1. 컨벤션에 대해서

(각 2점)

식별자 이름의 형태를 통해 직관적으로 이것이 함수인지, 변수인지 혹은 클래스명을 바로 알 수 있습니다.

저희가 아직 실질적으로 함수, 클래스를 만드는 방법을 배우지는 않았지만, 미리 컨벤션 규칙을 익히려 합니다.

아래 단어를 활용하여 변수명, 함수명, 클래스명을 컨벤션 규칙에 따라 작성해주세요.

customer list

In [0]:

```
# (1) 아래에 %%%% 를 지우고 위 단어로 컨벤션에 맞게 함수명을 작성해주세요.

def customer_list():
  pass
```

In [0]:

```
# (2) 아래에 %%%% 를 지우고 위 단어로 컨벤션에 맞게 변수명을 입력해주세요.

customer_list = 900
```

In [0]:

```
# (3) 아래에 %%%% 를 지우고 위 단어로 컨벤션에 맞게 클래스명을 입력해주세요.
# class 아래에 def, self 등 코드는 지금은 신경쓰지 않으셔도 됩니다.

class CustomerList(object):
  def __init__(self):
    self.name = "Hello"
```

2. 주피터 노트북 (Colab) 에서 Shell command (bash) 사용하기

(3 점)

분석 업무를 본인 PC 에서 작업을 한다면, 파일을 찾거나 저장, 이동을 윈도우 탐색기(macOS 는 Finder) 로 편리하게 작업이 가능합니다.

그러나 마우스를 활용할 수 없는 환경에서는 Shell command 를 통해 원하는 파일을 가져오거나, 저장, 삭제, 이동 등을 할 수 있습니다.

여기에서는 주피터 노트북에서 Shell command 를 사용하는 방법을 간단히 알아보겠습니다.

In [0]:

```
# 여기에 있는 코드를 그냥 실행해 주시고, 답은 다음칸에 입력하시면 됩니다.

# 아래 명령어들은 지금 아시지 않아도 되지만, 혹시 궁금하신 분들이 계실 것 같아 추가로 명령어를 남겨

# 아래는 Linux, macOS 에서 빈 파일을 만드는 명령어 입니다.
!touch 설명서.txt

# 아래는 Windows 에서 빈 파일을 만드는 명령어 입니다.
!type nul > 설명서.txt
```

In [0]:

```
# 여기에 동영상 강좌에서 다룬 명령어를 이용하여 설명서.txt 파일명이 보이도록 해주세요.

# 아래 방법으로 모두 볼 수 있습니다.
!ls
!ls 설명서.txt # 이 명령어는 파일/디렉토리(폴더)가 있으면 파일/디렉토리(폴더)명을 그대로 보여주며,
             # No such file or directory 등으로 존재하지 않는다는 의미의 에러를 표시합니다.

# Windows 는 아래와 같이 확인하실 수 있습니다.
!dir
!dir 설명서.txt


# 틀린것은 아니나 앞으로 Shell command 를 사용하실 때 주의 사항

# echo 는 Python 에서 print() 와 같은 기능을 합니다.
# 따라서 문제에서는 파일명이 보이도록 하면 된다에서는 echo 를 써도 되지만, 파일을 찾을 때에는
# echo 를 쓸 수 없습니다.
# 이게 무슨 말인지 이해가 잘 안가신다면 echo 를 쓰고 뒤에 아무런 단어를 쓰면 입력한 그대로 출력하는
```

3. 숫자(int, float) 와 문자(str) 를 합치는 방법

(5 점)

아래 a + b 를 실행 했을 때 1문자 로 합칠 수 있게 해주세요.

