Python 기초와 실습 2

KOREATECH 한국기술교육대학교

KOREATECH 능력개발교육원

CONTENTS

01 기초와 실습 3

02 함수 (function)

03 클래스 (class)

KOREATECH 한국기술교육대학교

KOREATECH HRDI 능력개발교육원

01 기초와 실습 3

01-1 리스트(list)

KOREATECH 한국기술교육대학교

KOREATECH HRDI 능력개발교육원

List 사용

```
list1 = [1, 2, 3, 4, 5]
```

- 여러개의 값을 담을 수 있는 변수
- 값 읽어오기
  - 리스트를 사용할때는 0번째가 첫번째
  - 첫번째 값 list1[0]
  - 두번째 값 list1[1]
  - 뒤에서 첫번째 값 list1[-1]
  - 뒤에서 두번째 값 list1[-2]
  - 리스트에 들어있는 값 보다 큰 값을 읽어오려고 하면 에러
  - 예. 위의 list1에서 list1[5] 또는 list1[-6]은 에러
- 값 쓰기
  - 변수와 같이 list1[0]=10이라고 하면 list의 첫번째 값이 10으로 변경

List 사용

list의 길이는 len() 함수를 사용

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

```
5
```

실습

List 사용

```
animal_list = ['고래', '염소', '라마', '카피바라']
```

1. 첫번째 값 '고래'를 출력
2. 두번째 값 '염소'를 출력
3. 뒤에서 첫번째 값 '카피바라'를 출력
4. 뒤에서 두번째 값 '라마'를 출력
5. animal_list[5]를 출력
6. '라마'값을 '코끼리'로 변경.

실습

List 사용

```
[4]  rainbow=['빨강','주황','노랑','초록','파랑','남색','보라']
```

1. first_color이란 변수를 만드세요.
2. rainbow를 이용해서 first_color에 값을 저장하세요.
3. '무지개의 첫번째 색은 ___ 이다'의 형식으로 출력하세요.

***print() 함수의 {}와 .format() 함수**

List 수정

리스트에 새로운 값을 추가하는 방법

- `list1=[1,2,3]` 이라고 할 때
- append를 사용
  - `list1.append(4)`
  - append를 이용하면 리스트에 새로운 값이 추가된다.
- 뒤에 새로운 리스트를 더하기
  - `list2 = list1 + [4]`
  - list1은 그대로 두고, 새로운 리스트를 만들어 낸다.

List 수정

리스트에 값이 들어있는지 확인하는 방법

- in 연산을 이용

```
#12라는 값이 리스트에 들어있는지 확인하는 코드
n=12
if n in list1:
    print('{}가 리스트에 있다.'.format(n))
```

List 수정

리스트에서 필요 없는 값을 지우는 방법

- del을 이용해서 특정 위치의 값을 지우기
  - `del list1[10]` 리스트의 10번째 값을 지워라
- remove를 이용해서 특정 값을 지우기
  - `list1.remove(40)` 을 하면 리스트에 40이라는 값이 있는경우 삭제
  - 여러개의 값이 있는 경우 가장 앞에 있는 하나만 지워짐

실습

List 수정

1. list1에 1, 2, 3 리스트를 만듭니다.
2. append() 함수를 사용해서 4를 추가합니다.
3. list1를 출력해서 4가 잘 들어갔는지 확인합니다.

실습

List 수정

1. grade_list에 'A', 'B', 'C' 리스트를 만듭니다.
2. missing_grade_list에 'D' 리스트를 만듭니다.
3. 이 2개의 리스트를 합치면서, total_grade_list 변수에 지정합니다.
4. total_grade_list 를 출력합니다.

문제풀기

List 수정

1. numbers에 5가 들어있을때, “5가 있다”를 출력하도록 빈칸을 채워 보세요.

```
1  numbers = [1,2,3,4,5]
2  if            in            :
3      print("5가 있다")
```

실습

List 수정

1. list1에 1, 2, 3 리스트를 만듭니다.
2. **del**를 사용해서 list1에서 2를 지웁니다.
3. list1를 출력하여 2가 빠졌는지 확인합니다.

