# [스파르타코딩클럽] 파이썬 문법 뽀개기

**매 주차 강의자료 시작에 PDF파일을 올려두었어요!**

▼ PDF 파일

**[수업 목표]**

1. 파이썬 기초 문법을 익히고,
2. 여러가지 퀴즈로 빠르게 숙달한다.
3. 파이썬 심화 문법들을 다뤄봅니다.
4. 머릿속 어딘가에 담아두고, 나중에 생각나면 써먹기!

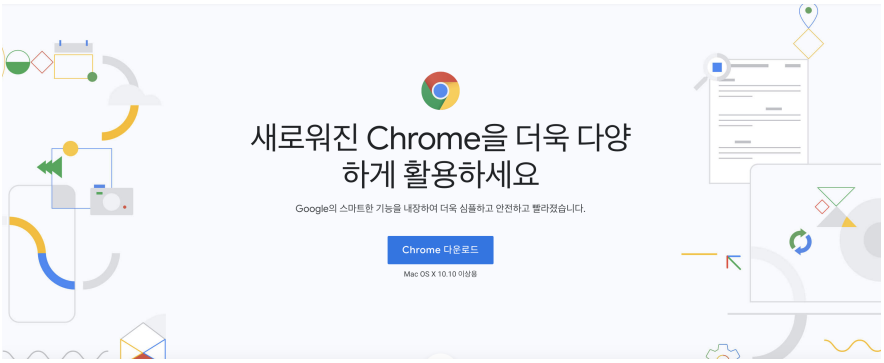
**[목차]**



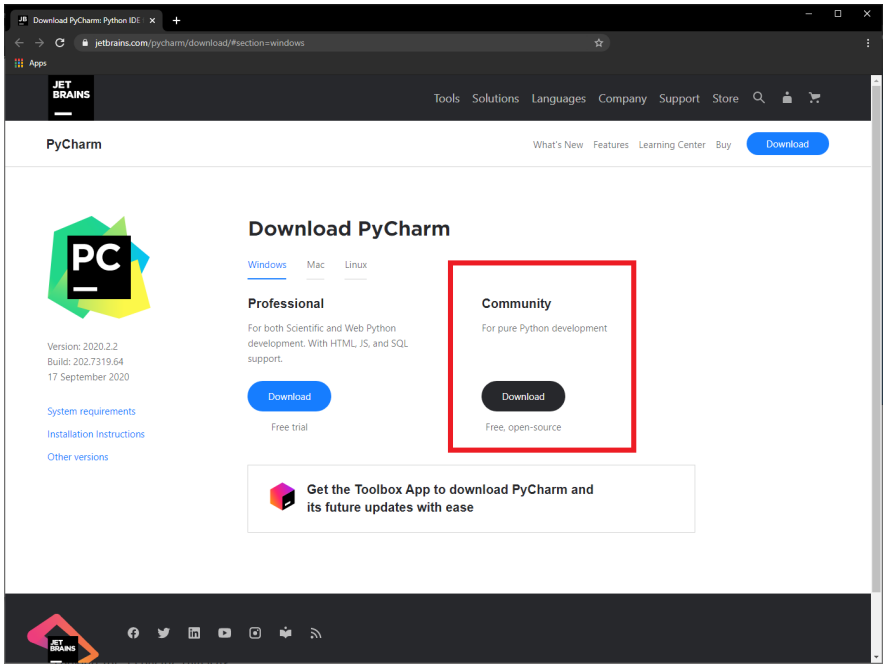
모든 토글을 열고 닫는 단축키
Windows : `Ctrl` + `alt` + `t`
Mac : `⌘` + `⌥` + `t`

# 1-0. 필수 프로그램 설치

▼ **Chrome : (다운로드 링크)**

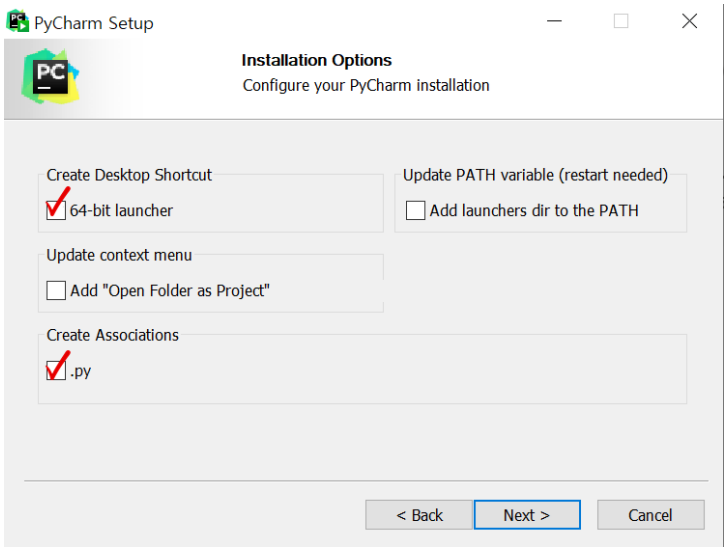



▼ **PyCharm Community : (다운로드 링크)**




위 그림과 같이 Community 버전을 다운로드 받은 후 설치까지 마무리해주세요.

- **"설치"만** 해두세요!
  설치 중 Installation Options는 아래와 같이 체크해주세요.