In [0]:

```
a = 1
b = "문자"

a + b
```

In [0]:

```
# (아래와 비슷하게 작성하시면 정답)
# 위 처럼 숫자형을 문자형(str) 로 변환한다면 더하기 연산을 통해 두 개 변수의 문자열을 하나로 합치는
# (아래와 비슷하게 코드를 작성하셔도 정답)

a = "1"
b = "문자"

a + b

a = str(1)
b = "문자"

a + b
```

Out[1]:

'1문자'

4. 전화번호에서 지역번호만 가져오기

(5 점)

아래 tel 이라는 변수에 전화번호가 저장되어 있습니다. 여기에서 **인덱스 값**을 활용하여 지역번호만 code 라는 변수에 저장하는 코드를 작성해주세요.

In [0]:

```
tel = "031-1111-2222"

# 아래 코드를 완성해주세요.
# 코드 일부분을 입력해놓았습니다. 활용하셔도 되고 혹은 지우고 원하는 방법으로 만들어보세요.

# 아래와 같이 작성하시면 됩니다.
code = tel[:3]
code = tel[0:3]


# 결과를 확인해보는 코드입니다.
print(code)
```

031

5. 반대로 순서를 뒤집기

(5 점)

order 라는 변수에 1,2,3,4,5,6,7,8,9 문자열이 저장되어 있습니다. 이를 거꾸로 출력하는 코드를 작성해주세요.

반드시 인덱스 기능을 활용해야 합니다.

In [0]:

```
order = "1,2,3,4,5,6,7,8,9"

# 아래에 코드를 완성해주세요.
# 코드 일부분을 입력해놓았습니다. 활용하셔도 되고 혹은 지우고 원하는 방법으로 만들어보세요.
order = order[::-1]


# 결과를 확인해보는 코드입니다.
print(order)
```

Out[6]:

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

6. 리스트에서 가장 마지막 값을 뽑아내기(출력 후 삭제)

(5 점)

노트북을 구매하기 위해 4개의 제품을 선택했고 이를 리스트에 저장했습니다.

products = ["A", "B", "C", "D"]

여기에서 리스트에 있는 **함수**를 활용하여 가장 마지막 제품을 하나 뽑아서 c 라는 변수에 저장해주세요.

In [0]:

```
products = ["A", "B", "C", "D"]

# 아래에 코드를 완성해주세요.
# 코드 일부분을 입력해놓았습니다. 활용하셔도 되고 혹은 지우고 원하는 방법으로 만들어보세요.
c = products.pop()

"""
pop() 안에 인덱스 번호를 넣을 수 있습니다. 예 : products.pop(1)
번호를 생략하면 리스트의 마지막 항목을 삭제 후 돌려줍니다.
관련 내용은 공식문서
https://docs.python.org/ko/3/tutorial/datastructures.html 에서도 확인하실 수 있습니다.
"""

print(c)
```

```
D
```

7. 연산자를 본격적으로 사용해보기 전에 기능을 복습해봅시다.

(각 2점)

아래 칸에 있는 설명에 맞는 코드를 하나씩 작성해주세요.

참고 : 주피터 노트북에서는(Colab 포함) print() 없이 코드를 작성하면 하단에 마지막 코드에 대한 결과값만 출력됩니다. 한 칸에서 모든 변수의 값을 출력하고 싶으시면 print() 함수를 꼭 써주세요.

In [0]:

```
# 아래 3줄 코드들은 수정하지 마세요.
a = 100
b = 200
datas = ["김", "박", "이", "홍", "정"]

# 문제1) a 변수에 100 을 더해서 저장해주세요. (할당 연산)
a += 100
print(a)

# 문제2) a 와 b 값이 다른지를 확인해보세요. (비교 연산)
print(a == b)

# 문제3) datas 리스트안에 "이" 라는 값이 있는지 확인해주세요. (멤버 연산)
print("이" in datas)
```

```
200
True
True
```

8. 형변환

(3 점)

아래 4개의 변수에는 각각 설문 응답여부를 1(응답), 0(미응답) 형태로 저장되어 있습니다. 이를 boolean 타입을 변경하려 합니다. 형변환을 통해 boolean 타입으로 같은 이름의 변수에 덮어써주세요.

