Python 중간고사

자기 주도 학습

이름	김주희
학번	20190709
학과	컴퓨터정보공학과
과목명	Python

강의계획서

동양미래대학교 DONGYANG MIRAE UNIVERSITY

강 의 계 획 서

아시아 직업교육 허브대학

2020 학년도 1학기	전공	컴퓨터정보공학과	학부	컴퓨터공학부
과 목 명	파이썬프로그래밍(2019009-PD)			

강의실 과 강의시간	수:6(3-217),7(3-217),8(3-217)	학점	3
교과분류	이론/실습	시수	3

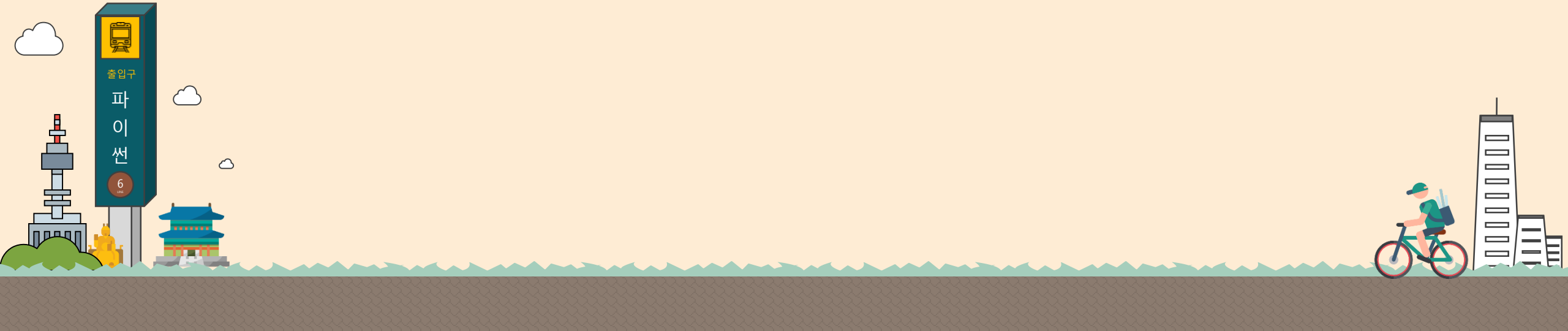
담당 교수	강환수 + 연구실 : 2호관-706 + 전 화 : 02-2610-1941 + E-MAIL : hskang@dongyang.ac.kr + 면담가능기간 : 화요일 13~16

학과 교육목표	
과목 개요	2010년 이후 파이썬의 폭발적인 인기는 제4차 산업혁명 시대의 도래와도 밀접한 연관성이 있다. 컴퓨팅 사고력은 누구나가 가져야할 역량이며, 인공지능, 빅데이터, 사물인터넷 등의 첨단 정보기술이 제4차 산업혁명 시대의 기술을 이끌고 있다. 제4차 산업혁명 시대를 주도하는 핵심 기술은 데이터과학과 머신러닝, 딥러닝이며, 이러한 분야에 적합한 언어인 파이썬은 매우중요한 언어가 되었다. 본 교과목은 파이썬 프로그래밍의 기초적이고 체계적인 학습을 수행한다. 본 교과목을 통하여 데이터 처리 방법에 대한 효율적인 파이썬 프로그래밍 방법을 학습한다.
학습목표 및 성취수준	1. 컴퓨팅 사고력의 중요성을 인지하고 4차 산업혁명에서 파이썬 언어의 필요성을 이해할 수 있다. 2. 기본적인 파이썬 문법을 이해하고 데이터 처리를 위한 자료구조를 이해하여 적용할 수 있다. 3. 문제 해결 방법을 위한 알고리즘을 이해하고 데이터 처리에 적용 할 수 있다. 4. 파이썬 프로그램을 이용하여 실무적인 코딩 작업을 할 수 있다.

	도서명	저자	출판사	비고
주교재	파이썬으로 배우는 누구나 코딩	강환수, 신용현	홍릉과학출판사	

수업시 사용도구	파이썬 기본 도구, 파이참, 아나콘다와 주피터 노트북
평가방법	중간고사 30%, 기말고사 40%, 과제물 및 퀴즈 10% 출석 20%(학교 규정, 학업성적 처리 지침에 따름)
수강안내	1. 파이썬의 개발환경을 설치하고 활용할 수 있다. 2. 파이썬의 기본 자료형을 이해하고 조건과 반복 구문을 활용할 수 있다. 3. 파이썬의 주요 자료인 리스트, 튜플, 딕셔너리, 집합을 활용할 수 있다. 4. 파이썬의 표준 라이브러리와 외부 라이브러리를 이해하고 활용할 수 있다. 5. 파이썬으로 객체지향 프로그래밍을 수행할 수 있다.

Contents



Chapter01 – 파이썬 언어란?

▶ 파이썬은 배우기 쉬운 프로그래밍 언어

- 누구나 무료로 사용할 수 있는 오픈 소스 프로그래밍 언어다.
- 1991년 네덜란드의 귀도 반 로섬이 개발하였다.
- 현재는 비영리 단체인 파이썬 소프트웨어 재단이 관리한다.

▶ 현재 파이썬의 인기는 최고다.

- 비전공자 컴퓨팅 사고력을 키우기 위한 프로그래밍 언어로도 많이 활용되고 있다.
- 배우기 쉽고 간결하며, 개발 속도가 빠르고 강력하다.
- 파이썬은 프로그래밍 교육 분야에서뿐 아니라 실무에서도 사용이 급증하고 있다.

Chapter01 – 컴퓨팅 사고력과 파이썬

▶ 지금은 지능 . 정보화 혁명 시대인 제4차 산업혁명 시대

- 제 4차 산업혁명은 모든 사물이 연결된 초연결 사회에서 생산되는 빅데이터를 기존 산업과 융합해 인공 지능, 클라우드 등 첨단 기술로 처리하는 정보 . 지능화 혁명 시대이다.

▶ 제 4차 산업혁명 시대 인재의 핵심역량은 문제 해결 능력과 창의 . 융합 사고 능력

- 제 4차 산업혁명을 이끌 인재가 갖춰야 할 덕목으로는 문제 해결 능력, 창의 . 융합 사고 능력, 의사소통 능력과 협업 능력, 자기 주도 학습 능력을 들 수 있다.

▶ 문제 해결 능력과 창의 . 융합 사고 능력에 필요한 컴퓨팅 사고력

- 미래 인재는 컴퓨터 과학 원리와 개념을 활용해 자신의 영역과 융합할 수 있는 역량을 갖춰야 한다. 이러한 능력은 컴퓨팅 사고력이라 한다.
- 컴퓨팅 사고력은 컴퓨터 분야의 문제 해결은 물론 일상생활에서의 문제 해결에 효율적으로 사용될 수 있는 방법을 제공하고, 창의성을 높이는 데에도 도움이 된다.
- 컴퓨팅 사고력 교육은 제 4차 산업혁명 시대의 인재 교육에서 가장 중요하다.

