차례

학습내용

LAB

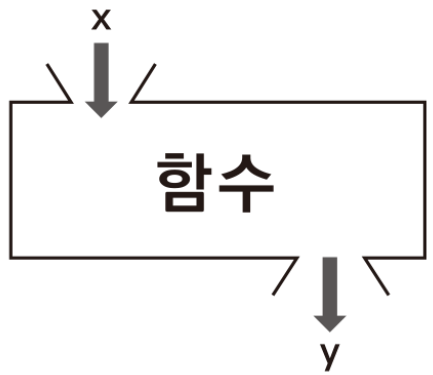
01. 함수가 무엇인가요?

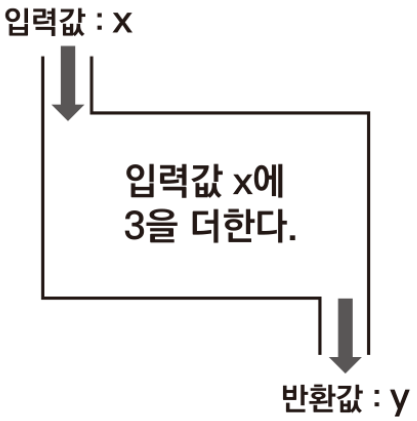
- 파이썬에서 프로그램을 쪼개는 3가지의 방법
  - ✓ 함수(function) : 여러 개의 명령을 하나의 단위로 묶어 놓은 것
  - ✓ 클래스(class) : 클래스는 코드 중에서, 관련된 변수와 함수를 하나로 묶은 것
  - ✓ 모듈(module) : 여러 개의 함수를 모아 놓은 것으로 프로그램 일부를 가지고 있는 독립적인 파일로 관리하는 것입니다.

함수 클래스 모듈

01. 함수가 무엇인가요?

- 특정 기능을 구현하는 부분을 따로 떼어 구현한 것
- 반복적으로 사용되는 의미 있는 부분
- 서브루틴(subroutine), 루틴(routine), 메서드(method), 프로시저(procedure)

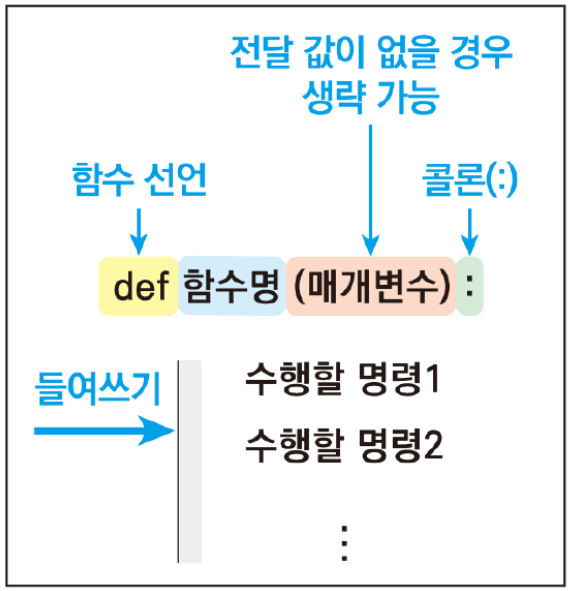




02. 함수를 작성하고 호출하기

- 함수를 만드는 과정 = 함수를 정의한다
- 함수를 사용하는 것 = 함수를 호출한다
- 한 번만 함수를 정의하면 언제든지 필요할 때 함수를 호출하여 문제를 처리할 수 있음
- 프로그램을 함수화하면 프로그램의 흐름을 일목요연하게 볼 수 있음

02. 함수를 작성하고 호출하기



콜론(:)
코드 블록이 이어서 등장한다는 것을 의미해요. (if, for, while 뒤에도 콜론이 있어요.)

02. 함수를 작성하고 호출하기

함수는 def 키워드를 이용하여 정의합니다.

코드

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

print_address( )          # 함수의 호출
```

실행 결과

```
서울특별시 종로구 1번지
파이썬 빌딩 7층
홍길동
```

03. 왜 우리는 함수를 작성하는가?

우리 집의 주소를 2번 인쇄하려고 합니다.

코드 1

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

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

코드 2