즉 1(응답) 이면 True, 0(미응답) 은 False 로 출력 해주세요.

In [0]:

```
# 아래 4줄 코드는 수정하지 마세요.
check1 = 1
check2 = 0
check3 = 0
check4 = 1

# 아래에 코드를 작성해주세요.
check1 = bool(check1)
check2 = bool(check2)
check3 = bool(check3)
check4 = bool(check4)

# 아래 코드는 boolean 형으로 제대로 변환되었는지 확인해보는 코드입니다.
print(type(check1))
print(type(check2))
print(type(check3))
print(type(check4))
```

```
<class 'bool'>
<class 'bool'>
<class 'bool'>
<class 'bool'>
```

9. dict 는 정말 순서가 없는 데이터 타입인가요?

(3 점)

앞서 강의에서 dict 는 순서가 없는 데이터 타입으로 배웠습니다.

먼저 아래 datas 라는 dict 데이터 타입을 생성하는 코드를 실행해주세요.

In [0]:

```
datas = {
    0: "바나나",
    1: "딸기",
    2: "망고"
}
```

In [0]:

```
datas[0]
```

Out[13]:

'바나나'

In [0]:

```
# 문제) datas[0] 를 입력하여 '바나나' 가 출력되었는데,
# 앞에 배운 인덱스 개념으로 값을 가져온건지 key 이름으로 가져온것인지
# 코드가 아닌 일반 문장으로 적어주세요.

# 아래에 답을 적어주세요.
# 답 : key 이름으로 가져왔습니다.
# 정말 key 이름으로 가져왔는지 궁금하시면, 아래와 같이 해보시고 시도해 볼 수 있습니다.

# 에러가 발생한다면 Key 를 통해 Value 를 가져왔다고 알 수 있습니다. 오류 메시지에도 'KeyError' 가 보

datas = {
  1: "바나나",
  2: "딸기",
  3: "망고"
}
datas[0]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-9-7584abf2f6d5> in <module>()
      12   3: "망고"
      13 }
---> 14 datas[0]
      15

KeyError: 0
```

10. 나이에 따른 버스 요금

(5 점)

사용자에게 나이(변수명 : age)를 입력받은 후, 아래의 운임요금(변수명 : fare) 기준에 맞게 버스 요금을 계산해주세요.

- 7세 이하 : 0원
- 8세 이상 ~ 20세 이하 : 700원
- 21세 이상 : 1200원

참고 : 변수명은 예시이며 Python 변수명 규칙에 따라 자유롭게 작성하시면 됩니다.

In [0]:

```
# 여기에 코드를 작성해주세요.

age = int(input("나이 : "))

if age <= 7:
  fare = 0
elif age <= 20:
  fare = 700
else:
  fare = 1200

print("요금 : {}".format(fare))
```

11. 4번째 고객마다 상품권 증정

(5 점)

방문하는 고객 중 5번째 마다 상품권을 증정(status = "상품권 증정") 하는 코드를 작성해주세요.

In [0]:

```
count = 10

if count % 5 == 0:
  status = "상품권 증정"
else:
  status = "증정하지 않음"

print(status)
```

In [0]:

```
# 테스트 코드
# 위에 코드를 작성하여 아래와 같이 count 변수에 각각 숫자를 입력하시고, 아래와 같은 결과가 나오면 됩

# 참고 : 아래는 예시 이며 아래와 같이 작성하는게 정답이 아닙니다. 참고만 해주시면 됩니다.

counts = [1, 4, 5, 10, 15, 20, 8, 7]

for i in counts:
  if i % 5 == 0:
    status = "상품권 증정"
  else:
    status = "증정하지 않음"
  print("count : {} , {}".format(i, status))
```

```
count : 1 , 증정하지 않음
count : 4 , 증정하지 않음
count : 5 , 상품권 증정
count : 10 , 상품권 증정
count : 15 , 상품권 증정
count : 20 , 상품권 증정
count : 8 , 증정하지 않음
count : 7 , 증정하지 않음
```

반복문(특히 while) 을 작성하시다가 무한 루프가 발생하면

코드를 실행 하셨다가 강제로 중단하려면

- Colab
  - ctrl + M, I 누르면 중단됩니다.
  - 단축키가 작동되지 않으면 '런타임' -> '실행 중단' 을 눌러도 됩니다.
- Jupyter Notebook/Lab
  - esc 키를 누릅니다.
  - 키보드 i 를 두 번 누릅니다.

