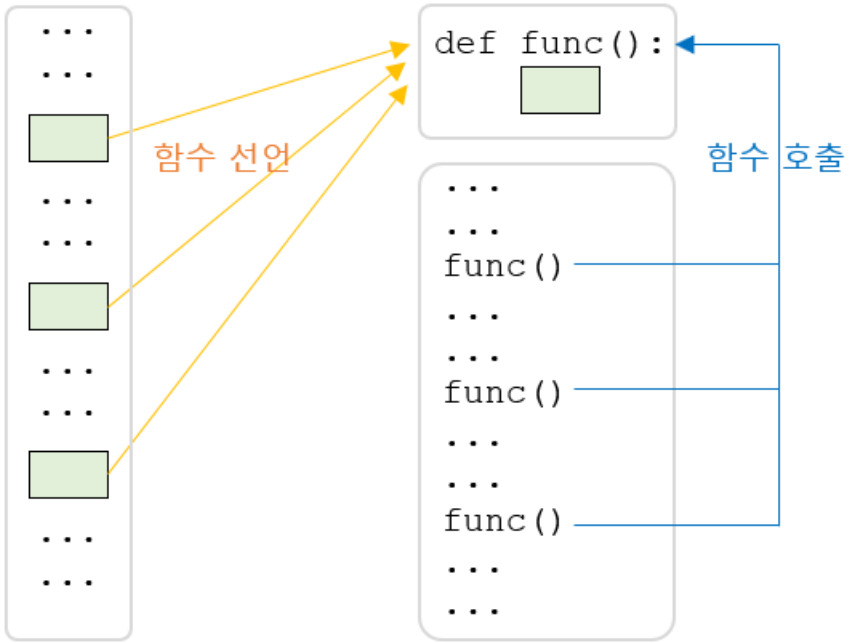
함수

함수(Function)

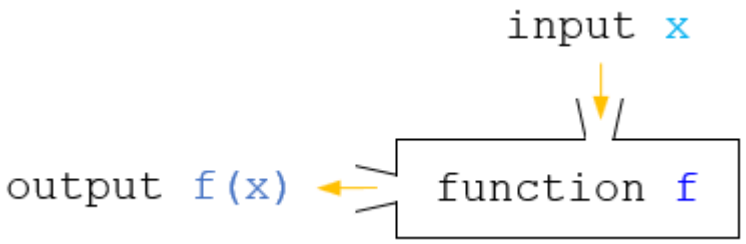
❖ 함수(function)의 필요성

- 프로그램 코드를 작성하는 과정에서 특정 기능을 수행하는 코드 부분을 여러 곳에서 자주 사용하는 경우가 있음
- 특정 기능의 코드 부분을 한데 묶어 이름을 붙여 둔 후, 필요한 곳에서 이름만을 사용하여 특정 기능의 코드 부분을 사용할 수 있음



함수(Function)

- 함수(function)
  - 특정 기능을 수행하는 코드의 묶음에 이름을 붙여 놓은 것
  - 큰 프로그램의 작은 프로그램 조각(모듈)과 같음
  - 함수는 입력을 받아 함수 내부에서 계산 등의 처리를 한 후 결과를 함수 밖으로 반환함



함수(Function)

❖함수의 구분

- **사용자 정의 함수(user-defined function)**
  - 사용자가 직접 만들어 사용하는 함수
- **내장함수(built-in function)**
  - 파이썬에서 미리 만들어져 제공되는 input(), print() 함수 등

abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	hex()	next()	slice()
ascii()	divmod()	id()	object()	sorted()
bin()	enumerate()	input()	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	

https://docs.python.org/3/library/functions.html

함수의 선언 및 호출

- 사용자 정의 함수
  - def 예약어를 이용하여 정의
  - 매개변수(parameter)가 사용되지 않고 함수 결과를 반환하지 않는 사용자 정의 함수의 기본 형태

```
def 함수이름():
    문장들
```

함수의 선언 및 호출

❖사용자 정의 함수

- **"Python" 문자열과 "파이썬" 문자열을 각각 출력하는 fpython() 함수의 정의**
  - 함수이름은 함수를 호출할 때 사용
  - 들여쓰기에 의해 함수의 시작과 끝 정의
  - 들여쓰기가 된 첫 번째 print() 함수부터 같은 들여쓰기가 된 두 번째 print() 함수까지 함수의 몸체 구성

```
def fpython():
    print("Python")
    print("파이썬")
```