```
for i in range(2):
    print("서울특별시 종로구 1번지")
    print("파이썬 빌딩 7층")
    print("홍길동")
```

코드 3

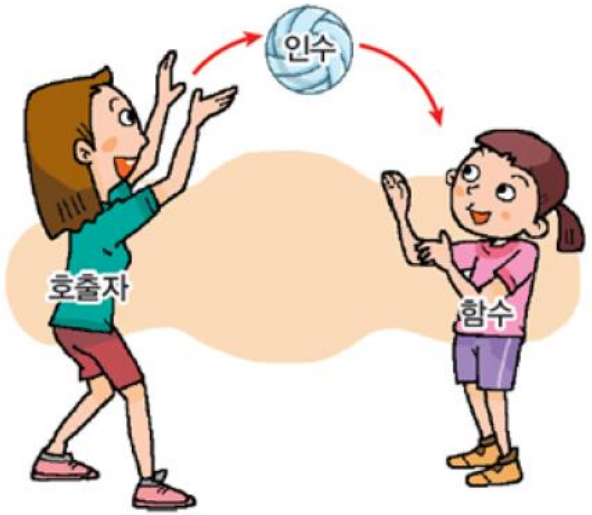
```
def print_address():
    print("서울특별시 종로구 1번지")
    print("파이썬 빌딩 7층")
    print("홍길동")

print_address()
print_address()
```

코드1, 코드2, 코드3 실행 결과

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

04. 함수에 1개의 인수 전달하기

- 사용자는 함수를 사용할 때 작업에 필요한 정보(값)을 전달할 수 있습니다.
- 인수(argument) : 호출 프로그램에 의하여 함수에 실제로 전달되는 값
- 매개변수(parameter) : 인수를 담아 함수 내에서 쓰이는 변수



04. 함수에 1개의 인수 전달하기

우편물을 받는 사람의 주소를 인쇄하는 print_address( ) 함수가 있는 프로그램을 작성해 봅시다.

코드

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

print_address("홍길동")
print_address("김코드")
print_address("나함수")
```

함수의 인수가 매개변수에 전달

실행 결과

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

05. 함수에 여러 개의 인수 전달하기

2개의 정수 start에서 end까지의 합을 계산하는 함수 get_sum( )을 작성해 보겠습니다.

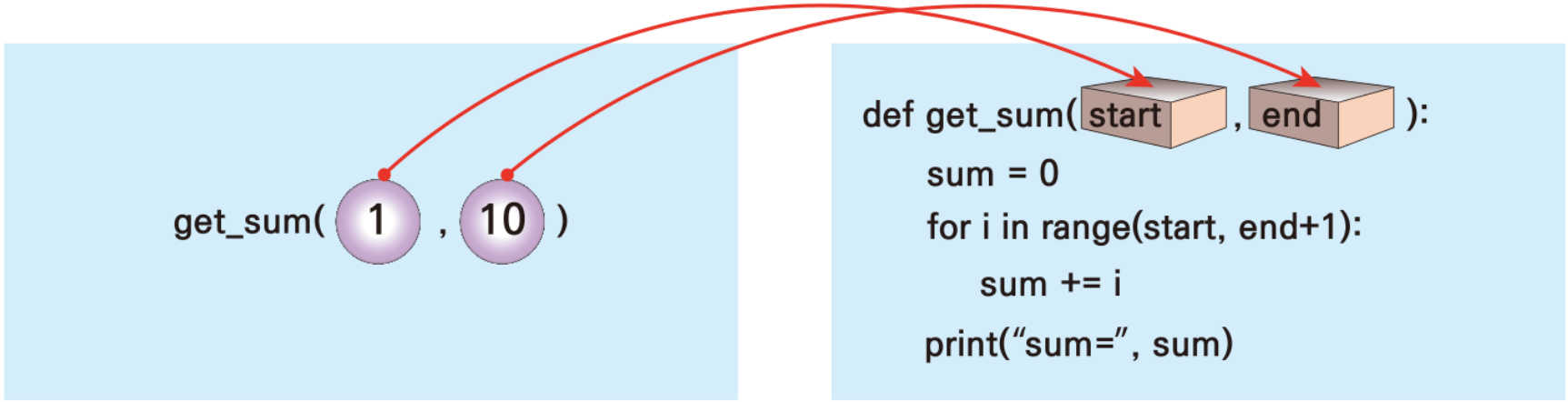
코드

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

get_sum(1, 10)   # 1과 10이 get_sum( )의 인수가 됩니다.
get_sum(1, 20)   # 1과 20이 get_sum( )의 인수가 됩니다.
```

실행 결과

