

파이썬 기본

1. 데이터 타입과 변수, 그리고 출력
2. 문자열 다루기 기본과 리스트 데이터 구조
3. 조건문
4. 반복문
5. 함수
6. 다양한 데이터 구조: 튜플, 딕셔너리, 집합

이외에 필요한 부분은 실제 프로그래밍을 하며 필요할 때 익힙니다.!

- 객체와 클래스, 라이브러리, 문자열등등

반복문 (for, while)

```
for 변수 in (리스트 or 문자열):  
    실행문1  
    ...
```

```
for i in ["python", "java", "golang"]:  
    print(i)
```

반복문 (for, while)

1부터 10까지 합한 값은?

```
sum = 0
for i in range(1,11):
    sum += i
    sum = sum + i
    print(sum)
```

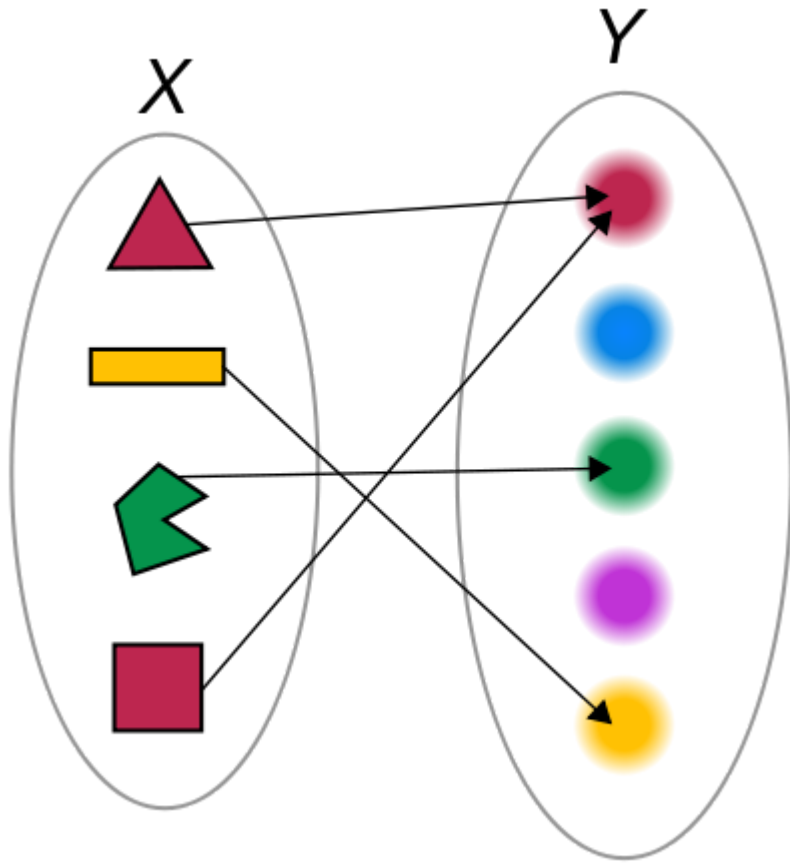
반복문 (for, while)

```
while 조건:  
    실행문1  
    ...
```

```
while name != "foo bar":  
    name = input("What's your name? ")  
    print("Hi, " + name + "So, where is foo bar?")
```

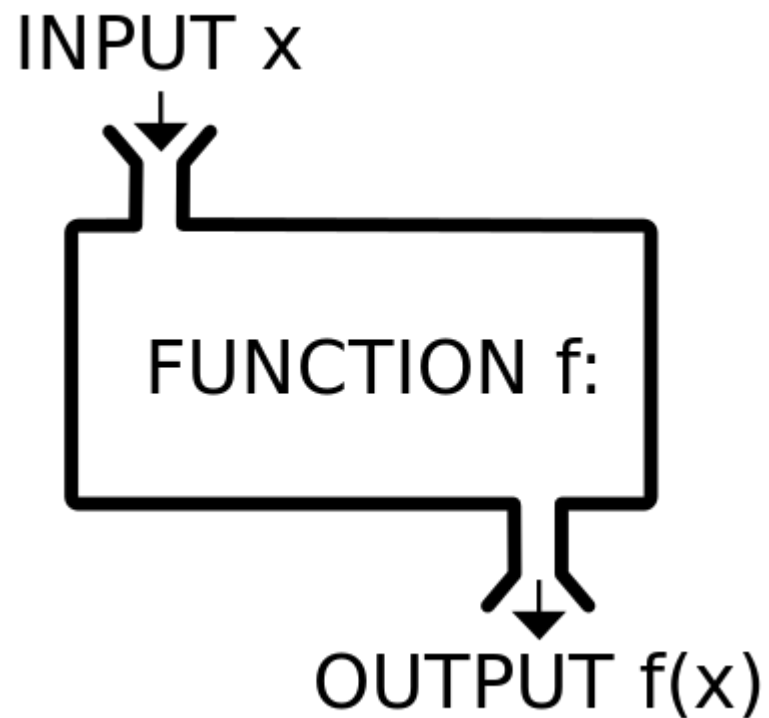
```
while 1:  
    print("Hello world!")
```