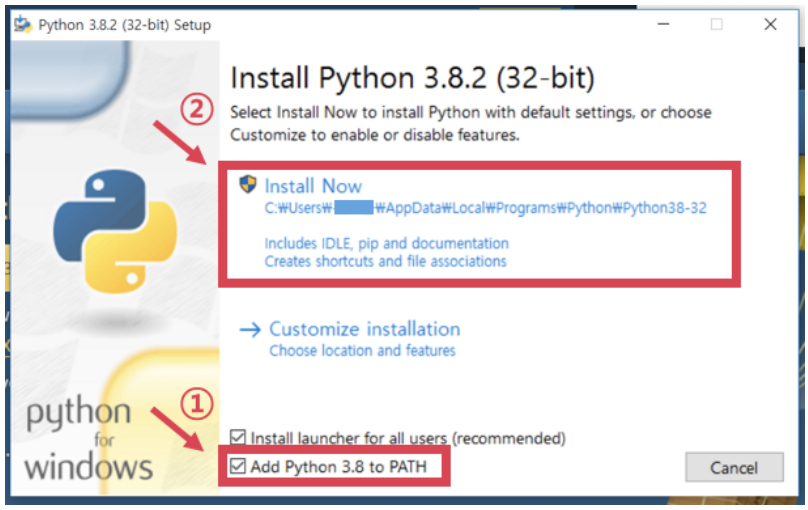


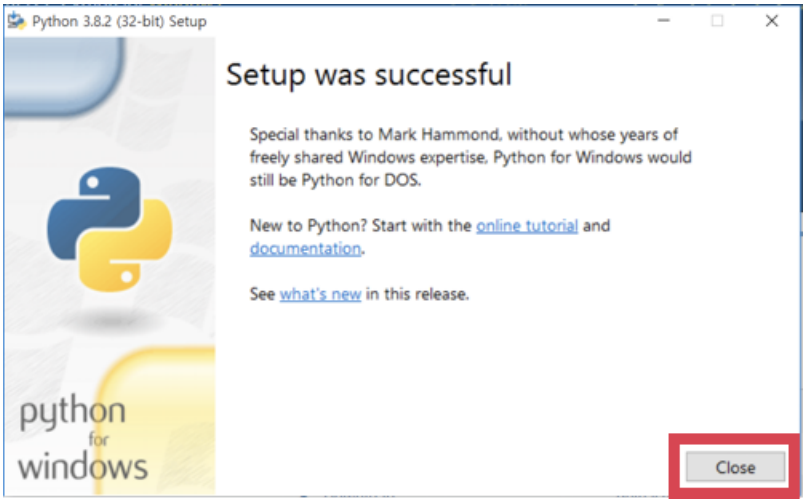

▼ **Python (다운로드 링크)**

- **윈도우**

1. 다운로드된 파이썬 파일을 열어 **설치**합니다.

> **주의) Add Python 3.8 to PATH 에 체크해야 합니다!!**



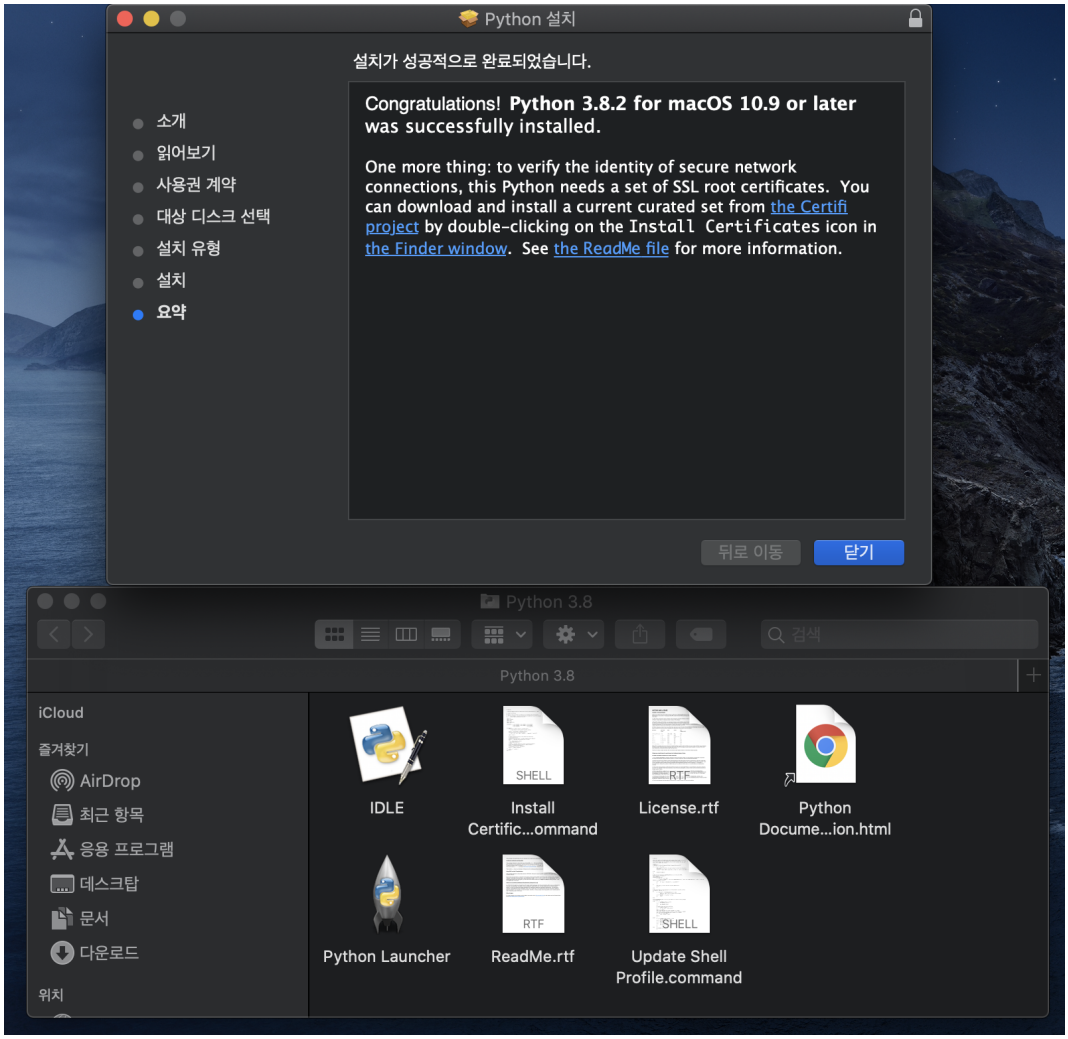



- **맥**

> Mac에는 기본적으로 파이썬 2.x 버전이 설치되어 있지만, 버전3으로 업데이트되면서 많은 변화가 있었습니다. 본 강의에서는 3.x 버전의 파이썬이 필요하니 아래 가이드에 따라 꼭 설치해주세요.

