

# LLM 활용 인공지능 서비스 개발자 양성과정

도봉 SeSAC 캠퍼스 X **Saltlux**

강사 최동혁

# 파이썬: 변수

## 변수란 무엇인가?

- 파이썬에서 변수는 "객체"의 별칭(별명)
- 객체는 메모리에 저장되어 있다. 변수는 이 사물함에 붙어 있는 이름표라고 생각하면 된다.
  - (참고1) 사실 이 이름표도 어딘가에 저장되어 있어 사물함 근처에 이름표가 있는 것은 아닙니다. 이해의 편의상 이 둘이 붙어 있다고 생각해 주세요.
  - (참고2) 이름표라고 표현한 것은 사물함의 이름표만 따로 떼어서 다른 사물함에 옮겨 붙일 수도 있기 때문입니다. 즉, 사물함과 이름표는 서로 독립적입니다.

## id 함수

- 객체가 저장된 메모리 주소를 반환
- (사실은 주소가 아니지만 그렇게 생각해도 무방)
- <https://docs.python.org/3/library/functions.html#id>

```
a = 3  
b = 2
```

```
print(id(a), id(b))
```

✓ 0.0s

4520201400 4520201368

- a=b 를 하면 id값은 어떻게 될까?

## id 함수

- 파이썬은 0 ~ 256까지는 변수가 달라도 같은 객체로 취급한다.
  - 왜 그럴까요?

```
a = 3
b = 3
```

```
print(id(a), id(b))
```

✓ 0.0s

4520201400 4520201400

```
a = 257
b = 257
```

```
print(id(a), id(b))
```

✓ 0.0s

4550000432 4549998992

## (참고) 파이썬의 내장함수들

1. `help()` 객체의 사용법이 궁금하면 `help()` 함수를 사용하거나 블록 지정 또는 마우스 오버하여 나오는 설명을 읽어 본다.
2. `print()` 값을 출력할 때 사용하는 기본적인 함수이다. 문자열, 숫자 등 다양한 타입의 데이터를 화면에 출력할 수 있다.
3. `len()` 리스트, 문자열, 튜플 등의 길이를 반환한다. 데이터 구조의 요소의 수를 알고 싶을 때 유용하게 사용된다.
4. `type()` 변수나 값의 데이터 타입을 확인하기 위해 사용한다.
5. `sorted()` 리스트나 튜플과 같은 반복 가능한 객체를 정렬하여 반환한다.
6. `input()` 사용자로부터 입력을 받기 위한 함수이다. 기본적으로 문자열 형태로 값을 반환한다.
7. `int()`, `float()`, `str()` 이들 함수는 각각 정수, 실수, 문자열로 타입을 변환하는 데 사용된다.
8. `list()`, `tuple()`, `set()`, `dict()` 이들 함수는 각각 리스트, 튜플, 집합, 딕셔너리로 데이터 타입을 변환하는 데 사용된다.
9. `max()`, `min()` 숫자나 문자열로 이루어진 리스트, 튜플 등에서 최대값 또는 최소값을 반환한다.
10. `range()` 연속된 숫자의 시퀀스를 생성한다. 주로 for 루프에서 반복을 위해 사용된다.