1. list1에 1, 2, 3 리스트를 만듭니다.
2. **remove()** 함수를 사용해서 list1에서 2를 지웁니다.
3. list1를 출력하여 2가 빠졌는지 확인합니다.

List 2차원

- 한 줄로 늘어선 1차원 리스트, 평면 구조의 2차원 리스트
- 2차원 리스트는 다음과 같이 가로×세로 형태로, 행(row)과 열(column) 모두 0부터 시작

List 2차원

2차원 리스트는 리스트 안에 리스트를 넣어서 만들 수 있으며 안쪽의 각 리스트는 ,(콤마)로 구분

- 리스트 = [[값, 값], [값, 값], [값, 값]]

```
a = [[10, 20], [30, 40], [50, 60]]
a
```

```
[[10, 20], [30, 40], [50, 60]]
```

List 2차원

- 가로 2, 세로 3의 2차원 리스트
- 리스트를 한 줄로 입력했지만 가로, 세로를 알아보기 쉽게 세 줄로 입력 가능

```
a = [[10, 20],
     [30, 40],
     [50, 60]]
a
```

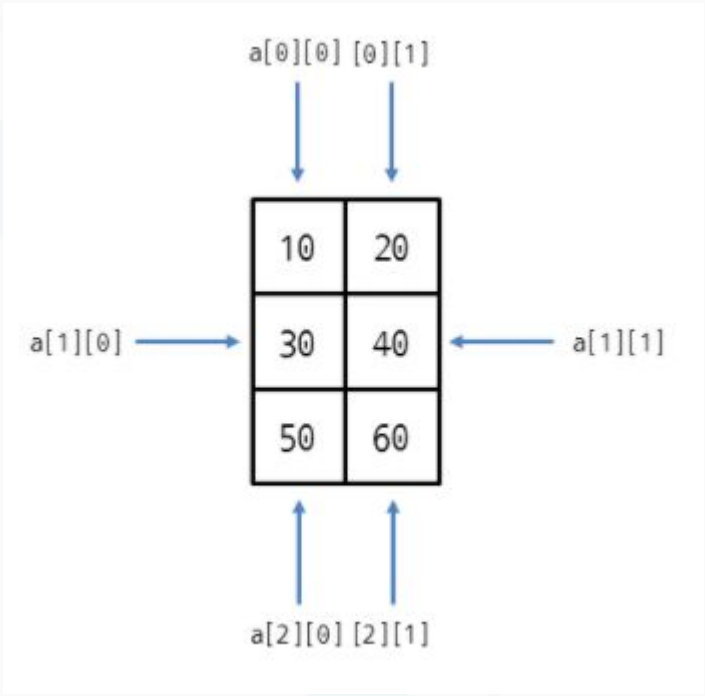
```
[[10, 20], [30, 40], [50, 60]]
```

List 2차원

2차원 리스트의 요소에 접근하거나 값을 할당할 때는 리스트 뒤에 [](대괄호)를 두 번 사용하며 [] 안에 세로(row) 인덱스와 가로(column) 인덱스를 지정

- **리스트[세로인덱스][가로인덱스]**
- **리스트[세로인덱스][가로인덱스] = 값**

2차원 리스트도 인덱스는 0부터 시작 따라서 리스트의 가로 첫 번째, 세로 첫 번째 요소는 `a[0][0]`




```
a = [[10, 20],
     [30, 40],
     [50, 60]]
a
```

```
[[10, 20], [30, 40], [50, 60]]
```

List 2차원

```
a = [[10, 20],
     [30, 40],
     [50, 60]
    ]
print('a[0][0] = ',  a[0][0])
print('a[1][1] = ', a[1][1])
a[2][1] = 1000
print('a = ', a)
```

```
a[0][0] =  10
a[1][1] =  40
a =  [[10, 20], [30, 40], [50, 1000]]
```

List 2차원

톱니형 리스트

2차원 리스트 [[10, 20], [30, 40], [50, 60]]은 가로 크기가 일정한 사각형 리스트입니다. 하지만, 파이썬에서는 가로 크기가 불규칙한 톱니형 리스트(jagged list)도 만들 수 있습니다.

append() 함수도 사용 가능

```
a = [[10, 20],
     [500, 600, 700],
     [9],
     [30, 40],
     [8],
     [800, 900, 1000]]
```

실습

List 2차원

아래의 예제로 2차원 리스트를 만들어 봅시다.