1. **버튼을 눌러 다운로드** 해주세요. (3.8.x 이상의 최신 버전을 설치!)


2. 다운로드된 파이썬 파일을 열어 **설치**합니다. 아래와 같은 폴더가 나타나면 설치가 완료된 것입니다.




# 1-1. 파이썬 문법 뽀개기 - 기초

▼ 수업의 목적

> 👉 본 강의의 목적은 파이썬 문법을 보다 체계적으로 이해할 수 있도록, 빠르게 이론을 습득하고 퀴즈로 숙달하게 만드는 데 있습니다.
>
> 또, 향후 마주칠 난이도 있는 문법들을 눈으로, 손으로 먼저 경험해보겠습니다. 🔥

- 그러나 `기본만 익히고, 찾아가며 하는 것이 코딩` 이라는 스파르타의 철학에는 변함이 없습니다. 본 강의로 문법을 완전히 숙달하고 코딩을 시작하겠다는 마음가짐 보다는, 어디까지나 '보조자료'로서의 활용에 그 목적이 있음을 인지해주시길 바랍니다!

▼ 배우는 순서

1. 기초적인 문법을 차근차근
2. 퀴즈를 풀며 익숙해지기

3. 약간 심화된 문법/쓰임새들

## 1-2. 파이썬 시작하기

▼ 1) 파이썬을 설치한다는 것의 의미

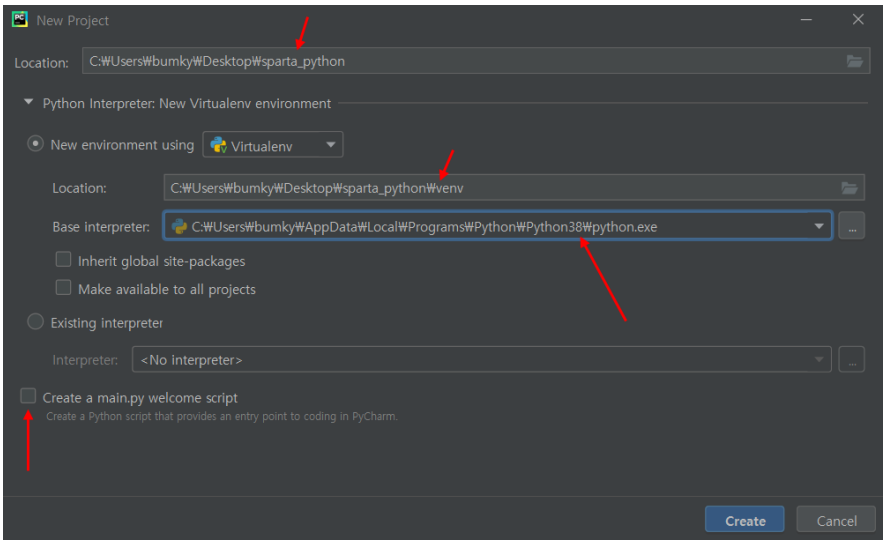
> 👉 **파이썬을 설치한다?**
>
> → 일종의 번역팩을 설치한다고 생각하면 됩니다. 컴퓨터는 101010001 과 같은 언어만 알아듣는다고 했지요? 파이썬 문법으로 된 것을 101010001로 변환해줄 수 있도록, 번역 패키지를 설치하는 것입니다.

> 📖 **토막 상식**
>
> 네덜란드의 프로그래머 Guido van Rossum이 '읽기 쉬운 코드'에 중점을 두어 개발한 프로그래밍 언어입니다.

- '읽기 쉽게' 만든 언어이기 때문에, 문법이 쉬워 초보자가 배우기 좋은 언어입니다.
- 무료/오픈소스로 운영되고 커뮤니티가 크기 때문에 자연어처리, 머신러닝 등 직접 구현하기 어려운 기능들도 다른 개발자의 라이브러리를 이용해 쉽게 코드를 작성할 수 있습니다.

▼ 2) 파이참에서 새 프로젝트 만들기

1. 바탕화면에 `sparta_python` 폴더를 만들어줍니다.
2. 파이참 실행!
3. Create New Project 를 선택해주세요.
4. 아래와 같은 화면으로 만들어주기!



- 주의! base interpreter - 파이썬 버전이 3.8로 되어있는지 확인해주세요.
- 주의! Location - 끝이 `\venv` 로 끝나는지 확인해주세요.
- 주의! main.py 자동 생성은 체크 해제!

5. `Create`를 클릭해서 새 프로젝트를 만들어줍니다.

▼ 3) 첫 파이썬 파일 실행하기!

- sparta 폴더 안에 hello.py라는 이름으로 파일을 하나 만들어줍니다.
- 다음 내용을 붙여넣어주세요.

```
print('hello sparta')
```

- 마우스 우클릭 → Run → hello.py 실행!

▼ 4) 파이썬 문법을 시작하기에 앞서..

- 파이썬은 매우 직관적인 언어이고, 할 수 있는 것도 많습니다. 그런데, 개발자들도 모든 문법을 기억하기란 쉽지 않습니다. 오늘 배우는 것 외에 필요한 것들은 구글링해서 찾아보면 됩니다!
- 수업을 다 들어도 한번에 모든 게 외워지지 않는 게 당연하니, 걱정하지 마세요!
  `"그런 개념이 있었지! 강의자료를 다시 뒤적여볼까?" 정도면 아주 충분합니다.`
  (개발자들도 오랜만에 안 쓰던 언어로 코딩하면 마찬가지예요!)