```
sum= 55
sum= 210
```

05. 함수에 여러 개의 인수 전달하기



05. 함수에 여러 개의 인수 전달하기

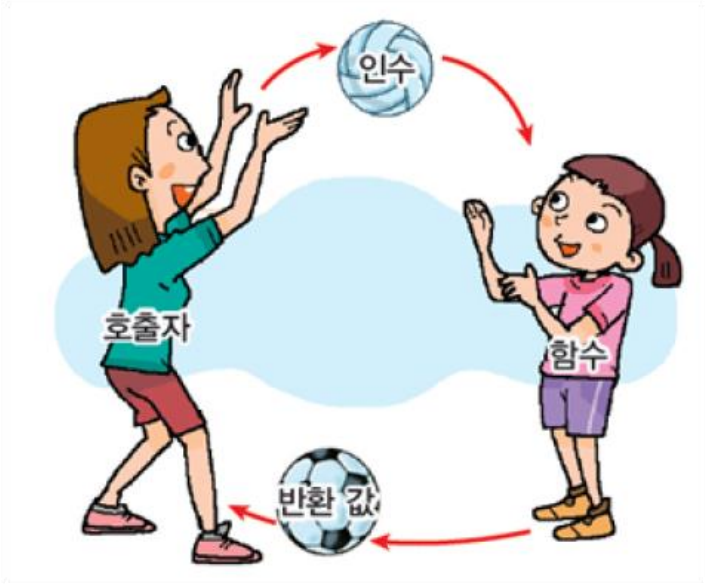
기본 매개변수(default parameter)

함수를 호출할 때 인수를 넘겨주지 않아도 인수가 자신의 기본값을 갖도록 하는 기능입니다.

```
def inc(a, step = 1) :
    print(a + step)

inc(10)
inc(10, 50)
```

06. 함수의 값 반환하기

- 반환 값(return value) : 함수로부터 되돌아오는 값



06. 함수의 값 반환하기

호출할 때 원의 반지름 값을 인수로 보내면 원의 면적을 계산해서 반환하는 함수를 작성해 보겠습니다.

코드

```
def calculate_area (radius):  # 함수의 정의
   area = 3.14 * radius**2
   return area                # 함숫값의 반환

c_area = calculate_area(5.0)  # 함수의 호출
print(c_area)

area_sum = calculate_area(5.0) + calculate_area(10.0)
                              # 수식에 적용된 함수 호출
print(area_sum)
```

실행 결과

```
78.5
392.5
```

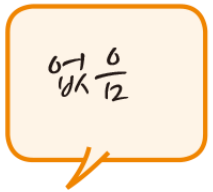
06. 함수의 값 반환하기

함수를 호출하여 수행하게 했어도, 그 반환 값을 사용하지 않을 수도 있습니다.

코드

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

calculate_area(5.0)           # 함수의 호출
```

실행 결과



06. 함수의 값 반환하기

파이썬에서는 함수가 여러 개의 값을 반환할 수 있습니다.

몇몇 다른 프로그래밍 언어는 함수가 반환하는 값을 단 한 개로 제한하기도 합니다. 하지만 파이썬에서는 다음과 같이 여러 개의 값을 반환할 수 있습니다.

코드

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

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

실행 결과

```
2 , 3
```

06. 함수의 값 반환하기

return만 홀로 쓸쓸히(?) 있는 경우도 있습니다.

코드

```
def judge(num) :
   if num % 2 == 0 :
      print("짝수")
      return      # 이 부분에서 종료됩니다.
   print("홀수")

num = int(input("자연수를 입력하세요: "))
judge(num)
```

실행 결과 1

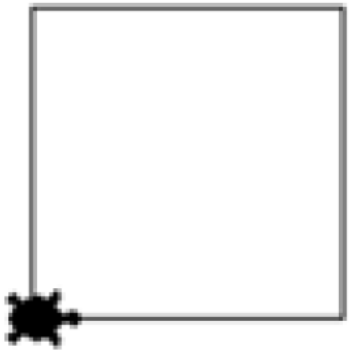
```
자연수를 입력하세요: 10
짝수
```

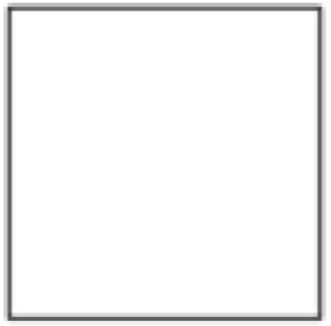
실행 결과 2

```
자연수를 입력하세요: 11
홀수
```

07. 함수를 좀 더 이해시켜 줄 예제

사각형 그리기 함수를 작성해 봅시다.