▶ 컴퓨팅 사고력 구성 요소

- 분해(decomposion) : 데이터, 프로세스 또는 문제를 작고 관리 가능한 부분으로 나눔
- 패턴인식(pattern recongnition) : 데이터의 패턴, 추세 및 정규성 관찰
- 추상화(abstraction) : 패턴을 생성하는 일반 원칙을 규정
- 알고리즘 설계(algorithm design) : 이 문제와 유사한 문제 해결을 위한 단계별 지침을 개발

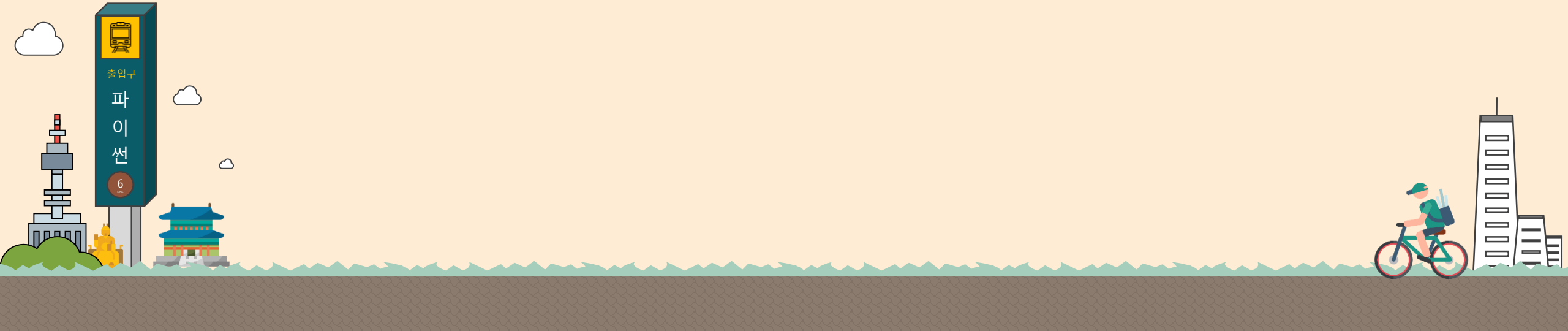
Chapter01 – 컴퓨팅 사고력과 파이썬

▶ 컴퓨팅 사고력 향상에 필요한 코딩 교육

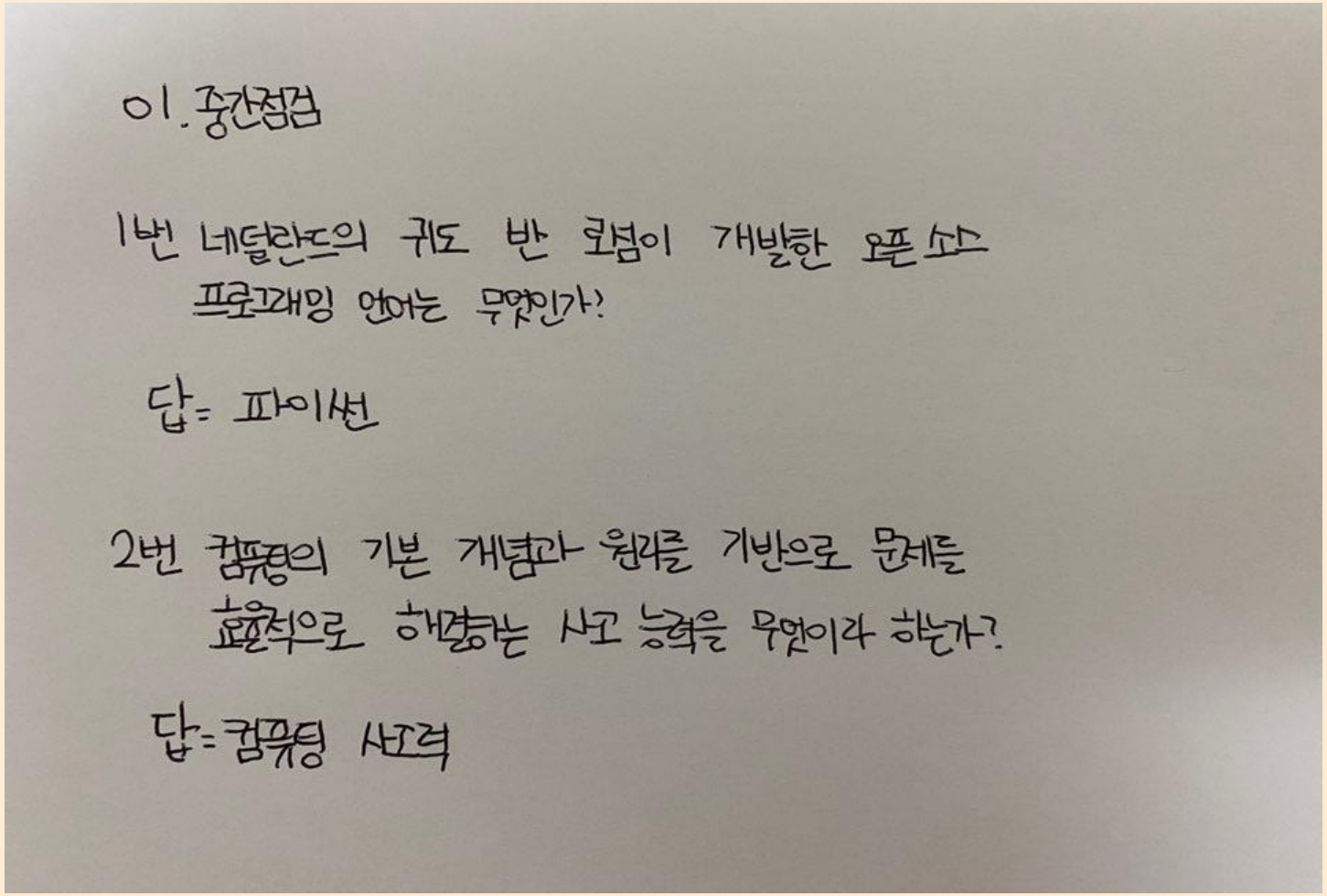
- 컴퓨팅 사고력은 간단하게 '컴퓨팅의 기본 개념과 원리를 기반으로 문제를 효율적으로 해결하는 사고 능력 ' 이라고 한다.
- 컴퓨팅 사고력을 향상시키는 방법 중 하나는 직접 프로그래밍하는 코딩 교육이다.

▶ 프로그래밍의 절차 : 이해, 설계, 구현, 공유

- 비전공자에겐 컴퓨팅 사고력을 교육하는 데 적합하다.
- 이해 : 주어진 문제를 이해하고 파악
- 이해(분할) : 문제를 좀 더 쉬운 작은 문제들로 분해
- 설계 (패턴인식) : 분해된 문제들 사이의 유사성 또는 패턴을 탐색
- 설계(추상화) : 문제에서 불필요한 부분은 제거하고 주요 핵심 요소를 추려 내는 과정
- 설계(알고리즘) : 프로그래밍 언어인 파이썬에 맞는 입력과 출력, 변수 저장 그리고 절차에 따라 구성
- 구현 : 문제 해결을 위해 파이썬으로 코드 개발, 작성된 코드의 실행과 테스팅, 디버깅 과정을 거쳐서 코드를 수정하고 필요하면 다시 설계
- 공유 : 자신이 구현한 프로그램을 발표하고 다른 학습자의 프로그램과 비교, 현재의 기능을 향상시키는 방향으로 프로그램 개선 연구, 교수자의 피드백 및 평가

