

Python

With 데이터 분석

왜 배워야해?



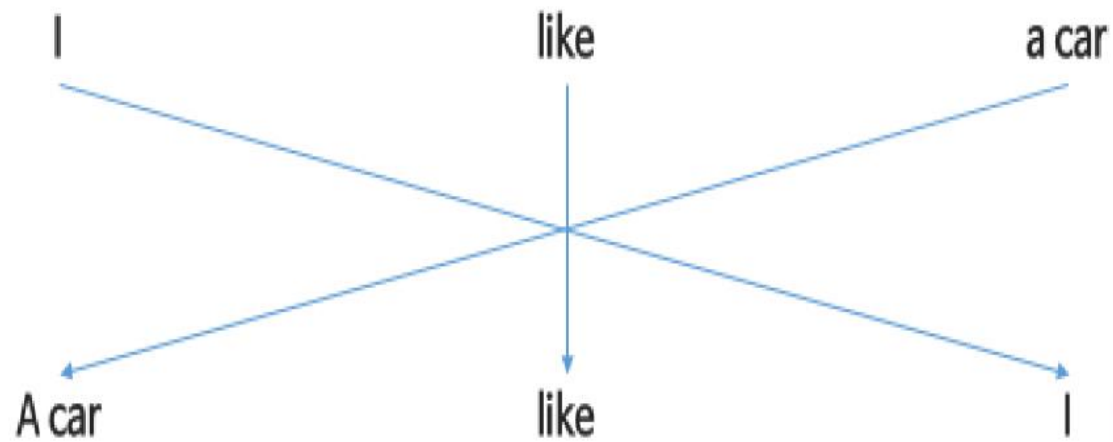
**Python
Environment**

얼마나 배워야해?

주어(S)

동사(V)

목적어(O)



이상해..

Python

기초강의

자료형

데이터가 어떤 종류인지 나타내는 분류

신규 회원 가입

이름

주소

전화번호

생년월일

☐

마케팅 정보 제공에 동의합니다.

신규 회원 가입

이름

지피티

주소

오픈에이아이

전화번호

012345678

생년월일

20220222



마케팅 정보 제공에 동의합니다.

신규 회원 가입

이름	지피티	문자 자료형
주소	오픈에이아이	
전화번호	012345678	숫자 자료형
생년월일	20220222	
<input type="checkbox"/> V	마케팅 정보 제공에 동의합니다.	불리안 자료형

숫자 자료형

Print(1)

Print(3.14)

문자 자료형

Print('하이')

Print("반가워")

문자 자료형

Print('하이')

Print("반가워")

Print("10")



불리안 자료형

Print(True)

Print(False)



형 변환

데이터의 자료형을 맞춰주는 것

$$2 + 1 = 3$$

2 + 'T' =

2 + 'T' = 'TT'

2 + 'T' = ~~'TT'~~

불가능

형 변환

숫자형

Int('2') → 2

float('1.5') → 1.5

float(2) → 2.0

int(2.5) → 2

문자형

str(2) → '2'

str(1.5) → '1.5'

str(False) → 'False'

str(True) → 'True'

변수

어떤 값을 저장하는 공간



봉투



봉투

= 변수

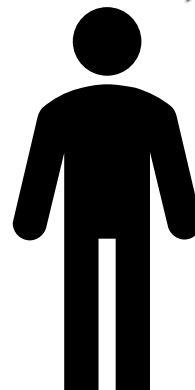
봉투 = 세뱃돈

변수 선언

변수 이름 = 값

variable = 100

name = 'Son'



프로그램 언어는
왼쪽에서 오른쪽

연산자

값을 연산하는 기호

산술 연산자

기호	의미	예시
+	더하기	<code>print(5 + 2)</code> >> 실행결과 7
-	빼기	<code>print(5 - 2)</code> >> 3
*	곱하기	<code>print(5 * 2)</code> >> 10
/	나누기	<code>print(5 / 2)</code> >> 2.5

기호	의미	예시
%	나머지	<code>print(5 % 2)</code> >> 실행결과 1
//	몫	<code>print(5 // 2)</code> >> 2
**	거듭 제곱	<code>print(5 ** 2)</code> >> 25

비교 연산자

기호	의미	예시
>	크다	<code>print(5 > 2)</code> >> 실행결과 True
>=	크거나 같다	<code>print(5 >= 2)</code> >> True
<	작다	<code>print(5 < 2)</code> >> False
<=	작거나 같다	<code>print(5 <= 2)</code> >> False

기호	의미	예시
==	같다	<code>print(5 == 2)</code> >> 실행결과 False
!=	같지 않다	<code>print(5 != 2)</code> >> True

논리 연산자

기호	의미	예시
and	둘다 참이면 True	<code>print(3 < 5 and 7 < 5)</code> >> 실행결과 False
or	하나라도 참이면 True	<code>print(3 < 5 or 7 < 5)</code> >> True
not	반전	<code>print(not 3 < 5)</code> >> False

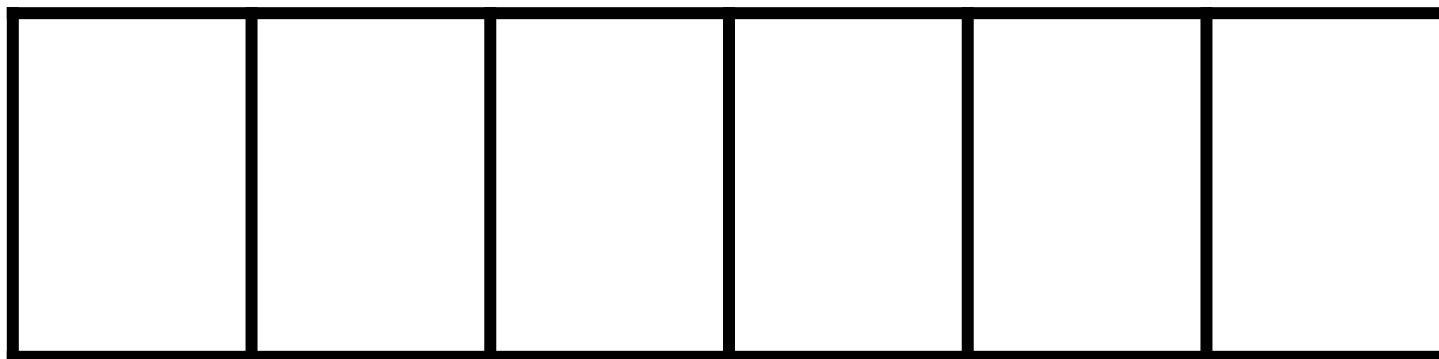
멤버 연산자

기호	의미	예시
in	포함	<code>print('c' in 'cat')</code> >> 실행결과 True
not in	미포함	<code>print('c' not in 'cat')</code> >> False