12. Dictionary 에 계좌 잔액을 모두 더하는 코드 작성

(3 점)

accounts 라는 딕셔너리의 잔액 (value) 을 total 이라는 변수에 모두 더하여 출력해주세요.

In [0]:

```
accounts = {"A계좌": 2322000, "B계좌": 34443000, "C계좌": 6888000}

# 여기에 코드를 작성해주세요.

total = 0
for i in accounts.values():
  total += i

print(total)
```

43653000

13. List Comprehension 으로 코드 작성해보기

(3 점)

아래와 같이 리스트 요소 중에 '음수' 가 있다면 0 으로 바꿔주는 코드를 List Comprehension 으로 작성해보세요.

In [0]:

```
numbers = [100, 200, -100, -50, 400, 700]

# 여기에 코드를 작성해주세요.
results = [i if i>0 else 0 for i in numbers]

print(results)
```

[100, 200, 0, 0, 400, 700]

14. 점심 메뉴 골라주는 함수

(3 점)

1. lunch_selector 라는 함수에 1개 이상의 음식을 입력받도록 해주세요.
2. 랜덤 함수를 활용하여 입력 받은 음식 중 하나를 랜덤으로 선택하여 리턴 해줍니다.

In [0]:

```
import random

def lunch_selector(*args):
  num = random.randint(0, len(args)-1) # 여기에 -1 을 한 이유는 아래 자세히 안내
  lunch = args[num]

  return lunch
```

In [0]:

```
# 예제 코드
lunch_selector("짜장면", "짬뽕", "탕수육")

# 위 '예제 코드' 에서 3가지 음식을 넣으면 len(args) 결과값으로 3이 출력됩니다.
# randint 에서 0부터 3으로 무작위로 숫자를 출력하게 되면 간혹 3이 출력되기도 합니다.
# 이러면 args[3] 을 하는것과 같은데, 보시다시피 음식은 3가지 입니다.
# 그러나 인덱스 번호는 0부터 시작하여 0 부터 2 까지 있는데 3을 넣으면 범위를 벗어나게 됩니다.
# 그래서 이를 방지하기 위해 -1 을 넣어서 3-1=2 와 같은 효과를 내개 하였습니다.
```

Out[42]:

'탕수육'

15. 중복되는 코드를 하나의 함수로 정리하기

(3 점)

학점을 계산하는 코드입니다. 두 명의 학생을 비교하는데 조건문을 각 학생별로 구현되어 있어 코드가 비효율적으로 작성되어 있습니다.

함수를 활용하여 코드를 개선해주세요.

In [0]:

```
student1_score = 70
student1_grade = ""

if student1_score > 90:
  student1_grade = "A"
elif student1_score >= 80:
  student1_grade = "B"
elif student1_score >= 70:
  student1_grade = "C"
elif student1_score >= 60:
  student1_grade = "D"
else:
  student1_grade = "F"

print(student1_grade)

student2_score = 50
student2_grade = ""

if student2_score > 90:
  student2_grade = "A"
elif student2_score >= 80:
  student2_grade = "B"
elif student2_score >= 70:
  student2_grade = "C"
elif student2_score >= 60:
  student2_grade = "D"
else:
  student2_grade = "F"

print(student2_grade)
```

```
C
F
```