Americano 1500

Latte 2000

Espresso 1700

Mocha 2500

식혜 2000

수정과 1900

실습

List 2차원

1. Mr. 바틀즈님은 아래의 학습 데이터를 리스트로 만들고 싶어 합니다.

이름	키	몸무게	성별
피리	40	19	M
노아	45	23	M
레이디	48	26	F

2. 피리가 살이 3 빠졌네요. 리스트를 수정하세요.
3. 다음 새로운 데이터를 추가하세요 - 허스키 / 65 / 38 / M
4. 다음 새로운 attribute/column(나이) 값을 입력하세요.
   a. 피리 - 17, 노아 - 15, 레이디 - 13, 허스키 - 13

List - 인덱스 슬라이싱 [:]

Index Slicing - 리스트에서 원하는 부분을 추출(슬라이싱) 하기

- 인덱스의 숫자를 사용
- 콜론(:) 앞과 뒤에 숫자를 사용
- 앞에 써주는 숫자 - 시작 인덱스(오프셋 offset)
- 뒤에 써주는 숫자 - 추출을 끝내려는 인덱스에 1을 더해준 값

중요!

콜론 왼쪽 숫자 = 추출하기 원하는 시작 인덱스

콜론 오른쪽에 써주는 숫자 = 추출하기 원하는 끝 인덱스 + 1

List - 인덱스 슬라이싱 [:]

Monday이 오프셋(인덱스)은 0, Tuesday는 1, Wednesday는 2, Thursday는 3, Friday는 4.
따라서 week[2:5] 란? week 2인 Wednesday 부터, 5 - 1 = 4 즉, 4는 Friday 이므로
['Wednesday', 'Thursday', 'Friday']가 출력 됨.

```
week = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']
week[0:2]
```

```
['Monday', 'Tuesday']
```

같이하기

List - 인덱스 슬라이싱 [:]

Nested indexing (중첩 인덱싱)

Nested List (중첩 리스트)에서도 원하는 값을 슬라이싱 (Slicing) 가능

아래의 `my_list[0:2]` 결과 값은?

```
my_list = ["Mitch", [3, 6, 7], ["yellow", 5, 6]]
my_list[0:2]
```

실습

List - 인덱스 슬라이싱 [:]

Nested indexing (중첩 인덱싱)

```
my_list = ["Mitch", [3, 6, 7], ["yellow", 5, 6]]
my_list[0:2]
```

위의 결과값을 확인했습니다.

이제, 중간의 `[6, 7]` 값을 출력하려고 하면, 어떻게 할까요?

실습

List - 인덱스 슬라이싱 [:]

Nested indexing (중첩 인덱싱)

중간의 `[6, 7]` 값을 가져오기 위해서,

- 먼저 `my_list[1]` 로 접근하면, `[ 3, 6, 7 ]`
- 여기서 6과 7만 가져와야 하므로, 6은 인덱스가 1이고, 7은 인덱스가 2
- 시작인덱스는 1 , 끝 인덱스는 2 + 1 = 3 으로 써주면
- => `my_list[ 1 ][ 1 : 3 ]`

```
week = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']
```

List - 인덱스 슬라이싱 [:]

숫자생략이 가능

[start:] start오프셋(인덱스)부터 끝까지

[:end] 처음부터 end-1 오프셋(인덱스)까지

- `week[3:]`
- `week[:2]`
- `week[::2]` ? 이건 값이 이상하네요?