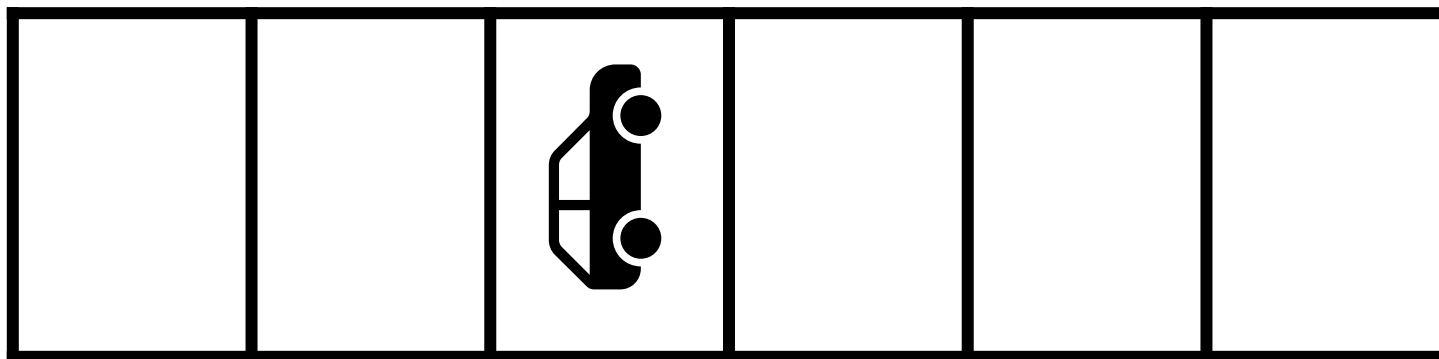
인덱싱

데이터를 위치로 찾는 방법

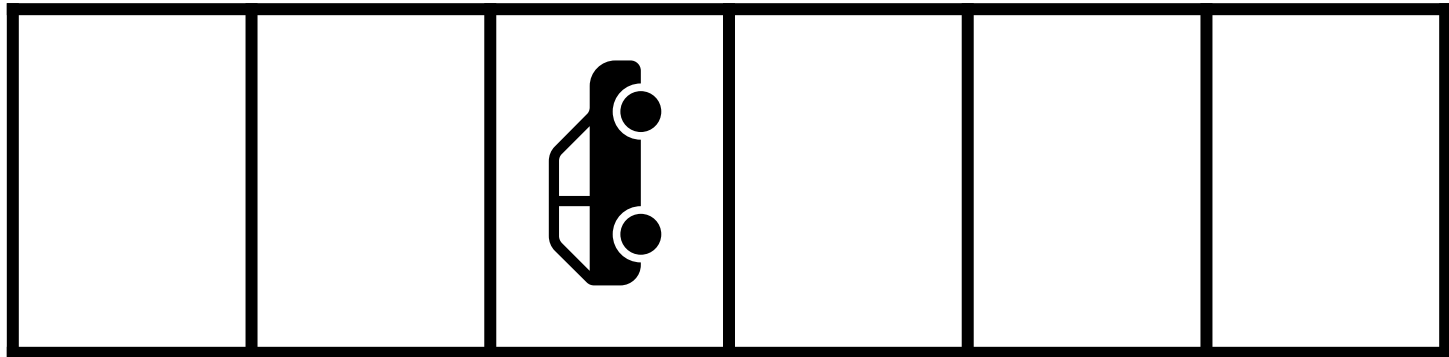
주차장



주차장

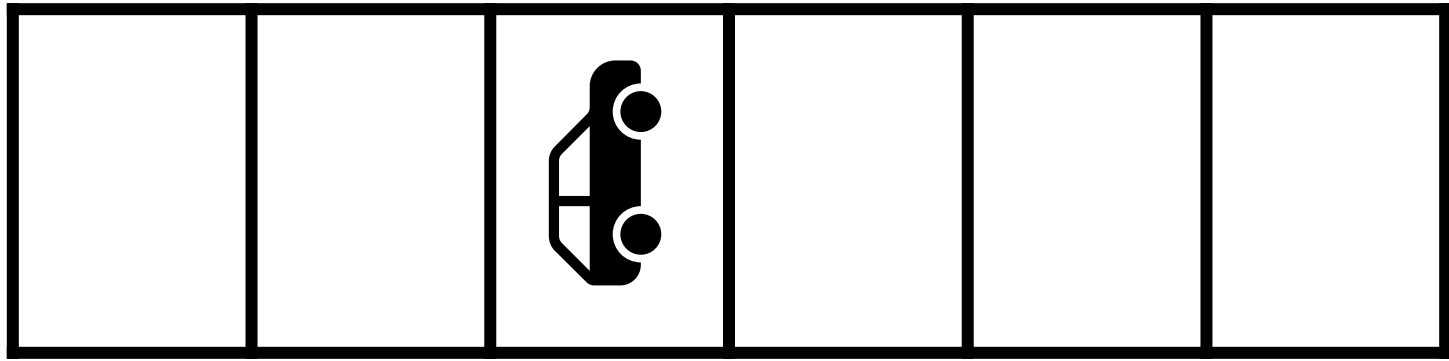


주차장



몇 번째 칸에 있을까?

주차장

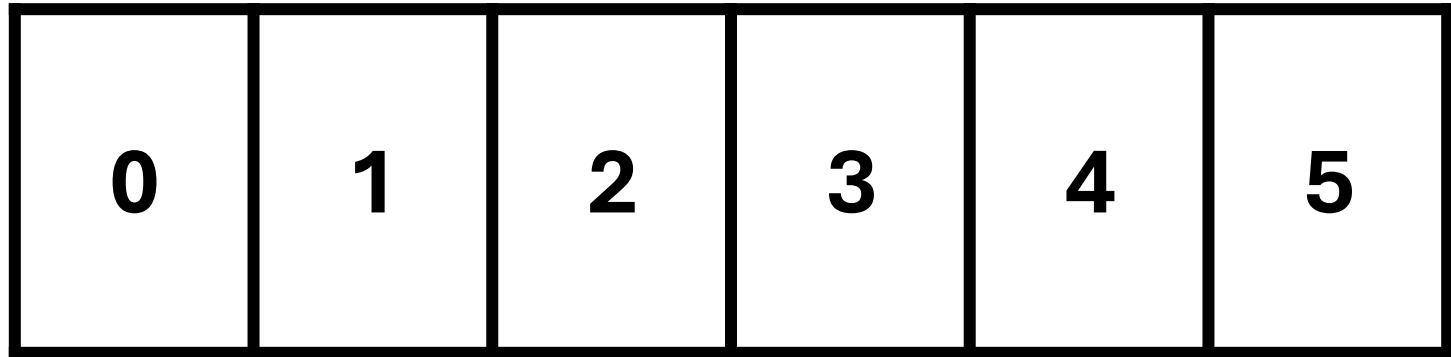


몇 번째 칸에 있을까?