# (참고) 파이썬의 내장함수들

11. `round()` 숫자를 주어진 소수점 아래로 반올림한다.
12. `sum()` 숫자 리스트나 튜플의 모든 항목의 합을 반환한다.
13. `zip()` 두 개 이상의 리스트나 튜플을 합쳐서 새로운 튜플의 시퀀스를 생성한다.
14. `any()`, `all()` 주어진 반복 가능한 객체의 요소 중 하나라도 참이면 `any()` 는 `True` 를 반환하고, 모든 요소가 참이면 `all()` 은 `True` 를 반환한다.
15. `enumerate()` 리스트, 튜플 등의 반복 가능한 객체의 인덱스와 값을 함께 반환한다. 주로 for 루프에서 인덱스와 함께 반복을 위해 사용된다.
16. `filter()` 주어진 함수를 사용하여 반복 가능한 객체의 요소를 필터링한다. 결과는 필터링된 요소들의 새로운 리스트이다.
17. `map()` 주어진 함수를 반복 가능한 객체의 모든 요소에 적용한다. 결과는 해당 함수에 의해 변환된 요소들의 새로운 리스트이다.
18. `lambda` 작은 익명 함수를 생성한다. `filter()`, `map()` 및 `sorted()` 와 같은 함수와 함께 사용되는 경우가 많다.
19. `open()` 파일을 열거나 생성할 때 사용한다. 다양한 모드(예: 읽기, 쓰기)로 파일을 열 수 있다.
20. `reversed()` 리스트나 튜플과 같은 시퀀스를 거꾸로 반환한다.
21. `slice()` 시퀀스를 슬라이스하여 특정 부분만 선택한다.
22. `chr()`, `ord()` `chr()` 는 주어진 숫자에 해당하는 문자를 반환하고, `ord()` 는 주어진 문자에 해당하는 숫자를 반환한다.
23. `bin()`, `hex()`, `oct()` 주어진 숫자를 각각 이진수, 16진수, 8진수 문자열로 변환한다.
24. `abs()` 숫자의 절대값을 반환한다.
25. `globals()`, `locals()` 현재의 전역 변수와 지역 변수를 딕셔너리 형태로 반환한다.

## (참고) 파이썬의 Builtin 객체들

- 내장함수 살펴보기

```
import builtins  
dir(builtins)
```

✓ 0.0s

```
'BlockingIOError',  
'BrokenPipeError',  
'BufferError',  
'BytesWarning',  
'ChildProcessError'
```



# 변수란 무엇인가?

- 파이썬에서 변수는 "객체"의 별칭(별명)
- 객체는 메모리에 저장되어 있다. 변수는 이 사물함에 붙어 있는 이름표라고 생각하면 된다.
  - (참고1) 사실 이 이름표도 어딘가에 저장되어 있어 사물함 근처에 이름표가 있는 것은 아닙니다. 이해의 편의상 이 둘이 붙어 있다고 생각해 주세요.
  - (참고2) 이름표라고 표현한 것은 사물함의 이름표만 따로 떼어서 다른 사물함에 옮겨 붙일 수도 있기 때문입니다. 즉, 사물함과 이름표는 서로 독립적입니다.
  - (참고3) **id()** 함수
    - 객체가 저장된 메모리 주소를 반환
    - (사실은 주소가 아니지만 그렇게 생각해도 무방)
    - <https://docs.python.org/3/library/functions.html#id>

```
a = 3
b = 2

print(id(a), id(b))
```

✓ 0.0s

4520201400 4520201368

# 파이썬: 자료형

# 자료형이란 무엇인가?

- 파이썬에서 모든 것은 객체. 즉, 클래스가 존재.
  - 어떤 클래스들은 너무어무 자주 쓰여서 특별한 이름으로 부르는데, 그것이 자료형
  - 예
    - 정수(int)
    - 문자열(str)
    - ...

# 숫자 자료형

- 정수
  - int: 32자리
  - long: 무제한의 자릿수(Python3에서 이것이 int가 됨.)
- 실수(float)
  - float: 32자리
  - double: 64자리(파이썬에서는 없음)

# 숫자 자료형

- 정수
  - int: 32자리
  - long: 무제한의 자릿수(Python3에서 이것이 int가 됨.)
- 실수(float)
  - float: 32자리
  - double: 64자리(파이썬에서는 없음)
  - 두 실수가 같은지 판단하기 위해서는 ==를 사용하면 안됨.

0.1 + 0.2

✓ 0.0s

0.30000000000000004

# 문자 자료형

- 숫자 자료형과 문자 자료형을 구분해서 쓰자

```
age = 33  
print(age)
```

```
age = '33'  
print(age)
```

# 문자 자료형

- 이스케이프 문자

- `\n` : 줄바꿈,
- `\t` : 탭,
- `\\` : `\` 문자,
- `\'` : `'` 문자,
- `\"` : `"` 문자,
- `\r` : 캐리지 리턴,
- `\f` : 폼 피드,
- `\a` : 벨 소리,
- `\b` : 백 스페이스,
- `\000` : 널 문자