  - 두 번째 print() 함수의 들여쓰기가 첫 번째 print() 함수와 다를 경우 함수의 몸체 부분은 print("Python") 만 해당됨

```
def fpython():
    print("Python")
print("파이썬")
```

함수의 선언 및 호출

❖사용자 정의 함수

- **함수의 호출(call)**
  - 함수 이름에 의한 함수의 사용
  - 정의한 fpython() 함수를 호출하면 fpython() 함수 내의 두 print() 함수가 순서대로 실행되어 값 출력
  - 들여쓰기를 서로 다르게 작성한 ifpython() 함수의 경우 print("파이썬") 함수가 실행되고, ifpython() 함수가 호출되어 실행

```
def fpython():
    print("Python")
    print("파이썬")
fpython()
```

실행결과

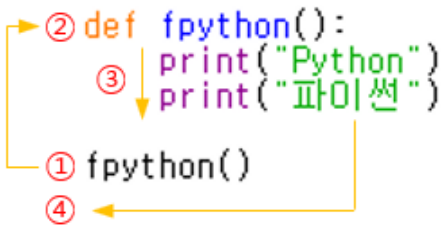
```
Python
파이썬
```

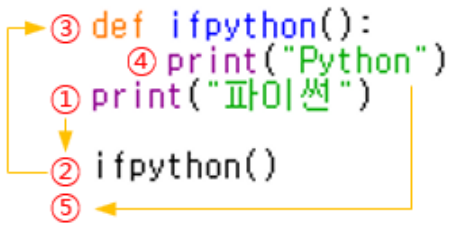
```
def ifpython():
    print("Python")
print("파이썬")

ifpython()
```

실행결과

```
파이썬
Python
```





함수의 선언 및 호출

TIP

함수 정의 전에 함수를 호출한 경우 오류 발생

함수를 정의하고 호출하는 과정에서 주의할 점은 함수를 호출하기 전에 위의 본문과 같이 반드시 먼저 함수가 정의되어야 한다. 만약 다음과 같이 함수 호출이 함수 정의보다 먼저 나타나면 해당 함수가 정의되지 않았다는 오류 메시지가 나타난다.

```
fpython()

def fpython():
    print("Python")
    print("파이썬")

=====
NameError: name 'fpython' is not defined
```

함수의 선언 및 호출

프로그램 학번, 성명 출력하기

다음 동작들을 순서대로 작성해보자.
① print() 함수를 이용하여 순서대로 학번과 성명을 2회 출력하시오.
② for문을 이용하여 학번과 성명을 2회 출력하시오.
③ 학번과 성명을 출력하는 sn( )함수를 만든 후 sn( )함수를 2회 호출하시오.
④for문을 이용하여 sn( )함수를 2회 호출하시오.

①
```
print("12345678")
print("홍길동")
print("12345678")
print("홍길동")
print("")
```

②
```
for i in range(2):
    print("12345678")
    print("홍길동")
print("")
```

실행결과

①
```
12345678
홍길동
12345678
홍길동
```

②
```
12345678
홍길동
12345678
홍길동
```

③
```
def sn():
    print("12345678")
    print("홍길동")

sn()
sn()
print("")
```

④
```
for i in range(2):
    sn()
```

③
```
12345678
홍길동
12345678
홍길동
```

④
```
12345678
홍길동
12345678
홍길동
```

함수의 선언 및 호출

>>> 잠깐! Coding

for문을 이용하여 1부터 9까지 출력하는 함수 print19( )를 작성하고 호출해보자.

for 문을 이용하여 1부터 9까지 출력하는 함수를 def print19()로 정의하고, 정의된 함수 print19()를 호출한다.

```
def print19():
    for i in range(1, 10):
        print(i, end=" ")
    print("")

print19()
```

실행결과

```
1 2 3 4 5 6 7 8 9
```

함수의 선언 및 호출

❖ 함수에 값 전달하기

- **인수(argument)**
  - 함수를 호출할 때 함수로 전달되는 값
- **매개변수(parameter)**
  - 함수에서 전달된 값을 받는 변수
- **인수와 매개변수는 함수를 호출할 때 데이터를 주고받기 위하여 필요**

```
def 함수이름(매개변수1, 매개변수2, ...):
    문장들
```