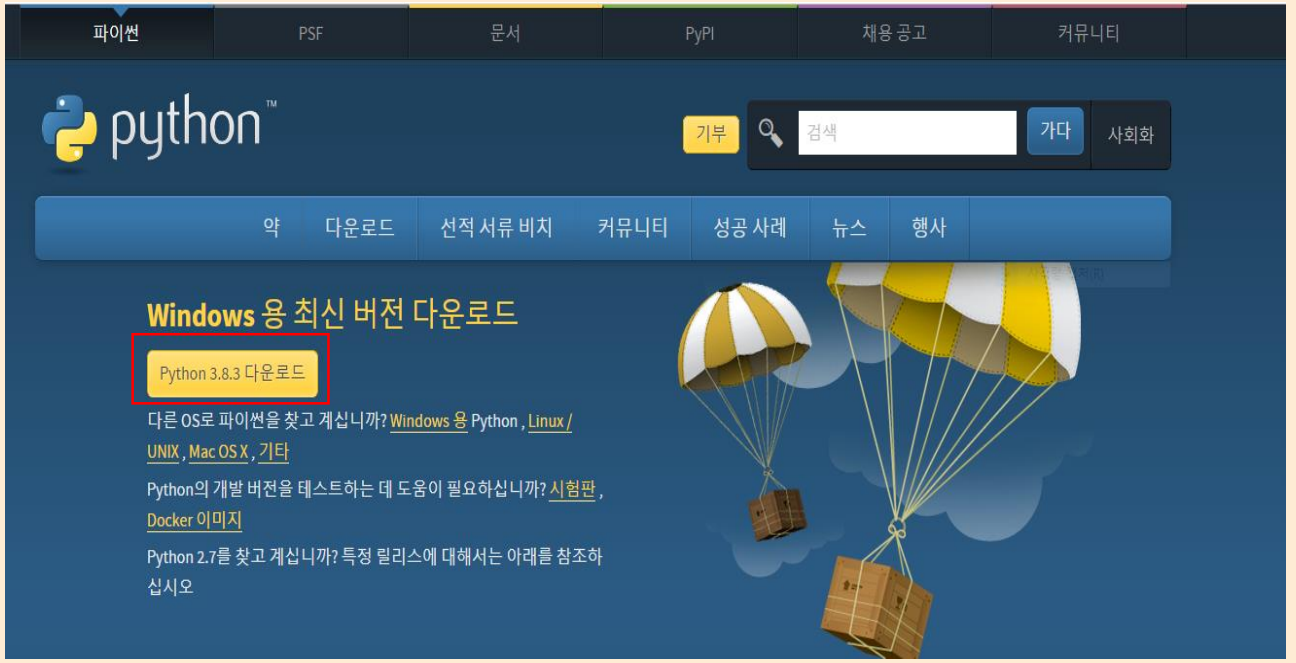


Chapter01 – 중간점검



Chapter01+02 – 파이썬 설치와 파이썬 쉘 실행

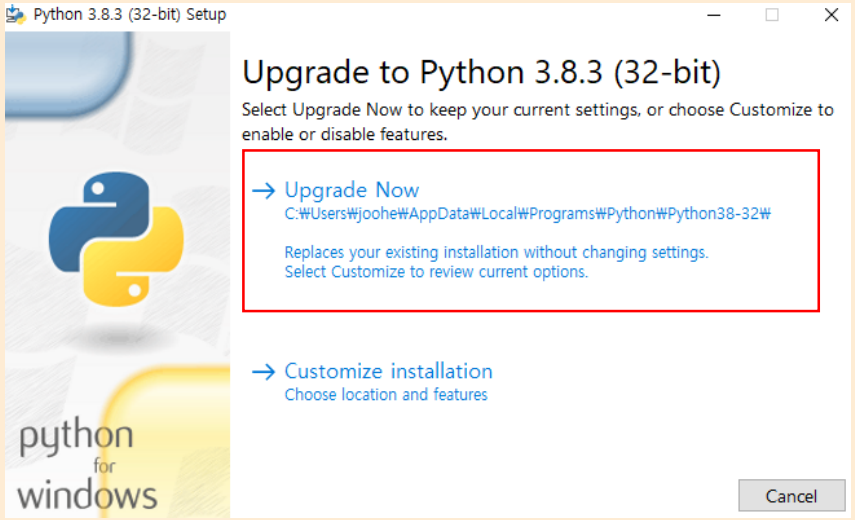
▶ 파이썬 설치

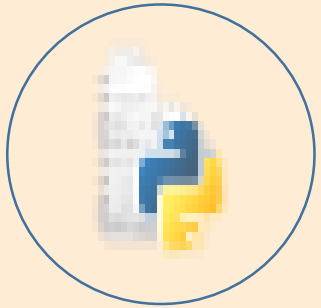


www.python.org로 들어가 다운로드를 누르게 되면 2번째 사진처럼 나온다.
다음에는 Python3.8.3다운로드를 눌러 다운을 받으면 된다.

