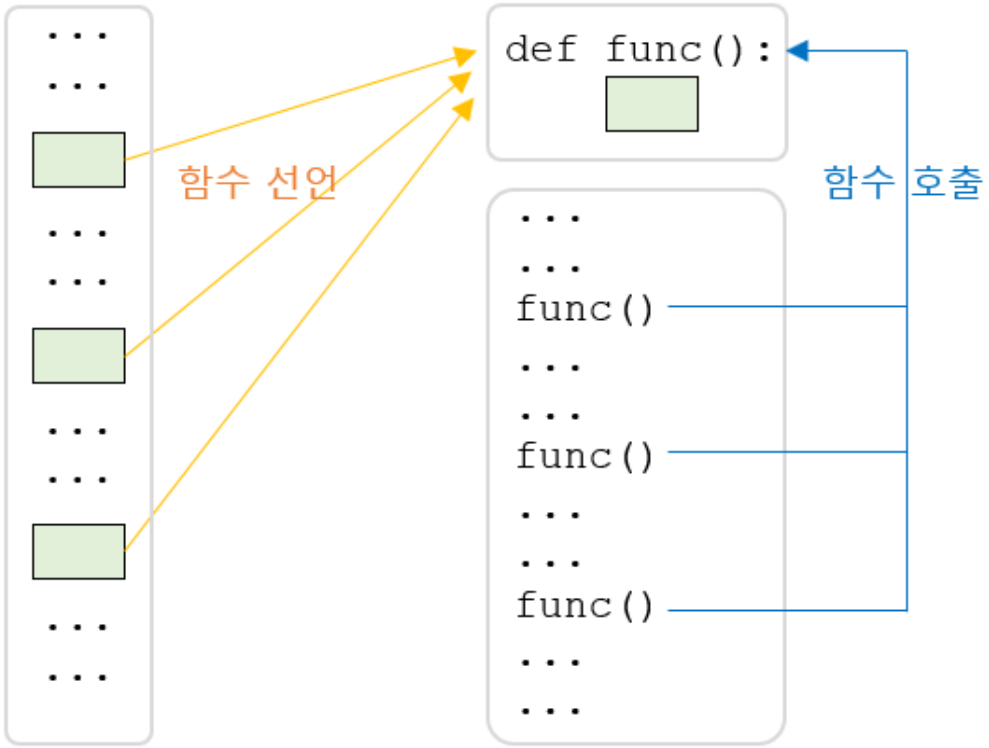
알고리즘 사고와 함수

09강

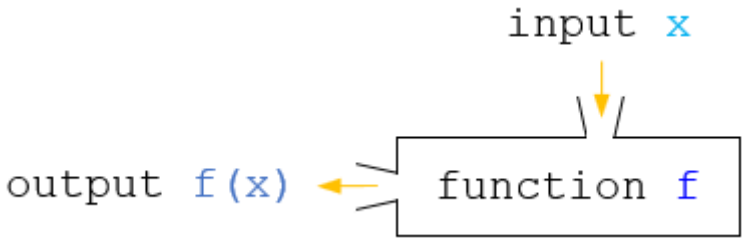
함수의 필요성

- 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)

- 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



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()	dict()	help()	min()	setattr()
all()	dir()	hex()	next()	slice()
any()	divmod()	id()	object()	sorted()
ascii()	enumerate()	input()	oct()	staticmethod()
bin()	eval()	int()	open()	str()
bool()	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()	
delattr()	hash()	memoryview()	set()	

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

사용자 정의 함수 생성

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

```
def 함수이름():
    문장들
```

- "Python"과 "파이썬" 문자열을 출력하는 fpython() 함수의 정의
  - 함수 이름은 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

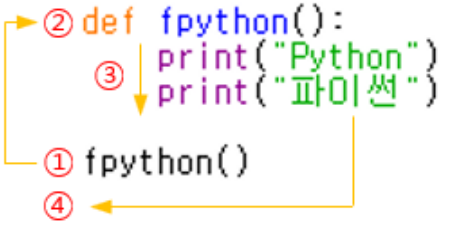
```
def ifpython():
    print('Python')
print('파이썬')
```

사용자 정의 함수 호출 (call)

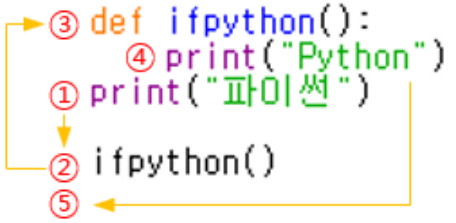
- 함수 이름에 의한 함수의 사용
- 정의한 fpython() 함수를 호출하면 fpython() 함수 내의 두 print() 함수가 순서대로 실행되어 값 출력
- 들여쓰기를 서로 다르게 작성한 ifpython() 함수의 경우 print("파이썬") 함수가 실행되고, ifpython() 함수가 호출되어 실행

```
fpython()

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

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

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




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



사용자 정의 기본함수 실습

- 다음 동작을 순서대로 작성하시오.
  - ① print() 함수를 이용하여 순서대로 학번과 성명을 2회 출력하시오.
  - ② for 문을 이용하여 학번과 성명을 2회 출력하시오.
  - ③ 학번과 성명을 출력하는 sn() 함수를 만든 후 sn() 함수를 2회 호출하시오.
  - ④ for 문을 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("")
```

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

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

④
```
for i in range(2):
    sn()
print("")
```

사용자 정의 기본함수 실습

- 1부터 9까지 출력하는 함수 print19()를 for문을 이용하여 작성하여 2회 호출하시오.