함수의 선언 및 호출

❖함수에 값 전달하기

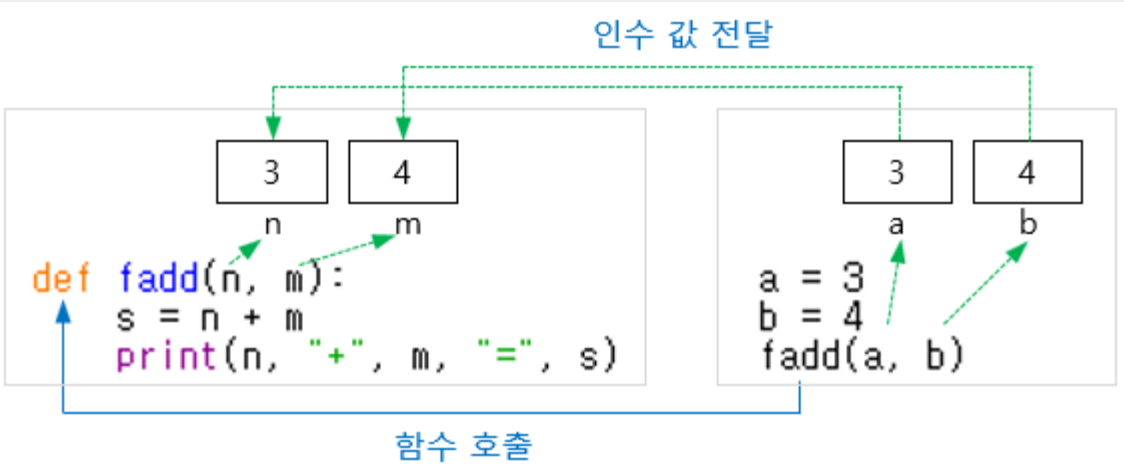
- **fadd() 함수**
  - 두 정수 값을 전달 받아 더한 후 값을 출력하는 사용자 정의 함수
  - fadd(a, b)로 함수를 호출하여 실행

```
def fadd(n, m):
    s = n + m
    print(n, "+", m, "=", s)

a = 3
b = 4
fadd(a, b)
```

실행결과

```
3 + 4 = 7
```



함수의 선언 및 호출

프로그램 구구단의 단 계산하기

사용자로부터 계산할 단을 입력받아 해당 구구단의 단을 계산하여 출력해보자.

```
def calc_gugudan(dan):
    for i in range(1, 10):
        print(dan, "*", i, "=", dan*i, "")

d = int(input("단 : "))
calc_gugudan(d)
```

실행결과

```
단 : 3
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
```

함수의 선언 및 호출

>>> 잠깐! Coding

앞의 프로그램을 이용하여 입력되는 단의 값을 1부터 9까지로 제한하도록 변경해 보자.

단의 값을 1부터 9까지 제한하는 방법
1. 입력한 후, 바로 if문에 의해 입력된 값을 검사한다.

```
def calc_gugudan(dan):
    for i in range(1, 10):
        print(dan, "*", i, "=", dan*i, "")

d = int(input("단 : "))
if d >=1 and d <= 9:
    calc_gugudan(d)
else:
    print("단은 1~9까지 입력해주세요.")
```

실행결과

```
단 : 10
단은 1~9까지 입력해주세요.
```

함수의 선언 및 호출

>>> 잠깐! Coding

앞의 프로그램을 이용하여 입력되는 단의 값을 1부터 9까지로 제한하도록 변경해 보자.

단의 값을 1부터 9까지 제한하는 방법
1. 입력한 후, 바로 if문에 의해 입력된 값을 검사한다.
2. 함수 내에서 검사한다.

```
def calc_gugudan(dan):
    if dan >=1 and dan <= 9:
        for i in range(1, 10):
            print(dan, "*", i, "=", dan*i,"")
    else:
        print("단은 1~9까지 입력해주세요.")
d = int(input("단 : "))
calc_gugudan(d)
```

실행결과

```
단 : 0
단은 1~9까지 입력해주세요.
```

함수의 선언 및 호출

>>> 잠깐! Coding

시작에 해당하는 정수를 입력받아 변수 s에 대입하고, 끝에 해당하는 정수를 입력받아 e에 대입한 후, for문을 이용하여 start부터 end까지 출력하는 함수 print_se(start,end)를 작성하고 호출해보자.(단, 입력한 후 변수 s의 값이 변수 e의 값보다 작을 때 만 함수를 호출한다.)

```
def print19(st, ed):
    for i in range(st, ed+1):
        print(i, end=" ")
    print("")

s = int(input("시작값 : "))
e = int(input("끝값 : "))
if s < e:
    print19(s, e)
else:
      print("시작값이  끝값보다  작아야
합니다.")
```