함수 (function)



- 수학적 정의: 첫 번째 집합의 임의의 한 원소를 두 번째 집합의 오직 한 원소에 대응시키는 대응 관계
- x : 정의역 y : 공역

함수 (function)



- 프로그래밍에서의 함수: 입력값을 내부에서 어떤 처리를 통해 결과값을 출력하는 것
- 용어: 인자(argument, parameter) 또는 인풋, 결과값 (또는 리턴값)

함수 (function)

```
def function(parameter):  
    실행문1  
    실행문2  
    ...  
    return output
```

함수 (function)

```
def awe_sum(a,b):  
    result = a + b  
    return result  
  
a = 2  
b = 3  
print(awe_sum(a,b))
```


함수 (function without input)

```
def print_hello():  
    return "hello"  
  
result_hello = print_hello()  
print(result_hello)
```

함수 (function without return)

```
def func_wo_return(a):  
    print("This is function without return for " + str(a) + " times.")  
  
func_wo_return(1)
```

함수 (function with multiple return)

```
def mul_return(a):  
    b = a + 1  
    return a, b
```

함수 (자주 사용하는 return 사용법)

```
def id_check(id):  
    if id == "admin":  
        print("invalid id: admin")  
        return  
    print("valid id: ", id)
```

데이터 구조 (List, Tuple)

List

```
animals = [' ', ' ', ' ']
```

Tuple

```
animals = (' ', ' ', ' ')
```

데이터 구조 (Tuple)

데이터 구조: 리스트, 튜플, 딕셔너리, 집합

Tuple은 괄호를 이용해 선언할 수 있습니다.

```
tuple1 = (1, 2, 3, 4)
```

tuple은 삭제나 추가가 불가능합니다.

```
del tuple[1]  
tuple1[1] = 'c'
```

tuple끼리 더하거나 반복하는 것은 가능합니다.

```
tuple2 = (5, 6)
```

```
print(tuple1 + tuple2)
```

```
print(tuple1 * 3)
```

tuple 덕분에 변수끼리 값을 편하게 바꿀 수 있습니다.

```
x = y
y = x (x)

temp = x
x = y
y = temp

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

tuple 덕분에 함수에서 하나 이상의 값을 반환할 수도 있습니다.

```
def quot_and_rem(x, y):
    quot = x // y
    rem = x % y
    return (quot, rem)

(quot, rem) = quot_and_rem(3, 10)
```

List <-> Tuple

```
list((1,2))  
tuple([1,2])
```


데이터 구조 (dictionary의 선언)

```
dict1 = {}  
print(dict1)
```

dictionary는 **key**와 **value**로 이루어져 있으며, 추가하는 법은 다음과 같습니다.

```
dict1 = {'name': 'foo bar'}  
print(dict1)
```

```
dict1 = {'korean': 95, 'math': 100, 'science': [80, 70, 90, 60]}  
print(dict1)
```

```
dict1['english'] = "pass"  
print(dict1)
```

요소 삭제는 **del**을 활용합니다.

```
del dict1['math']  
print(dict1)
```

key를 활용해 value를 출력하는 법을 알아보시다.

```
print(dict1['korean'])
```

key만 출력하는 법을 알아보시다.

```
print(dict1.keys())
```

value만 출력할때 이렇게 합니다.

```
print(dict1.values())
```

key와 value를 함께 출력합니다.

```
print(dict1.items())
```

데이터 구조 (집합 또는 set)

- 수학 집합 연산을 쉽게 하기 위해 만든 자료형
- 순서없음
- 중복없음

데이터 구조 (set)

Set 선언

```
ppap = {'pen', 'apple', 'pineapple', 'pen'}  
print(ppap)
```

```
'apple' in ppap  
'applepen' in ppap
```

```
pineapple = set('pineapple')  
pineapple
```

데이터 구조 (set)

`A = set('golang')`

`B = set('python')`

`A ∪ B == A | B`

`A ∩ B == A & B`

`A - B == A - B`

`A Δ B == A ^ B`

데이터 구조 (set)

참고: 추가와 삭제

- 변수명.add(특정 데이터): 하나의 데이터를 추가할 때는 add 함수 사용
- 변수명.update(여러 데이터): 여러 데이터를 추가할 때는 리스트 형태로 update 함수 사용
 - 여러 데이터는 리스트 형태로, 넣어야 함
 - data.update([2, 3, 4])
- 변수명.remove(특정 데이터): 특정 데이터를 삭제할 때는 remove 함수 사용