파이썬 전체 기본부터 클래스 리뷰하기

학습 내용 및 목표

- 파이썬의 전반적인 내용을 살펴봅니다.
  - 입출력 함수 및 연산자
  - 자료형(리스트, 튜플, 딕셔너리)
  - if, for
  - 함수, 클래스에 대해서 살펴본다.

1-1 Python 시작하기

In [1]:

```
### Hello 문자열 출력
print("Hello")
print()
### 'end' 매개변수의 역할
print("Hello", end='\n')
print("Hello", end='')
print("Hello")
print()
### print를 이용하여 여러줄 문자열
# '\n'은 줄을 바꿈의 명령의 내용을 전달한다.
print("Line1 \nLine2 \nLine3 \n")
```

```
Hello

Hello
HelloHello

Line1
Line2
Line3
```

1-2 변수 사용하기

In [2]:

```
### 동물의 다리의 개수는?
animal_leg = 4
print(animal_leg)

str1 = 'my'
str2 = 'name'
print(str1, str2)
```

```
4
my name
```

여러줄의 문자열을 저장 및 출력

- ''' 내용 '''
- """ 내용 """

In [3]:

```
str1 = '''
line1
line2
line3
'''

print(str1)
```

```
line1
line2
line3
```

문자열과 문자열을 어떻게 연결시킬까?

In [4]:

```
str12 = "my" + "name"
print(str12)
```

```
myname
```

1-3 문자열을 입력받고, 내용을 출력시키기

In [5]:

```
str1 = "my" + " name is"
name = input("당신의 이름은 ?")
print(str1, name)
```

```
당신의 이름은 ?toto
my name is toto
```

- 다른 형태 출력

나이도 입력받아, 출력시켜보자.

In [6]:

```
age = input("나이는 ?")
print("당신의 나이는 " + age + "살 입니다.")
```

```
나이는 ?30
당신의 나이는 30살 입니다.
```

- age의 자료형은 무엇일까

In [7]:

```
print(type(age))
```

```
<class 'str'>
```

(생각해보기) 그렇다면 나이에 10을 더해서 출력해 보기

In [8]:

```
age = int( input("나이는 : ") ) # 입력 후, 정수형 변환

# sep는 ','가 갖는 한칸 공백을 없애준다.
print("당신의 10년 후의 나이는 " , age+10 ,"살입니다.", sep="")
```

```
나이는 : 30
당신의 10년 후의 나이는 40살입니다.
```

1-4 연산자

- 사칙 연산자 : +, -, *, /
- 기타 연산자 : **(제곱), %(나머지), //(몫)

In [9]:

```
print(9 + 10) # 더하기
print(9 - 10) # 빼기
print(30 / 5) # 나누기
print(5 * 3) # 곱하기
print(2**5) # 제곱 연산 2*2*2*2*2
print(17 % 4) # 나머지 1
print(17//4) # 몫 4
```

```
19
-1
6.0
15
32
1
4
```

- 비교 연산자

- 같다 "=="
- 다르다 "!="
- 크거나 같다 ">="
- 작거나 같다 "<="
- 크다 ">"
- 작다 "<"

In [10]:

```
print( 3 == 3 )   # True (1이상)
print( 3 != 3 )   # False (0)
print( 3 >= 1 )
print( 3 <= 1 )
# 1 => 1 # 에러 => =이 먼저오면 에러 발생
```

```
True
False
True
False
```

In [11]:

```
print(5 != 5) # False
age = 18
print(age >= 15) # True
age = 13
print(age >= 15) # False
```

```
False
True
False
```

연산자 and, or, not, in

- True는 참을 의미, False는 거짓을 의미
- A and B : A와 B가 참이면 True, 둘 중 하나라도 False이면 False
- A or B : A또는 B가 True이면 True, 둘 다 False일 경우 False
- not A : A가 True이면 False, A가 False이면 True
- 임의의값 in [자료형] : 임의의 값이 해당 자료형에 있으면 True, 아니면 False