  => 다음 페이지에서 설명

```
[5]  week[3:]

     ['Thursday', 'Friday']
```

```
[6]  week[:2]

     ['Monday', 'Tuesday']
```

```
[8]  week[::2]

     ['Monday', 'Wednesday', 'Friday']
```

같이하기

List - 인덱스 슬라이싱 [:]

step

[start : end : step] step만큼 문자를 건너뛰면서 추출

예)

```
rainbow=['빨강','주황','노랑','초록','파랑','남색','보라']
rainbow[0:8:2]
```

01 기초와 실습 3

01-2 for 반복문

KOREATECH 한국기술교육대학교

KOREATECH 능력개발교육원

for in list

for in 반복문

- 코드를 필요한만큼 반복해서 실행

```
for pattern in patterns:
    print (pattern)
```

- 리스트 patterns의 값을 하나씩 꺼내 pattern으로 전달
- 리스트의 길이만큼 print (pattern) 실행
- : (콜론)은 꼭 사용해야 하며, 들여쓰기도 중요

문제풀기

for in list

1. 리스트 `list = ['가위','바위','보']` 를 만들어 봅니다.
2. **`for in`** 문을 활용해서 list의 내용을 한줄씩 출력하는 프로그램을 작성해 보세요.

for in range

range() 함수

- 필요한 만큼의 숫자를 만들어내는 유용한 기능

```
for i in range(5):
  print(i)
```

```
0
1
2
3
4
```

for in range

enumerate() 함수

- 리스트가 있는 경우 순서와 리스트의 값을 전달하는 기능

```
names = ['철수', '영희', '영수']
for i, name in enumerate(names):
    print('{}번: {}'.format(i + 1, name))
```

```
1번: 철수
2번: 영희
3번: 영수
```

실습

for in range

1. 빈 리스트 a를 만드세요.
2. range() 함수와 append() 함수를 사용해서 아래와 같은 리스트를 만든 후, 출력합니다.

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

실습

for in range

1. range() 함수 사용해서 0부터 3까지 출력해봅니다.

실습

for in range

1. range() 함수 사용해서 무지개의 순서와 색을 출력해 보세요.
    - 단, rainbow에 새로운 값이 추가되더라도 그 값을 모두 출력할 수 있도록 len을 이용해야 합니다.

hint1: `rainbow=["빨","주","노","초","파","남","보"]`

hint2: `print('{}번째 색은 {}'.format(i+1,color))`

실습

for in range

1. enumerate() 함수 사용해서 무지개의 순서와 색을 출력해 보세요.
    - len() 함수를 사용할 필요가 없습니다.

hint1: `rainbow=["빨","주","노","초","파","남","보"]`

hint2: `print('{}번째 색은 {}'.format(i+1,color))`

실습

for in range

1. 변수 days에는 1월부터 12월까지 그 달에 포함된 날짜수가 정리되어 있습니다.
2. for in문과 range() 또는 enumerate()를 이용해서 다음과 같이 출력되도록 만들어 보세요.

```
1월의 날짜수는 31일 입니다.
2월의 날짜수는 29일 입니다.
...
```

hint1: 위와 같은 형식으로 12월까지 출력하세요. 출력 형식은 코드의 print문을 활용해서 format()의 괄호 안을 채워 넣으면 됩니다.

hint2: 순회할 리스트가 정해져 있고, 그 리스트에서 하나씩 꺼내 쓰기만 되는 상황이라면 for in list를, 순회할 횟수가 정해져 있거나 1씩 증가하는 숫자가 필요하다면 for in range()를 사용하는 것이 좋습니다.

실습

for문으로 2차원 리스트 만들기

1. 빈 리스트 a를 만드세요.
2. for 문을 두번 사용해야 합니다.
3. 열과 행의 2개의 변수가 필요합니다.
4. append() 함수를 두번 사용해야 합니다.

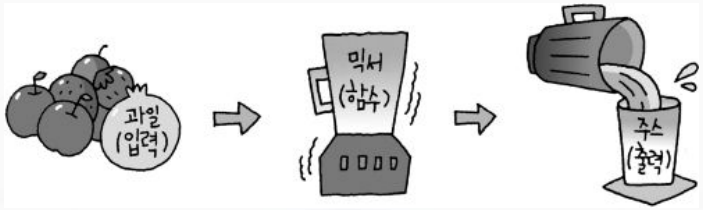
```
[[0, 0], [0, 0], [0, 0]]
```

02 함수 (function)

KOREATECH 한국기술교육대학교

KOREATECH 능력개발교육원

함수란?



함수를 설명하기 전에 믹서를 생각해 보자. 우리는 믹서에 과일을 넣는다. 그리고 믹서를 사용해서 과일을 갈아 과일 주스를 만든다. 우리가 믹서에 넣는 과일은 "입력"이 되고 과일 주스는 "출력(결과값)"이 된다.

