리자, 격자(grid) 배치 관리자. 절대(place)위치 배치 관리자 입니다.

수고하셨습니다



컴퓨팅 사고와 프로그래밍



CH 12.

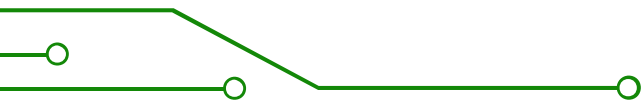
GUI 프로그래밍2



학습목표



1. tkinter 를 이용해 그림판 프로그램을 만들 수 있다.
2. tkinter 를 이용해 계산기 프로그램을 만들 수 있다.

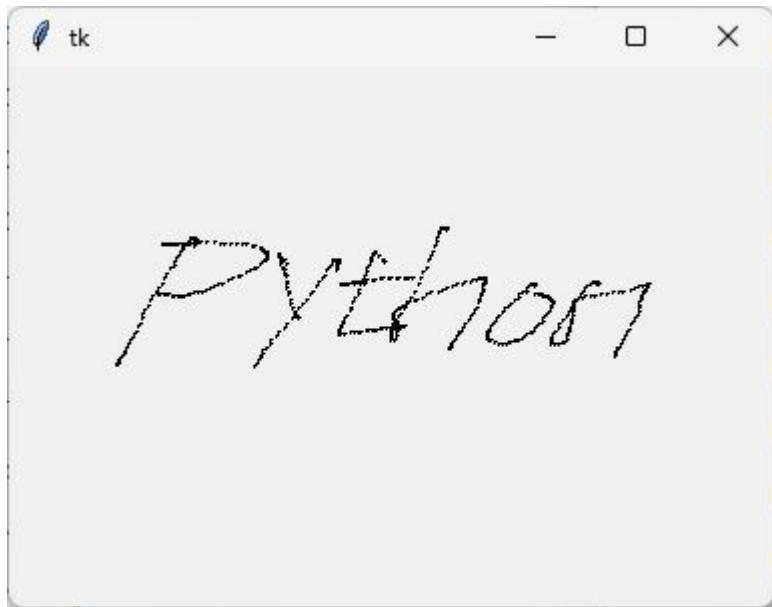




이론수업

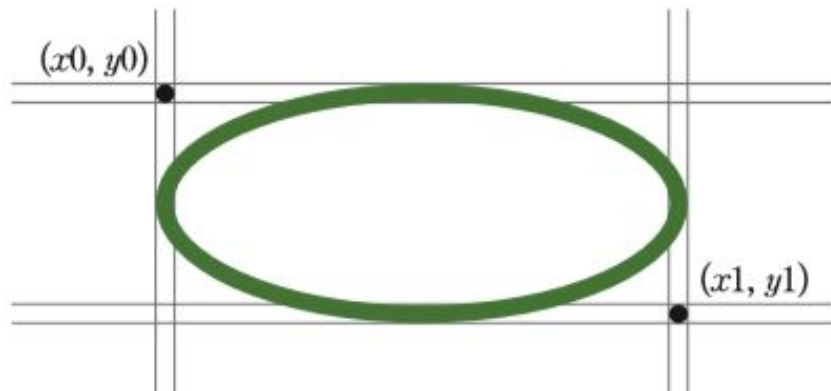


다음과 같이 마우스를 움직여서 화면에 그림을 그리는 윈도우의 그림판과 비슷한 프로그램을 작성해보자.



tkinter에서 그림을 그리려면 캔버스(canvas)라는 위젯이 필요하다. Canvas 위젯을 사용하면 많은 그래픽 기능을 사용할 수 있다.

```
window = Tk()  
...  
canvas = Canvas(window, width=300, height=200)  
canvas.create_oval(x0, y0, x1, y1, option, ...)
```



마우스 Motion 이벤트

이름	의미
<Motion>	마우스가 움직일 때
<B1-Motion>	마우스 왼쪽 버튼을 누르면서 움직일 때
<B2-Motion>	마우스 휠 버튼을 누르면서 움직일 때
<B3-Motion>	마우스 오른쪽 버튼을 누르면서 움직일 때