In [12]:

```
print(True and True)
print(1 and 3)
print(1 and 0)
print(True and False)
print(1 in [1,2,3,4])
```

```
True
3
0
False
True
```

In [13]:

```
print(    (1 == 1) and (10 > 3)    )  # True and True
print(    (1 == 1) & (10 > 3)    )  # True and True
```

```
True
True
```

In [14]:

```
print((1 == 1) or (1 > 0) )
print((1 != 1) | (1 < 0) )   # False or False

## A or B : A와 B중에 하나라도 True가 있으면 결론적으로 True가 된다.그 외에는 False
print(5!=5 or 5>=3) # False and True => True
```

```
True
False
True
```

In [15]:

```
print( not 1 ) # False
print( not False) # True
print( not 0) # True (0는 0을 의미)
print( not 'A') # 0을 제외한 모든 값은 True
```

```
False
True
True
False
```

19세이하 영화 입장 금지 프로그램

In [16]:

```
## 19세 입장 금지 영화
## 출입조건 19세 이상이고, 남자 입장 가능
age = 17
sex = "M"
print(age >= 19 and sex=="M") # True : 입장 불가
```

```
False
```

- 예외조건 : 19세 이상 또는 아이가 부모와 함께 올 경우, 가능

In [17]:

```
## 출입조건 19세 이상이거나(또는) 아이가 있는 친구는 가능한 영화
age = 17
gubun = "P" # 부모
print(age >= 19 or gubun=="P") # False or True => True : 입장
```

```
True
```

1-5 연산자를 이용한 계산

In [18]:

```
a = 5
b = 2
plus = a + b
mul = a * b
div = a / b # 결과값 형변환 5/2 = 2.5
remainder = a % b
power = a ** b

print("합은 %d 입니다." % a)
print("%d 와 %d의 합은 %d 입니다." % (a, b, plus))
print("{} 와 {}의 합은 {}입니다.".format(a, b, plus))
print("{} 와 {}의 나눈 값은 {:.2f}입니다.".format(a, b, div))
```

```
합은 5 입니다.
5 와 2의 합은 7 입니다.
5 와 2의 합은 7입니다.
5 와 2의 나눈 값은 2.50입니다.
```

print문의 여러가지 출력

In [19]:

```
print("{2} 와 {1}의 합은 {0}입니다.".format(2, 5, 7))
```

```
7 와 5의 합은 2입니다.
```

In [20]:

```
print("{0} 와 {1}의 나눈 값은 {2}입니다.".format(a, b, div))
```

```
5 와 2의 나눈 값은 2.5입니다.
```

1-6 if문 실습해보기

숙박 인원이 10명 이상이면

- 두개의 방으로 나누어서 예약해야 합니다.
- 10명 이하이면 하나의 방으로 숙박 가능합니다.

In [21]:

```
people = 10

if people >= 10:
  print("두개의 방을 예약해야 합니다.")
else:
  print("305호실 예약 가능합니다.")
```

두개의 방을 예약해야 합니다.

1-7 for문 실습해 보기

In [22]:

```
for one in ['하나', '둘']:
  print(one)
```

하나
둘

생각해보기 & 실습 1

- 나이를 입력받고, 8세 이상이면 학교에 입학이 가능합니다. 아니면 아직 입학이 힘듭니다. 출력해 보기

2-1 if

- 기본 구조1

```
if (조건식):
  실행문1
```

- 기본 구조2

```
if (조건식):
  실행문1
else:
  실행문2
```

- 기본 구조3

```
if (조건식) & (조건식):
  실행문1
  실행문2
else:
  실행문3
  실행문4
```

- 기본 구조4

```
if (조건식) & (조건식):
  실행문1
  실행문2
elif (조건식):
  실행문3
elif (조건식):
  실행문4
else:
  실행문3
  실행문4
```