Chapter01+02 – 파이썬 설치와 파이썬 쉘 실행

▶ 파이썬 설치 및 사용 방법

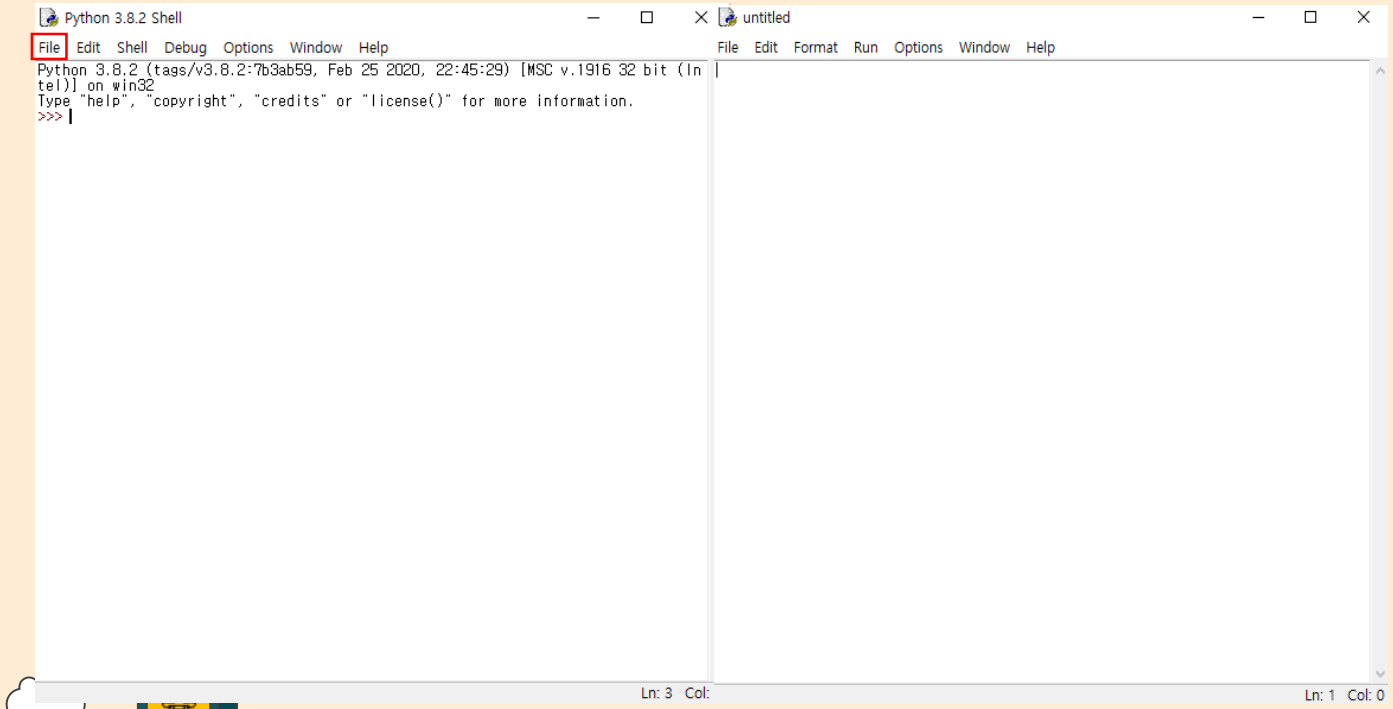




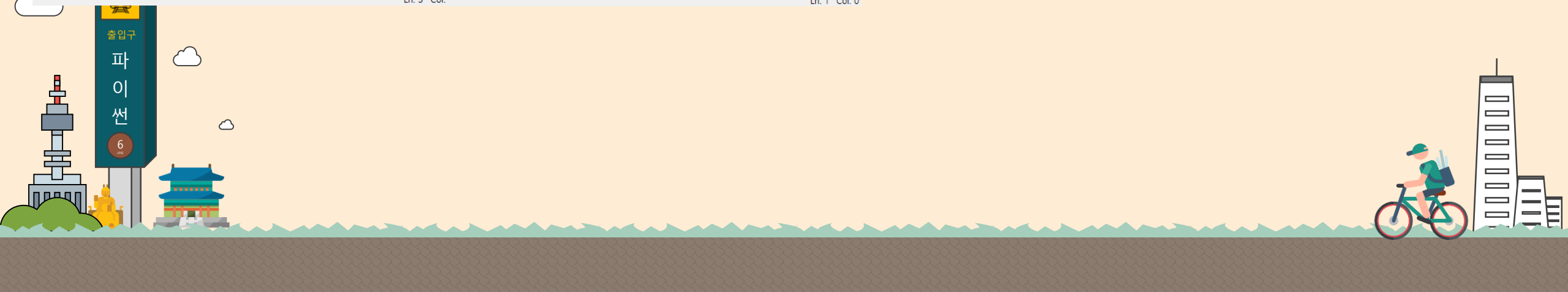
처음 사진을 누르면 다운이 받아진다.
다운이 다 받아지면 Python – IDLE을
누르면 실행이 된다.

Chapter01+02 – 파이썬 설치와 파이썬 쉘 실행

▶ 파이썬 사용 방법

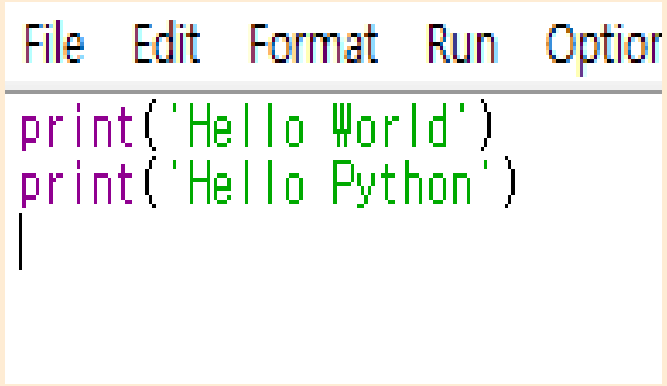


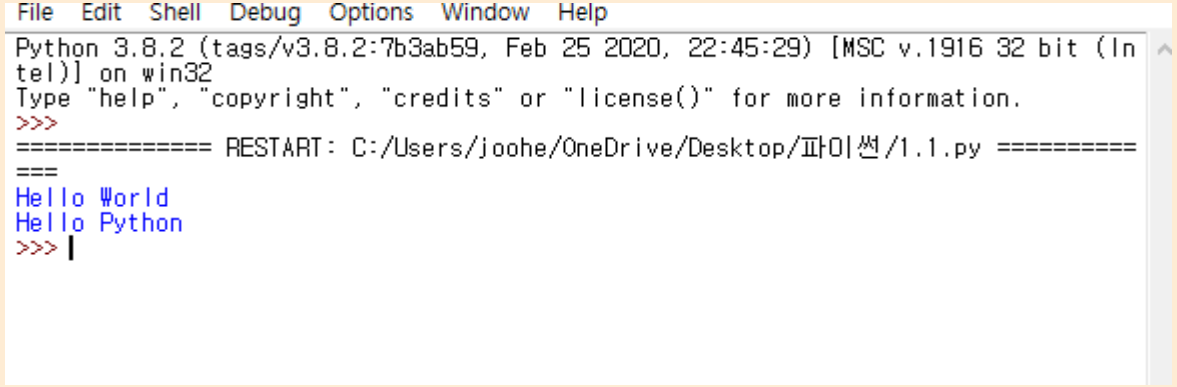
- 첫번째 사진에서 파일을 누르고 New File을 누른다.
- 그럼 새로운 창이 뜨는데 그 창에 파이썬을 코딩하면 된다.
- 꼭 코딩을 다 하고 나면 파일을 누르고 Save as 눌러 저장을 한다.
- 저장은 (01.py) 이런 식으로 저장하면 되고, 결과를 보기 위해서는 Run을 눌러 Run Module 누르면 첫번째 사진에 결과가 나온다.



Chapter01+02 – 파이썬 설치와 파이썬 쉘 실행

▶ 파이썬 코딩





- 첫번째 사진은 코딩을 한 사진이고,
- 두번째 사진은 코딩 한 결과가 나온 사진이다.
- 두번째 사진에서 >>>에 코딩할 수 있지만 중요한 점은 >>>
다음 첫 칸을 공간을 두고 코딩을 하게 되면 오류가 난다.

Chapter01+02 – 중간점검

02. 중간점검

1번 글자 '파이썬'을 출력하는 프로그램을 작성하세요.

답: Print('파이썬')

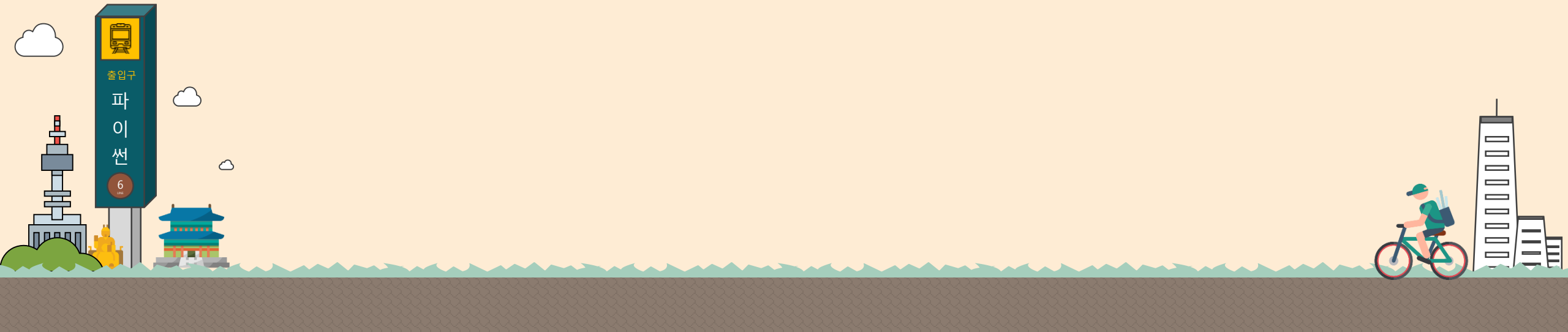
2번 두줄에 글자 '파이썬 개발 도구'와 '파이썬 쉘 IDLE'을 출력하는 프로그램을 작성하세요.