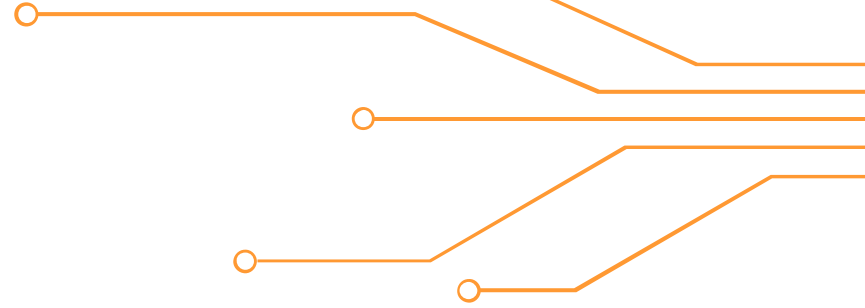



```
from tkinter import *

def paint(event):
    x1, y1 = ( event.x-1 ), ( event.y-1 )
    x2, y2 = ( event.x+1 ), ( event.y+1 )
    canvas.create_oval( x1, y1, x2, y2, fill = "black")

window = Tk()
canvas = Canvas(window)
canvas.pack()
canvas.bind("<B1-Motion>", paint)
window.mainloop()
```



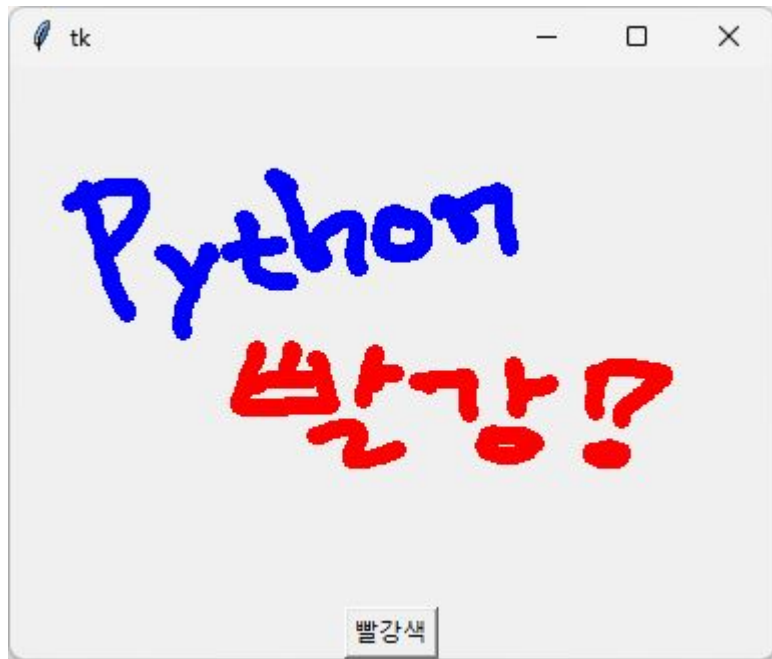


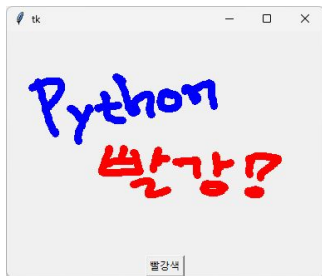


실습



색상 변경 버튼 추가





```
from tkinter import *
```

```
mycolor = "blue"
```

```
def paint(event):  
    x1, y1 = ( event.x-1 ), ( event.y+1 )  
    x2, y2 = ( event.x-1 ), ( event.y+1 )  
    canvas.create_oval( x1, y1, x2, y2, fill = mycolor, outline=mycolor)
```

```
def change_color():  
    global mycolor  
    mycolor="red"
```

```
window = Tk()  
canvas = Canvas(window)  
canvas.pack()  
canvas.bind("<B1-Motion>", paint)  
button = Button(window, text="빨강색", command=change_color)  
button.pack()  
window.mainloop()
```

