Python Cheat Sheet

by 코드잇

1 기본

콘솔창 출력

```
print("Hello World")
print(12)
```

코멘트 작성

```
# 작성하고 싶은 코멘트
```

변수: 값에 붙이는 이름표

- 변수명 = 값

```
burger_price = 4990
user_name = “Emily”
```

함수: 특정 코드를 모아 이름을 붙인 것

- 파라미터: 함수에 넘겨주는 값
- return문: 함수의 반환값, 함수 종료시킴

```
def sum(n1, n2):          ← 함수 정의
    res = n1 + n2
    return res

num = sum(2, 4)           ← 함수 호출
print(num)                → 6 출력
```

- 옵셔널 파라미터: 파라미터에 기본값을 설정하여, 함수 호출 시 파라미터에 값을 꼭 넘겨주지 않아도 됨. 꼭 마지막에 써야 함

```
def myself(name, nationality="한국"):
    print(name)
    print(nationality)

myself("코드잇")  → 코드잇 한국 출력
```

2 자료형 (Data Type)

숫자형

- 정수(Integer) `-1, -2, 0, 1, 2`
- 소수(Floating Point) `3.14, -7.3, 2.0`
- 숫자형 연산
  - 덧셈 `7 + 4`
  - 뺄셈 `7 - 4`
  - 곱셈 `7 * 4`
  - 나눗셈 `7 / 4`
  - 몫 `7 // 4`
  - 나머지 `7 % 4`
  - 거듭제곱 `7 ** 4`
  - 축약연산자: +=, -=, *=, /=
    ex) `x = x + 3` ⟶ `x += 3`
- 반올림 함수
  - round(숫자, n): 특정 숫자를, 소수점 n번째까지 표기하면서 반올림
    ex) `round(5.1479, 2)` ⟶ `5.15`

문자열(String)

- 기본 형태 `"Hello" 'Hello'`
  `"I'm \"excited\" to learn Python!"` ← 문자열 안에 따옴표를 포함하고 싶을 때
- 문자열 연결 `"Hello" + "World"` → `"HelloWorld"`
  `"2" + "5"` → `"25"`
  `"Hello" * 3` → `"HelloHelloHello"`
- 문자열 포매팅
  - format( ) 메소드

```
"오늘은 {}년 {}월 {}일입니다".format(2024, 2, 23) → 오늘은 2024년 2월 24일입니다
"저는 {1}, {0}, {2}를 좋아합니다".format("아이브", "에스파", "르세라핌")
                                  → 저는 에스파, 아이브, 르세라핌을 좋아합니다
"{} 나누기 {}은 {:.3f}입니다".format(17, 4, 17/4) → 17 나누기 4는 4.250입니다
                                  → 소수의 형태로 소숫점 셋째자리까지 표기하라는 뜻
```

  - f-string 방식

```
name = "장원영"
age = 21

print(f"제 이름은 {name}이고 {age}살입니다") → 제 이름은 장원영이고 21살입니다 출력
```

- strip( ) 메소드: 문자열 양 끝의 화이트 스페이스(ex. " ", "\t", "\n") 제거

```
"    \t    오늘 날  씨 되게 좋다!   \n     ".strip() → 오늘 날  씨 되게 좋다!
```

- split( ) 메소드: 특정 문자를 기준으로 문자열을 나눠서 리스트 생성

```
"1.2.3.4.5.6".split(".")   마침표를 기준으로 문자열 나누기 → ["1", "2", "3", "4", "5", "6"]
"밥은 먹고 다니냐".split()  공백을 기준으로 문자열 나누기 → ["밥은", "먹고", "다니냐"]
```

형 변환

- 정수로 변환 `int(3.8)` → 3
  `int("2")` → 2
- 소수로 변환 `float(3)` → 3.0
- 문자열로 변환 `str(3)` → "3"

불린형(Boolean)