답: Print('파이썬 개발 도구')
Print('파이썬 쉘 IDLE')

Chapter01+03 – 제4차 산업혁명 시대, 모두에게 필요한 파이썬

▶ 쉽고 강력한 언어

- **간결하고 배우기 쉽다**
+ 파이썬은 C나 자바 언어에 비해 문법이 쉽고, 표현 구조가 인간의 사고 체계와 닮아 있어 사용하기가 쉽다.
+ 간결해 초보자도 쉽게 배울 수 있다.
- **무료이며, 다양한 자료 구조의 제공으로 생산성이 높다.**
+ 코드의 양이 줄어들고, 소스의 개발과 테스트도 빨라 실제 프로젝트 수행 시 생산성이 높다.
- **라이브러리가 독보적이고, 강력하며, 데이터과학과 머신 러닝 등과 같은 다양한 분야에 적용할 수 있다.**
+ 파이썬의 기본 철학인 '건전지 포함'에서 알 수 있는 것처럼 다양한 라이브러리를 쉽게 사용할 수 있도록 표준 라이브러리를 제공한다.
+다양한 파이썬의 라이브러리 저장소인 파이썬 패키지 색인을 제공해 파이썬 커뮤니티에서 공유하는 강력한 라이브러리를 활용할 수 있다.
+ 2019년 3월 기준으로 120만 개의 다양한 기능을 가진 패키지 버전을 제공하고 있다.
- **다른 모듈을 연결해 사용할 수 있는 풀 언어로도 자주 활용된다.**
+ 다른 언어로 개발하거나 이미 개발돼 쓰이고 있는 모듈들을 연결하는 풀 언어로도 자주 활용된다.



Chapter01+03 – 제4차 산업혁명 시대, 모두에게 필요한 파이썬

▶ 빅데이터 처리와 머신 러닝 등 다양한 분야에 적합한 언어

- 교육과 학술, 실무 등 다양한 분야에 사용

+ 파이썬의 외부에 풍부한 라이브러리가 있어 다양한 용도로 확장하기 좋기 때문이다.
+ 대학 프로그래밍 교양 수업부터 스타트업, 대형 글로벌 기업에 이르기까지 다양한 분야에 활용되고 있다.

- 인공 지능의 구현과 빅데이터 분석 처리 분야에 사용

+넘파이(NumPy), 싸이파이(Scipy), 심파이(SymPy)등을 이용하면 수식을 빠르게 연산할 수 있어서 수학, 과학, 공학 기본으로 이용한다.
+ 판다스(Pandas), 멧플롯리브(Matplotlib), 씨본(Seaborn), 보케(Bokeh) 등은 데이터 처리와 고급 통계 차트를 그리기 위한 통계용 시각화 기능을 제공한다.
+ 싸이킷런(Scikit-Learn), 텐서플로(TensorFlow), 케라스(Keras)는 머신 러닝에서 신경망 모형 등과 같은 딥러닝 모형을 위한 파이썬 패키지다.

▶ 파이썬은 좀 속도가 느리다는 단점

- 대부분의 인터프리터 언어가 그렇듯이 실행속도가 느리다는 것과 모바일 앱 개발 환경에는 아직 사용하기 힘들다는 것을 들 수 있다.

▶ 다양한 종류의 개발 환경

- 기본 IDE에 추가 : 비주얼 스튜디오, 파이썬 도구, PyDev 설치 이클립스
- 파이썬 전용 IDE : PyCharm, Spyder, Jupyter Notebook, Jupyter Lab, Comodo, Wing python IDE, PyScripter, Thonny 등
- 편집기 전문 개발 환경 : Sublime Text, Visual Studio Code, Notepad++, Atom, Vim 등

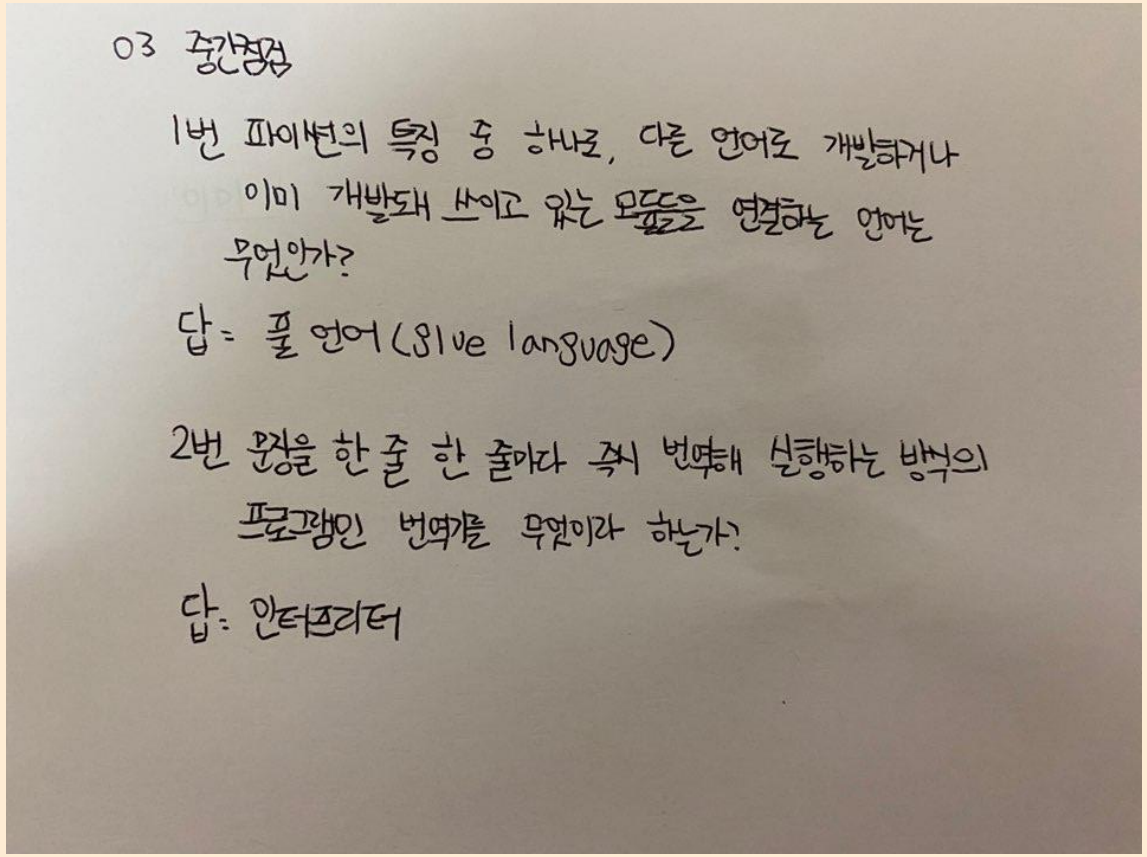
▶ 인터프리터 방식의 언어, 파이썬

- 인터프리터 방식이란 ?

동시 통역처럼 파이썬의 문장을 한 줄 한 줄마다 즉시 번역해 실행하는 방식이다.

- 인터프리터는 한 줄 한 줄의 해석을 담당하고 컴파일러는 컴파일을 담당하는 개발 도구 소프트웨어다.