버튼을 추가해보자

- 녹색, 노란색, 파란색 버튼 추가

```
from tkinter import *
```

```
mycolor = "blue"
```

```
def paint(event):
```

```
    r=4
```

```
    x1, y1 = ( event.x-r ), ( event.y-r )
```

```
    x2, y2 = ( event.x+r ), ( event.y+r )
```

```
    canvas.create_oval( x1, y1, x2, y2, fill = mycolor, outline=mycolor)
```

```
def change_red():
```

```
    global mycolor
```

```
    mycolor="red"
```



```
def change_blue():  
    global mycolor  
    mycolor="blue"
```

```
def change_green():  
    global mycolor  
    mycolor="green"
```

```
def change_yellow():  
    global mycolor  
    mycolor="yellow"
```

```
window = Tk()  
canvas = Canvas(window)  
canvas.pack()  
canvas.bind("<B1-Motion>", paint)
```



```
button = Button(window, text="빨강색", command=change_red)
button.pack(side='left')
```

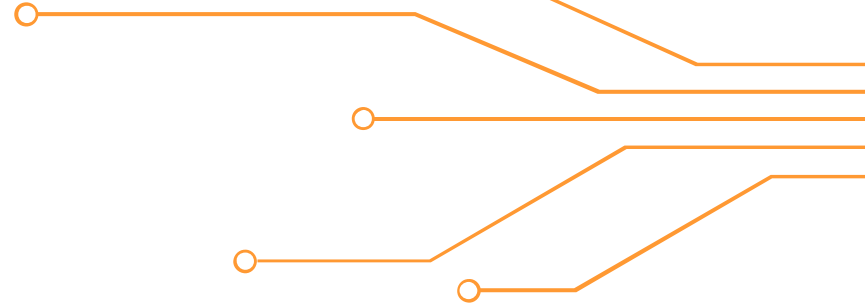
```
button = Button(window, text="파란색", command=change_blue)
button.pack(side='left')
```

```
button = Button(window, text="초록색", command=change_green)
button.pack(side='left')
```

```
button = Button(window, text="노란색", command=change_yellow)
button.pack(side='left')
```

```
window.mainloop()
```



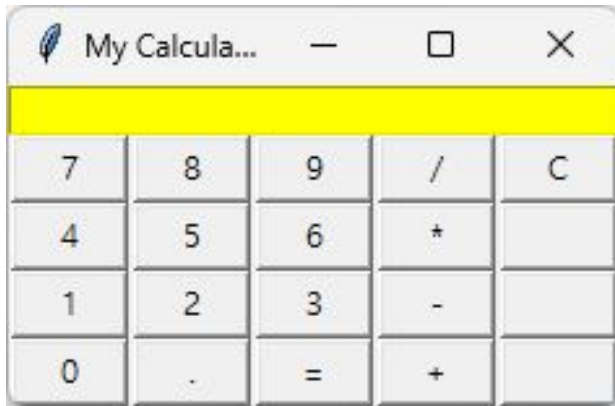
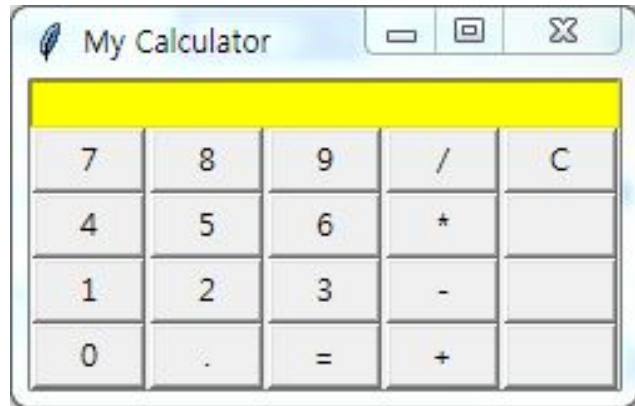




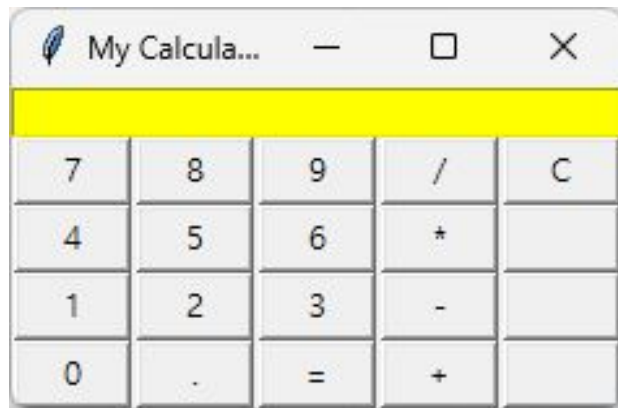
이론수업



다음과 같은 계산기를 작성해보자.