- True, False
- 논리 연산자
  - `A and B` A, B 둘 다 True여야 True
  - `A or B` A, B 둘 중에 하나만 True여도 True
  - `not A` 반대의 불린값
- 비교 연산자 `<, >, <=, >=, ==, !=`

ex) `7 != 7 or not(4 < 3 and 12 > 10)` → True

3 리스트와 사전

리스트(List)

- 변수에 리스트 저장하기 `numbers = [2, 3, 5, 7, 11, 13]`
  `names = ["카리나", "윈터", "닝닝", "지젤"]`

인덱스	0	1	2	3
마이너스 인덱스	-4	-3	-2	-1

- 인덱싱(Indexing): 인덱스를 통해 리스트 요소를 받아오는 것

  `numbers[1]` ⟶ `3` `names[-3]` ⟶ `"윈터"`

- 슬라이싱(Slicing): 리스트의 일부를 통째로 잘라서 받아오는 것

  `numbers[0:3]` 인덱스 0부터 인덱스 2까지 ⟶ `[2, 3, 5]`
  `numbers[3:]` 인덱스 3부터 끝까지 ⟶ `[7, 11, 13]`
  `numbers[:3]` 처음부터 인덱스 2까지 ⟶ `[2, 3, 5]`

- 리스트에 있는 요소 변경

  `numbers[0] = 15` numbers의 인덱스 0의 요소가 15로 변경됨

- 리스트 내장 함수 *아래 항목들은 각각 numbers = [2, 3, 5, 7, 11, 13]을 기준으로 실행

  `len(numbers)` 리스트 요소 개수 ⟶ `6`
  `numbers.append(15)` 리스트 마지막에 요소 추가 ⟶ `[2, 3, 5, 7, 11, 13, 15]`
  `numbers.insert(2, 9)` 특정 인덱스에 요소 삽입 ⟶ `[2, 3, 9, 5, 7, 11, 13]`
  `del numbers[2]` 인덱스를 통해 리스트 요소 삭제 ⟶ `[2, 3, 7, 11, 13]`
  `numbers.remove(5)` 값을 통해 리스트 요소 삭제(첫 번째로 5의 값을 갖고 있는 요소 삭제) ⟶ `[2, 3, 7, 11, 13]`

- 리스트 정렬 *아래 항목들은 각각 numbers = [4, 6, 10, 8, 2]을 기준으로 실행

  `sorted(numbers)` 오름차순으로 정렬된 새로운 리스트 반환 ⟶ `[2, 4, 6, 8 ,10]`
  `sorted(numbers, reverse=True)` 내림차순으로 정렬된 새로운 리스트 반환 ⟶ `[10, 8, 6, 4, 2]`
  `numbers.sort( )` 리스트를 오름차순으로 정렬 ⟶ `[2, 4, 6, 8 ,10]`
  `numbers.sort(reverse=True)` 리스트를 내림차순으로 정렬 ⟶ `[10, 8, 6, 4, 2]`

- 리스트에서 값의 존재 확인하기

  `numbers = [2, 4, 6, 8, 10]`

  `4 in numbers` ⟶ `True`
  `7 not in numbers` ⟶ `True`

- 리스트 복사

  `x = [2, 3, 5, 7, 11]`
  `y = list(x)` ⟶ 변수 y에도 [2, 3, 5, 7, 11]이 저장됨

- 리스트와 문자열

  `alphabet_string = "ABCDEFG"`

  `alphabet_string[0]` 문자열 인덱싱 ⟶ `"A"`
  `alphabet_string[0:5]` 문자열 슬라이싱 ⟶ `"ABCDE"`
  `len(alphabet_string)` 문자열 길이 ⟶ `7`
  `alphabet_string[0] = "Z"` (X) ⟵ 리스트와 달리, 문자열은 수정 불가능

사전(Dictionary)