||

인덱스

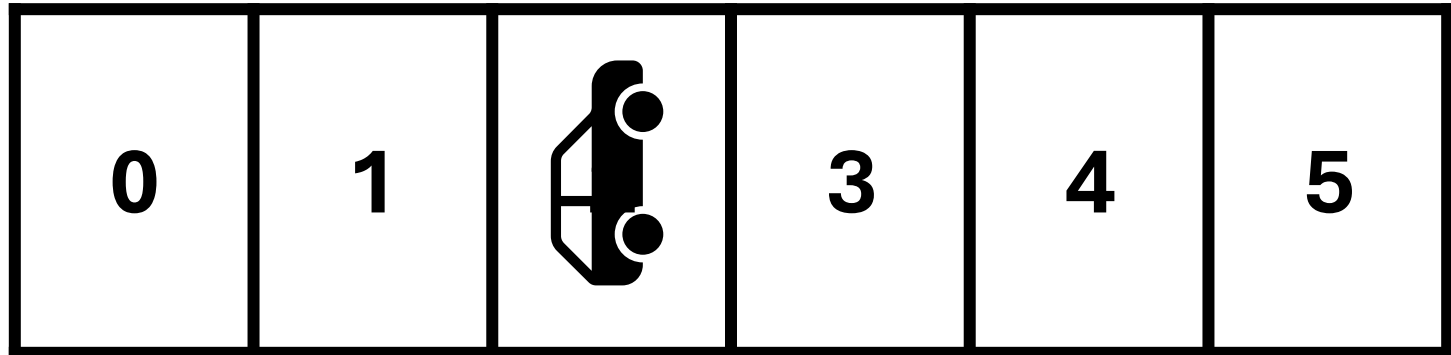
파이썬 주차장



Python의
주차장 번호는
0부터 시작



파이썬 주차장



3번째

칸에 있음

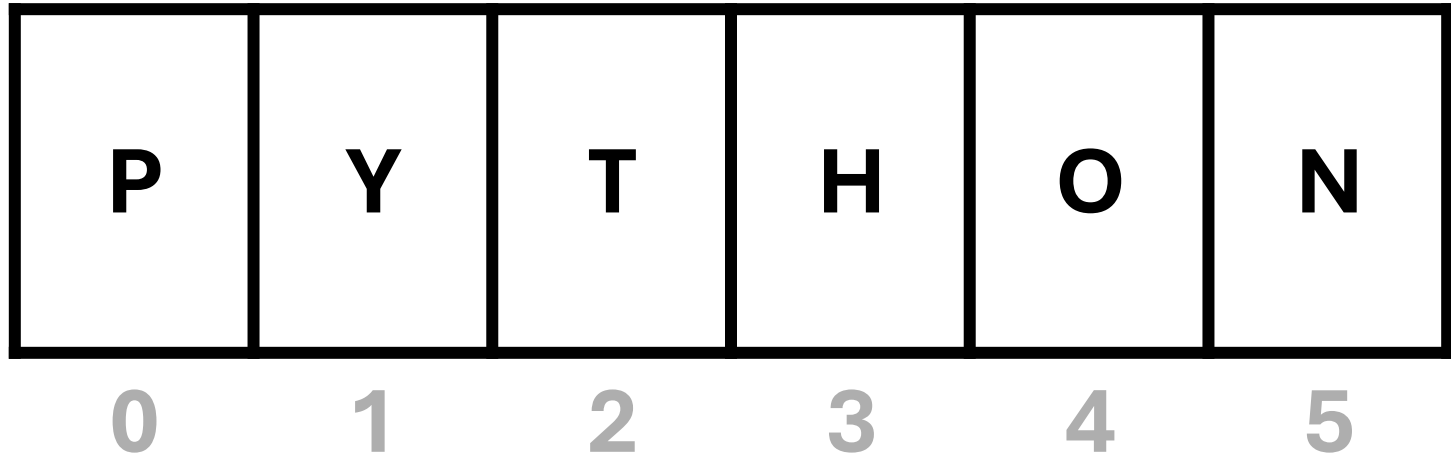
||

인덱스

2

```
lang = 'PYTHON'
```

lang 주차장

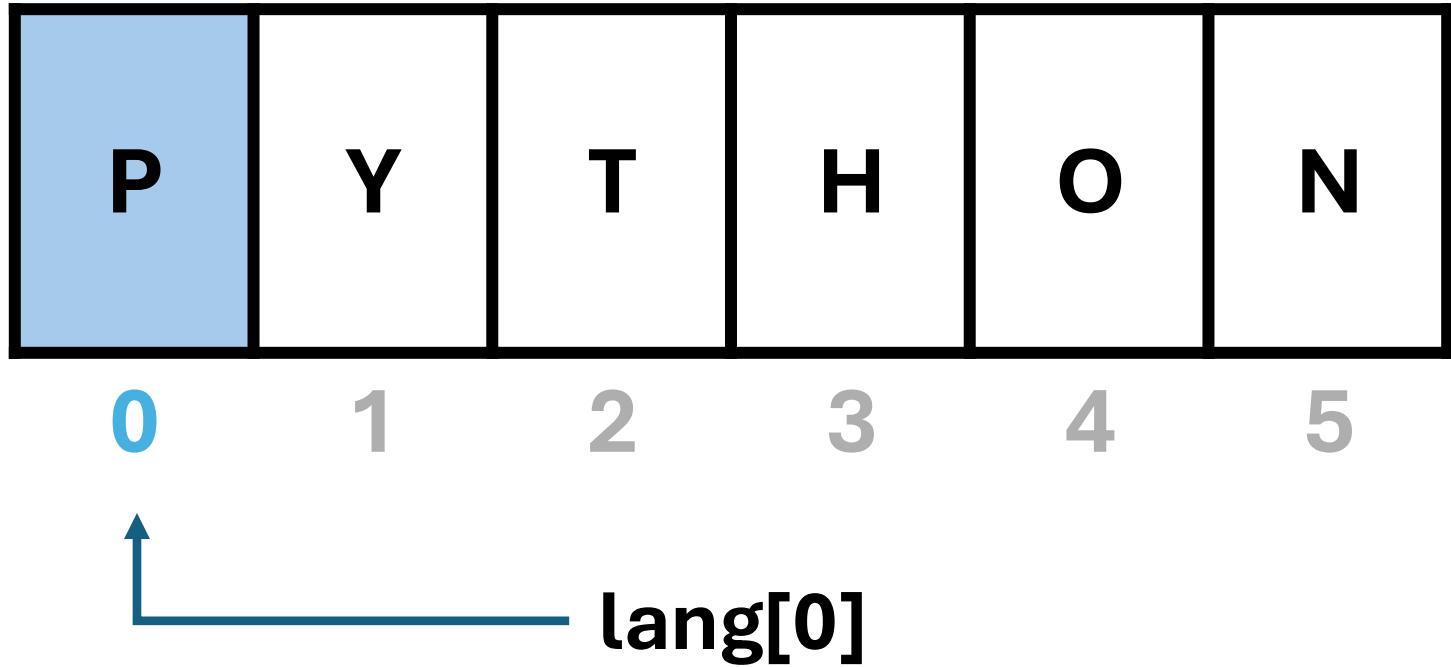


문자형을
변수에 할당시
주차장 생성



```
lang = 'PYTHON'
```

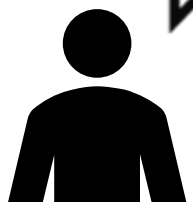
lang 주차장



슬라이싱

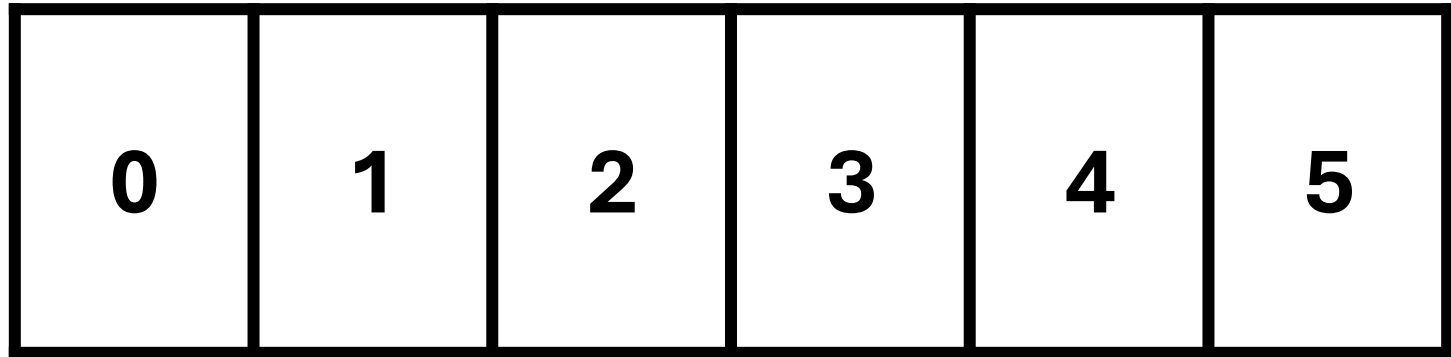
데이터를 범위로 찾는 방법

시작부터: 끝 직전까지



데이터의
범위를
지정하는 표현

파이썬 주차장

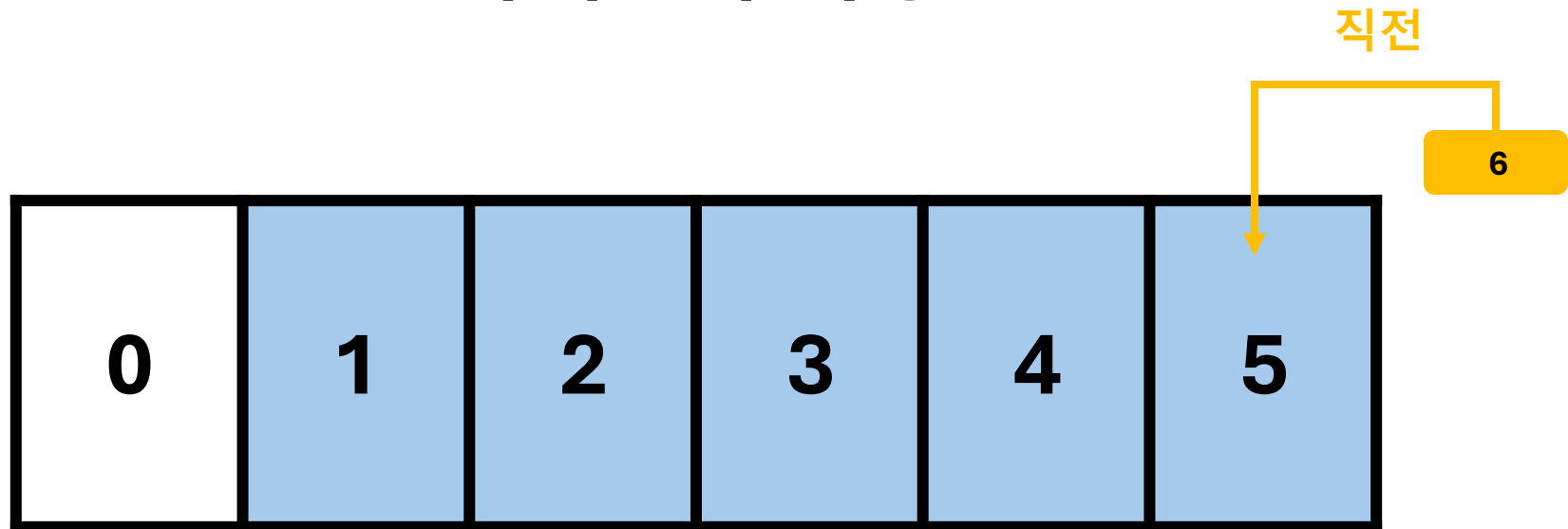


나 어디에
주차해?



인덱스 1부터
6직전까지
주차할 수 있어

파이썬 주차장



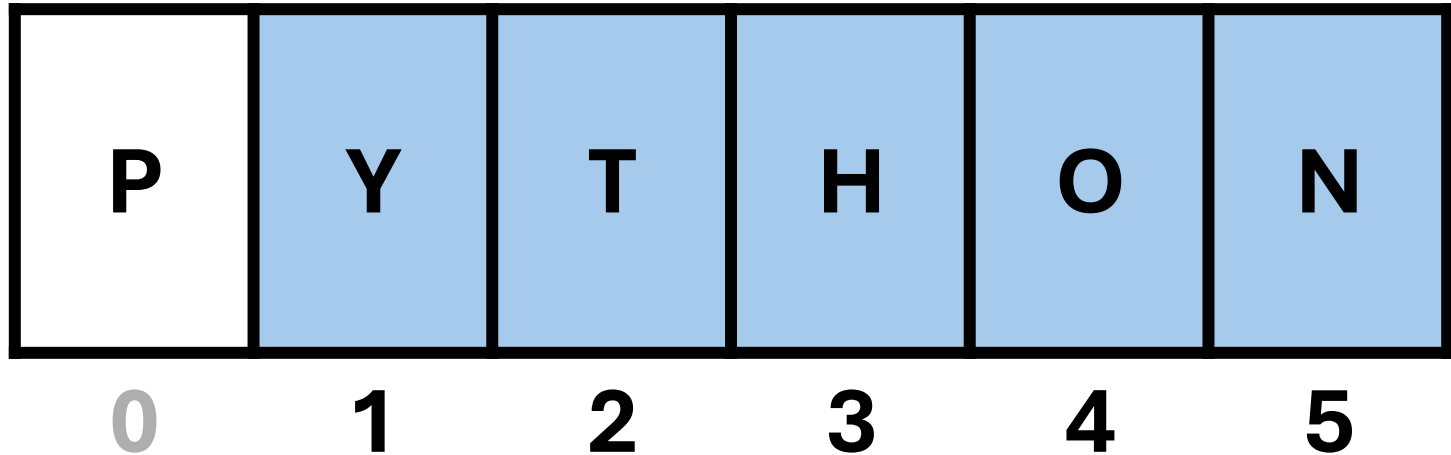
내가 주차할 수
있는 범위는..?