에러 발생

- indent의 간격이 다를 경우,

In [25]:

```
people = 10

# 인덴트가 다를 경우, 에러가 발생.
if people >= 10:
  print("사람이 10명이상입니다.") # indent라고 한다. 동일하게 해 준다.
   print("사람이 10명이상입니다.") # 이 줄은 인덴트(들여쓰기) 공간이 다르다. 에러 발생.
  print("사람이 10명이상입니다.")
```

```
  File "<ipython-input-25-65478a503485>", line 6
    print("사람이 10명이상입니다.") # 이 줄은 인덴트(들여쓰기) 공간이 다르다. 에러 발
생.
    ^
IndentationError: unexpected indent
```

In [26]:

```
people = 10

# 인덴트가 다를 경우, 에러가 발생.
if people >= 10:
  print("사람이 10명이상입니다.") # indent라고 한다. 동일하게 해 준다.
  print("사람이 10명이상입니다.") # 이 줄은 인덴트(들여쓰기) 공간이 다르다. 에러 발생.
  print("사람이 10명이상입니다.")
```

```
사람이 10명이상입니다.
사람이 10명이상입니다.
사람이 10명이상입니다.
```

In [27]:

```
num = int( input("사람의 수는 :") )
if num >= 10:
 print("사람이 10명이상입니다.") # indent라고 한다. 동일하게 해 준다.
 print("10명 이상의 경우는 추가 비용을 지불해야 합니다.")
else:
 print("사람이 10명 미만입니다")
```

```
사람의 수는 :20
사람이 10명이상입니다.
10명 이상의 경우는 추가 비용을 지불해야 합니다.
```

In [28]:

```
age = int(input("나이가 어떻게 되나요?"))

if (age>=15):
 print("입장 가능합니다. 어떤 영화 티켓을 구매하시겠어요?")
else:
 print("입장이 불가합니다. 15세이상만 입장이 가능합니다.")
```

```
나이가 어떻게 되나요?20
입장 가능합니다. 어떤 영화 티켓을 구매하시겠어요?
```

2-2 학점 프로그램 만들기

- 학점 판정 프로그램 만들어보기
  - 4.5 이면 A+
  - 4.0 이상이면 A
  - 3.5 이상이면 B+
  - 3.0 이상이면 B
  - 2.5 이상이면 C+
  - 나머지 F

In [30]:

```
score = float( input("학점입력해 주세요 : ") )

if score==4.5:
 grade = 'A+'
elif score>=4.0:
 grade = 'A'
elif score>=3.5:
 grade = 'B+'
else:
 grade = 'F'

print("당신의 학점은 ", grade , "입니다.")
```

```
학점입력해 주세요 : 4.2
당신의 학점은  A 입니다.
```

(생각해보기) 만약 학점을 숫자가 아닌 경우가 입력되었을때, 어떻게 수정 보완할 수 있을까?

실습 - 아래와 같이 판정하는 프로그램을 작성해 보자.

- 점수가 90점 이상이면 A
- 점수가 80점 이상이면 B
- 점수가 70점 이상이면 C
- 나머지 점수는 F

2-3 for 문 알아보기

기본 구조1

```
for 변수 in range(시작값, 끝값, 증감값):
  실행문1
```

In [31]:

```
for num in range(1,11,1):
  print(num)
```

```
1
2
3
4
5
6
7
8
9
10
```

In [32]:

```
for num in range(11,1, -1):  # 마지막 증감값 1은 기본값으로 생략되면 1씩 증가
  print(num)
```

```
11
10
9
8
7
6
5
4
3
2
```

기본 구조2

- [변수]의 값은 for문 루프안에서 사용이 가능.
- range 부분에 자료형(리스트, 딕셔너리)이 들어가는 것이 가능.

```
for [변수] in [리스트 or 튜플 등]:
  실행문1
  실행문2
```

In [33]:

```
al = ['하나', '둘', '셋', '넷', '다섯']

for one in al:
  print(one)
```

```
하나
둘
셋
넷
다섯
```