07. 함수를 좀 더 이해시켜 줄 예제

코드1-반복문 사용

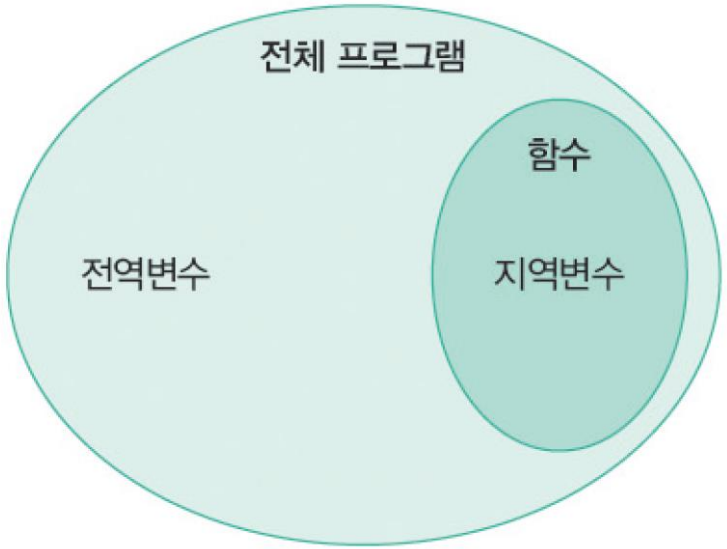
```
import turtle
t = turtle.Turtle( )
t.shape("turtle")

for i in range(4 ):
   t.forward(100)
   t.left(90)
t.up( )              # 펜을 든다.
t.goto(-200, 0)  # (-200, 0)으로 이동한다.
t.down( )           # 펜을 내린다.

for i in range(4 ):
   t.forward(100)
   t.left(90)
```

코드2-함수 사용

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

def square(length) :  # length는 한 변의 길이
   for i in range(4 ):
      t.forward(length)
      t.left(90)

square(100)      # square( ) 함수를 호출한다.
t.up( )              # 펜을 든다.
t.goto(-200, 0)  # (-200, 0)으로 이동한다.
t.down( )           # 펜을 내린다.
square(100)      # square( ) 함수를 호출한다.
```

08. 지역변수와 전역변수

- 변수가 사용되는 범위에 따라 지역변수(local variable)와 전역변수(global variable)로 나눔
- 지역변수 : 함수 안에서 생성되어 사용되는 변수, 함수가 종료되면 사라짐
- 전역변수 : 프로그램의 어디서나 사용할 수 있음
- 변수의 범위는 변수의 사용 가능 영역과 변수가 생존하는 시간과 관련됨



08. 지역변수와 전역변수

원의 면적을 계산하려는 다음 프로그램을 살펴보겠습니다.

코드

```
def calculate_area( ):
    result = 3.14 * r **2
    return result

r = float(input("원의 반지름: "))
area = calculate_area( )
print(area)
```

실행 결과

```
원의 반지름: 5.0
78.5
```

08. 지역변수와 전역변수

코드도 줄일 겸 'calculate_area( )' 안의 변수 result도 변수 r처럼 밖에서 함께 써보겠습니다.

코드

```
def calculate_area( ):
    result = 3.14 * r **2

r = float(input("원의 반지름: "))
calculate_area( )
print(result)
```

실행 결과

```
원의 반지름: 5.0
Traceback (most recent call last):
... (생략) ...
NameError: name 'result' is not defined
```

08. 지역변수와 전역변수

그렇다면 result를 전역변수로 만들어 보겠습니다.

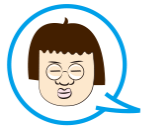
코드

```
result = 0                    # 전역변수 result
def calculate_area( ):
   result = 3.14 * r **2      # 지역변수 result