## 1-3. 변수 선언과 자료형

▼ 5) 변수 선언

- 파이썬에서 새 변수를 만들 때는 `변수이름 = 값`의 형태로 써요. `a = b`와 `b = a`는 다르다는 사실! 출력할 때는 위에서 본 것처럼 `print()`를 씁니다 🙂

```
a = 3         # 3을 a에 넣는다.
print(a)
b = a         # a에 들어 있는 값인 3을 b에 넣는다.
print(b)
a = 5         # a에 5라는 새로운 값을 넣는다.
print(a, b)   # 5 3
```

▼ 6) 숫자형 자료형

> 📖 자료형이란 프로그래밍을 할 때 쓰이는 숫자, 문자열 등 자료 형태로 사용하는 모든 것을 뜻합니다! 파이썬에서 어떤 '값'을 쓰는지 알아야 코딩을 할 수 있겠죠?

- 다양한 형태의 숫자를 쓸 수 있습니다.

```
a = 5
b = 4.8
```

- 숫자 간에는 사칙연산이 가능합니다. 몫과 나머지도 구할 수 있어요.

```
a = 7
b = 2

a+b   # 9
a-b   # 5
a*b   # 14
a/b   # 3.5

a+3*b         # 13 (여러 연산을 한 줄에 할 경우 사칙연산의 순서대로!)
(a+3)*b       # 20 (소괄호를 이용해서 먼저 계산할 부분을 표시해줄 수 있어요!)
```

- 변수에 저장된 값에 연산을 한 후 다시 그 값을 같은 변수에 저장할 수 있습니다.

```
a = 5
a = a + 3 # 5에 3을 더한 값을 다시 a에 저장
print(a)  # 8

a += 3    # 줄여 쓸 수도 있다. 같은 의미!
```

- 예 - 배우지 않은 나눗셈의 나머지를 알아보려면?

> 👉 모르는 것은 무조건 검색! 검색!
> 구글에 `파이썬 나머지 구하기`, 또는 `파이썬 거듭제곱`이라고 쳐보기!

```
a//b  # 3 (몫)
a%b   # 1 (나머지)
a**b  # 49 (거듭제곱)
```

▼ 7) Bool 자료형

- 특별한 자료형으로 참/거짓을 나타내는 불(Boolean) 자료형이 있습니다.

```
x = True   # 참
y = False  # 거짓

# 소문자로 쓰면 자료형으로 인식하지 않고 변수명이라 생각해 에러가 납니다~
z = true   # name 'true' is not defined

True = 1   # True/False는 변수명으로 쓸 수 없어요!
```

- 보통 아래처럼 '비교연산자'의 결과로 나타내기 위해 쓰여요.

```
4 > 2      # True  크다
5 < 1      # False 작다
6 >= 5     # True  크거나 같다
4 <= 4     # True  작거나 같다
3 == 5     # False 같다
4 != 7     # True  같지 않다
```

- 불 자료형에는 논리연산자를 이용할 수 있습니다.

```
a = 4 > 2  # True
not a      # False    NOT 연산자로 참을 거짓으로, 거짓을 참으로 바꿔준다.

a and b    # False    AND 연산자로 모두 참이어야 참을 반환한다.
a or b     # True     OR 연산자로 둘 중 하나만 참이면 참이다.
```

▼ 8) ✍️ Q. 숫자들의 평균 구하기

```
a = 24
b = 16
c = 26
```

▼ A. 풀이

```
print((a+b+c)/3)
```

# 1-4. 문자열 다루기

▼ 9) 문자열

▼ 문자열 기초

- 파이썬에서는 '글'도 데이터로 사용할 수 있는데요, 이것을 '문자열'이라고 합니다.

```
# 작은 따옴표 또는 큰 따옴표. 둘 다 같아요!

a = "aa"
b = 'aa'
```

- 따옴표로 감싸지 않으면 변수이름을 뜻하기 때문에 꼭 구분해서 써야합니다!

```
a = 1
b = "a"
c = a
print(a, b, c) # 1 "a" 1
```

- 이렇게 다양한 방법으로 만들 수 있기 때문에, 문자열에 따옴표를 포함해야할 때 쓰면 편해요 🙂

```
print("I'm happy :)")
```

▼ 문자열 연산

- 문자열 간의 더하기는 두 문자열을 이어붙인 문자열을 반환합니다.

```
first_name = "Harry"
last_name = "Potter"

first_name + last_name  # HarryPotter
first_name + " " + last_name  # Harry Potter

a = "3"
b = "5"
a + b    # 35
```

- 문자열과 정수를 더하면 에러!

```
a = "3"
a + 5    # 문자열과 숫자형은 더할 수 없어서 에러!
```

- 문자열의 길이는 `len()` 함수를 써서 구할 수 있습니다!

```
print(len("abcde"))           # 5
print(len("Hello, Sparta!"))  # 14
print(len("안녕하세요."))      # 6
```

▼ 그 외에도 다양한 기능을 쓸 수 있어요!

> 💡 이렇게 특정 자료형 뒤에 `.`을 붙이고 쓰는 내장 함수들을 '메소드(method)'라고 합니다! 다음주에 '클래스'를 배울 때 다시 살펴볼게요~

- 모든 알파벳을 대문자/소문자로 바꾸기

```
sentence = 'Python is FUN!'

sentence.upper()  # PYTHON IS FUN!
sentence.lower()  # python is fun!
```

- 특정 문자를 기준으로 문자열을 나누기

```
# 이메일 주소에서 도메인 'gmail'만 추출하기
myemail = 'test@gmail.com'

result = myemail.split('@') # ['test','gmail.com'] (뒤에 배울 '리스트'라는 자료형이에요 :))

result[0] # test (리스트의 첫번째 요소)
result[1] # gmail.com (리스트의 두 번째 요소

result2 = result[1].split('.') # ['gmail','com']

result2[0] # gmail -> 우리가 알고 싶었던 것
result2[1] # com

# 한 줄로 한 번에!
myemail.split('@')[1].split('.')[0]
```

- 특정 문자를 다른 문자로 바꾸기

```
txt = '서울시-마포구-망원동'
print(txt.replace('-', '>')) # '서울시>마포구>망원동'
```

