2025.04.01. SR 파이썬 스터디 1주차

Jupyter Notebook, 조코딩 02(1). 숫자형, 문자열, 리스트

✅ Jupyter 설치 및 사용법

Jupyter Keymap Extension 설치

: vscode에서 주피터를 실행했을 때

Jupyter Notebook이란?

- 파이썬 파일 편집 프로그램.
- 데이터과학에서 많이 사용됨.
- .py 확장자명으로 저장되는 기존 파이썬과 달리, .ipynb 확장자명으로 저장됨.
- 코드를 구획별로 묶어서 짤 수 있음.
- 마크다운(markdown) 기능을 활용하면 마치 노트를 정리하거나 워드파일을 쓰듯이 메모해가며 파이썬 코드를 짤 수 있음.

Jupyter 사용법

- shift+enter: 현재 셀 실행하고 다음 셀로 이동
- ctrl+enter: 현재 셀 실행
- esc: 읽기 모드로 변경(방향키로 셀 이동 가능, 셀 더블클릭이나 enter로 다시 편집 가능)
- b: 현재 셀 아래에 셀 생성
- a: 현재 셀 위에 셀 생성
- dd: 현재 셀 삭제
- z: 셀 삭제 취소
- m: 현재 셀 코드에서 마크다운으로 바꾸기
- y: 현재 셀 마크다운에서 코드로 바기
- c: 셀 복사
- v: 셀 붙여넣기
- x: 셀 잘라내기

마크다운(markdown) 주요 문법

셀 실행하면 적용됨

- # : 제목(#가 많아질수록 작아짐)
- * : 점 말머리 기호 '•'
  - tab* : 말머리 기호 안의 말머리 기호 '▪'
- 숫자. : 숫자 말머리 기호
- *~~~* : 기울이기
- **~~~** : 두껍게

- 스페이스바 두 번 + 엔터 : 엔터
- ₩~: 마크다운 문법 무시
- [Google](http://www.google.co.kr): 링크 삽입
  예시: Google

파이썬은 인터프리터 언어인 덕분에 Jupyter와 잘 어울림.

인터프리터란?

C++ 등의 언어는 코드를 써 놓고 실행하면 일단 코드를 전부 다 한 번에 CPU가 이해할 수 있는 언어로 바꿈. 이후 코드를 돌림.
그러나 파이썬은 코드를 한 줄 한 줄씩 번역하고 돌림. 한 줄 번역하고 돌리고, 다음 줄 번역하고 돌리고, ...
이러한 방식 또는 이걸 가능하게 해 주는 프로그램을 '인터프리터'라고 함.
참고로 C++ 처럼 일단 전부 번역부터 하는 경우 '컴파일러'라고 함.

조코딩 점프 투 파이썬 2강: 파이썬 프로그래밍의 기초, 자료형 (1)

조코딩 02(1). 숫자형, 문자열, 리스트

✅ 자료형

- int: 정수(integer)
- float: 실수, 파이썬에서 실수는 부동소수점(floating point) 방식으로 저장함.
- str: 문자열(string)
- list: 리스트
- tuple: 튜플(tuple)
- dict: 딕셔너리(dictionary)
- set: 집합
- bool: 불(boolean)

a: 변수 또는 자료
파이썬에서 변수란, 자료가 들어있는 공간이다.

- type(a): a 자료형 파악
- int(a), float(a), str(a), list(a): a 특정 자료형으로 변환. int로 만드는 게 불가능하면 에러 남.
- print(a): a 출력

In [ ]:

Out[ ]: 1

✅ 숫자형(int, float)

숫자형 연산

- a + b: 더하기
- a - b: 빼기
- a * b: 곱하기
- a ** b: 거듭제곱(power)
- a / b: 나누기
- a // b: 몫
- a % b: 나머지

In [ ]:

✅ 문자열(str)

문자열 표현법

'asdf'

"asdf"

'''
asdf
asdf
asdf
'''

"""
asdf
asdf
asdf
"""

In [ ]:

문자열 연산

- a + b: a, b 문자열 더하기
- a * n: a 문자열 n번 반복하기

In [ ]:
```
a = 'apple'
b = 'banana'
a * 3
```

Out[ ]:
```
'applebanana'
```

문자열 인덱싱

주의: 컴퓨터는 순서를 0부터 센다!!!!