r = float(input("원의 반지름: "))
calculate_area( )
print(result)
```

실행 결과

```
원의 반지름: 5.0
0
```

08. 지역변수와 전역변수



calculate_area( ) 안에서 계산된 값을 전역변수 result에 저장해 보겠습니다.

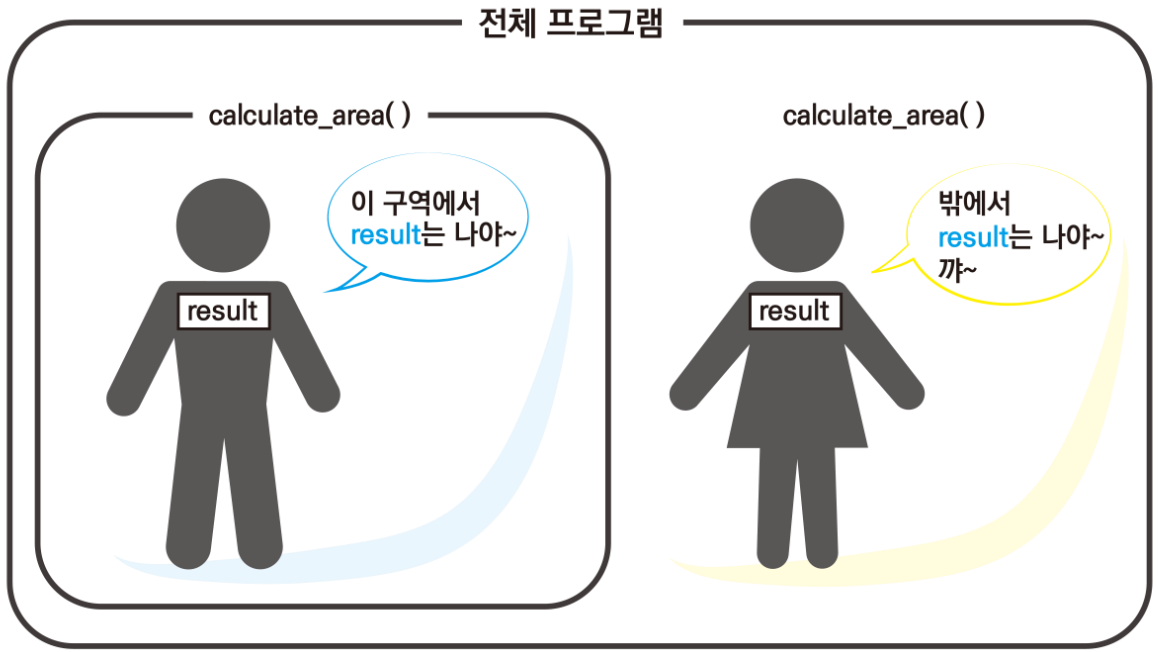
코드

```
result = 0
def calculate_area( ):
    global result
    result = 3.14 * r **2

r = float(input("원의 반지름: "))
calculate_area( )
print(result)
```

실행 결과

```
원의 반지름: 5.0
78.5
```

08. 지역변수와 전역변수



09. 디폴트 인수

우리에게 반갑게 인사를 하는 함수 greet( )를 살펴보겠습니다.

코드 1

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

greet("철수", "좋은 아침!")
```

실행 결과

```
안녕  철수, 좋은 아침!
```

코드 2

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

greet("철수")      # 매개변수 부족
```

실행 결과

```
... (생략) ...
TypeError: greet( ) missing 1 required positional argument: 'msg'
```

09. 디폴트 인수

만약 인수의 개수가 부족한 경우, 어느 정도는 기본값으로 매개변수를 처리해주는 장치가 있다면 편리할 것입니다.
바로 이것이 디폴트 인수입니다.

코드

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

greet("영희")
```

실행 결과

```
안녕  영희, 잘 지내죠?
```

10. 키워드 인수

- 위치 인수(positional argument) : 인수를 위치에 의해 구분
- 키워드 인수(keyword argument) : 인수 앞의 키워드에 따라 구분

코드

```
def calc(x, y):
   return x - y

print( calc(10, 20) )
print( calc(20, 10) )
```

실행 결과

```
-10
10
```

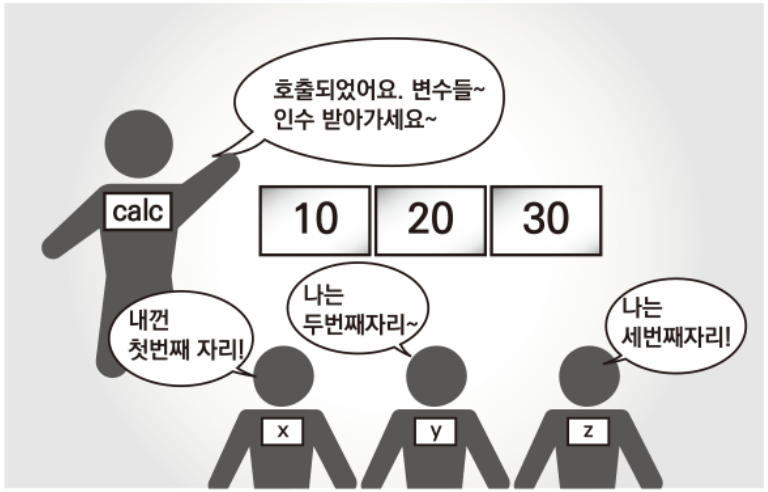
코드

```
def calc(x, y):
   return x - y

print( calc(x=10, y=20) )
print( calc(y=20, x=10) )
```