파이썬 주차장[1:6]

```
lang = 'PYTHON'
```

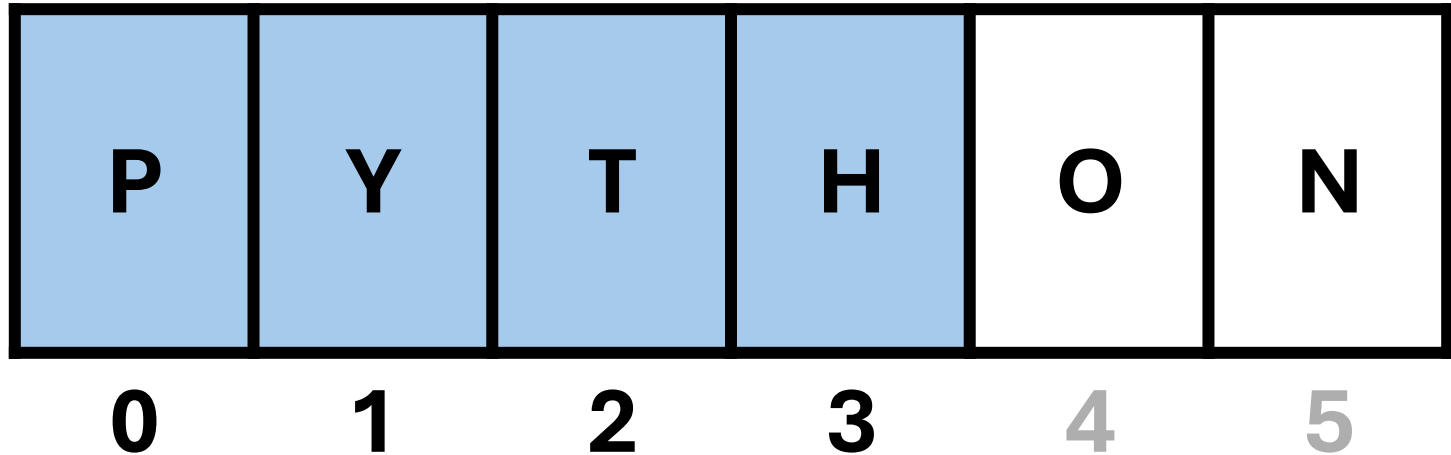
lang 주차장



```
lang[1:6] > 'YTHON'
```

```
lang = 'PYTHON'
```

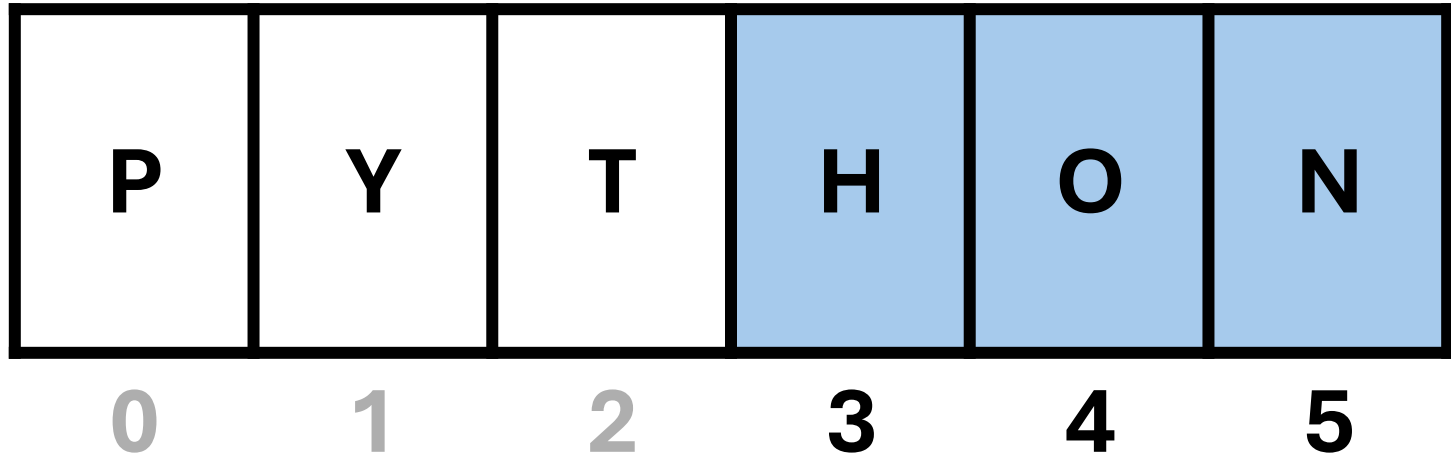
lang 주차장



```
lang[:4] > 'PYTH'
```

```
lang = 'PYTHON'
```

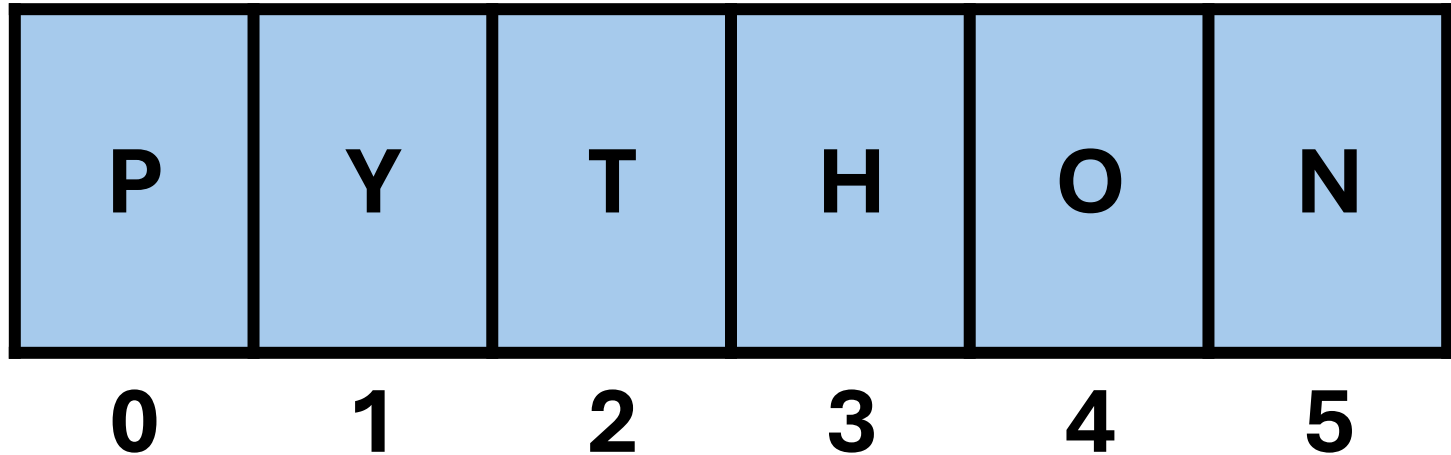
lang 주차장



```
lang[3:] > 'HON'
```

```
lang = 'PYTHON'
```

lang 주차장



```
lang[:] > 'PYTHON'
```

리스트

데이터를 저장하는 방법1



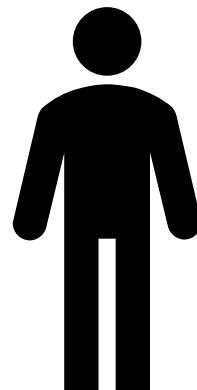
리스트

리스트

리스트=[값, 값, ...]

`list = [1, 2, True, False, '아무거나']`

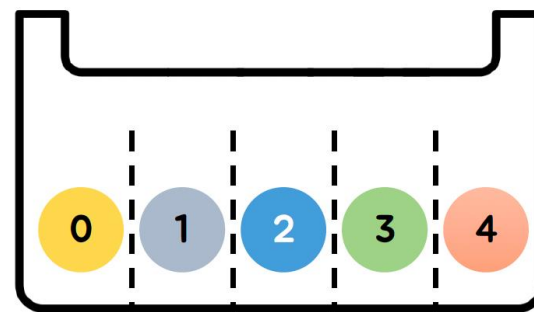
`empty_list = []`



골라 잡아~~



X



← 인덱스

순서대로 관리

리스트

리스트=[값, 값, ...]

list = [1, 2, True, False, '아무거나']

인덱싱 복습

Print(list[2])

>

리스트

리스트=[값, 값, ...]

list = [1, 2, True, False, '아무거나']

인덱싱 복습

Print(list[2])

> True

리스트

리스트=[값, 값, ...]

list = [1, 2, True, False, '아무거나']

슬라이싱 복습

Print(list[2:4])

>

리스트

리스트=[값, 값, ...]

list = [1, 2, True, False, '아무거나']

슬라이싱 복습

Print(list[2:4])

> [True, False]

리스트

리스트=[값, 값, ...]

list = [1, 2, True, False, '아무거나']

연산자 복습

Print('아무거나' in list)

>

리스트

리스트=[값, 값, ...]

list = [1, 2, True, False, '아무거나']

연산자 복습

Print('아무거나' in list)

> True

리스트

리스트=[값, 값, ...]

list = [1, 2, True, False, '아무거나']

길이는?