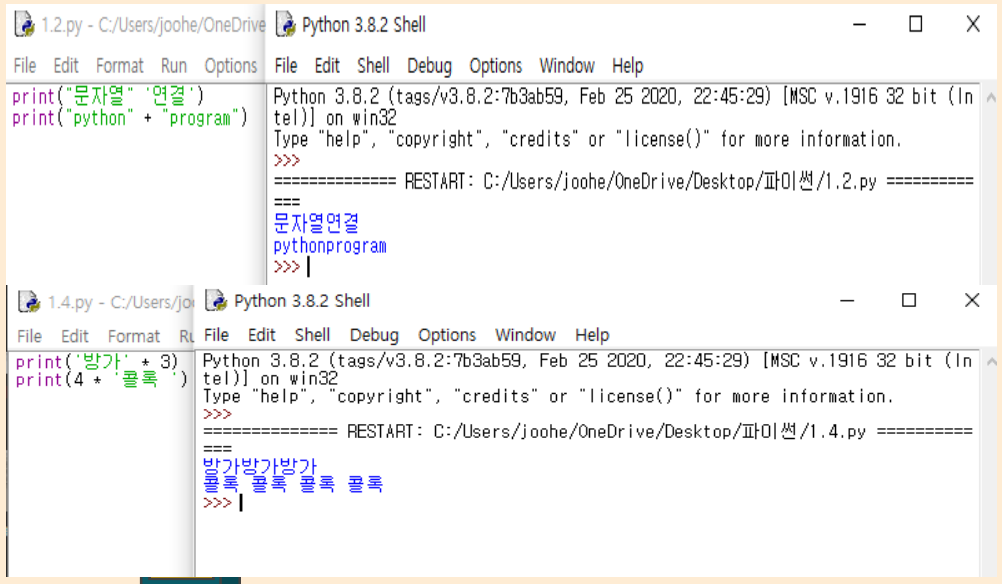
Chapter01+03 – 중간점검



Chapter02 – 다양한 자료 : 문자열과 수

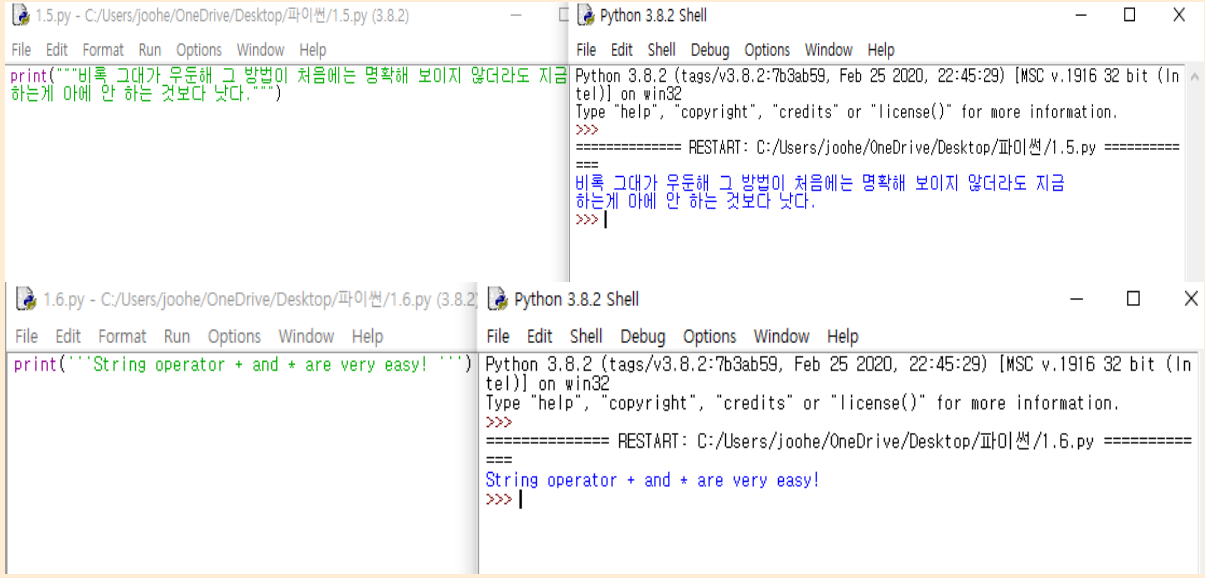
▶ 문자열의 연결연산자 +와 반복 연산자*

- 컴퓨팅 사고력은 간단하게 '컴퓨팅의 기본 개념과 원리를 기반으로 문제를 효율적으로 해결하는 사고 능력 ' 이라고 한다.
- 컴퓨팅 사고력을 향상시키는 방법 중 하나는 직접 프로그래밍하는 코딩 교육이다.
- 정수가 아닌 'str' 유형으로 시퀀스를 곱할 수 없다.



▶ 여러 줄의 문자열의 처리에 사용되는 삼중 따옴표

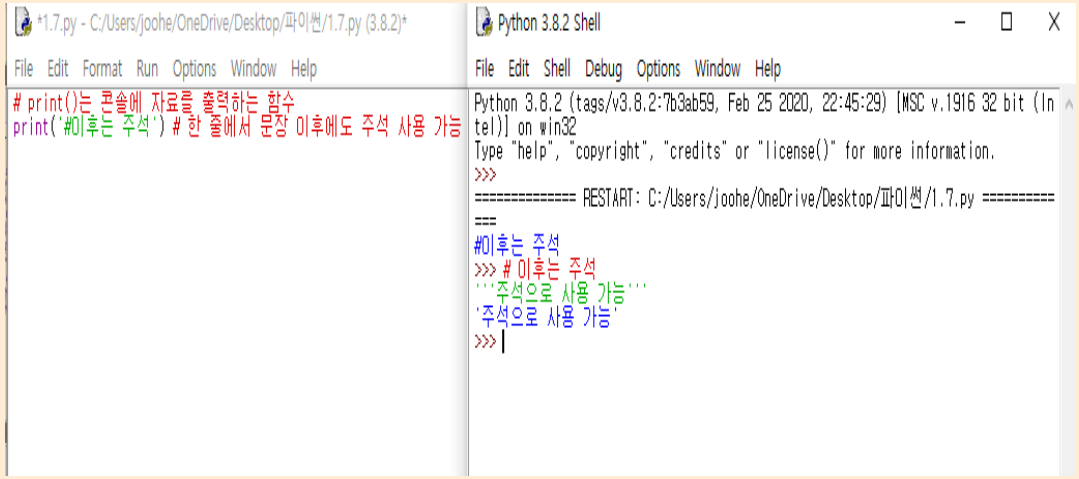
- 문자열이 길거나 필요에 의해 여러 줄에 걸쳐 문자열을 처리하기 위해서는 삼중 따옴표를 사용한다.
- 삼중 따옴표란, 작은따옴표나 큰따옴표를 연속적으로 3개씩 문자열 앞 뒤에 둘러싸는 것을 말한다.