- 격자 배치 관리자를 사용
- 버튼과 엔트리 위젯 사용



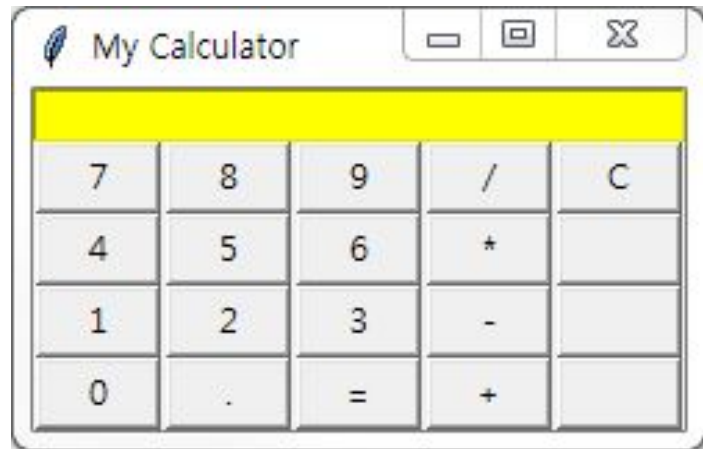
엔트리 위젯

버튼을 격자 모양으로 배치

```
from tkinter import *  
  
window = Tk()  
window.title("My Calculator")  
display = Entry(window, width=33, bg="yellow")  
display.grid(row=0, column=0, columnspan=5)
```

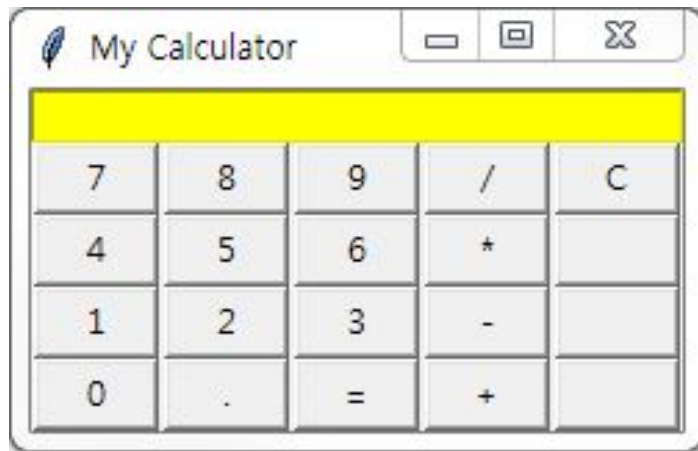


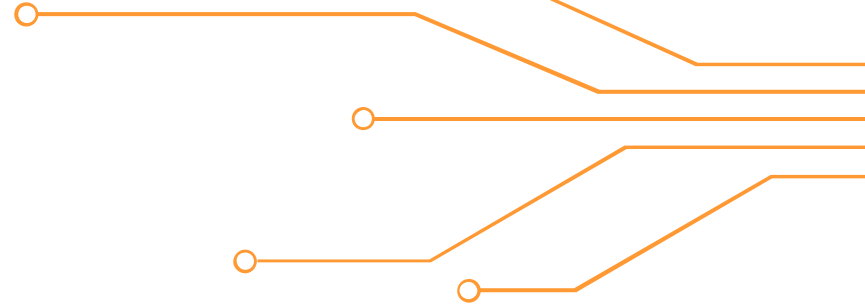
```
button_list = [  
    '7', '8', '9', '/', 'C',  
    '4', '5', '6', '*', ' ',  
    '1', '2', '3', '-', ' ',  
    '0', '.', '=', '+', ' ' ]
```



```
row_index = 1  
col_index = 0
```

```
for button_text in button_list:  
    Button(window, text=button_text, width=5).grid(row=row_index, column=col_index)  
    col_index += 1  
    if col_index > 4:  
        row_index += 1  
        col_index = 0
```

