1. 반복, 조건문

- for a in b:
- while aaaa:

In [1]:

```
#1) 홀수만 출력하기 (for, while문 모두 한번씩)
#for문은 자동으로 하나씩 카운트 업 되지만, while은 조건부터 카운트업 모두 새로 설정해줘야 함

a = [1, 10, 9, 24, 25, 26]
```

In [4]:

```
#2) 1-7중 5만 제외하고 출력하라
# continue 문 사용 (pass사용해도 같은 값 나오긴 함)
```

In [5]:

```
#3) 1부터 100까지 합
# while문 사용
```

In [6]:

```
#4) for에서 index사용해서 index와 value를 모두 출력해보아라
# 특정함수 사용
# for문은 순회 시, 인덱스를 추출하는 게 아닌, 값을 추출해 순회한다. while과 다른 점.
```

In [7]:

```
#5) while문으로 구구단 작성 (2 - 9단)
```

In [9]:

```
#6) 1~20중 5의 배수만으로 구성된 리스트를 작성하라 ( 1줄로 )
# range와 indexing 에서 increment 의 활용
```

2. 함수

In [10]:

```
# parameter = 함수에 입력하는 값
# parameter : 함수에 입력하는 값
# output을 return으로 내보내고, caller가 그 값을 받는다.
# a = add(3,30) 이라면, 3, 30은 parameter / a = caller / output은 아마 33

# multiple return : 여러개의 값을 한번에 반환하는 함수일 경우, 그 여러 반환 값은 튜플로 반환됨
# return a,b,c 일 때, (a,b,c) 로 튜플로 묶여 반환됨
```


In [11]:

```
# 2-1. 가변길이 인자함수

# 2-1-1. 튜플형 ( 인자가 튜플형 )

#어떻게 하면 가변길이인자 함수를 정의할 수 있나
#변수 앞에 별표'*' 만 추가하면 됨
#관례적으로, 가변길이 인자는 이름을 args라 함.  즉,   *args

def test(*args):
    for item in args:
        print(item)

test(10,20,30,40,50,60)
```

```
10
20
30
40
50
60
```

In [15]:

```
# 2-1. 가변길이 인자함수

# 2-1-2. 딕셔너리형 ( 인자가 key = value, key = value 형 )

# 파라미터 : key = value 로 입력
# 함수 정의 시 : **kwargs 로 변수 사용
# ** 가 붙은 경우, 키워드 파라미터로 인식
# 파라미터의 이름과 값을 함께 전달 가능
# keyword = value 로 호출해준다
# **kwargs 라 네이밍해 사용

def test2(**kwargs):
    for key, value in kwargs.items():
        print('key: ', key,', value:',value)

test2(a=1,b='man')

# 딕셔너리형이 함수의 parameter로 들어갈 때는,
# def func(key = 'value', key2 = 'value2', ... )이런 식으로 들어간다!
# 원래 딕셔너리형은 {'key' = 'value', 'key2' = 'value2', ... } 이런식. 잘 기억
```

```
key:  a , value: 1
key:  b , value: man
```

In [14]:

```
# 2-1-3. format 형

# '  { placeholder }  '.format(, , , , ,)
```

In [17]:

```
# default parameter

# 함수 정의 시, 인자를 명시해서 사용
# def test(x=10,y)
```

In [18]:

```
# keyword parameter

# 함수 사용 시, 인자를 명시
# test(x=1,y,z)
```

3. lambda 함수

한줄로 표현되는 익명함수 (함수의 이름이 없다)

코드에서 한번만 사용될 때. 1회성으로 만들어서 사용

return 사용하지 않음.

lambda 입력값:출력값 구조

내장함수(sort, filter, map, reduce) 인자로 함수를 사용할 때 : lambda 함수를 주로 사용

In [25]:

```
# sort 의 경우  :  정렬의 key를 람다이용해서 정의
# strings.sort(key = 정렬의 기준)
# strings.sort(key = lambda x:len(x))

# filter 의 경우  :  뽑아낼 기준을 람다이용해서 정의
# filter(key, 대상 변수)
# key = 필터링 기준
# filter(lambda x:x%2==0, num)

# map 의 경우 : 주어진 리스트를 변형 + 반환
# map(key, 대상 변수)
# key = 변형식
# map(lambda x:x**2, num)

# reduce 의 경우 : 앞 2개의 원소로 연산, 결과는 다시 뒷 원소와 함께 연산됨. 결국엔 결과값 하나만 남게됨, reduce는 functools모듈의 함수
# reduce(key, 대상 변수)
# key = 두 수의 연산식
# import functools
# functools.reduce(lambda x,y:x+y, num)
```


함수 연습문제

- 1. 평균 구하기 (reduce 활용)
- 1. 소수 판별여부
- 1. 주어진 수들 사이에 소수가 몇개인지 출력

In [54]:

```
# 0. 평균구하기 ( 1-10 )

a = list(range(1,11))
import functools
b = (functools.reduce(lambda x,y:x+y,a)/len(a))
b
```

Out[54]:

```
5.5
```

In [55]:

```
# 1. 소수 판별 여부
# 꼭 해보기!   def find(x):    로 시작
```

In [15]:

```
# 2. 주어진 수들 사이에 소수가 몇개인지 출력 --> 소수면 cnt+=1 하면 될 듯

lista = []

# A = True 면 소수. 카운트 업 / False 면 소수가 아님
def count(x):  #count함수는 소수일 경우 1씩 추가하는 함수
    if x == 1:
        pass
    elif x == 2:
        return 1
    else:
        cnt = 0
        for i in range(2,x+1):
            # i의 소수판별. 소수이면 cnt+=1
            A = True
            for j in range(2,i):
                if i % j == 0:
                    A = False
            if A:
                cnt += 1
                lista.append(i)

    return cnt

print('소수의 갯수 :',count(3),'/','소수는 :',lista)
```

```
소수의 갯수 : 2 / 소수는 : [2, 3]
```

In [ ]:

```

```