Python - 문법

Index

Arithmetic

사칙연산

구분	예제	결과
덧셈	1 + 2	3
뺄셈	1 - 2	-1
곱셈	5 * 2	10
나눗셈	5 / 2	2.5
나눗셈 (몫)	5 // 2	2
나눗셈 (나머지)	9 % 2	1
제곱	2 ** 3	8
괄호 (우선순위)	(2 + 3) * 2	10

if - else

선택문

if – else

- 여러 프로그래밍 언어에서 사용하는 가장 기본적인 조건 판단문
- 다중 조건 시 if, elif, else로 구분하여 사용

```
If 조건식:
_ _ _ _ 코드
```

공백 or 탭
공식 4칸

```
>> score = 75
>> if score > 90:
>>     print('참 잘했어요')
>> elif score > 70:
>>     print('잘했어요')
>> else:
>>     print('더 노력해봐요')
```

결과 : 잘했어요

match / case

- python 3.10 버전에서 추가됨.
- 특정 변수값의 여러 사례(case)를 미리 정의해놓고 분기처리
- 구조적 패턴 매칭

```
match 변수:
_ _ _ _ case 1:
_ _ _ _ _ _ _ _ _ 코드
```

```
>> command = 'load'
>> match command:
>>     case 'quit':
>>         print('프로그램 종료')
>>     case 'load':
>>         print('파일 로드')
>>     case _:
>>         print('명령 정의되지 않음')
```

결과 : 파일 로드

for, while

반복문

for
- 특정 코드를 반복할 때 사용
- 주로 한정된 범위만큼 반복할 필요가 있을 경우
- enumerate를 사용해 index에 접근가능

for 변수 in iterable 변수:
_ _ _ _ 코드

공백 or 탭
공식 4칸

```
>> for i in range(10)
>>     print("현재값: ", i)

>> for index, value in enumerate(range(10))
>>     print("인덱스", index, "값", value)
```

while
- 마찬가지로 반복 시 사용
- 주로 반복횟수가 정해져 있지 않을 경우, 조건에 따라 반복

while 조건식:
_ _ _ _ 코드

공백 or 탭
공식 4칸

```
>> i = 0
>> while i < 10:
>>     print("현재 값: ", i)
>>     i += 1
```

break, continue
- break : 해당 반복문을 중단
- continue : 현 위치에서 반복문의 처음으로 돌아가 다음반복 수행

etc

기타

내장함수

- 최대값, 최소값 max(), min()

```
>> a = [1,7,9,15]
>> max(a)
결과 : 15
>> min(a)
결과 : 1
```

- 합계 sum()

```
>> a = [1,7,9,15]
>> sum(a)
결과 : 32
```

- 문자열 분리 split()

```
>> number_str = '13 22 38 44'
>> number_str.split()
결과 : ['13', '22', '38', '44']
```

- 문자열 합치기 join()

```
>> fruits = ['apple', 'banana', 'watermelon']
>> '_'.join(fruits)
결과 : 'apple_banana_watermelon'
```

- 리스트 합치기 zip()

```
>> keys = ["name", "age", "address"]
>> user = ["jason", "29", "seoul"]
>> info = dict(zip(keys, user))
>> print(info)
결과 : {'name': 'jason', 'age': 29, 'address': 'seoul}
```

lambda

- lambda 변수 : 식

```
>> (lambda x: x+10)(5)
```
결과 : 15

map

- map(함수, 리스트)
- 리스트 내 요소들을 하나씩 함수에 넣어,
  계산된 결과를 새 리스트에 담아 반환

```
>> list(map(lambda x: x**2, range(5)))
```
결과 : [0,1,4,9,16]

filter

- filter(함수, 리스트)
- 리스트 내 요소들을 하나씩 함수에 넣어,
  True인 경우만 새 리스트에 담아 반환

```
>> list(filter(lambda x: x < 5, range(10)))
```
결과 : [0,1,2,3,4]

List comprehension
- [식 for 변수 in 리스트]
- for문, list comprehension 속도 비교

```
>> a = [i for i in range(10)]
>> a
```
결과 : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

- [식 for 변수 in 리스트 if 조건]
- 조건식에서 True인 값만 식을 실행

```
>> a = [i for i in range(10) if i%2 == 0]
>> a
```
결과 : [0, 2, 4, 6, 8]

- [식 for 변수 in 리스트 for 변수 in 리스트]
- 중첩해서 활용 가능

```
>> a = [i * j for i in range(2, 10) for j in range(1, 10)]
>> a
```
결과 : [2, 4, 6, 8, 10, ···, 81]