실행결과

```
시작값 : 10
끝값 : 2
시작값이 끝값보다 작아야 합니다.

시작값 : 2
끝값 : 10
2 3 4 5 6 7 8 9 10
```

함수의 선언 및 호출

❖ 함수의 결과 반환받기

- **return 예약어**
  - 함수 밖으로 함수의 결과 값을 반환

```
def 함수이름(매개변수1, 매개변수2, ...):
    문장들
    return 결과값
```

함수의 선언 및 호출

❖함수의 결과 반환 받기

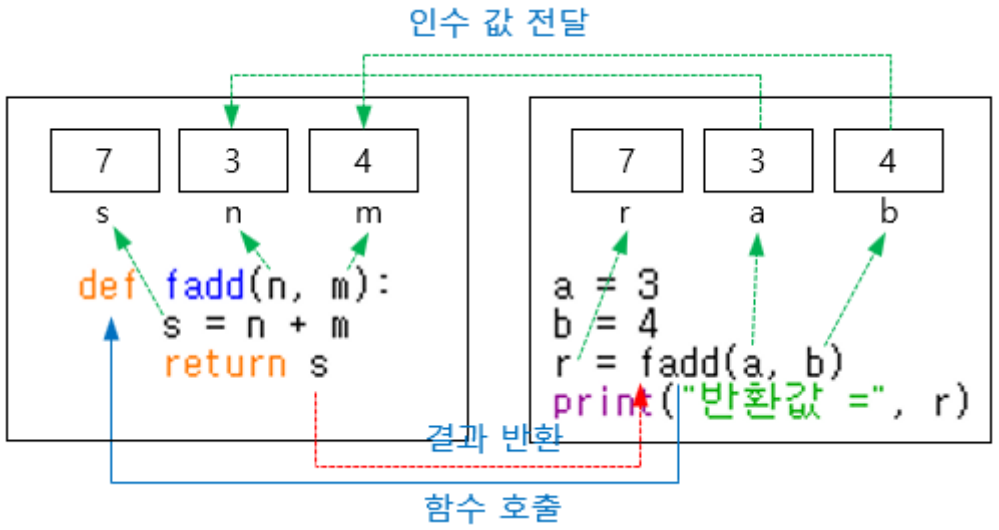
- **fadd() 함수**
  - 두 정수 값을 전달 받아 더한 후 값을 반환하는 사용자 정의 함수
  - fadd(a, b)로 함수를 호출하여 결과를 반환 받아 출력

```
def fadd(n, m):
    s = n + m
    return s

a = 3
b = 4
r = fadd(a, b)
print("반환값 =", r)
```

실행결과

```
반환값 = 7
```



함수의 선언 및 호출

❖함수의 결과 반환 받기

- **fadd() 함수**
  - a=3, b=4일 때
  - r = fadd(a, b) → 결과값 7이 반환되어 변수 r에 대입됨
  - print("반환값 =", fadd(a, b)) → 결과값 7이 print()의 인수로 사용
  - c = 2 + fadd(a, b) → 결과값 7이 수식의 피연산자로 사용되어 계산됨
  - fadd(a, b) → 결과값 7이 반환되었지만 값을 사용하지 않게 됨

함수의 선언 및 호출

프로그램 두 수의 평균값 구하기

사용자로부터 입력받은 두 수의 평균값을 구하여 반환하는 avg( )함수를 만들고 호출해보자. 또한 avg( ) 함수의 결과를 반환받아 출력해보자.

```
def avg(a, b):
    s = (a + b) / 2
    return s

in1 = int(input("값1 : "))
in2 = int(input("값2 : "))
r = avg(in1, in2)
print("평균 =", r)
```

실행결과

```
값1 : 3
값2 : 7
평균 = 5.0
```

함수의 선언 및 호출

>>> 잠깐! Coding

앞의 프로그램을 이용하여 입력값이 3개일 때의 평균을 구하도록 avg( )함수를 변경해보자.

입력값이 3개일 때 평균을 구하려면, 3번째 입력을 저장할 변수를 추가하고,avg(a,b,c)형태로 함수 선언을 변경해야한다.

```
def avg(a, b, c):
    s = (a + b + c) / 3
    return s

in1 = int(input("값1 : "))
in2 = int(input("값2 : "))
in3 = int(input("값3 : "))
r = avg(in1, in2, in3)
print("평균 =", r)
```

실행결과

```
값1 : 3
값2 : 7
값3 : 5
평균 = 5.0
```