▼ 인덱싱과 슬라이싱

- 문자열은 '문자들의 모임'이기 때문에 그 일부를 따로 떼어 부르는 방법이 있습니다. 한 글자 씩 부를 때는 몇 번째인지 '인덱스'를 넣어 불러서 인덱싱이라고 합니다.

```
f="abcdefghijklmnopqrstuvwxyz"
f[1]   # b 파이썬은 숫자를 0부터 셉니다
```

- 문자열의 일부를 잘라낼 때는 슬라이싱이라고 해요.

```
f[4:15]  # efghijklmno           f[4]부터 f[15] 전까지, 총 15-4=11개!

f[8:]    # ijklmnopqrstuvwxyz    f[8]부터 끝까지, 앞의 8개 빼고!
f[:7]    # abcdefg               시작부터 f[7] 전까지, 앞의 7개!

f[:]     # abcdefghijklmnopqrstuvwxyz  처음부터 끝까지
```

- 특정 문자열로 자르고 싶을 때! `split('문자열')`을 활용합니다.

```
myemail = 'abc@sparta.co'

domain = myemail.split('@')[1].split('.')[0]
print(domain)
```

▼ 10) ✍️ Q. 문자열의 앞의 반만 출력하기

`"sparta"`의 앞의 3글자인 `"spa"`만 출력해봅시다.

▼ A. 풀이

```
text = "sparta"
print(text[:3])
```

▼ 11) ✍️ Q. 전화번호의 지역번호 출력하기

```
phone = "02-123-1234"
```

힌트: `.split()`을 써보세요!

▼ A. 풀이

```
phone = "02-123-1234"
print(phone.split("-")[0])
```

# 1-5. 리스트와 딕셔너리

▼ 12) 리스트 (list)

▼ 리스트 기초

- 순서가 있는, 다른 자료형들의 모임!

```
a = [1, 5, 2]
b = [3, "a", 6, 1]
c = []
d = list()
e = [1, 2, 4, [2, 3, 4]]
```

- 리스트의 길이도 `len()` 함수를 사용해서 잴 수 있어요.

```
a = [1, 5, 2]
print(len(a))   # 3

b = [1, 3, [2, 0], 1]
print(len(b))   # 4
```

- 순서가 있기 때문에, 문자열에서처럼 인덱싱과 슬라이싱을 사용할 수 있습니다!

```
a = [1, 3, 2, 4]
print(a[3])  # 4
print(a[1:3]) # [3, 2]
print(a[-1]) # 4 (맨 마지막 것)
```

- 리스트의 요소가 리스트라면? 중첩해서!

```
a = [1, 2, [2, 3], 0]
print(a[3])      # [2, 3]
print(a[3][0])   # 2
```

▼ 리스트의 더 많은 기능들!

- 덧붙이기

```
a = [1, 2, 3]
a.append(5)
print(a)     # [1, 2, 3, 5]

a.append([1, 2])
print(a)     # [1, 2, 3, 5, [1, 2]]


# 더하기 연산과 비교!
a += [2, 7]
print(a)     # [1, 2, 3, 5, [1, 2], 2, 7]
```

- 정렬하기

```
a = [2, 5, 3]
a.sort()
print(a)   # [2, 3, 5]
a.sort(reverse=True)
print(a)   # [5, 3, 2]
```

- 요소가 리스트 안에 있는지 알아보기

```
a = [2, 1, 4, "2", 6]
print(1 in a)      # True
print("1" in a)    # False
print(0 not in a)  # True
```

▼ 13) 딕셔너리 (dictionary)

▼ 딕셔너리 기초

- 딕셔너리는 키(key)와 밸류(value)의 쌍으로 이루어진 자료의 모임입니다. 영한사전에서 영어 단어를 검색하면 한국어 뜻이 나오는 것처럼요!

```
person = {"name":"Bob", "age": 21}
print(person["name"])
```

- 딕셔너리를 만드는 데는 여러가지 방법을 쓸 수 있습니다.

```
a = {"one":1, "two":2}

# 빈 딕셔너리 만들기
a = {}
a = dict()
```

- 딕셔너리의 요소에는 순서가 없기 때문에 인덱싱을 사용할 수 없어요.

```
person = {"name":"Bob", "age": 21}
print(person[0])   # 0이라는 key가 없으므로 KeyError 발생!
```

- 딕셔너리의 값을 업데이트하거나 새로운 쌍의 자료를 넣을 수 있습니다.

```
person = {"name":"Bob", "age": 21}

person["name"] = "Robert"
print(person)  # {'name': 'Robert', 'age': 21}

person["height"] = 174.8
print(person)  # {'name': 'Robert', 'age': 21, 'height': 174.8}
```

- 딕셔너리의 밸류로는 아무 자료형이나 쓸 수 있어요. 다른 딕셔너리를 넣을 수도 있죠!

```
person = {"name":"Alice", "age": 16, "scores": {"math": 81, "science": 92, "Korean": 84}}
print(person["scores"])            # {'math': 81, 'science': 92, 'Korean': 84}
print(person["scores"]["science"])  # 92
```

▼ 딕셔너리의 더 많은 기능들!

- 딕셔너리 안에 해당 키가 존재하는지 알고 싶을 때는 `in`을 사용합니다.

```
person = {"name":"Bob", "age": 21}

print("name" in person)       # True
print("email" in person)      # False
print("phone" not in person)  # True
```

▼ 14) 리스트와 딕셔너리의 조합

- 딕셔너리는 리스트와 함께 쓰여 자료를 정리하는 데 쓰일 수 있습니다.

```
people = [{'name': 'bob', 'age': 20}, {'name': 'carry', 'age': 38}]

# people[0]['name']의 값은? 'bob'
# people[1]['name']의 값은? 'carry'

person = {'name': 'john', 'age': 7}
people.append(person)

# people의 값은? [{'name':'bob','age':20}, {'name':'carry','age':38}, {'name':'john','age':7}]
# people[2]['name']의 값은? 'john'
```