In [34]:

```
# 딕셔너리 값 출력하기
num = {'one':'하나', 'two':'둘', 'three':'셋', 'four':'넷', 'five':'다섯'}

for one in num:
    print(one, num[one])
```

```
one 하나
two 둘
three 셋
four 넷
five 다섯
```

3-1 자료형 - 문자열

In [35]:

```
str1 = "Hello World!"
print( str1[0] )   # 첫번째 선택 (파이썬은 숫자 0부터 시작)

# Hello 선택
# : 을 기준으로 앞이 시작값, 뒤에값이 끝나는 값.
print( str1[0:5] ) # 첫번째(0)부터 다섯번째(4)까지 선택 (파이썬은 숫자 0부터 시작)
```

```
H
Hello
```

In [36]:

```
# Hello 선택
print( str1[ :5] ) # 처음부터 다섯번째(4)까지 선택 (파이썬은 숫자 0부터 시작)
print( str1[:] )
print( str1[2:5] )  # 세번째(2)부터 다섯번째(4)까지 선택 (파이썬은 숫자 0부터 시작)
```

```
Hello
Hello World!
llo
```

In [37]:

```
## 인덱싱(Indexing) : 값을 하나 하나 선택
## 슬라이싱(slicing) : 값을 조각 단위로 선택
```

실습해 보기

- 나의 이름을 입력받아(input), 앞의 3자리를 출력해 보자.

3-2 자료형 - 리스트

In [38]:

```
a = [1,2,3,4,5]
print(a)

a = [1,2,'a',3,'b']
print(a)

print( a[1] )  # 0부터 시작하여 1은 두번째 요소를 가르킴
print( a[2:5] )

a = [1,2,'a',3,'b']
print( a[-1] )

# 뒤에서부터 3개 가져오기
print( a[-3] )
```

```
[1, 2, 3, 4, 5]
[1, 2, 'a', 3, 'b']
2
['a', 3, 'b']
b
a
```

In [39]:

```
a = [ [1,2],
      [2,3],
      [4,5]
 ]
print(a[1]) # 2번째 값([2,3])
print(a[1][1]) # 2번째 값중의 2번째 값을 가져오기
```

```
[2, 3]
3
```

하나의 값을 추가하기

In [44]:

```
list1 = [1,2,3]
print( list1 )

list1.append(4)
print( list1)
```

```
[1, 2, 3]
[1, 2, 3, 4]
```

In [45]:

```
[1,1] + [2,2]
```

Out[45]:

```
[1, 1, 2, 2]
```

하나의 값을 삭제하기

In [46]:

```
list2 = [1,2,3,2]
list2.remove(2)      # 요소 중에 2의 값을 하나 삭제(앞에서부터 봤을때)
list2
```

Out[46]:

```
[1, 3, 2]
```

In [47]:

```
## 두개의 값을 한꺼번에 추가하기
list3 = [1,2,3]
list3.append([3,4])
print(list3)
```

```
[1, 2, 3, [3, 4]]
```

In [48]:

```
## 두개의 값을 한꺼번에 추가하기 - extend
list4 = [1,2,3]
list4.extend([3,4])
print(list4)
```

```
[1, 2, 3, 3, 4]
```

3-3 리스트와 for문

In [49]:

```
list1 = [1,2,3,4,5,11,22,33,44,55]
for i in list1:
  print(i, end=" ")  # 출력 후, 한칸을 띄우고 다음번 진행
```

```
1 2 3 4 5 11 22 33 44 55
```

In [50]:

```
# range(시작값,끝값,증가값)
for i in range(0,10,1):
  print(i, end=" ")
```

```
0 1 2 3 4 5 6 7 8 9
```

In [51]:

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

```
0 1 2 3 4 5 6 7 8 9
```

In [52]:

```
for i in range(5, 0,-1):
  print(i, end=" ")
```

```
5 4 3 2 1
```

In [53]:

```
season = ['봄','여름','가을', '겨울']
print("season의 요소의 개수 : ", len(season))
for one in season:
  print(one)
```

```
season의 요소의 개수 :  4
봄
여름
가을
겨울
```

3-4 딕셔너리(dict)

- [ ] : 리스트 => 값의 변경이 가능하다. 수정이 가능하다. 추가가능
- ( ) : 튜플 => 값의 변경이 불가. 속도가 좀 더 빠르다.
- { } : 딕셔너리 => 한쌍의 데이터가 이루어져 있다. 키:값
- 딕셔너리는 {}로 둘러싸이며, 키:값이 한쌍을 이루고, ','를 기준으로 구분된다.