Print(len(list))

> 5

리스트

리스트=[값, 값, ...]

```
list = [1, 2, True, False, '아무거나']
```

리스트 값 변경

```
list[-1] = '수정됨'
```

```
Print(list[3:])
```

```
> [False, '수정됨']
```

-1은 제일
마지막을
의미해요



리스트

리스트=[값, 값, ...]

list = [1, 2, True, False, '아무거나']

리스트 값 추가

list.append('ㅋㅋㅋ')

Print(list[-1])

> ['ㅋㅋㅋ']

리스트

리스트=[값, 값, ...]

```
list = [1, 2, True, False, '아무거나']
```

리스트 값 제거

```
list.remove('아무거나')
```

```
Print(list)
```

```
> [1, 2, True, False]
```

remove의 값이
여러개면 모두
삭제됩니다



리스트

리스트=[값, 값, ...]

```
list = [1, 'ㅋ']
```

```
list2 = [False]
```

리스트 연산

```
list3 = list + list2
```

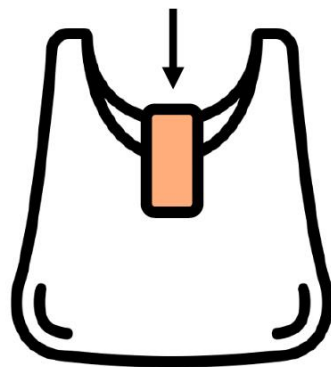
```
Print(list3)
```

```
> [1, 'ㅋ', False]
```

튜플

데이터를 저장하는 방법2

개봉금지 스티커



투플

튜플

🚫 수정불가

튜플=(값, 값, ...)

`tuple = (1, '2', True)`

`Print(list[2])`

`> True`

튜플

🚫 수정불가

튜플=(값, 값, ...)

`tuple = (1, '2', True)`

`Print(list[1:])`

`> ('2', True)`

튜플

🚫 수정불가

튜플=(값, 값, ...)

`tuple = (1, '2', True)`

`Print(1 in tuple)`

`> True`

튜플

🚫 수정불가

튜플=(값, 값, ...)

`tuple = (1, '2', True)`

`Print(len(tuple))`

`> 3`

딕셔너리

데이터를 저장하는 방법3

공부

: [명사] 학문이나 기술을 배우고 익힘

공부

key

: [명사] 학문이나 기술을 배우고 익힘 value

딕셔너리

(key, value)

중복 불가

KEY	VALUE
이름	지피티
생년월일	20220222
정보동의	True

딕셔너리

딕셔너리 = {'키1': 값1, '키2': 값2, ...}

user = {'이름' : '지피티', '생년월일':20220222, '정보동의': True}

KEY	VALUE
이름	지피티
생년월일	20220222
정보동의	True

딕셔너리

딕셔너리 = {'키1': 값1, '키2': 값2, ...}

```
user = {'이름' : '지피티', '생년월일':20220222, '정보동의': True}
```

값 확인

```
Print(user['이름'])
```

```
> '지피티'
```

KEY	VALUE
이름	지피티
생년월일	20220222
정보동의	True

딕셔너리

딕셔너리 = {'키1': 값1, '키2': 값2, ...}

```
user = {'이름' : '지피티', '생년월일':20220222, '정보동의': True}
```

데이터 추가

```
user['주소'] = 'OpenAI'
```

KEY	VALUE
이름	지피티
생년월일	20220222
정보동의	True
주소	OpenAI

딕셔너리

딕셔너리 = {'키1': 값1, '키2': 값2, ...}

```
user = {'이름' : '지피티', '생년월일':20220222, '정보동의': True}
```

```
# 데이터 삭제
```

```
user.pop('생년월일')
```

KEY	VALUE
이름	지피티
정보동의	True

딕셔너리

딕셔너리 = {'키1': 값1, '키2': 값2, ...}

```
user = {'이름' : '지피티', '생년월일':20220222, '정보동의': True}
```

데이터 변경

```
user['이름'] = '지피티4'
```

KEY	VALUE
이름	지피티4
생년월일	20220222
정보동의	True

딕셔너리

딕셔너리 = {'키1': 값1, '키2': 값2, ...}

```
user = {'이름' : '지피티', '생년월일':20220222, '정보동의': True}
```

키 확인

```
Print(user.keys())
```

```
> ['이름', '생년월일', '정보동의']
```

KEY	VALUE
이름	지피티
생년월일	20220222
정보동의	True

딕셔너리

딕셔너리 = {'키1': 값1, '키2': 값2, ...}

```
user = {'이름' : '지피티', '생년월일':20220222, '정보동의': True}
```

값 확인

```
Print(user.values())
```

```
> ['지피티', 20220222, True]
```

KEY	VALUE
이름	지피티
생년월일	20220222
정보동의	True

딕셔너리

딕셔너리 = {'키1': 값1, '키2': 값2, ...}

```
user = {'이름' : '지피티', '생년월일':20220222, '정보동의': True}
```

```
# 키:값 확인
```

```
Print(user.items())
```

```
> [('이름', '지피티'), ('생년월일', 20220222), ('정보동의', True)]
```


IF

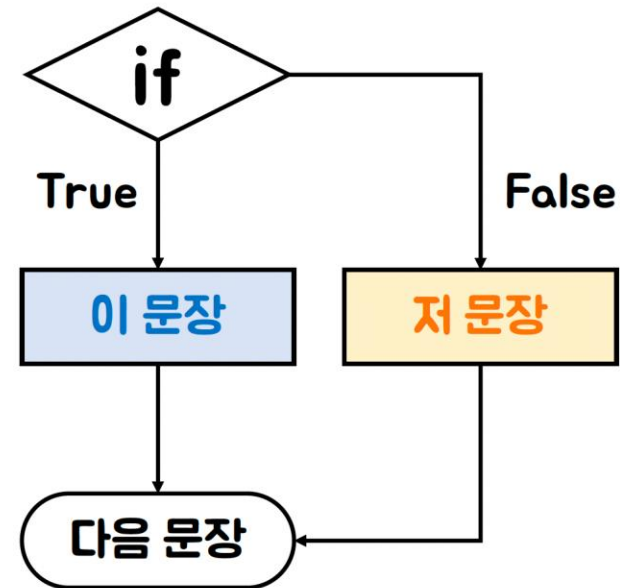
알고리즘 기초1

If, else
만약 ~ 라면, 그렇지 않다면

조건문 if

```
today = '주말'
if today == '주말':
    print('행복한')
else:
    print('지옥같은')
print('인생')
```

> 행복한
> 인생



elif

아니야? 그럼 혹시 ~라면

if, elif, elif, elif, else

만약 ~ 라면

아니야? 그럼 혹시 ~라면

아니야? 그럼 혹시 ~라면

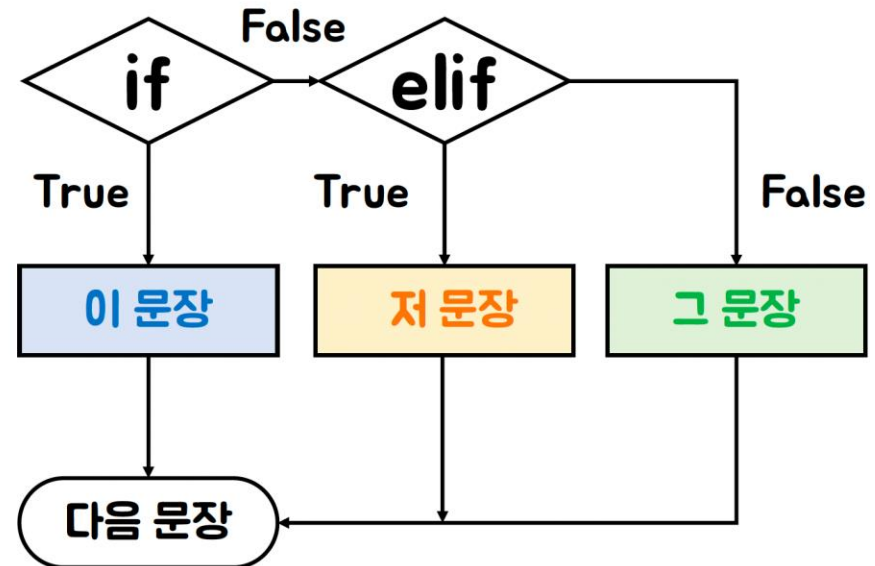
아니야? 그럼 혹시 ~라면

그렇지 않다면

조건문 if

```
today = '토요일'
if today == '일요일':
    print('미묘한')
if today == '토요일':
    print('행복한')
else:
    print('지옥같은')
print('인생')
```

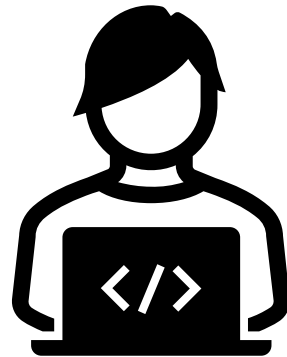
> 행복한
> 인생



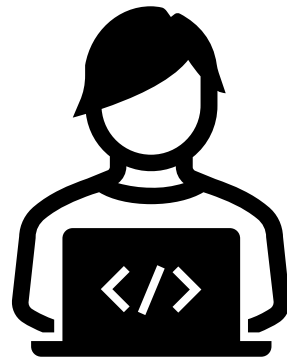
FOR

알고리즘 기초2

**파이썬이
스퀘트를하면
건강해질까?**

[illegible]

몸짱 파이썬을
만들어보고
싶다능.. 헤헤



`print('스쿼트!!!')`

`X 1000...?`

반복문
for

반복문 for

**for 변수 in 반복 대상:
반복 수행**

```
For x in range(1000):  
    print('스쿼트!!!')
```

```
> 스쿼트!!  
> 스쿼트!!  
> 스쿼트!!  
... (1000번 수행중)  
> 스쿼트!!
```

range는 0 부터
적힌 숫자
미만까지!