▼ 15) ✍️ Q. 딕셔너리에서 원하는 정보를 찾아보기

▼ **[코드스니펫] - 딕셔너리 퀴즈**

```
people = [
    {'name': 'bob', 'age': 20, 'score':{'math':90,'science':70}},
    {'name': 'carry', 'age': 38, 'score':{'math':40,'science':72}},
    {'name': 'smith', 'age': 28, 'score':{'math':80,'science':90}},
    {'name': 'john', 'age': 34, 'score':{'math':75,'science':100}}
]
```


```
people = [
    {'name': 'bob', 'age': 20, 'score':{'math':90,'science':70}},
    {'name': 'carry', 'age': 38, 'score':{'math':40,'science':72}},
    {'name': 'smith', 'age': 28, 'score':{'math':80,'science':90}},
    {'name': 'john', 'age': 34, 'score':{'math':75,'science':100}}
]
```


smith의 science 점수를 출력해보세요

▼ A. 풀이

```
print(people[3]['score']['science'])
```

## 1-6. 조건문

▼ 16) `if` 문

- 조건을 만족했을 때만 특정 코드를 실행하도록 하는 문법입니다.

```
money = 5000
if money > 3800:
    print("택시 타자!")
```

- 파이썬에서는 어디까지 구문에 포함되는지를 들여쓰기로 구분하기 때문에 아주 중요합니다.

> 👉 조건에는 불 자료형이 들어갑니다. money > 3800은 `True`

```
money = 5000
if money > 3800:
    print("택시 타자!")
```

▼ 17) `else` 와 `elif`

- 조건을 만족하지 않을 때 다른 코드를 실행하고 싶을 때 쓰는 문법입니다.

```
money = 2000
if money > 3800:
    print("택시 타자!")
else:
    print("걸어가자...")
```

- 다양한 조건을 판단할 때는 `elif` 를 쓰면 좋아요!

```
age = 27
if age < 20:
    print("청소년입니다.")
elif age < 65:
    print("성인입니다.")
else:
    print("무료로 이용하세요!")
```

## 1-7. 반복문

▼ 18) `for` 문

- 0부터 9까지 숫자를 출력해볼까요?

```
print(0)
print(1)
print(2)
...
print(9)
```

- 파이썬에서는 아래처럼 쓰는 것이 더 자연스럽습니다.

```
fruits = ['사과', '배', '감', '귤']

for fruit in fruits:
    print(fruit)
```

▼ 19) 함께하기 - Q. 사람의 나이 출력하기

▼ **[코드스니펫] - 나이출력하기**

```
people = [
    {'name': 'bob', 'age': 20},
    {'name': 'carry', 'age': 38},
    {'name': 'john', 'age': 7},
    {'name': 'smith', 'age': 17},
    {'name': 'ben', 'age': 27},
    {'name': 'bobby', 'age': 57},
    {'name': 'red', 'age': 32},
    {'name': 'queen', 'age': 25}
]
```

```
people = [
    {'name': 'bob', 'age': 20},
    {'name': 'carry', 'age': 38},
    {'name': 'john', 'age': 7},
    {'name': 'smith', 'age': 17},
    {'name': 'ben', 'age': 27},
    {'name': 'bobby', 'age': 57},
    {'name': 'red', 'age': 32},
    {'name': 'queen', 'age': 25}
]
```

이 리스트에서 나이가 20보다 큰 사람만 출력합니다.

▼ A. 풀이

```
for person in people:
    if person['age'] > 20:
        print(person['name'])
```

▼ 20) `for` 문 - enumerate, break

▼ **[코드스니펫] - 나이 출력하기**

```
fruits = ['사과', '배', '감', '귤','귤','수박','참외','감자','배','홍시','참외','오렌지']
```

```
fruits = ['사과', '배', '감', '귤','귤','수박','참외','감자','배','홍시','참외','오렌지']
```

```
for i, fruit in enumerate(fruits):
    print(i,fruit)
```

예를 들어 앞에 5개만 출력해보고 싶다면?

```
for i, fruit in enumerate(fruits):
    print(i, fruit)
    if i == 4:
        break
```

## 1-8. 반복문 - 연습문제

▼ 21) ✍️ Q. 리스트에서 짝수만 출력하는 함수 만들기

▼ **[코드스니펫] - 짝수 출력하기**

```
num_list = [1, 2, 3, 6, 3, 2, 4, 5, 6, 2, 4]
```

```
num_list = [1, 2, 3, 6, 3, 2, 4, 5, 6, 2, 4]
```

이 리스트에서 짝수인 요소만 출력해봅시다.

▼ A. 풀이

```
num_list = [1, 2, 3, 6, 3, 2, 4, 5, 6, 2, 4]

for num in num_list:
    if num % 2 == 0:
        print(num)
```

▼ 22) ✍️ Q. 리스트에서 짝수의 개수를 출력하기

```
num_list = [1, 2, 3, 6, 3, 2, 4, 5, 6, 2, 4]
```

이 리스트에서 짝수가 몇 개인지 출력하기

▼ A. 풀이

```
num_list = [1, 2, 3, 6, 3, 2, 4, 5, 6, 2, 4]

count = 0

for num in num_list:
    if num % 2 == 0:
```

```
        count += 1

print(count)
```

▼ 23) ✍️ Q. 리스트 안에 있는 모든 숫자 더하기

```
num_list = [1, 2, 3, 6, 3, 2, 4, 5, 6, 2, 4]
```

▼ A. 풀이

```
result = 0
for num in num_list:
    result += num

print(result)
```

```
print(sum(a_list))
```

▼ 24) ✍️ Q. 리스트 안에 있는 자연수 중 가장 큰 숫자 구하기

```
num_list = [1, 2, 3, 6, 3, 2, 4, 5, 6, 2, 4]
```

▼ A. 풀이

```
max_value = 0
for num in num_list:
    if num > max_value:
       max_value = num

print(max_value)
```

```
print(max(a_list))
```

# 1-9. 함수

▼ 25) 함수 사용 방법

- 함수는 반복적으로 사용하는 코드들에 이름을 붙여놓은 것입니다.

```
def hello():
    print("안녕!")
    print("또 만나요!")


hello()
hello()
```