In [54]:

```
dictdat1 = { 'key1':"value1", 'key2':"value2", 'key3':'value3'}
dictdat2 = { 11:"value1", 22:"value2", 33:'value3'}
dictdat3 = { 'key1':"value1", 'key2':(1,2,3,4,5), 'key3':[1,2,3,4,5] }

# dictdat3['키값']
print( dictdat1['key3'] )

# dictdat3['키값']
print( dictdat3['key3'] )
```

```
value3
[1, 2, 3, 4, 5]
```

3-5 튜플(tuple)

- 리스트와 달리 값이 변경과 수정이 어렵다.
- 장점은 필요한 기능이 줄기때문에 리스트에 비해 상대적으로 빠르다.

In [55]:

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

print( tuple1    )  # 전체 값 출력
print( tuple1[1] )  # 두번째 값 출력
```

```
(1, 2, 3)
2
```

튜플은 값의 수정이 안됩니다.

In [56]:

```
tuple1[1] = 20  # 에러 발생
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call
 last)
<ipython-input-56-4eb5dd8d8af4> in <module>
----> 1 tuple1[1] = 20  # 에러 발생

TypeError: 'tuple' object does not support item assignment
```

In [57]:

```
tuple1 = (1,2,3)
list1 = [1,2,3]
```

리스트와 튜플의 사용 객체 확인

In [58]:

```
print(tuple1.__sizeof__())
dir(tuple1)
```

```
48
```

Out[58]:

```
['__add__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__getnewargs__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__mul__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__rmul__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'count',
 'index']
```

In [59]:

```
print(list1.__sizeof__())
dir(list1)
```

```
64
```

Out[59]:

```
['__add__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__delitem__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__gt__',
 '__hash__',
 '__iadd__',
 '__imul__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__mul__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__reversed__',
 '__rmul__',
 '__setattr__',
 '__setitem__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'append',
 'clear',
 'copy',
 'count',
 'extend',
 'index',
 'insert',
 'pop',
 'remove',
 'reverse',
 'sort']
```

3-6 함수

In [60]:

```
def two_num_plus(a,b):
  print("두 값의 합은 :", a+b)
```

In [61]:

```
print( two_num_plus(3,5) )
```

```
두 값의 합은 : 8
None
```

In [65]:

```
def num_plus(a,b):
  print("두 값의 합은 :", a+b)
  return 1   # 실행 성공
```

In [67]:

```
result = num_plus(3,5)
if result == 1:
    print("정상적으로 수행 종료")
```

```
두 값의 합은 : 8
정상적으로 수행 종료
```

여러개의 값을 더하기

In [72]:

```
def multi_plus(*arg):
   sum = 0
   for one in arg:
     # print(one)
     sum += one
   return sum
```

In [74]:

```
value = multi_plus(3,5,6,2,3)
print("전체 합은 : ", value)
```

```
전체 합은 :  19
```

In [78]:

```
robot = {"안녕":'반가워 난 에프라고해',
         "날씨":'그래 날씨도 좋고, 너도 만나서 기쁘다 좋은 날이야',
         "이름":"난 에이미라고 해. 너는"}

print(robot.keys())
print(robot.values())
print(robot.items())
```

```
dict_keys(['안녕', '날씨', '이름'])
dict_values(['반가워 난 에프라고해', '그래 날씨도 좋고, 너도 만나서 기쁘다 좋은 날이
야', '난 에이미라고 해. 너는'])
dict_items([('안녕', '반가워 난 에프라고해'), ('날씨', '그래 날씨도 좋고, 너도 만나
서 기쁘다 좋은 날이야'), ('이름', '난 에이미라고 해. 너는')])
```

In [79]:

```
for i in range(3):
  word = input("대화를 입력해 주세요?(종료:q) ")
  if word in robot.keys():
    print( robot[word] )
  else:
    print("무슨이야기인지 아직 모르겠어. 미안")
```

```
대화를 입력해 주세요?(종료:q) q
무슨이야기인지 아직 모르겠어. 미안
대화를 입력해 주세요?(종료:q) 이름
난 에이미라고 해. 너는
대화를 입력해 주세요?(종료:q) 이름
난 에이미라고 해. 너는
```

실습 3

- 위의 실습2에 종료기능을 추가해 보자. q를 입력하면 더 이상 묻지 않고, 끝내기

4-1 클래스

In [80]:

```
# 클래스 기본 선언 형태
class Cal:
    pass
```

In [81]:

```
### 인스턴스
### 클래스를 선언 후, 다음과 같이 2대의 계산기를 만들 수 있다.
a = Cal()
b = Cal()
```

4-2 함수를 이용한 계산기 구현하기

- global을 이용하여 전역변수의 형태로 변수를 사용할 수 있다.

In [82]:

```
# 첫번째 계산기
result = 0
def plus1(num):
    global result
    result += num
    return result

print( plus1(3) )  # 기본값 0 + 3
print( plus1(7) )  # 기본값 0 + 3 + 7
```

```
3
10
```

In [83]:

```
# 첫번째 계산기
result1 = 0
def plus1(num):
    global result1
    result1 += num
    return result1

print( plus1(3) )
print( plus1(7) )

# 두번째 계산기
result2 = 0
def plus2(num):
    global result2
    result2 += num
    return result2

print( plus2(2) )
print( plus2(10) )

# 세번째 계산기
result3 = 0
def plus3(num):
    global result3
    result3 += num
    return result3

print( plus3(2) )
print( plus3(10) )
```

```
3
10
2
12
2
12
```

4-3 클래스를 활용한 계산기 만들기

In [84]:

```
class Cal:
    result = 0

    def plus(self, num):
        self.result += num
        return self.result

    def minus(self, num):
        self.result -= num
        return self.result

    def divide(self, num):
        self.result /= num
        return self.result
```

다섯대의 계산기 만들기

In [85]:

```
# 인스턴스 (객체를 생성)
c1 = Cal()
c2 = Cal()
c3 = Cal()
c4 = Cal()
c5 = Cal()
```

In [86]:

```
# 계산기1에 3을 두번 더한다.
print( c1.plus(3) )
print( c1.plus(3) )

# 계산기2에 4을 두번 더한다.
print( c2.plus(4) )
print( c2.plus(4) )

# 계산기3에 5, 5을 연속으로 더한다.
print( c3.plus(5) )
print( c3.plus(5) )

# 계산기4에 6, 6을 연속으로 더한다.
print( c4.plus(6) )
print( c4.plus(6) )
```

```
3
6
4
8
5
10
6
12
```

In [87]:

```
print( "계산기1 현재 결과 : " , c1.result )  # 0 + 3 + 3
print( "계산기2 현재 결과 : " , c2.result )  # 0 + 4 + 4
print( "계산기3 현재 결과 : " , c3.result )  # 0 + 5 + 5
print( "계산기4 현재 결과 : " , c4.result )  # 0 + 6 + 6
print( "계산기5 현재 결과 : " , c5.result )  # 0
```

```
계산기1 현재 결과 :  6
계산기2 현재 결과 :  8
계산기3 현재 결과 :  10
계산기4 현재 결과 :  12
계산기5 현재 결과 :  0
```

In [88]:

```
## 예외 상황 발생
print( c3.divide(0) )
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call
 last)
<ipython-input-88-b2a8c255e44d> in <module>
      1 ## 예외 상황 발생
----> 2 print( c3.divide(0) )