함수

어떤 동작을 수행하는 코드의 묶음



주가 조회 함수



주가조회 코드



주가 출력 코드

함수 정의

```
def 함수명():  
    수행할 문장
```

```
def show_stock_price():  
    print('해당 종목의 주가가 왜 궁금한데요')
```

함수 호출

```
def show_stock_price():  
    print('해당 종목의 주가가 왜 궁금한데요')
```

```
stock_code = '005930'  
print(f'{stock_cde}종목의 주가를 알려드리겠습니다. .')  
show_stock_price()
```

```
> 005930 종목의 주가를 알려드리겠습니다..  
> '해당 종목의 주가가 왜 궁금한데요'
```

반환값

함수 내에서 처리된 결과를 반환

함수 정의

```
def 함수명():  
    수행할 문장  
    return 반환값
```

```
def get_stock_price():  
    return 3000
```

함수 호출

```
def get_stock_price():  
    return 3000
```

```
stock_code = '005930'  
stock_price = get_stock_price()  
print(f'{stock_code}종목의 주가는 {stock_price}원 입니다.')
```

```
> 005930 종목의 주가는 3000원 입니다'
```

전달값

함수 내에서 처리하기 위한 값

함수 정의

```
def 함수명(전달값):  
    수행할 문장  
    return 반환값
```

```
def get_stock_price(stock_code):  
    price = # 종목코드로 주가를 가져오는 코드  
    return price
```

함수 호출

```
def get_stock_price(stock_code):  
    price = # 종목코드로 주가를 가져오는 코드  
    return price
```

```
stock_code = '005930'
```

```
stock_price = get_stock_price(stock_code)
```

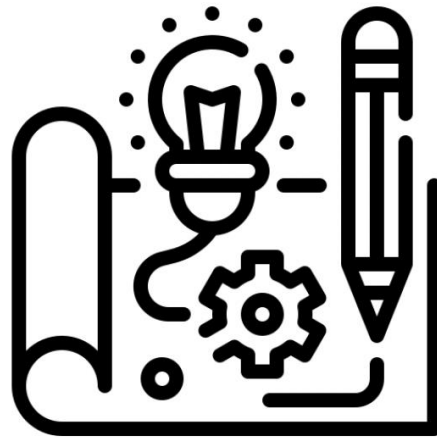
```
print(f'{stock_code}종목의 주가는 {stock_price}원 입니다.')
```

> 005930 종목의 주가는 **80,000** 원 입니다'

★ 호출시마다 매번 최신의 주가를 가져올 수 있음

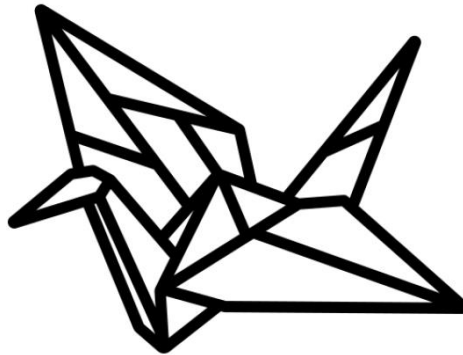
클래스

설계도

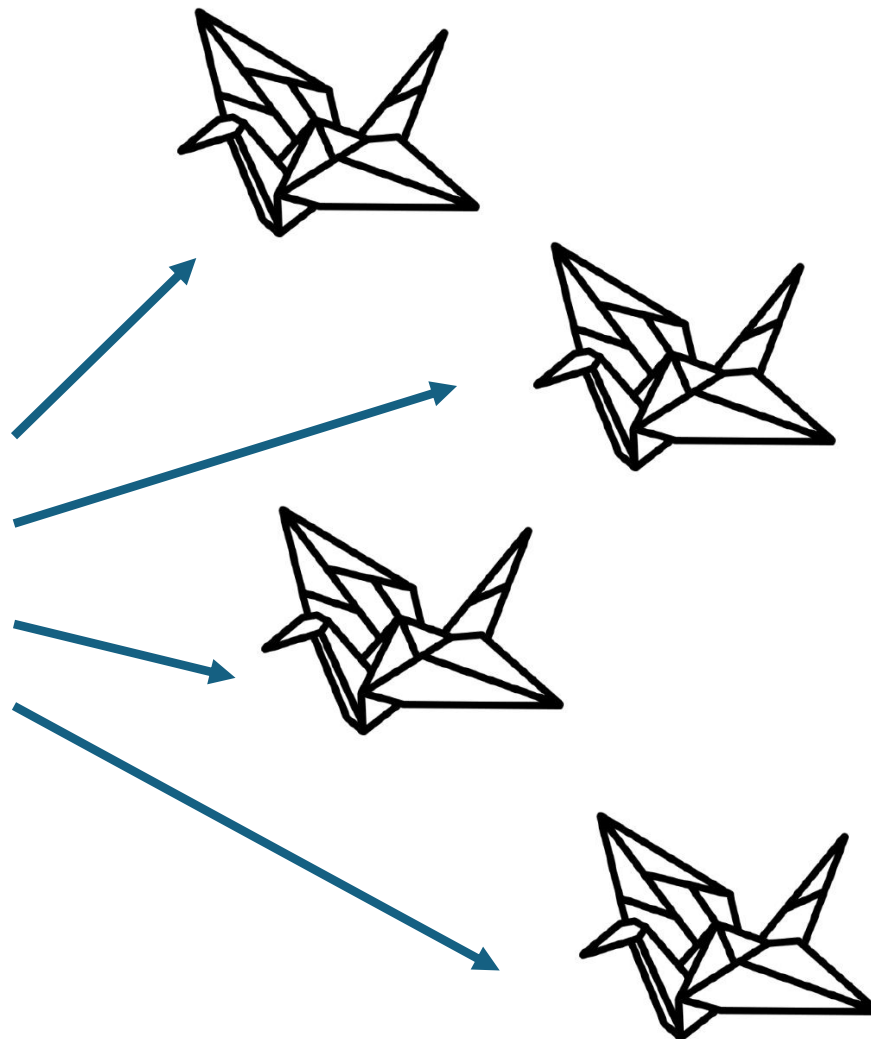
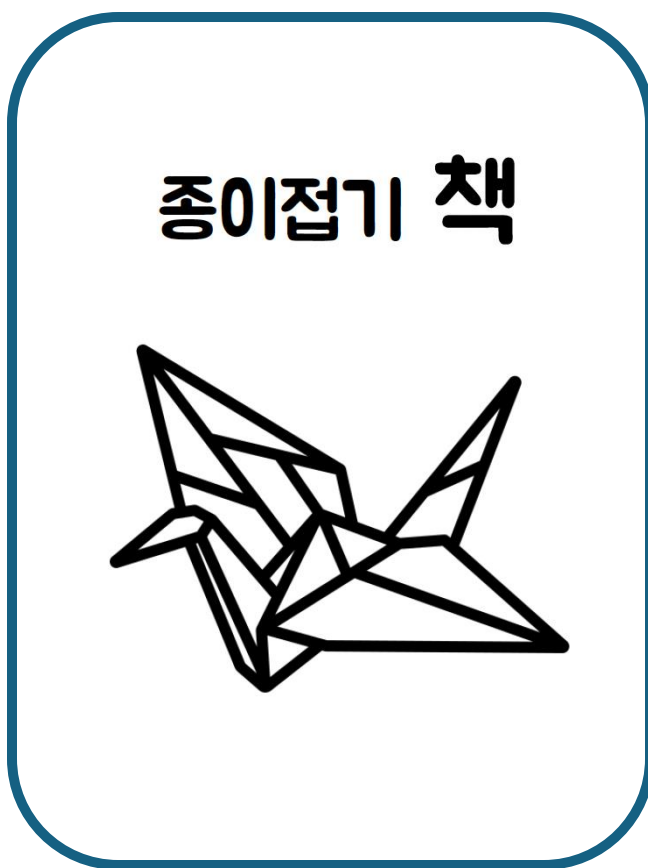