Chapter02 – 다양한 자료 : 문자열과 수

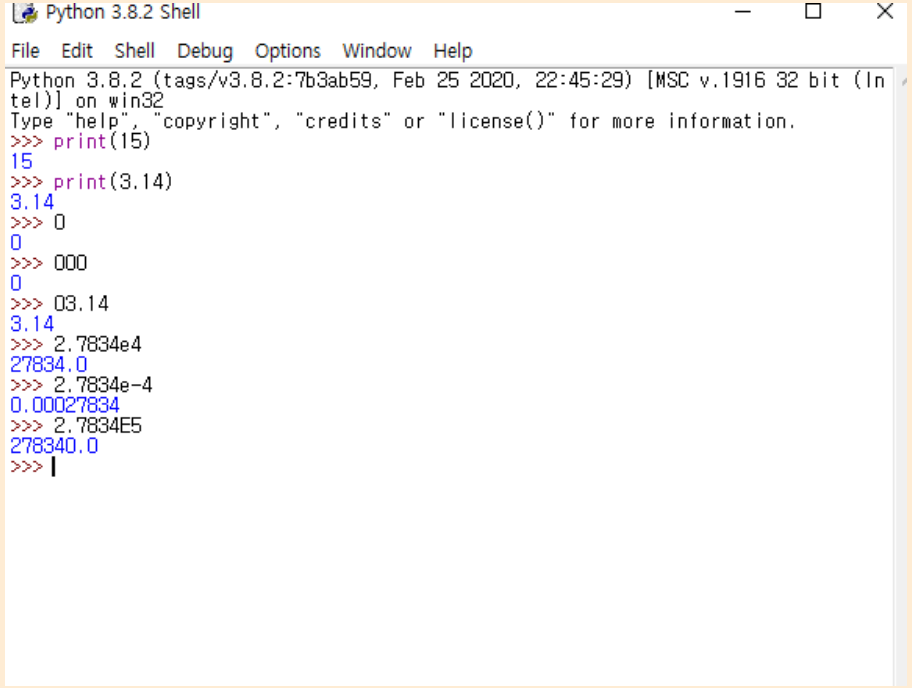
▶ 문법과 상관 없는 주석

- 주석은 소스 설명으로, 인터프리터는 주석을 무시한다.
- 파이썬 주석은 #으로 시작하고 그 줄의 끝까지 유효하다.



▶ 수학에서 사용하던 정수와 실수

- 숫자는 간단히 정수화 실수로 나눈다.
- 15, 7, 20 등은 정수, 소수점이 있는 3.14, 2.718 등은 실수다.



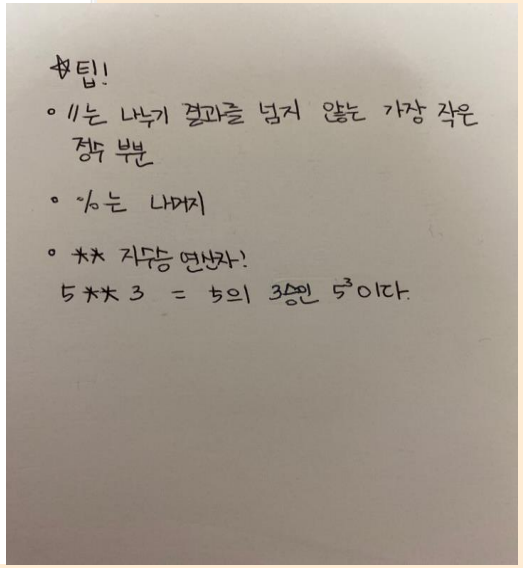
Chapter02 – 다양한 자료 : 문자열과 수

▶ 수의 더하기, 빼기, 곱하기, 나누기, 연산자

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 5 + 3
8
>>> 5 - 3
2
>>> 5 * 3
15
>>> 5 / 3
1.6666666666666667
>>>
```

▶ 수의 몫, 나머지, 지수승 연산자

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 8 / 5
1.6
>>> 8 // 5
1
>>> 2.8 / 1.2
2.3333333333333335
>>> 2.8 // 1.2
2.0
>>> -1.5 / 0.2
-7.5
>>> -1.5 // 0.2
-8.0
>>> 5 // 3
1
>>> 5 % 3
2
>>> 5 ** 3
125
>>>
```



2 ** 2 ** 3의 계산 순서는 오른쪽에서 왼쪽으로 계산한다.
나머지 연산자는 왼쪽에서 오른쪽

Chapter02 – 다양한 자료 : 문자열과 수

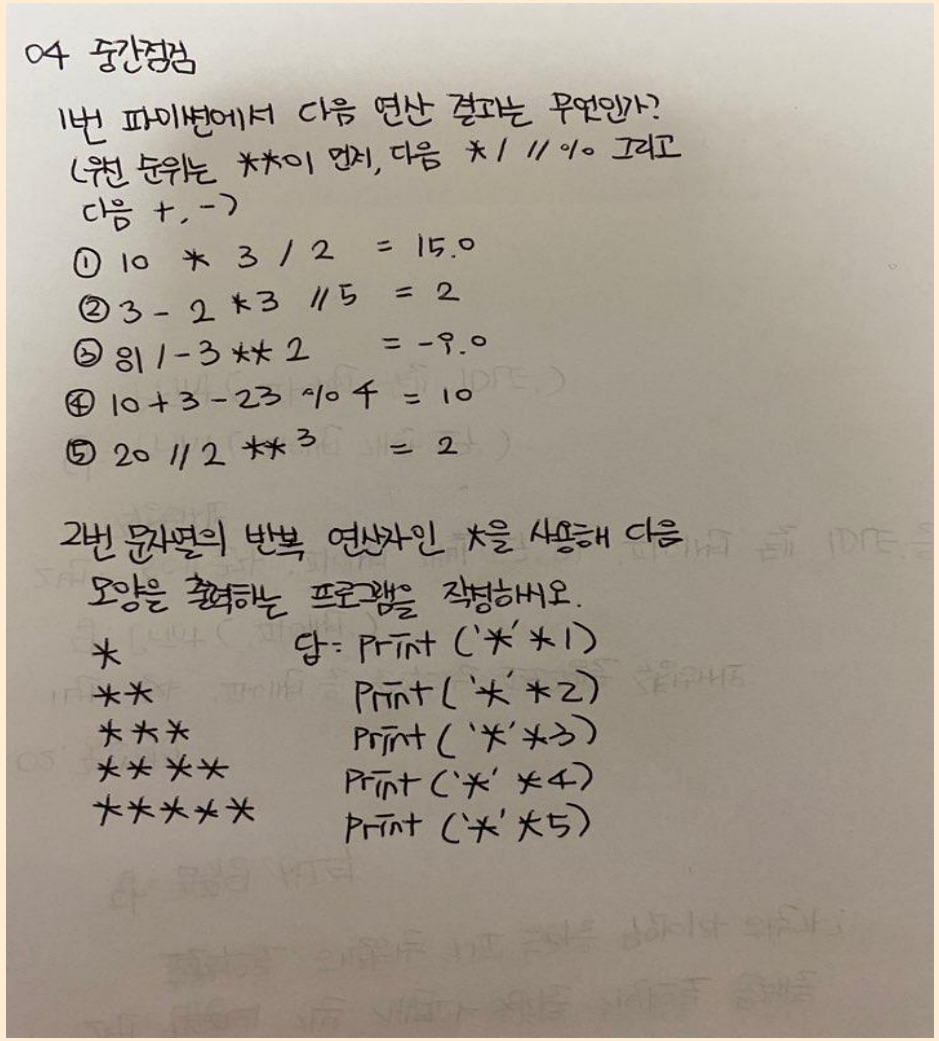
▶ 언더 스코어(_)

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 60
60
>>> _
60
>>> 3*_
180
>>> _
180
>>> '파'
'파'
>>> _
'파'
>>> '파이' + '썬' * 3
'파이썬썬썬'
>>> _
'파이썬썬썬'
>>>
```

▶ 표현식 문자열 실행 함수 eval()

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> eval('3 + 15 / 2')
10.5
>>> eval('4 + 3 % 5')
2
>>> eval('"java " * 3')
'java java java '
>>>
```

Chapter02 – 중간점검



04 중간점검

1번 파이썬에서 다음 연산 결과는 무엇인가?
(우선 순위는 **이 먼저, 다음 * / // % 그리고 다음 +, -)

① 10 * 3 / 2 = 15.0
② 3 - 2 * 3 // 5 = 2
③ 의 1 - 3 ** 2 = -8.0
④ 10 + 3 - 23 % 4 = 10
⑤ 20 // 2 ** 3 = 2

2번 문자열의 반복 연산자인 *을 사용해 다음 모양을 출력하는 프로그램을 작성하세요.