그렇다면 믹서는 무엇인가? (믹서는 과일을 입력받아 주스를 출력하는 함수와 같다.)

우리가 배우려는 함수가 바로 믹서와 비슷하다. 입력값을 가지고 어떤 일을 수행한 다음에 그 결과물을 내어놓는 것, 이것이 바로 함수가 하는 일이다. 우리는 어려서부터 함수에 대해 공부했지만 함수에 관해 깊이 생각해 본 적은 별로 없다. 예를 들어 y = 2x + 3도 함수이다. 하지만 이를 수학 시간에 배운 직선 그래프로만 알고 있지 x에 어떤 값을 넣었을 때 어떤 변화에 의해서 y 값이 나오는지 그 과정에 대해서는 별로 관심을 두지 않았을 것이다.

이제 우리는 함수에 대해 조금 더 생각해 보는 시간을 가져야 한다. 프로그래밍에서 함수는 정말 중요하기 때문이다. 자, 이제 파이썬 함수의 세계로 깊이 들어가 보자.

함수는 왜 사용하는가?

- 똑같은 내용을 반복해서 작성할때
- "반복적으로 사용되는 가치 있는 부분"을 한 뭉치로 묶음
- 어떤 입력값을 주었을 때 어떤 결과값을 돌려준다
- 프로그램을 함수화하면 프로그램 흐름을 일목요연하게 볼 수 있음.
- 프로그램 흐름도 잘 파악할 수 있고 오류가 어디에서 나는지도 알기 쉬움
- 함수를 잘 사용하고 함수를 적절하게 만들 줄 아는 사람이 능력 있는 프로그래머

파이썬 함수의 구조

```
def 함수명(매개변수):
    <수행할 문장1>
    <수행할 문장2>
    ...
```

- **def**는 함수를 만들 때 사용하는 예약어
- 함수 이름은 자유
- 함수 이름 뒤 괄호 안의 parameter(매개변수)는 이 함수에 입력으로 전달되는 값을 받는 변수

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

- "이 함수의 이름(함수명)은 add이고 입력으로 2개의 값을 받으며 결과값은 2개의 입력값을 더한 값이다."
- 여기에서 return은 함수의 결과값을 돌려주는 명령어이다.

함수란?

1. 함수는 코드의 덩어리에 이름을 붙인 것이다.
2. 새 함수를 정의할 수 있다.
3. print()는 미리 만들어진 함수이다.
4. 함수를 한번 만들고 나면, 그 안은 잊어버려도 좋다.

```
[35] def hello():  # 함수의 정의
       print('안녕, 함수!')
```

```
print('첫줄 실행')
hello()       # 함수의 호출
print('끝줄 실행')
```

```
첫줄 실행
안녕, 함수!
끝줄 실행
```

매개변수 (parameter)

- 매개변수 - 함수를 정의할 때 사용하는 이름
- 실행인자 - 함수를 실행할 때 넘기는 변수, 값
- 매개변수와 실행 인자
  - 매개변수와 실행 인자의 개수는 동일해야 한다.
  - 여러 개일 경우 쉼표로 구분

매개변수

```
def print_round(number):     # 함수의 정의
    rounded = round(number)
    print(rounded)

print_round(4.6)         # 함수의 호출
print_round(2.2)
```

```
5
2
```

- 매개변수 - 함수를 정의할 때 사용하는 이름
- 실행 인자 - 함수를 실행할 때 넘기는 변수, 값
- 매개변수와 실행 인자
  - 매개변수와 실행 인자의 개수는 동일해야 한다.
  - 여러 개일 경우 쉼표로 구분

함수의 값 (return)

- return을 이용해 값을 돌려줌
- 여러 값 반환 - return 뒤에 여러 값을 쉼표로 구분해서 값을 보내고, 받을때도 쉼표로 구분하여 받는다.

```
def add_10(value):
  result = value + 10
  return result

n = add_10(5)
print(n)
```

```
15
```

실습

함수 (function)

1. 입력값이 없는 함수를 만드세요.
2. 결과값이 없는 함수를 만드세요.
3. 입력값도 결과값도 없는 함수를 만드세요.