설계도

종이접기 책



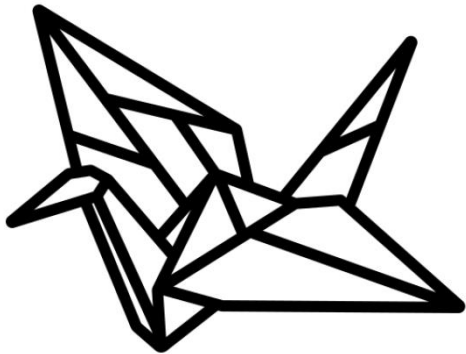
종이를 통해 무언가를 만들어 낼 수 있음



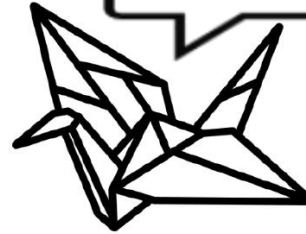
종이(자원)만 있으면 무한으로 생성 가능

부모 클래스라고
부르렴

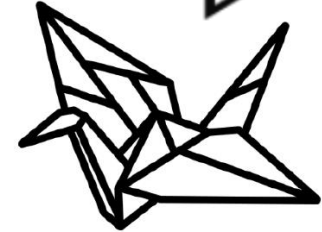
종이접기 책



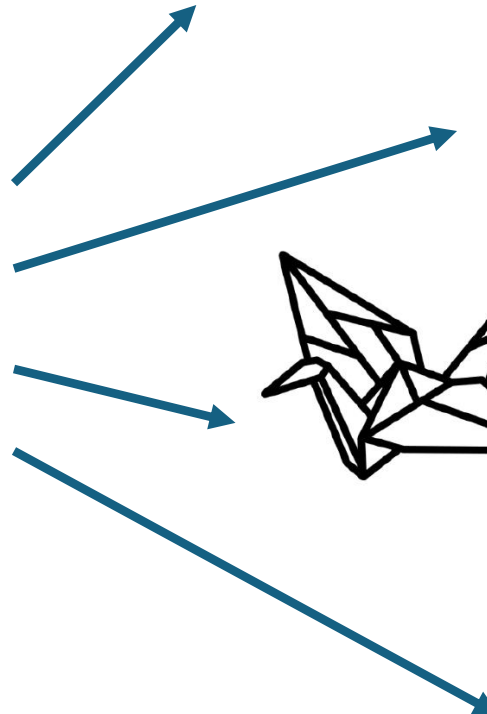
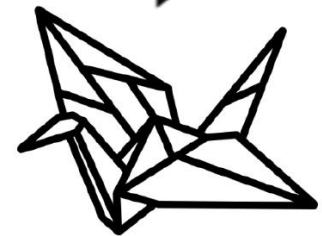
엄마..?



우리는 객체



당신의
인스턴스



클래스

```
class 클래스명:  
    def __init__(self, 전달값1, ...):  
        수행 기능  
  
    def 기능1(self, 전달값, ...):  
        수행기능
```

```
class User:  
    def __init__(self, name):  
        self.name = name  
    def get_name(self):  
        return self.name
```

클래스

```
class User:
    def __init__(self, name):
        self.name = name
    def get_name(self):
        return f"{self.name} 입니다!"
```

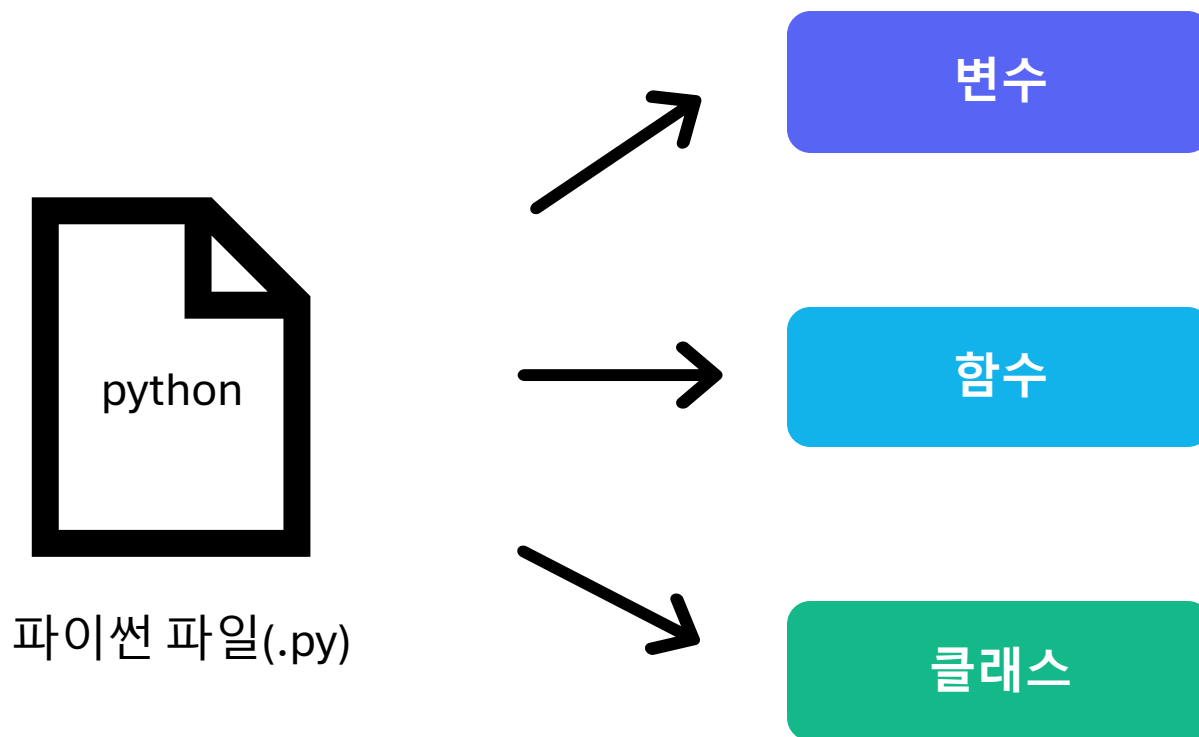
```
user = User(name = '지피티')
print(user.name)
print(user.get_name())
```

```
> 지피티
```

```
> 지피티 입니다!
```

모듈

하나의 파이썬 파일(.py)



- 1) **import** 모듈
- 2) **from** 모듈 **import** 변수, 함수, 클래스

신규파일

```
import stock  
stock_code = stock.종.알.함()  
print(stock_code)
```

> 종목코드는 005293입니다.

```
from stock import 주.알.함  
주.알.함(stock_code)
```

> 주가는 80,000원 입니다.

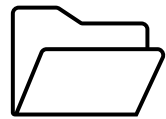
stock.py

주가를 알려주는
함수

종목코드를
알려주는 함수

패키지

비슷한 모듈의 집합(폴더)



패키지



모듈1

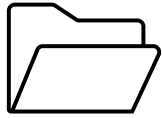
하나의 파이썬 파일(.py)

모듈2

하나의 파이썬 파일(.py)

모듈3

하나의 파이썬 파일(.py)



주가

stock.py

주가를 알려주는
함수

종목코드를
알려주는 함수

chart.py

차트를 그려주는
함수

신규파일

```
from 주가 import stock  
stock_code = stock.종.알.함()  
print(stock_code)
```

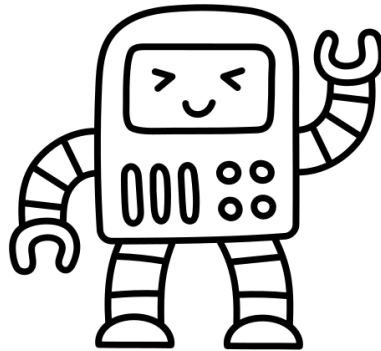
> 종목코드는 005293입니다.

```
import 주가.chart  
chart.차.그.함(stock_code)
```

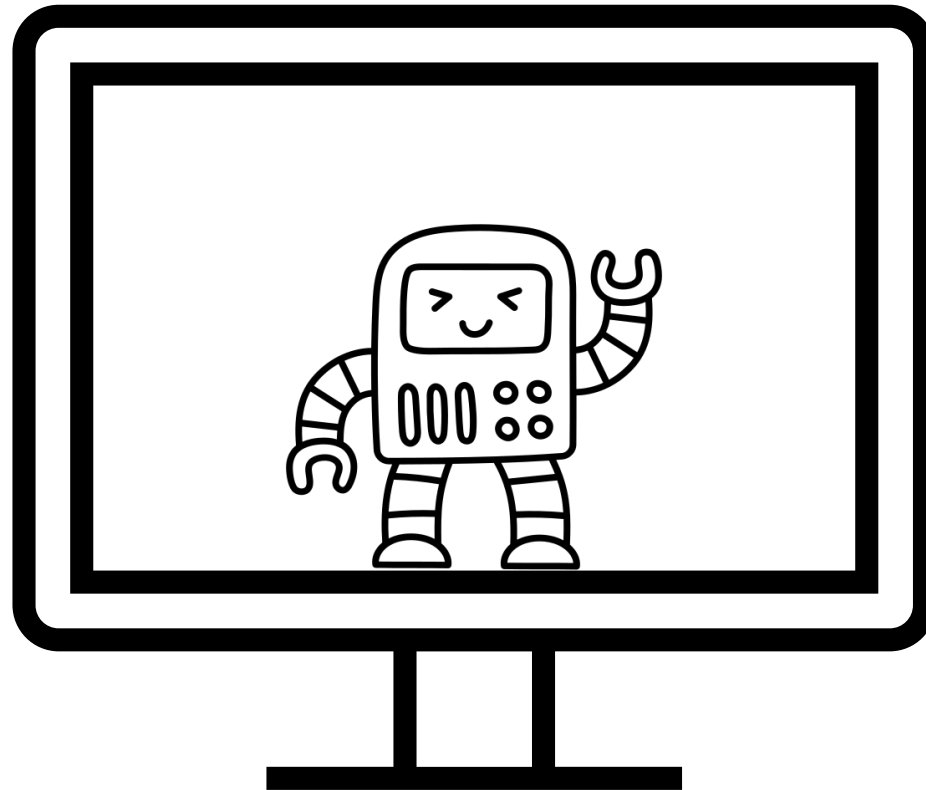
**주가****stock.py**주가를 알려주는
함수종목코드를
알려주는 함수**chart.py**차트를 그려주는
함수