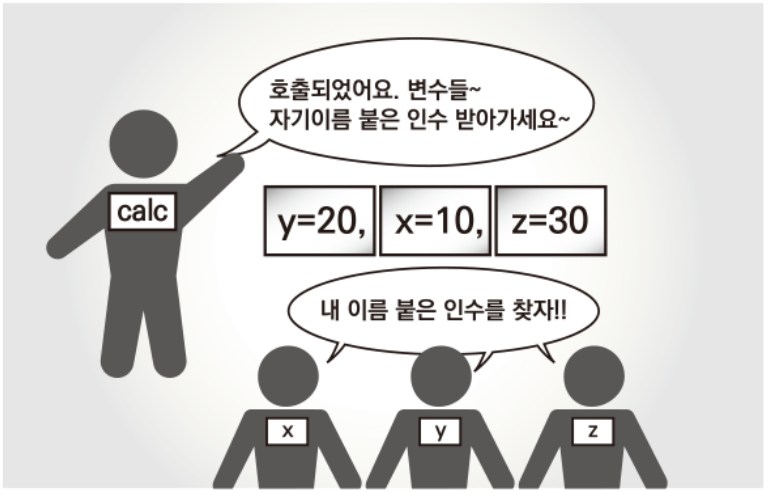
실행 결과

```
-10
-10
```

10. 키워드 인수



〈위치 인수 전달〉



〈키워드 인수 전달〉

10. 키워드 인수

키워드 인수 사용 시 주의사항!

코드

```
def calc(x, y, z) :
  return x + y + z

print( calc(10, y = 20, z = 30) )
```

실행 결과

```
60
```

코드

```
def calc(x, y, z) :
  return x + y + z

print( calc(x = 10, 20, 30) )
```

실행 결과

```
SyntaxError: positional argument
follows keyword argument
```

10. 키워드 인수

키워드 인수 전달의 경우 총정리!

키워드 인수의 내용을 예를 들어 정리하면 다음과 같습니다.

```
def func1(x, y, z) :
    return x * y * z
```

(호출 예)

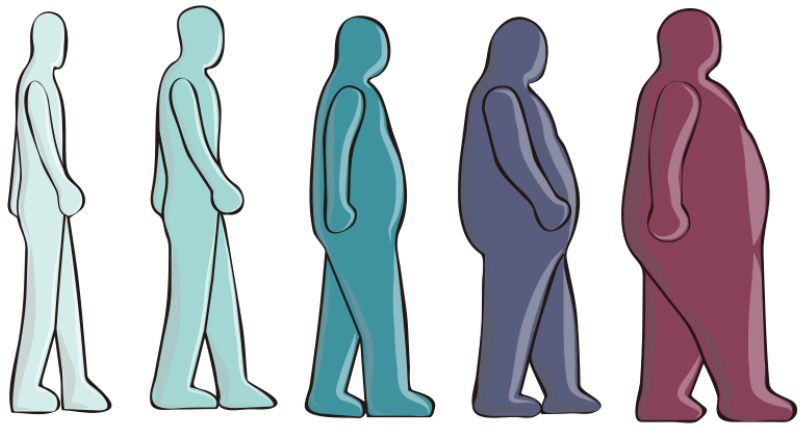
func1(1, 3, 5)	#15
func1(y = 7, z = 5, x = 2)	#70
func1(z = 2, x = 4, y = 5)	#40
func1(5, z = 10, y = 2)	#100
func1(z = 10, 20, x = 2)	#에러(위치 인수가 키워드 인수 뒤에 오면 안 됨)
func1(5, x = 2, z = 20)	#에러(x에 두 개의 값이 중복으로 할당된다는 에러)

BMI 계산기

체질량지수(BMI) 공식

$$BMI = \frac{몸무게(kg)}{키(m) \times 키(m)}$$

구분	BMI
저체중	<18.5
정상	18.5~22.9
과체중	23~24.9
경도비만	25~29.9
고도비만	30 이상



저체중, 정상, 과체중, 경도비만, 고도비만

BMI 계산기

실행 결과

```
키를 m단위로 입력하세요: 1.60
몸무게를 kg단위로 입력하세요: 47
당신은 저체중입니다.
```

생각 1 : 프로그램의 순서를 생각해 봅니다.