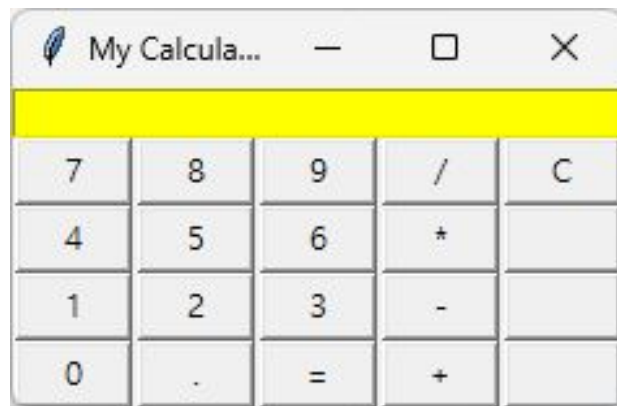
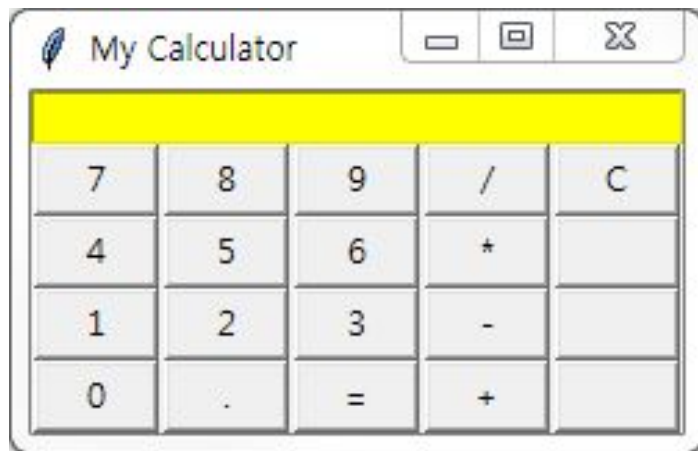






실습





calc1.py

```
from tkinter import *

window = Tk()
window.title("My Calculator")
display = Entry(window, width=33, bg="yellow")
display.grid(row=0, column=0, columnspan=5)

button_list = [ '7', '8', '9', '/', 'C',
                 '4', '5', '6', '*', ' ',
                 '1', '2', '3', '-', ' ',
                 '0', '.', '=', '+', ' ' ]

row_index = 1
col_index = 0

def click(key):
    display.insert(END, key)    # 엔트리 위젯의 끝에 key를 추가

for button_text in button_list:
    Button(window, text=button_text, width=5,
           command=click(button_text)).grid(row=row_index, column=col_index)
    col_index += 1
    if col_index > 4:
        row_index += 1
        col_index = 0
window.mainloop()
```

calc2.py

```
from tkinter import *

window = Tk()
window.title("My Calculator")
display = Entry(window, width=33, bg="yellow")
display.grid(row=0, column=0, columnspan=5)

button_list = [ '7', '8', '9', '/', 'C',
                 '4', '5', '6', '*', ' ',
                 '1', '2', '3', '-', ' ',
                 '0', '.', '=', '+', ' ' ]

row_index = 1
col_index = 0

def click(key):
    display.insert(END, key)    # 엔트리 위젯의 끝에 key를 추가

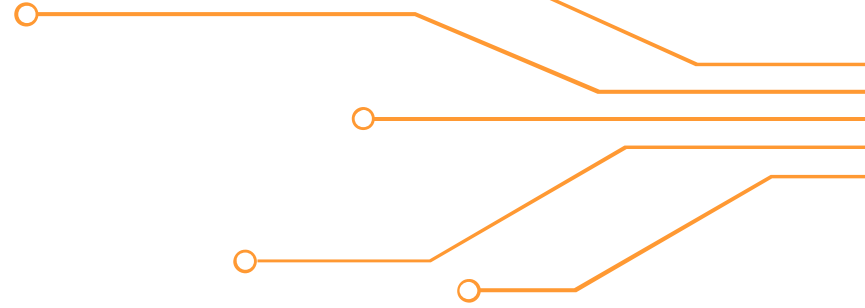
for button_text in button_list:
    def process(t=button_text):
        click(t)
    Button(window, text=button_text, width=5,
           command=process).grid(row=row_index, column=col_index)
    col_index += 1
    if col_index > 4:
        row_index += 1
        col_index = 0
window.mainloop()
```