실행환경

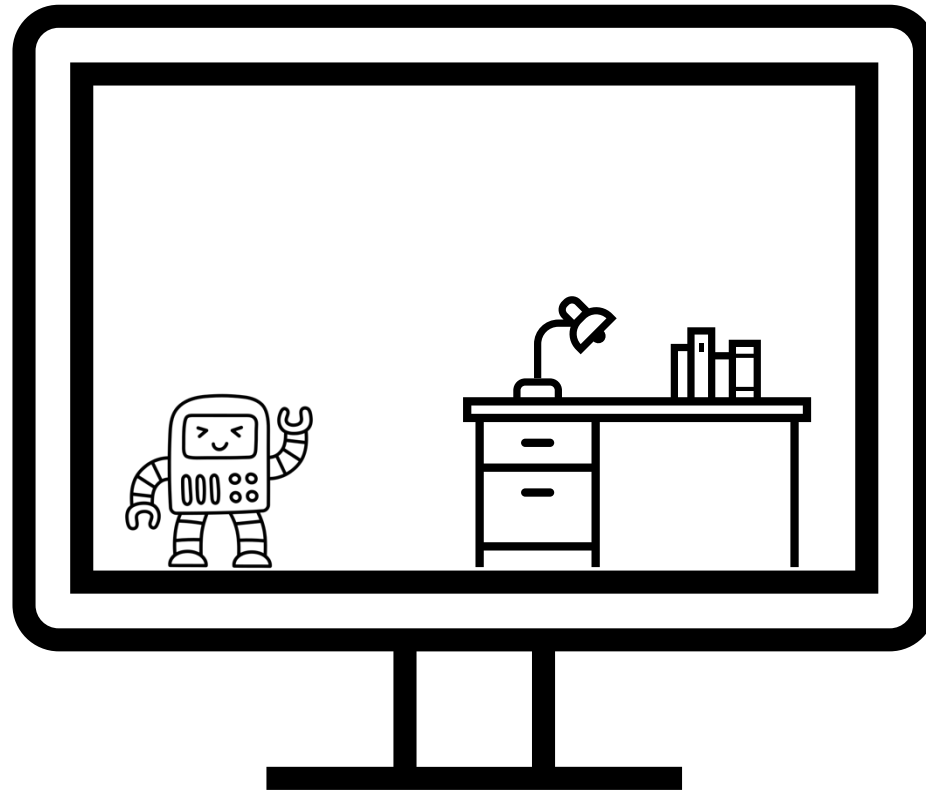
Python environments



Python



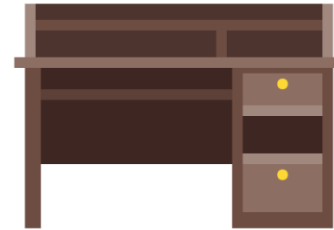
Computer



Environment

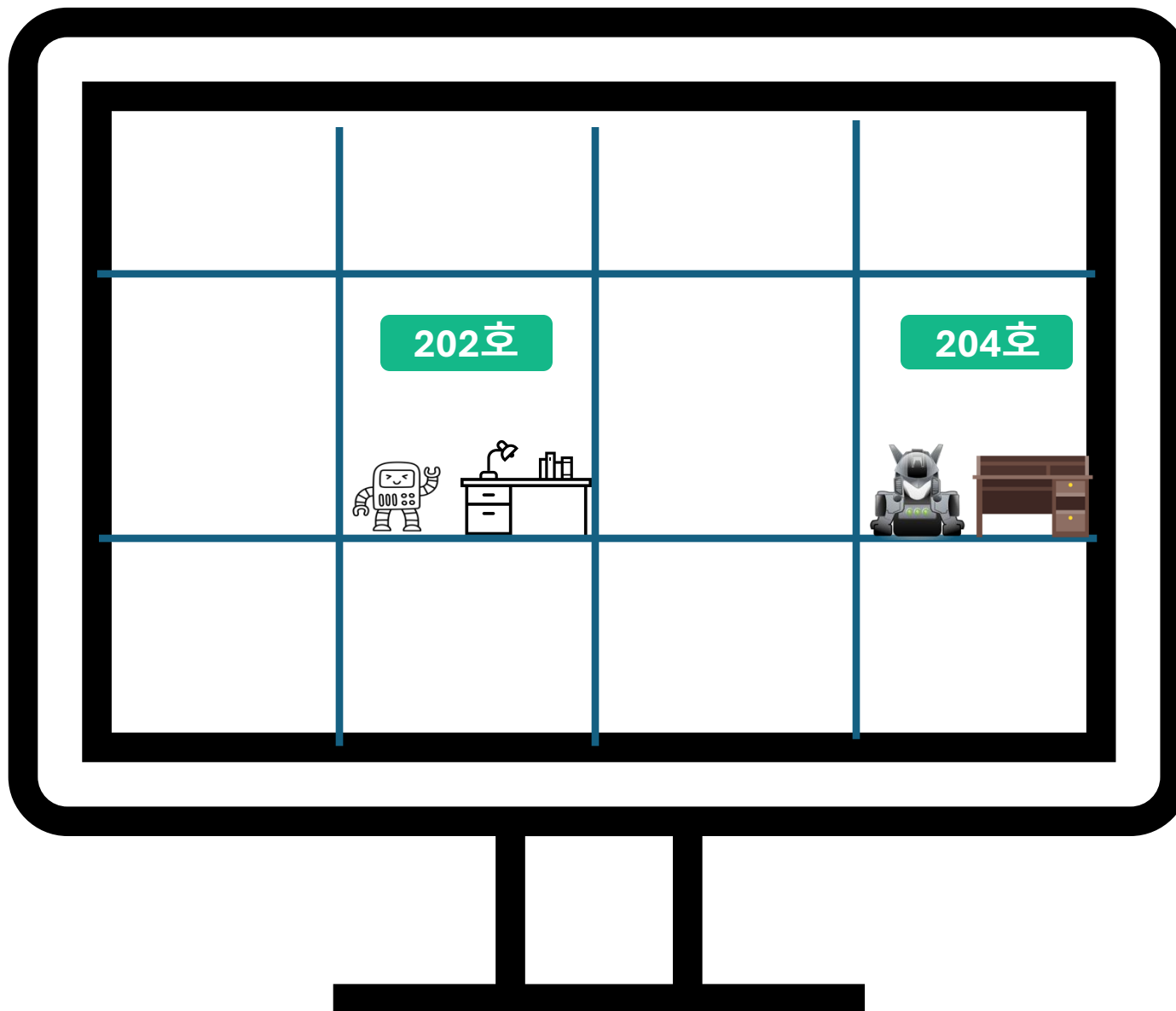


New python



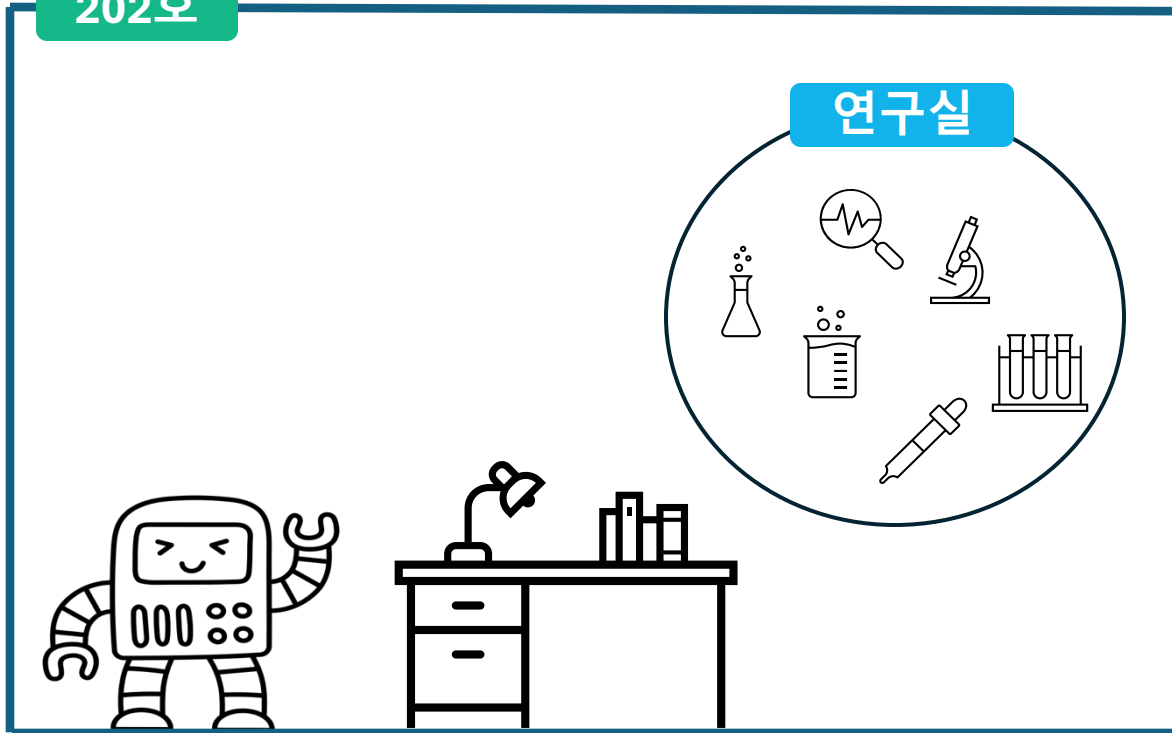
**New
Environment**

Apartment



202호

연구실



Jupyter Lab

202호

연구실

실험 결과는
3입니다!!!

1+1이 뭔가요?

Interaction