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

print19()
print19()
```

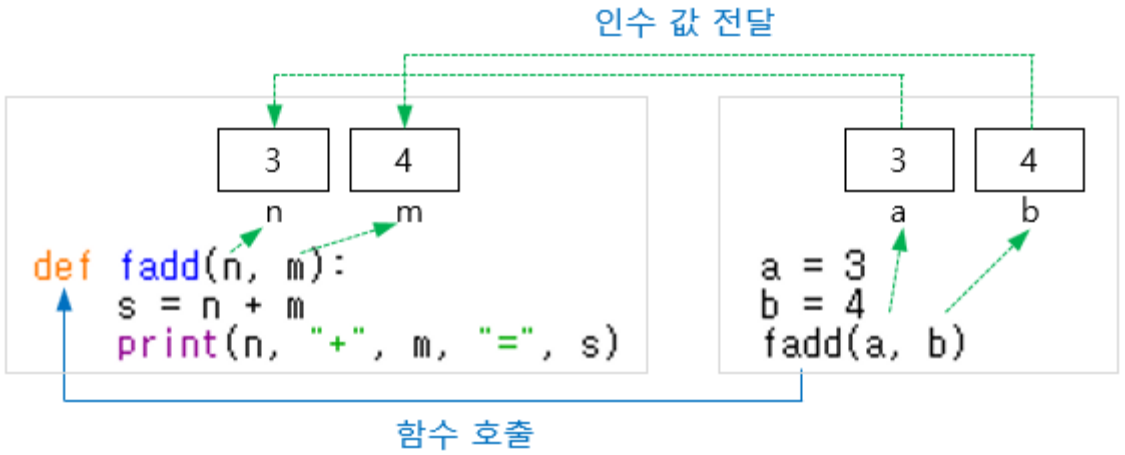
함수에 값 전달하기

- **인수(argument)**
  - 함수를 호출할 때 함수로 전달되는 값
- **매개변수(parameter)**
  - 함수에서 전달된 값을 받는 변수

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

- 인수와 매개변수는 함수를 호출할 때 데이터를 주고받기 위하여 필요
- 예제: 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)
```



함수 인수 전달 실습

- 사용자로부터 계산할 구구단의 단수를 입력 받아 해당 구구단을 계산 출력하시오.

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

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

- 입력되는 단의 값을 1부터 9까지로 제한하도록 변경하시오.

```
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 입력해 주세요.')
```

```
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 입력해 주세요.')

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

함수 인수 전달 실습

- 시작에 해당되는 정수를 입력 받아 변수 s에 대입하고, 끝에 해당하는 정수를 입력받아 변수 e에 대입한 후 for문을 이용하여 start부터 end까지 출력하는 함수 print19(start, end)를 작성하고 호출해 보자. (단, 입력한 변수 s의 값이 변수 e의 값 보다 작을 때만 함수를 호출함)

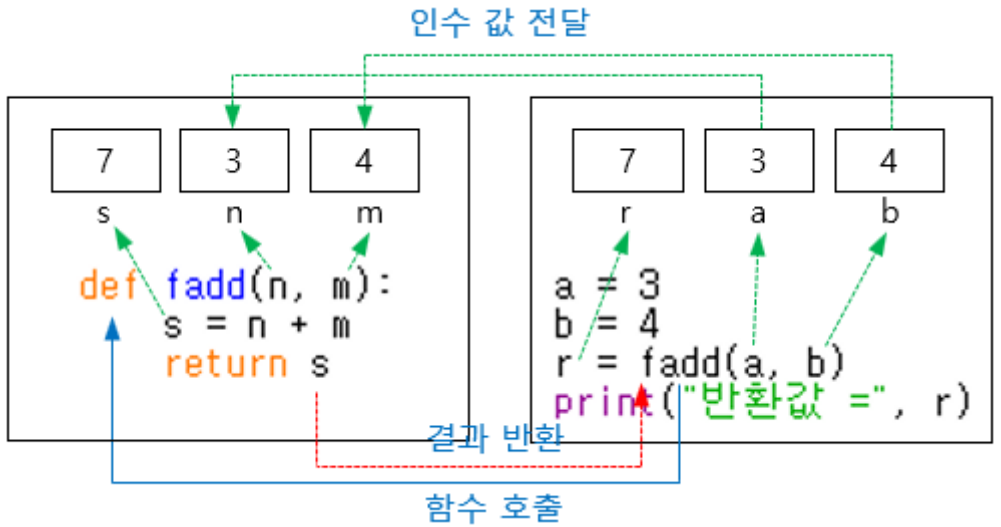
```
def print19(st, ed):
    for i in range(st, ed+1):
        print(i, end=" ")
    print("")

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

함수 결과 반환 (return)

- return 예약어 : 함수 밖으로 함수의 결과 값 반환
- 예제 :
  - fadd() 함수
  - 두 정수 값을 전달 받아 더한 후 값을 반환하는 사용자 정의 함수
  - fadd(a, b)로 함수를 호출하여 결과를 반환 받아 출력
  - 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

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

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

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



함수 결과 반환 실습

- 사용자로부터 입력 받은 두 수의 평균값을 구하여 반환하는 avg() 함수를 만들고 호출. 또한 avg() 함수의 결과를 반환 받아 출력

```
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)
```

- 입력 값이 3개일 때의 평균을 구하는 avg() 함수로 변경하여 평균 계산을 수행

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

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

EOD