calc3.py

```
from tkinter import *
window = Tk()
window.title("My Calculator")
display = Entry(window, width=33, bg="yellow")
display.grid(row=0, column=0, columnspan=5)
button_list = [ '7', '8', '9', '/', 'C',
                 '4', '5', '6', '*', ' ',
                 '1', '2', '3', '-', ' ',
                 '0', '.', '=', '+', ' ' ]

def click(key):
    if key == "=":
        result = eval(display.get())
        s = str(result)
        display.insert(END, "=" + s)
    else: display.insert(END, key)

row_index = 1
col_index = 0
for button_text in button_list:
    def process(t=button_text):
        click(t)
    Button(window, text=button_text, width=5,
           command=process).grid(row=row_index, column=col_index)
    col_index += 1
    if col_index > 4:
        row_index += 1
        col_index = 0
window.mainloop()
```





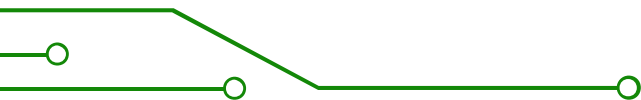
퀴즈



1. Tkinter를 사용해서 버튼을 생성하는 함수는 무엇인가요?

- 1) Button()
- 2) Label()
- 3) TextField()
- 4) Combobox()

30초



1. Tkinter를 사용해서 버튼을 생성하는 함수는 무엇인가요? [1]

1) **Button()**

2) Label()

3) TextField()

4) Combobox()



1. Tkinter를 사용해서 버튼을 생성하는 함수는 무엇인가요? [1]

1) **Button()**

2) Label()

3) TextField()

4) Combobox()

`button = Button(window, text='클릭하세요', command=process)` 와 같은 형태로 `Button()` 함수를 사용하여 버튼을 생성할 수 있습니다.



1. Tkinter를 사용해서 Label 을 생성하는 함수는 무엇인가요?

- 1) Button()
- 2) Label()
- 3) TextField()
- 4) Combobox()

30초



1. Tkinter를 사용해서 Label 을 생성하는 함수는 무엇인가요? [2]

1) Button()

2) Label()

3) TextField()

4) Combobox()



1. Tkinter를 사용해서 Label 을 생성하는 함수는 무엇인가요? [2]

1) Button()

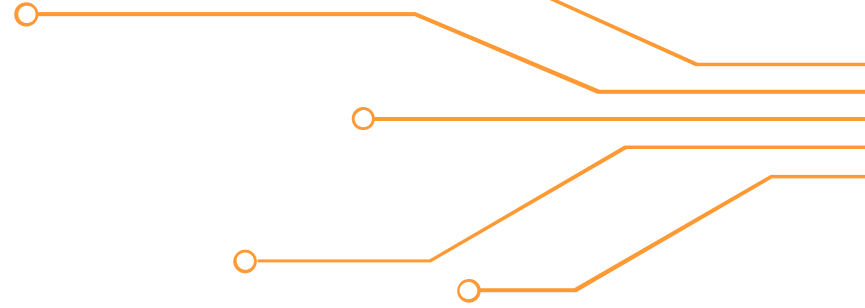
2) Label()

3) TextField()

4) Combobox()

`lb11 = Label(window, text='섭씨')` 와 같은 형태로 `Label()` 함수를 사용할 수 있습니다.



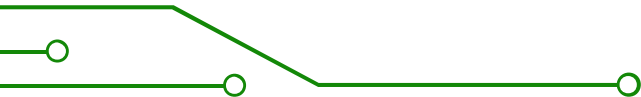




핵심정리



- ❑ tkinter 로 그림판 프로그램을 만들기 위해서 캔버스 객체를 사용하고, 캔버스 객체에서 발생하는 드래그 이벤트를 처리합니다. 드래그 이벤트가 발생하면 타원과 같은 도형을 그릴 수 있습니다.
- ❑ tkinter 를 이용하여 계산기 프로그램을 만들 수 있습니다.
- ❑ 엔트리, 버튼, 격자 배치 관리자, 텍스트, 엔트리 위젯 등의 GUI 관련 용어를 이해하고, 이벤트를 이용하여 프로그래밍할 수 있습니다.



수고하셨습니다