사용자로부터 키와 몸무게 입력받기 → BMI(키, 몸무게) 함수 체질량지수 계산 후 결과 반환 →

result(체질량지수) 호출 결과 출력

환전 계산기

세계지도 | 세계시간 | 국가번호 | 환율

원달러 미국USD

1182.5 ▲3.30 (+0.27%)

3개월 1년 3년 10년

하나은행

하나은행 신한은행

매매기준율 환율우대없음

미국 USD

1

1 달러

대한민국 KRW

1182.5

1,182.5 원

통화명	매매기준율	전일대비	등락률
미국 USD	1,182.5	▲3.30	+0.27%
일본 JPY 100	1,078.14	▲15.57	+1.41%
유럽연합 EUR	1,286.74	▲10.60	+0.80%
중국 CNY	169.22	▲0.48	+0.28%

환전 계산기

$$\text{환전금액} = \frac{\text{한국돈}}{\text{매매기준율}}$$

실행 결과 1

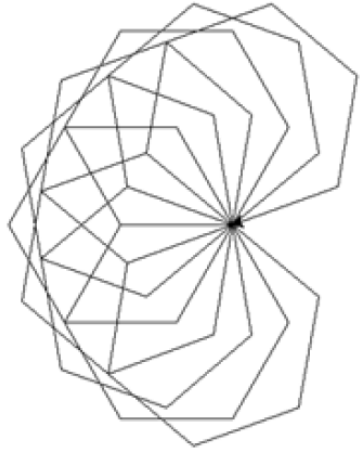
환전 금액(원)을 입력하세요: 1000
국가를 입력하세요: 중국
1000 원은 5.91 위안 입니다

실행 결과 2

환전 금액(원)을 입력하세요: 1000
국가를 입력하세요: 태국
해당 국가 정보가 없습니다.

N각형을 그리는 함수 작성하기

n각형을 그리는 함수를 작성하여 활발하게 사용해보겠습니다. 함수의 이름은 n_polygon( )라고 하여 n각형으로 아래와 같이 10번 그려주는 프로그램을 작성해 보겠습니다.

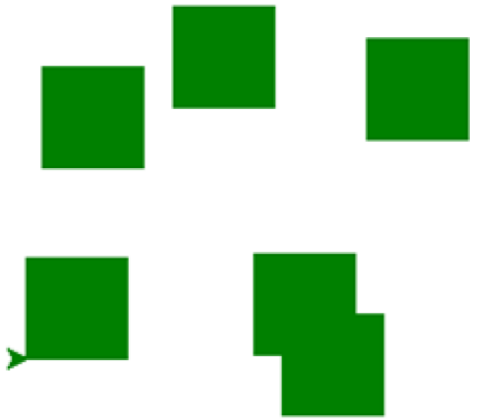


생각 1 : 프로그램의 순서를 생각해 봅니다.

터틀 그래픽을 사용할 준비 작업 → n_polygon( ) 함수 정의

→ n_polygon( ) 함수 n번 반복 호출

클릭하는 곳에 사각형 그리기

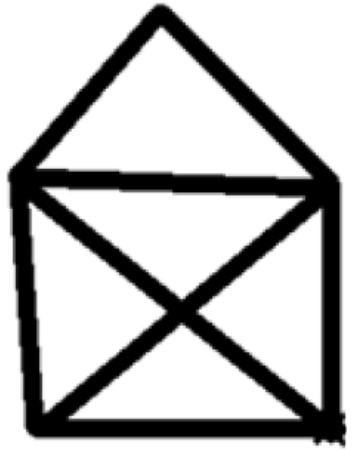


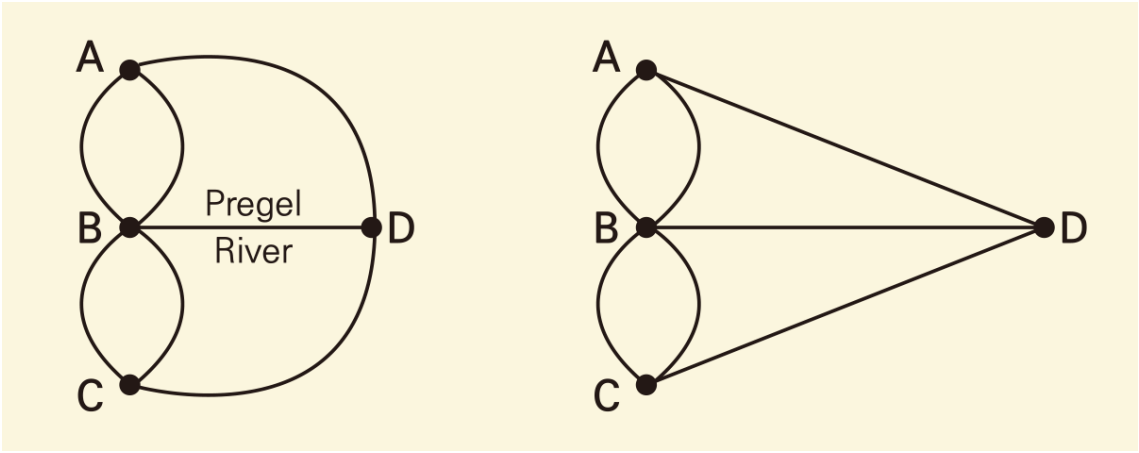
생각 1 : 프로그램의 순서를 생각해봅니다.