- a[0]: a 문자열에 있는 0번째 문자 불러오기 (우리말: 첫 번째 문자 불러오기)

- a[1]: a 문자열에 있는 1번째 문자 불러오기 (우리말: 두 번째 문자 불러오기)
- a[-1]: a 문자열에 있는 마지막 문자 불러오기
- a[-2]: a 문자열에 있는 마지막에서 두 번째 문자 불러오기

문자열 슬라이싱

주의: 컴퓨터는 처음은 포함하고, 끝은 포함 안 한다!!!!

- a[0:1]: a 문자열에 있는 문자 0번째**부터** 1번째 **전까지** 불러오기
- a[0:2]: a 문자열에 있는 문자 0~1번째 것 불러오기
- a[:2]: 처음**부터** 2번째 **전까지** 불러오기
- a[1:]: 1번째부터 마지막까지 불러오기
- a[1:-1]: 1번째부터 마지막 **전까지** 불러오기
- a[::2]: 문자열의 모든 문자를 2칸 간격으로 불러오기
- a[::-1]: 문자열의 모든 문자를 거꾸로 불러오기

In [22]:
```
a = 'abcde'
a[0:-1]
```

Out[22]:
```
'abcd'
```

문자열 포매팅

- s: 문자열(str)
- %d: 10진수(decimal) = int
- %f: 실수(float, 부동소수점으로 표현한 실수)
- '~{}~'.format(k) {}위치에 k 집어넣기 -> 이때 '{' 또는 '}'를 문자열에 입력하고 싶으면 두 번 쓴다.'{{'
- f'~{a}~': 문자열 안에 변수 문자열로 바꿔서 집어넣기 -> 이것도 마찬가지

In [ ]:
```

```

문자열 관련 각종 함수

- a.len(): 문자열 a의 길이 세기
- a.count(b): 문자열 a에 문자열 b가 몇 개 있는지 세기
- a.find(b): 문자열 a에 문자열 b가 있으면, b가 처음 나온 위치(인덱스) 알려 줌. 없으면 -1 반환
- a.index(b): a.find(b)와 기능은 같음. 단 b가 없으면 에러 냄
- a.strip(): 양쪽 공백 제거
- a.rstrip(): 오른쪽 공백 제거
- a.lstrip(): 왼쪽 공백 제거
- a.replace(b, c): a에서 b를 c로 바꿈
- a.split(): 공백 기준으로 문자열을 나눠 리스트로 변환
- a.split(k): k 기준으로 문자열을 나눠 리스트로 변환

In [ ]:
```

```

리스트(list)

인덱싱, 슬라이싱 방법은 str과 비슷함.

리스트는 리스트 안의 원소를 직접 바꾸거나 삭제할 수 있음.

- a[0] = b : 리스트 a 안의 첫 자료를 b로 바꿈.
- del a[0] : 리스트 a 안의 첫 자료를 삭제함.
- a.pop(): 리스트 a 안의 마지막 자료를 반환하고 삭제함.
- a.remove(b): 리스트 안에 b와 일치하는 자료가 있으면 그 첫 번째 걸 삭제함.

리스트 안의 원소를 추후에 추가할 수도 있음.

- a.append(b): 리스트 a의 끝에 b를 추가함.
- a.insert(b, c): 리스트 a의 b 위치에 c를 추가함.

In [30]:
```
asdf = ['a', 'b', 'asdf', 1, 34289127489134]
asdf[1] = 'k'
asdf
```

Out[30]:
```
['a', 'k', 'asdf', 1, 34289127489134]
```

리스트 관련 각종 함수

- len(a): 리스트가 가지고 있는 자료 개수 반환
- a.sort(): 리스트 순서 오름차순으로 정렬
- a.sort(reverse=True): 리스트 순서 내림차순으로 정렬

In [ ]:
```
```