# 문자 자료형

- 줄바꿈 문자는 실습을 해봅시다.

```
multiline = "산산이 부서진 이름이여!\n허공 중에 헤어진 이름이여!\n불러도 주인 없는 이름이여!\n부르다가 내가 죽을 이름이여!"  
print(multiline)
```



# 문자 자료형

- 문자열 포매팅
  - (주의) 포매팅은 원래 값을 바꾸는 것이 아니다.
- 다음 3가지 방법이 있다.
  1. % 연산자
    - %s, %d, %f
  2. f-string
    - f"문자열 {변수명}"
  3. format() 메소드
    - "문자열".format()

# 문자 자료형

## 1. % 연산자

```
season = 16  
print("나는 솔로 %d기" % season)
```

]

# 문자 자료형

## 2. f-string

```
brand = "Starbucks"  
menu = "Americano"  
f'제가 제일 좋아하는 카페는 {brand}입니다. 그 중에서도 {menu}가 가장 좋아요.'
```

```
age = 34  
print(f'제 나이는 윤석열 나이로 {age-1}살입니다.')
```

연산도 된다.

# 문자 자료형

## 3. format 메소드

```
mbti = 'INFP'  
blood = 'B'  
introduction = '제 MBTI는 {}이고, 혈액형은 {}형입니다.'.format(mbti, blood)  
print(introduction)
```

# 문자 자료형

## 따옴표

- 작은 따옴표, 큰 따옴표, 세 작은 따옴표, 세 큰 따옴표 모두 사용 가능하다.

### 1. 작은 따옴표

```
greeting = '안녕하세요?'  
print(greeting)
```

### 3. 작은 따옴표 안에 큰 따옴표 포함하기

```
quote = '미셸 투르니에: "일은 인간의 본성에 맞지 않는다. 하면 피곤해지는 게 그 증거다."'
```

### 2. 큰 따옴표

```
answer = "네, 안녕하세요"
```

### 4. 큰 따옴표 안에 작은 따옴표 포함하기

```
lyric = "What's your ETA? What's your ETA?"
```

# 문자 자료형

## 따옴표

- 작은 따옴표, 큰 따옴표, 세 작은 따옴표, 세 큰 따옴표 모두 사용 가능하다.

### 5. 세 작은 따옴표 또는 세 큰 따옴표로 여러 줄의 문자열 정의하기

```
multiline = '''저 폰은 바다에 버려요.  
깊은 데 빠뜨려서, 아무도 못 찾게 해요.'''  
print(multiline)
```

```
multiline2 = """그 시절은 지나갔고  
이제 거기 남은 건  
아무것도 없다.  
"""  
print(multiline2)
```

# 문자 자료형

문자열 연산은 다음과 같은 것들이 있다.

- 더하기
- 곱하기
- 길이 구하기
- 인덱싱
- 슬라이싱

# 문자 자료형

## 1. 문자열 더하기

```
first = "국경의 긴 터널을 빠져나오자, "  
second = "설국이었다."
```

```
sentence = first + second  
print(sentence)
```

## 2. 문자열 곱하기

```
arirang = "아리랑"  
print(arirang*3)
```

## 3. 문자열 길이 구하기

```
len(arirang)
```



# 문자 자료형

## 1. 문자열 더하기

```
first = "국경의 긴 터널을 빠져나오자, "  
second = "설국이었다."
```

```
sentence = first + second  
print(sentence)
```

## 2. 문자열 곱하기

```
arirang = "아리랑"  
print(arirang*3)
```

## 3. 문자열 길이 구하기

```
len(arirang)
```

## 문자 자료형

### 4. 문자열 인덱싱 (위치)

```
# 파이썬은 0부터 시작한다.  
string = 'Monty Python'  
string[0]
```

```
# 음수를 넣는다면?  
string[-1]
```

0	1	2	3	4	5	6	7	8	9	10	11
M	o	n	t	y		P	y	t	h	o	n
-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
[-12:-7]						[6:10]					

# 문자열 메소드

문자열 연산을 더욱 쉽게 하기 위한 메소드들이 있다.

1. `count()`
2. `join()`
3. `upper()`
4. `lower()`
5. `strip()`, `lstrip()`, `rstrip()`
6. `replace()`
7. `split()`

Jupyter Hub에서 실습해 봅시다.