실습

함수 (function)

1. a의 값은 5, b의 값은 7입니다.
2. a와 b를 **더한(+)** 값을 저장하고 결과를 출력 부분을 함수로 만들어 보세요.
3. 함수의 이름은 add로 만들어야 합니다.
4. 함수를 만들고 나면 함수를 사용해서 결과가 출력합니다.
5. a값을 15로 바꾼후, 함수를 다시 사용해서 바뀐 결과를 확인합니다.

실습

함수 (function)

1. a의 값은 5, b의 값은 7입니다.
2. a와 b를 **곱한(*)** 값을 저장하고 결과를 출력 부분을 함수로 만들어 보세요.
3. 함수의 이름은 mul로 만들어야 합니다. (mul은 multiply의 줄임말)
4. 함수를 만들고 나면 함수를 사용해서 결과가 출력합니다.
5. b값을 -3으로 바꾼후, 함수를 다시 사용해서 바뀐 결과를 확인합니다.
6. a값을 0으로 바꾼후, 함수를 다시 사용해서 바뀐 결과를 확인합니다.

문제풀기

함수 (function)

1. 전에 만들었던 2개의 함수를 합치면 더 좋은 함수가 되겠네요. add_and_mul이란 함수를 만들어봅니다
2. 이 함수는 2개의 return값이 있습니다.
   a. `return a + b`
   b. `return a * b`
3. 2번 사용해서 함수를 만들었는데 머가 좀 이상하네요. 어떤 문제가 있나요?

실습

함수 (function)

1. age 값을 받는 함수 print_stage()를 만들어 봅니다.
2. parameter는 age라고 정합니다.
3. 함수안에 조건문을 사용할 수 있습니다.
    a. age < 10 경우, 'kid' 을 출력
    b. age < 20 경우, 'teen' 을 출력
    c. age < 65 경우, 'adult' 을 출력
    d. age >= 65 경우, 'senior' 을 출력
4. 값을 5, 15, 44, 80의 나의 넣어 함수가 잘 만들어 졌는지 확인하세요.

03 클래스 (class)

KOREATECH 한국기술교육대학교

KOREATECH HRDI 능력개발교육원

자료형(type) 복습하기

class는 type()을 배울 때 잠깐 본적 있습니다.

```
print(type("string"))
print(type(1))
```

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

클래스란?

- 변수와 함수를 모아놓은 것
- 객체(Object)
- 인스턴스(instance)

조금 쉽게 이해해 보기

어린시절 뽑기를 해 본적이 있다면 아래 그림과 비슷한 모양의 뽑기 틀을 본적이 있을 것이다. 뽑기 아저씨가 뽑기를 불에 달군 후 평평한 바닥에 '탁'하고 소리나게 떨어뜨려 넓적하고 동그랗게 만든후에 아래와 비슷한 틀로 모양을 찍어준다. 찍어준 모양대로 모양을 만들어 오면 아저씨는 뽑기 한개를 더 해 준다.
이곳에서 설명할 클래스라는 것이 마치 위 뽑기의 틀(별모양, 하트모양)과 비슷하다. 별 모양의 틀(클래스)로 찍으면 별모양의 뽑기(인스턴스)가 생성되고 하트 모양의 틀(클래스)로 찍으면 하트모양의 뽑기(인스턴스)가 나오는 것이다.
클래스란 똑같은 무엇인가를 계속해서 만들어 낼 수 있는 설계도면 같은 것이고(뽑기 틀), 인스턴스란 클래스에 의해서 만들어진 피조물(별 또는 하트가 찍혀진 뽑기)을 뜻하는 것이다.

클래스 만들기

1. class 키워드를 사용
2. 끝에 :(콜론) 사용
3. 들여쓰기는 필수
4. class안에 변수를 저장

```
[7]  class Person:
         name = "철수"
```

```
person1 = Person()
person1.name
```

```
'철수'
```

클래스 속성 변경

```
class Person:
  name = "철수"
```

```
[9] person1 = Person()
    person1.name
```

```
'철수'
```

```
person1.name = "Pamela Harris"
person1.name
```

```
'Pamela Harris'
```

클래스 만들기

함수 만들기

```
[34] class Person:

        def print_name(self):
          print("print_name() 안에 Mr. 바틀즈")
```

```
bartles.print_name()
```

```
print_name() 안에 Mr. 바틀즈
```

self란?