- 조건문에 넣을 값을 바꿔가면서 결과를 확인할 때 쓰면 편합니다.

```
def bus_rate(age):
    if age > 65:
        print("무료로 이용하세요")
    elif age > 20:
        print("성인입니다.")
    else:
        print("청소년입니다")

bus_rate(27)
bus_rate(10)
bus_rate(72)
```

- 단순한 출력 뿐만 아니라 결과 값을 돌려주도록 함수를 만들 수도 있어요!

```
def bus_fee(age):
    if age > 65:
        return 0
    elif age > 20:
        return 1200
    else:
        return 0


money = bus_fee(28)
print(money)
```

▼ 26) ✍️ Q. 주민등록번호를 입력받아 성별을 출력하는 함수 만들기

```
def check_gender(pin):
    print('')

my_pin = '200101-3012345'
check_gender(my_pin)
```

주민등록번호 뒷자리의 맨 첫 번째 숫자가 1, 3 등 홀수이면 남성, 2, 4 등 짝수이면 여성이죠?

힌트! → "2"라는 문자열을 숫자로 바꾸려면? `int("2")` 이렇게, int로 감싸주세요!

▼ A. 풀이

```
def check_gender(pin):
    num = int(pin.split('-')[1][0])
    if num % 2 == 0:
        print('여성')
    else:
        print('남성')

my_pin = "200101-3012345"
check_gender(my_pin)
```

---

## 2-1. 파이썬 문법 뽀개기 - 심화

▼ 심화 수업에서 다룰 내용들

- 약-간 심화된 문법에 관해 다룰거예요!
- 아주 많은 내용이 나올 예정이지만, 바로 쓸만한 것들도 있고, `알아두기만` 해도 괜찮은 것들도 있어요! 그러니, 이 다음 수업들은 오히려 '맘 편안히' 들어주세요~!

## 2-2. 튜플, 집합

▼ 1) 튜플 (tuple)

- 튜플은 `리스트와 비슷하지만 불변`인 자료형 입니다. 마찬가지로 순서가 존재하죠!

```
a = (1,2,3)

print(a[0])
```

- 예를 들어 아래와 같은 작업이 불가합니다!

```
a = (1,2,3)
a[0] = 99
```

- 언제 주로 사용하냐면요, 아래와 같이, 딕셔너리 대신 리스트와 튜플로 딕셔너리 '비슷하게' 만들어 사용해야 할 때 많이 쓰입니다!

```
a_dict = [('bob','24'),('john','29'),('smith','30')]
```

▼ 2) 집합 (set)

- 집합은 말 그대로 '집합'을 구현하는 방법! 좋은점: `중복이 제거`됩니다.

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

a_set = set(a)

print(a_set)
```

- 교집합 / 합집합 / 차집합도 구할 수 있어요

```
a = ['사과','감','수박','참외','딸기']
b = ['사과','멜론','청포도','토마토','참외']

print(a & b)  # 교집합
print(a | b)  # 합집합
```

▼ 3) ✍️ 구글링문제 - Q. A가 들은 수업 중, B가 듣지 않은 수업을 찾아보기

**▼ [코드스니펫] - AB수업 문제**

```
student_a = ['물리2','국어','수학1','음악','화학1','화학2','체육']
student_b = ['물리1','수학1','미술','화학2','체육']
```


```
student_a = ['물리2','국어','수학1','음악','화학1','화학2','체육']
student_b = ['물리1','수학1','미술','화학2','체육']
```


a와 b의 '차집합' 구하는 방법을 구글링해보세요!

▼ A. 풀이

```
set_a = set(student_a)
set_b = set(student_b)

print(set_a-set_b)
```

## 2-3. f-string

▼ 4) 변수로 더 직관적인 문자열 만들기

- 예를 들어 아래 for문을 살펴보겠습니다.

**▼ [코드스니펫] - 점수 모음**

```
scores = [
    {'name':'영수','score':70},
    {'name':'영희','score':65},
    {'name':'기찬','score':75},
    {'name':'희수','score':23},
    {'name':'서경','score':99},
    {'name':'미주','score':100},
    {'name':'병태','score':32}
]
```


```
scores = [
    {'name':'영수','score':70},
    {'name':'영희','score':65},
    {'name':'기찬','score':75},
    {'name':'희수','score':23},
    {'name':'서경','score':99},
    {'name':'미주','score':100},
    {'name':'병태','score':32}
]
```


이름과 점수를 모두 출력해볼게요

```
for s in scores:
    name = s['name']
    score = str(s['score'])
    print(name,score)
```

아래와 같이 출력해보면 어떨까요?

```
for s in scores:
    name = s['name']
    score = str(s['score'])
    print(name+'는 '+score+'점 입니다')
```

f-stirng을 이용하면 훨씬 간단하게 가능하답니다.

```
for s in scores:
    name = s['name']
    score = str(s['score'])
    print(f'{name}은 {score}점입니다')
```

## 2-4. 예외처리

▼ 5) try - except 문

- 에러가 있어도 건너뛰게 할 수 있는 방법

**▼ [코드스니펫] - try-except 예제**

```
people = [
    {'name': 'bob', 'age': 20},
    {'name': 'carry', 'age': 38},
    {'name': 'john', 'age': 7},
    {'name': 'smith', 'age': 17},
    {'name': 'ben', 'age': 27},
    {'name': 'bobby', 'age': 57},
    {'name': 'red', 'age': 32},
    {'name': 'queen', 'age': 25}
]
```

👉 실제 프로젝트 남용하는 것은 금물! 어디서 에러가 났는지 알 수 없어요 💀

자, 다음과 같이 20세 보다 큰 사람만 출력한다고 해봅시다.

```
people = [
    {'name': 'bob', 'age': 20},
    {'name': 'carry', 'age': 38},
    {'name': 'john', 'age': 7},
    {'name': 'smith', 'age': 17},
    {'name': 'ben', 'age': 27},
    {'name': 'bobby', 'age': 57},
    {'name': 'red', 'age': 32},
    {'name': 'queen', 'age': 25}
]

for person in people:
    if person['age'] > 20:
        print (person['name'])
```

그런데 만약, bobby가 age를 갖고 있지 않다면? - 데이터 하나가 잘못 들어간거죠!