In [0]:

```
# 여기에 위 학점 계산 코드를 함수를 활용하여 중복 코드를 제거하여 코드를 개선해주세요.

# 아래 함수에 코드를 작성해주세요.
def set_grade(score):
  grade = ""
  if score > 90:
    grade = "A"
  elif score >= 80:
    grade = "B"
  elif score >= 70:
    grade = "C"
  elif score >= 60:
    grade = "D"
  else:
    grade = "F"

  return grade

set_grade(77)
```

Out[6]:

```
'C'
```

16. 리스트에 들어 있는 요소 구분하기

(5 점)

과일 선호도 조사를 하여 나온 결과가 리스트에 저장되어 있습니다. "사과", "바나나", "오렌지" 가 각각 3~5개씩 들어 있습니다. **조건문**을 활용하여 리스트에 있는 요소를 `apple` , `banana` , `orange` 의 리스트에 추가해주세요.

논리적인 문제해결 단계를 연습하기 위해 외부 패키지 사용을 하지 말아주세요.

ToDo

1. 먼저 코드를 작성하기 전에 강의영상에서 다룬 것 처럼 **논리적인 문제해결 순서**를 적어주세요.
2. 작성하신 논리적인 문제해결에 알맞은 코드를 작성해주세요.

결과 예

(아래 apple 출력시 '바나나' 가 출력되는 오류가 있습니다. 채점시에는 리스트명에 상관 없이 각 과일별로 분류만 되었다면 맞는 것으로 채점하였습니다.)

```
print(apple)
print(banana)
["바나나", "바나나", "바나나"]
```

In [0]:

```
fruits = ["바나나", "사과", "사과", "오렌지", "오렌지", "바나나", "바나나", "사과", "사과", "오렌지"

# 논리적인 문제해결 순서를 "주석 형태로" 작성하시고 그 아랫줄에 코드를 작성해주세요.

# 1. "사과" 는 apple, "바나나" 는 banana, "오렌지" 는 orange 라는 리스트에 값을 추가할 수 있도록 비
apple = list()
banana = list()
orange = list()

# 2. fruits 리스트 길이만큼 반복하는 반복문을 작성 합니다.
for fruit in fruits:

  # 3. fruits 요소를 하나씩 가져옵니다.
  # 위와 같이 for loop 문을 작성했다면 반복할 때 마다 fruit 에 각각 요소 (여기서는 "사과, "바나나",

  # 4. 조건문을 통해 "사과", "바나나", "오렌지" 와 같은지 비교 합니다.
  if fruit == "사과":

    # 5. 3가지 조건중에서 하나의 과일 종류에 해당하면 "사과" 는 apple, "바나나" 는 banana, "오렌지"
    apple.append(fruit)

  elif fruit == "바나나":
    banana.append(fruit)

  elif fruit == "오렌지":
    orange.append(fruit)

# (선택사항) apple, banana, orange 라는 리스트에 담긴 요소가 무엇인지 확인합니다.

print(apple)
print(banana)
print(orange)
```

```
['사과', '사과', '사과', '사과', '사과']
['바나나', '바나나', '바나나']
['오렌지', '오렌지', '오렌지', '오렌지']
```

17. 미국식 날짜 표기를 한국식으로 변환

(4 점)

월/일/연도 형태로 저장된 `usa_date` 를 한국식 표기인 **연도/월/일** 로 바꿔서 저장해주세요.

논리적인 문제해결 단계를 연습하기 위해 외부 패키지 사용을 하지 말아주세요.

ToDo

1. 먼저 코드를 작성하기 전에 강의영상에서 다룬 것 처럼 **논리적인 문제해결 순서**를 적어주세요.
2. 작성하신 논리적인 문제해결에 알맞은 코드를 작성해주세요.