- class의 인스턴스(instance)를 나타내는 변수
- self는 class내 들의 첫번째 인자로 전달

클래스 만들기

self를 사용해 함수로 내장된 변수 사용하기

- class안에 있는 변수를 함수에서 사용할 때 self를 사용

```
[36] class Student:
       name = "김철수"
       def info(self):
         print("제 이름은 " + self.name + "입니다.")
```

```
[37] inst = Student()
     type(inst)
```

```
__main__.Student
```

```
[38] inst.name
```

```
'김철수'
```

```
inst.info()
```

```
제 이름은 김철수입니다.
```

같이하기

클래스 만들기

1. Dog class 만드세요.
2. 이 Dog class안에, cry()난 함수를 만들고 “왈왈!”를 프린트하세요.
3. dog1은 이 클래스를 사용해서 인스턴스를 만드세요.
4. cry()함수를 부르세요
5. self를 필요 없을것 같으니, 뺀 후, 다시 실행해 보세요.
6. 왜 이렇게 되었을까요?

클래스 만들기

"cry()는 위치 인수(positional argument)를 아무것도 받지 않지만 한 개가 더 들어왔다는 얘기입니다. 우리는 인수를 써주지 않았는데 누가 넘겼다는 걸까요? 파이썬은 dog가 클래스 Dog의 인스턴스이고, dog.cry()와 같이 인스턴스 함수를 호출하려고 하면 파이썬은 내부적으로 'Dog.cry(dog)'와 같이 변형합니다."

```
class Dog:
  def cry():
    print("왈왈!")

dog = Dog()
dog.cry()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-42-ae91896b4daf> in <module>()
      4
      5 dog = Dog()
----> 6 dog.cry()

TypeError: cry() takes 0 positional arguments but 1 was given

SEARCH STACK OVERFLOW
```

같이하기

클래스 만들기

이런 문제를 해결하려면, 인스턴스를 만들지 않고 그냥 call하면 되겠네요.

어떤 결과가 나올까요?

인스턴스를 왜 중요할까요?

```
class Dog:
  def cry():
    print("왈왈!")

Dog.cry()
```

실습

아래 내용으로 클래스를 만드세요

- 클래스명 - Contact
- 이 클래스 안에 3개의 변수:
  - name - default(기본값)은 “이름”
  - phone - default는 ‘011-0000-0000’
  - email - default는 ‘example@example.com’
- 함수 print_name()은 name을 출력 - format은 `name: {}`
- 함수 print_phone()은 phone을 출력 - format은 `phone: {}`
- 함수 print_email()은 email을 출력 - format은 `email: {}`
- 함수 info는 name, phone, email 3개를 모두 출력
- 함수 change_name()은 name을 변경
  - tip: 함수에 parameter(매개변수)를 사용

실습

아래 내용으로 클래스를 만드세요

- 클래스명 - Calculator
- 전에 만들었던 add() 와 mul()의 함수를 사용
  - add(a, b)는 2개의 매개변수를 받고, 합을 출력
  - mul(a, b)는 2개의 매개변수를 받고, 곱을 출력

- class instance를 만든 후, add(a, b)를 사용
- mul(a, b)를 사용하여 계산이 맞는지 확인

Reference

https://dojang.io/mod/page/view.php?id=2291

https://vision-ai.tistory.com/entry/%ED%8C%8C%EC%9D%B4%EC%8D%AC-%EB%A6%AC%EC%8A%A4%ED%8A%B8-%EC%BD%9C%EB%A1%A0-%EC%8A%AC%EB%9D%BC%EC%9D%B4%EC%8B%B1-List-Slicing

https://velog.io/@kpl5672/python-2%EC%B0%A8%EC%9B%90%EB%A6%AC%EC%8A%A4%ED%8A%B8

https://wikidocs.net/24

http://hleecaster.com/python-class/

https://sjs0270.tistory.com/140

https://blog.hexabrain.net/284