- 키-값 쌍 (key - value pair)의 형태

```
my_dictionary = { 5: 25, 2: 4, 3: 9 }
my_family = {"엄마": "김자옥", "아빠": "이석진", "아들": "이동민"}

my_dictionary[2]      키를 통해 값 받아오기 ⟶ 4
my_dictionary[9] = 81 새로운 쌍 추가하기 ⟶ {5: 25, 2: 4, 3: 9, 9: 81}
my_family.values( )   값 목록 확인하기 ⟶ dict_values(['김자옥', '이석진', '이동민'])
my_family.keys( )     키 목록 확인하기 ⟶ dict_keys(['엄마', '아빠', '아들'])
```

4 제어문

if 조건문

- if 조건 부분: 수행 부분 — 조건 부분이 True이면 수행 부분 실행

```
if total >= 90:      ⟶ total이 90 이상이 아니면 다음 elif 구문으로 넘어감
    print("A")
elif total >=80:
    print("B")
elif total >=70:
    print("C")
else:
    print("F")
```

while 반복문

- while 조건 부분: 수행 부분 — 조건 부분이 True일 동안 수행 부분 실행

```
i = 1
while i <= 3:
    print("나는 잘생겼다!")  ⟶ 세 번 출력
    i += 1
```

for 반복문

- for 변수 in 리스트(또는 문자열): 수행 부분 — 리스트의 요소가 차례대로 변수에 대입되며 수행 부분 실행

```
my_list = [2, 3, 5]
for number in my_list:
    print(number)  ⟶ my_list의 요소가 순서대로 출력
```

- range 함수

```
range(stop)              0부터 stop-1까지의 범위의 숫자
range(start, stop)       start부터 stop-1까지의 범위의 숫자
range(start, stop, step) start부터 stop-1까지 step 간격의 숫자
```

x)
```
for i in range(3, 17, 3):
    print(i)  ⟶ 3 6 9 12 15 출력
```

break와 continue

- break: 반복문 종료

```
i = 100
while True:
    if i % 23 == 0:
        break  ⟶ i가 23의 배수면 반복문 종료
    i = i + 1
print(i)  ⟶ 115 출력
```

- continue: 다음 조건 부분으로 넘어감

```
for i in range(1, 11):
    if i % 2 == 1:
        continue ⟶ i가 홀수면 다음 조건 부분으로 넘어감
    print(i)  ⟶ 2 4 6 8 10 출력
```

5 사용자 입력

input 함수

- input("문자열") 해당 문자열은 사용자에게 입력을 받기 전 콘솔에 출력됨
- input 함수를 통해 사용자로부터 받은 값을 변수에 저장해 사용
- input 함수를 통해 사용자로부터 받은 값은 문자열이므로,
- 숫자 데이터를 받아 사용하고 싶을 경우 형변환 필요

```
number = int(input("숫자를 입력하세요: "))
print(number + 2)
```

```
숫자를 입력하세요: 5
7
```

사용자가 5를 입력한 후 Enter 누르면 7 출력

6 파일 읽고 쓰기

파일 읽기

- open(파일 경로, "r") read

```
with open("text_file.txt", "r") as f:
    for line in f:
        print(line)
```

→ 읽어들인 파일을 f라는 변수에 저장

→ 파일의 내용이 한 줄씩 출력

파일 쓰기

- open(파일 경로, "w") write

```
with open("text_file.txt", "w") as f:
    f.write("Hello World!\n")
    f.write("Nice to Meet You!")
```

→ 내용을 쓸 파일을 f라는 변수에 저장

→ Hello World!를 쓰고 줄바꿈

- text_file.txt가 없으면, text_file.txt를 생성 후 내용 입력
- text_file.txt가 이미 있으면, 해당 파일에 내용 덮어씀

- open(파일 경로, "a") append

```
with open("text_file.txt", "a") as f:
    f.write("Hello World!\n")
    f.write("Nice to Meet You!")
```

→ 내용을 쓸 파일을 f라는 변수에 저장

→ text_file.txt의 끝에 내용 추가

- text_file.txt가 없으면, text_file.txt를 생성 후 내용 입력
- text_file.txt가 이미 있으면, 해당 파일 마지막에 내용 추가됨