In [0]:

```
usa_date = "03/20/2020"

# 논리적인 문제해결 순서를 "주석 형태로" 작성하시고 그 아랫줄에 코드를 작성해주세요.

# 1. usa_date 를 / 를 기준으로 나눠서 temp 라는 이름의 리스트로 저장합니다. 문자열에서 split() 의 리
temp = usa_date = usa_date.split("/")

# 2. kor_date 에 연도/월/일 순서로 문자열 형태로 저장합니다.
kor_date = str(temp[2]) + "/" + str(temp[0]) + "/" + str(temp[1])

# (선택사항) kor_date 에 저장된 값을 확인합니다.
print(kor_date)
```

2020/03/20

18. docstring 작성해보기

(3 점)

강의영상에서 docstring 작성 방법을 다시 확인해보며 아래 함수에 알맞은 docstring 을 작성해주세요.

강의영상을 참고하거나 Google 검색에서 'python docstring' 를 찾아서 작성 방법을 확인해주세요. 일부 키워드를 제외하고 한글로 작성하셔도 됩니다.

작성할 내용

- 함수의 간단한 기능 설명
- 함수에 들어갈 parameter 설명 (변수명, 자료형)
- 함수 리턴시 리턴값 자료형

In [0]:

```
def calc(option, num1 = 100, num2 = 100):
  """
  더하기와 곱하기 기능을 수행하는 함수 입니다.

  Parameters:
    option (str): "더하기", "곱하기" 중 원하는 연산
    num1 (int, float): 형태의 값
    num2 (int, float): 형태의 값
  Returns:
    int, float : 계산 결과에 따라서 반환합니다.
  """
  if option == "더하기":
    total = num1 + num2
  elif option == "곱하기":
    total = num1 * num2
  else:
    total = 0

  return total
```

19. 함수 내에서 전역(global) 변수 값 가져오고 변경하기

(3 점)

editor() 함수를 호출하면, 전역 변수인 result 의 값 "apple" 를 "banana" 로 변경하도록 코드를 작성해주세요.

In [0]:

```
result = "apple"

def editor():

  global result # result 변수 앞에 global 을 넣어 함수 밖에 있는 result 변수와 연결합니다.
  result = "banana"

editor()
print(result)
```

banana

20. lambda 함수로 구현하기

(5 점)

아래 함수를 lambda 형식으로 구현해보세요.

In [0]:

```
def add(num):
  return num + 10

print(add(5))
```

15

In [0]:

```
# 여기에 위 add 함수와 같은 기능을 하는 lambda 함수를 구현해보세요.
add_lambda = lambda num: num + 10
print(add_lambda(5))
```

15

21. callback function 으로 계산기 만들기

(5 점)

지난 시간에 만들어 보았던 "더하기", "곱하기" 를 계산해주는 함수를 callback function 으로 구현해보세요.

코드

```
def calc(option, num1 = 100, num2 = 100):

  if option == "더하기":
    total = num1 + num2
  elif option == "곱하기":
    total = num1 * num2
  else:
    total = 0

  return total
```

callback function 으로 구현할 때 지켜야 할 사항

1. "더하기" 는 add(), "곱하기" 는 mul() 라는 함수를 만듭니다.
2. 실제 사용자가 더하기 혹은 곱하기 계산을 시키기 위해 사용할 함수 이름은 calc() 입니다.
3. 위 샘플 함수내에 조건문에 else 는 무시합니다.
4. 위 샘플 함수처럼 숫자값을 입력하지 않으면 기본적으로 각각 100 을 할당하게 해주세요.

In [0]:

```
# 여기에 코드를 구현해주세요.

def calc(func, a = 100, b = 100):

  return func(a, b)

def add(a, b):
  return a + b

def mul(a, b):
  return a * b
```

In [0]:

```
# 코드 실행 예

calc(add)
```

Out[2]:

200

In [0]:

```
# 코드 실행 예

calc(mul, 3, 3)
```

Out[4]:

9

22. map 함수로 반복문 없이 간편하게 리스트의 모든 요소를 사용

(5 점)

`purchased_dates` 라는 리스트에는 **년/월/일** 형태로 날짜가 저장되어 있습니다.