<ipython-input-84-d2d530d3388b> in divide(self, num)
     11
     12     def divide(self, num):
---> 13         self.result /= num
     14         return self.result

ZeroDivisionError: division by zero
```

4-4 나누기 에러, 기능 에러를 개선한 개선기

In [89]:

```
class Cal_change():
    pass
```

[기본] 기존의 클래스의 기능 상속이 가능하다.

- 기존의 클래스의 모든 기능에 대해 사용이 가능하다.
- 기본 문법

```
class 클래스명(상속받을 클래스명):
추가할 실행문1
추가할 실행문2
```

In [90]:

```
# 아무 것도 없다.
# 어떤 클래스를 상속을 받아서,
# 해당 클래스가 가진 변수 및 메소드를 사용 가능하다.
class Cal_change(Cal):
    pass
```

In [91]:

```
c_ch1 = Cal_change()

# 3을 더하고, 5를 빼기
print( "기능 개선 계산기 현재 결과 : " , c_ch1.result )
c_ch1.plus(3)
print( "기능 개선 계산기 현재 결과 : " , c_ch1.result )
c_ch1.minus(5)
print( "기능 개선 계산기 현재 결과 : " , c_ch1.result )
```

```
기능 개선 계산기 현재 결과 :  0
기능 개선 계산기 현재 결과 :  3
기능 개선 계산기 현재 결과 :  -2
```

오버라이딩(overridding)을 이해하기

- 기존의 메소드(함수)와 동일한 이름으로 정의할 경우, 상속받은 기능보다 우선적으로 기능이 수행된다.
- 기존에 0이 들어올 경우, 에러 발생하여, 이를 보완한 클래스 생성

In [92]:

```
# 오버라이딩
# 상속받아서 사용하는데, 기존에 상속받은 존재하는 메소드를 변경하는 것.
class Cal_change(Cal):
    def divide(self, num):
        if num==0:
            return "0으로 나눌 수 없습니다."
        else:
            self.result /= num
        return self.result
```

In [93]:

```
c_ch2 = Cal_change()

# 3을 더하고, 5를 빼기
print( "기능 개선 계산기 현재 결과 : " , c_ch2.result )
print( c_ch2.plus(3) )
print( "기능 개선 계산기 현재 결과 : " , c_ch2.result )
print( c_ch2.minus(5) )
print( "기능 개선 계산기 현재 결과 : " , c_ch2.result )
print( c_ch2.divide(0) )
```

```
기능 개선 계산기 현재 결과 :  0
3
기능 개선 계산기 현재 결과 :  3
-2
기능 개선 계산기 현재 결과 :  -2
0으로 나눌 수 없습니다.
```

(실습 4-2) Cal 클래스를 상속받아서,

- 해당 클래스에 하나의 메소드(곱하기)를 추가해 봅시다.
- 그리고 계산기에 C(0으로 세팅하는) 기능을 추가해 봅시다.
- 생성한 계산기로 아래 연산을 수행해 보자.
  - 초기값 + 10 / 0 * 5
  - C를 눌러 초기화
  - 결과값 + 5

In [94]:

```
# 오버라이딩
# 상속받아서 사용하는데, 기존에 상속받은 존재하는 메소드를 변경하는 것.
class Cal_change(Cal):
    def Czero(self):
        self.result = 0
        return self.result

    def mul(self, num):
        self.result *= num
        return self.result

    def divide(self, num):
        if num==0:
            return "0으로 나눌 수 없습니다."
        else:
            self.result /= num
        return self.result
```

In [95]:

```
c1 = Cal_change()
print( c1.plus(10) )
print( c1.divide(0) )
print( c1.mul(5) )
print( c1.Czero() )
print( c1.plus(5))
```

```
10
0으로 나눌 수 없습니다.
50
0
5
```

[실습] tv class를 생성해 보자. ¶

- tv class 이름 : Tv_Basic
- tv 채널 기본 : channel = 0
- tv 채널 변경 : change_channel(self, num) : num 채널로 변경