터틀 그래픽을 사용할 준비 작업 → 마우스가 클릭되는 것 감지 → square( ) 함수 호출

한 붓 그리기

한붓 그리기는 도형이나 그림을 한 점에서 그리기 시작하여 선을 한 번도 떼지 않고 같은 선을 두 번 반복해서 지나지 않도록 하여 그림을 완성하는 작업입니다. 터틀 그래픽에서 클릭된 위치까지 터틀이 직선을 그리게 하여 한붓 그리기를 시뮬레이션해보는 프로그램을 작성해 보겠습니다.





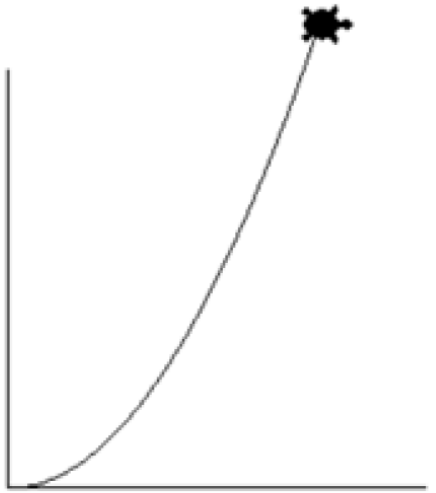
이차함수 그래프 그리기

함수 $f(x)=x^2+1$을 계산하는 함수를 작성하고 이 함수를 이용하여 화면에 f(x)의 그래프를 그리는 프로그램을 작성해 보겠습니다

생각 1 : 프로그램의 순서를 생각해 봅니다.

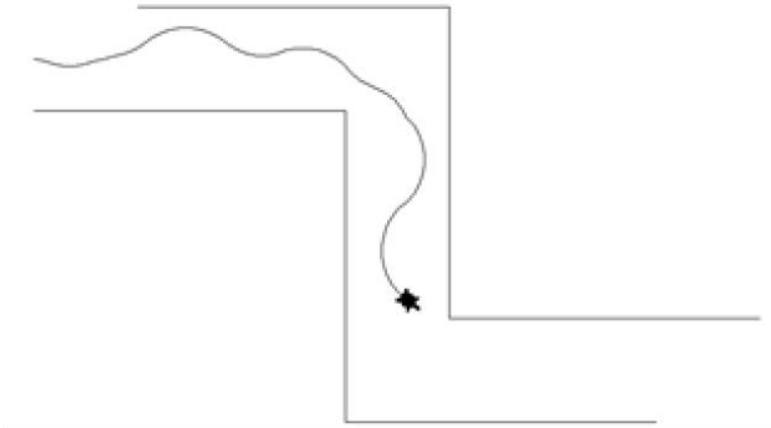
터틀 그래픽을 사용할 준비 작업 → x축 그리기

→ 이차함수 그래프 그리기



테세우스 터틀 미로 탈출 게임

반인반수의 괴물 미노타우로스가 있는 미로를 테세우스 터틀은 빠져나와야만 게임을 끝낼 수 있습니다. 테세우스 터틀이 화면에 그려진 미로의 벽에 닿지 않게 화살표 키로 조종하여 미로를 탈출하는 프로그램을 작성해 보겠습니다



재귀호출

- 함수는 내부에서 다시 자기 자신을 호출할 수 있습니다. 이것을 재귀호출(recursion)이라고 합니다
- 재귀(recursive) : 어떤 문제를 해결하기 위해 동일한 문제를 더 작은 경우로 쪼개고 쪼개어 문제를 바로 풀 수 있도록 간단해질 때까지 나누어 해결하는 방법
- 프로그래밍에서는 이 기법을 함수가 내부에서 문제의 범위를 줄여 자기 자신을 다시 호출하고 다시 그 안에서 다시 자기 자신을 호출하는 재귀호출이라는 방식으로 구현할 수 있습니다.

$$n!=n \times (n-1) \times (n-2) \times \cdots 3 \times 2 \times 1$$

프랙털 나무 그리기

- 프랙털(fractal) 이론은 1975년 망델브로(Mandelbrot)라는 수학자에 의해서 시작
- 프랙털이란 자신의 작은 부분에 자신과 닮은 모습이 나타나고 그 안의 작은 부분에도 자신과 닮은 모습이 무한히 반복되어 나타나는 현상
- 프랙털은 컴퓨터 그래픽 이론에서 출발하여 현대 물리와 수학에서 빼놓을 수 없는 부분