각각 날짜가 표현된 문자열(str) 을 년, 월, 일 로 분리하여 리스트 형태로 만들고, `map()` 함수로 리스트 형태로 문자열(str) 을 넣으면 각각 분리되어 아래와 같이 출력되게 코드를 작성해주세요.

함수, 람다함수 등 기능 구현은 자유롭게 하시되, map() 함수로 테스트한 결과도 함께 작성해주세요.

저는 아래와 같이 함수를 작성하여 구현해본 예제를 올려드립니다.

반드시 함수로만 구현해야 하는건 아닙니다.

In [0]:

```
purchased_dates = ["2020/01/03", "2019/12/11", "2019/11/10", "2018/07/22"]

# 위 purchased_dates 는 삭제하지 마시고 활용하여 아래에 코드를 구현해주세요.

def split_dates(dates):
  return dates.split("/")
```

In [0]:

```
# 실행 예 입니다. map() 만 사용하시면 위에 구현하신 코드에 맞게끔 출력하면 됩니다.

list(map(split_dates, purchased_dates))
```

Out[16]:

```
[['2020', '01', '03'],
 ['2019', '12', '11'],
 ['2019', '11', '10'],
 ['2018', '07', '22']]
```

23. 리스트에서 사과만 뽑기

(3 점)

'사과'와 다른 과일들이 섞여 있는 fruits 리스트가 있습니다.

이 중에서 '사과'만 뽑을 수 있도록 `filter()` 를 활용하여 구현해주세요.

In [0]:

```
fruits = ["사과", "배", "바나나", "딸기", "바나나", "사과", "배", "망고", "망고", "사과", "사과"]
```

In [0]:

```
# 아래 줄에 filter() 코드를 작성해서 '사과' 만 추출해주세요.
list(filter(lambda x : x == "사과", fruits))
```

Out[15]:

```
['사과', '사과', '사과', '사과']
```

24. reduce 로 1부터 100까지 더하는 코드 작성

(4 점)

1 부터 100을 더하여 총합이 5050 이 되도록 코드를 작성해주세요.

In [0]:

```
from functools import reduce

# 아래 줄에 reduce() 로 구현해주세요.
reduce(lambda x, y : x+y, range(1, 101))
```

Out[4]:

5050

25. Decorator 함수 내부 코드가 실행되는 순서

(5 점)

아래는 Decorator 함수를 활용하여 사용자 이름을 입력 받아 해당 이름이 nameList 에 있으면 "안녕하세요! ?? 님." 을 출력합니다.

구현해야 할 내용

- 사용자가 hello() 를 호출 합니다.
- 사용자에게 이름을 입력 받습니다. `input()`
- `nameList` 에서 입력된 이름과 비교 합니다.
- 해당 이름을 상단에 `name` 변수에 덮어씁니다.
- hello 함수에 `print()` 문에서 `"안녕하세요! ??님."` 을 출력합니다.

참고

- 코드를 구현하다보면 입력된 이름과 비교하여 정상적으로 출력했으나, 바로 다음줄에 "이름이 없습니다." 가 출력되는 문제가 생길 수도 있습니다.
- 아래 구현 방법보다 Decorator 함수를 활용하여 더 나은 방법으로도 코드를 구현할 수 있습니다. 그럴땐 아래 가이드를 삭제하고 위 **구현해야 할 내용**을 참고하여 직접 작성하셔도 좋습니다.

In [0]:

```
name = ""

def auth(func):
  def wrapper(*args, **kwargs):
    nameList = ["Park", "Kim", "Lee"]
    input_name = input("이름 : ")
    for n in nameList:
      if input_name == n:
        global name
        name = n
        result = func(*args, **kwargs)
        break
      else:
        result = "이름이 없습니다."
    return result
  return wrapper

@auth
def hello():
  print("안녕하세요! {} 님.".format(name))
```

In [0]:

```
# 실행 예
hello()
```

이름 : 123

Out[16]:

'이름이 없습니다.'

In [0]:

```
실행 예
hello()
```

이름 : Kim
안녕하세요! Kim 님.