```
people = [
    {'name': 'bob', 'age': 20},
    {'name': 'carry', 'age': 38},
    {'name': 'john', 'age': 7},
    {'name': 'smith', 'age': 17},
    {'name': 'ben', 'age': 27},
    {'name': 'bobby'},
    {'name': 'red', 'age': 32},
    {'name': 'queen', 'age': 25}
]

for person in people:
    if person['age'] > 20:
        print (person['name'])
```

그 때 아래와 같이 try except 구문을 이용하면 에러를 넘길 수 있답니다.

```
for person in people:
    try:
        if person['age'] > 20:
            print (person['name'])
    except:
        name = person['name']
        print(f'{name} - 에러입니다')
```

## 2-5. 파일 불러오기

▼ 6) 여러개 파일로 분리하려면

- main_test.py

```
from main_func import *

say_hi()
```

- main_func.py

```
def say_hi():
  print('안녕!')
```

## 2-6. 한줄의 마법

▼ 7) if문 - 삼항연산자

- 만약 조건에 따라 다른 값을 변수에 저장하고 싶다면?

```
num = 3

if num%2 == 0:
    result = "짝수"
else:
    result = "홀수"

print(f"{num}은 {result}입니다.")
```

- 이것을 한 줄에 적는 것이 파이썬의 유일한 삼항연산자인 조건식입니다.

```
num = 3

result = "짝수" if num%2 == 0 else "홀수"

print(f"{num}은 {result}입니다.")
```

💡 `(참일 때 값) if (조건) else (거짓일 때 값)`으로 항이 3개라 삼항 연산자입니다 🙂

▼ 8) for문 - 한방에 써버리기

- `a_list`의 각 요소에 2를 곱한 새로운 리스트를 만들고 싶다면?

```
a_list  = [1, 3, 2, 5, 1, 2]

b_list = []
for a in a_list:
    b_list.append(a*2)

print(b_list)
```

- 이것을 한 번에 쓸 수 있는 방법이 있죠!

```
a_list  = [1, 3, 2, 5, 1, 2]

b_list = [a*2 for a in a_list]

print(b_list)
```

🚨 **아래부터는 조-금 어려울 수 있으니, 가볍게 `나중을 위해 들어둔다` 라고 생각해두세요!**

## 2-7. map, filter, lambda식

▼ 9) map - 리스트의 모든 원소를 조작하기

▼ **[코드스니펫] - 리스트조작 연습**

```
people = [
    {'name': 'bob', 'age': 20},
    {'name': 'carry', 'age': 38},
    {'name': 'john', 'age': 7},
    {'name': 'smith', 'age': 17},
    {'name': 'ben', 'age': 27},
    {'name': 'bobby', 'age': 57},
    {'name': 'red', 'age': 32},
    {'name': 'queen', 'age': 25}
]
```

1. 1차 조작

```
def check_adult(person):
    if person['age'] > 20:
        return '성인'
    else:
        return '청소년'

result = map(check_adult, people)
print(list(result))
```

2. 2차 조작!

```
def check_adult(person):
    return '성인' if person['age'] > 20 else '청소년'

result = map(check_adult, people)
print(list(result))
```

3. 3차 조작!

```
result = map(lambda x: ('성인' if x['age'] > 20 else '청소년'), people)
print(list(result))
```

▼ 10) filter - 리스트의 모든 원소 중 특별한 것만 뽑기

- map과 아주 유사한데, True인 것들만 뽑기! (map보다 훨씬 쉬워요!)

```
result = filter(lambda x: x['age'] > 20, people)
print(list(result))
```

## 2-8. 함수 심화

▼ 10) 함수의 매개변수

💡 이 내용들은 직접 쓰는 것보단 알고 있으면 내장함수 등을 사용할 때 도큐먼트를 읽는 데 도움이 됩니다!

1. 함수에 인수를 넣을 때, 어떤 매개변수에 어떤 값을 넣을지 정해줄 수 있어요. 순서 상관 없음!

```
def cal(a, b):
    return a + 2 * b

print(cal(3, 5))
print(cal(5, 3))
print(cal(a=3, b=5))
print(cal(b=5, a=3))
```

2. 특정 매개변수에 디폴트 값을 지정해줄 수 있어요.

```
def cal2(a, b=3):
    return a + 2 * b

print(cal2(4))
print(cal2(4, 2))
print(cal2(a=6))
print(cal2(a=1, b=7))
```

3. 입력값의 개수를 지정하지 않고 모두 받는 방법!

```
def call_names(*args):
    for name in args:
        print(f'{name}야 밥먹어라~')

call_names('철수','영수','희재')
```

💡 이렇게 여러 개의 인수를 하나의 매개변수로 받을 때 관례적으로 `args`라는 이름을 사용합니다. arguments라는 뜻이에요!

4. 키워드 인수를 여러 개 받는 방법!

```
def get_kwargs(**kwargs):
    print(kwargs)

get_kwargs(name='bob')
get_kwargs(name='john', age='27')
```

## 2-9. 클래스

▼ 11) 우선, 클래스를 언제 사용하는지 생각해보시죠!

👉 예를 들어, 아주 많은 몬스터들의 HP를 관리해야 하면 어떻게 해야 편할까요?


방법1. → 리스트의 순서를 잘 지켜서 각 몬스터들의 hp를 잘 적어둔다.


방법2. → 몬스터마다 각자의 hp를 가질 수 있게 한다.


▼ 12) 클래스의 사용 방법을 눈으로 살펴보겠습니다.

```
class Monster():
    hp = 100
    alive = True

    def damage(self, attack):
        self.hp = self.hp - attack
        if self.hp < 0:
            self.alive = False

    def status_check(self):
        if self.alive:
            print('살아있다')
        else:
            print('죽었다')

m = Monster()
m.damage(120)

m2 = Monster()
m2.damage(90)

m.status_check()
m2.status_check()
```