*
**
***
****
*****

답: print('*' * 1)
print('*' * 2)
print('*' * 3)
print('*' * 4)
print('*' * 5)

Chapter02+02 – 변수와 키워드, 대입 연산자

▶ 자료형과 type() 함수

```
File  Edit  Shell  Debug  Options  Window  Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> type(3)
<class 'int'>
>>> pi = 3.14
>>> type(pi)
<class 'float'>
>>> type('python')
<class 'str'>
>>> type(3 + 4j)
<class 'complex'>
>>>
```

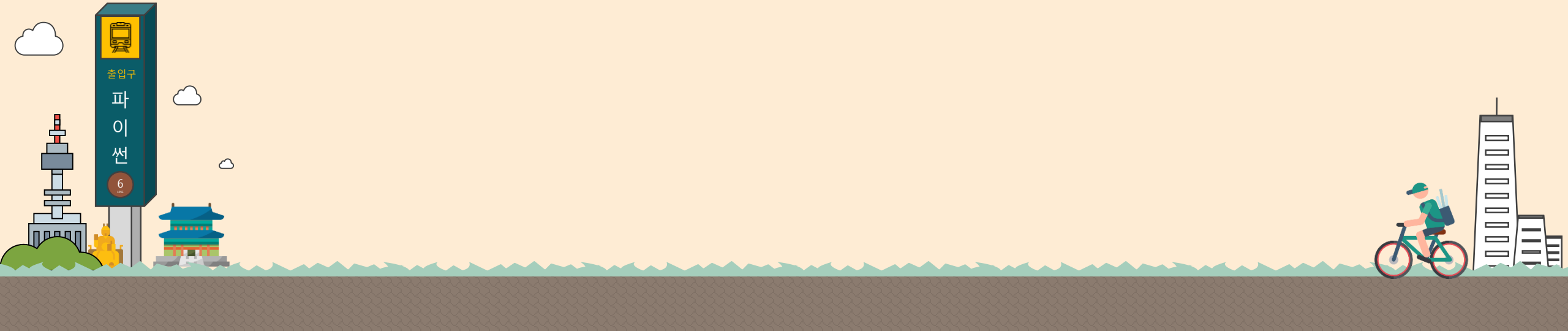
▶ 변수의 이해와 대입 연산자 =을 이용한 값의 저장

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> a = b = c = 5
>>> print(a)
5
>>> print(b)
5
>>> print(c)
5
>>> b = 3
>>> print(b)
3
>>>
```

Chapter02+02 – 변수와 키워드, 대입 연산자

- **프로그래밍 언어가 이미 정해놓은 단어, 키워드**

False	None	True	and	as	assert	break	class	continue
def	del	elif	Else	except	Finally	for	From	global
If	import	In	Is	Lambda	Nonlocal	Not	or	Pass
raise	return	try	while	With	yield			



Chapter02+02 – 변수와 키워드, 대입 연산자

▶ 변수 이름을 붙일 때의 규칙

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> value = 10
>>> Value = 200
>>> value
10
>>> Value
200
>>> total_price = 100000
>>> coffeePrice = 4500
>>>
>>> _month = 11
>>> 2020year = 2020
SyntaxError: invalid syntax
>>> sale@ = 20 # 오류 : 문자 @ 사용 불가
SyntaxError: invalid syntax
>>> #오류 : 공백 문자 사용 불가
>>> sale price = 16000
SyntaxError: invalid syntax
>>> import = 30 # 오류 : 키워드 사용 불가
SyntaxError: invalid syntax
>>>
```

Chapter02+02 – 변수와 키워드, 대입 연산자

Value	0	Value와는 구분
Total_price	0	여러 단어로 구성된 변수를 읽기 쉽게 하기 위해 중간에 _ 삽입
coffeePrice	0	여러 단어로 구성된 변수를 읽기 쉽게 하기 위해 중간에 _ 삽입
_month	0	맨 앞에 _ 도 가능하지만 특수한 사용이 아니라면 가능한 사용하지 말도록 권고
2020year	X	숫자가 가장 앞에 올 수 없음
sale@	X	영문자 이외의 문자는 올 수 없음
Sale price	X	공백 문자는 사용 불가
import	X	키워드도 사용 불가

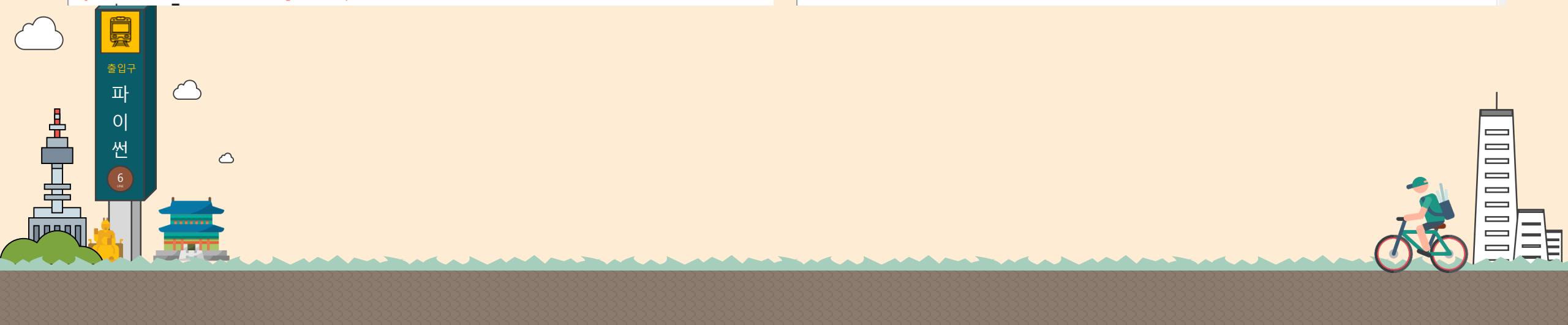
Chapter02+02 – 변수와 키워드, 대입 연산자

▶ 동일한 변수에 값을 수정하는 다양한 대입 연산자

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> data = 7
>>> data += 1
>>> print(data)
8
>>> data += 5
>>> print(data)
13
>>> var = 5
>>> var *= 10 // 3 % 5
>>> print(var)
15
```

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> value = 1
>>> value = value + 1 # value 값을 1증가
>>> print(value)
2
>>> value + 1 = value #오류
SyntaxError: cannot assign to operator
```

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> var = 5
>>> var += 2
>>> print(var)
7
>>> var -= 3
>>> print(var)
4
>>> var *= 5
>>> print(var)
20
>>> var //= 3
>>> print(var)
6
>>> var **= 2
>>> print(var)
36
>>> var %= 7
>>> print(var)
1
>>> var /= 5
>>> print(var)
0.2
>>>
```



Chapter02+02 변수와 키워드, 대입 연산자

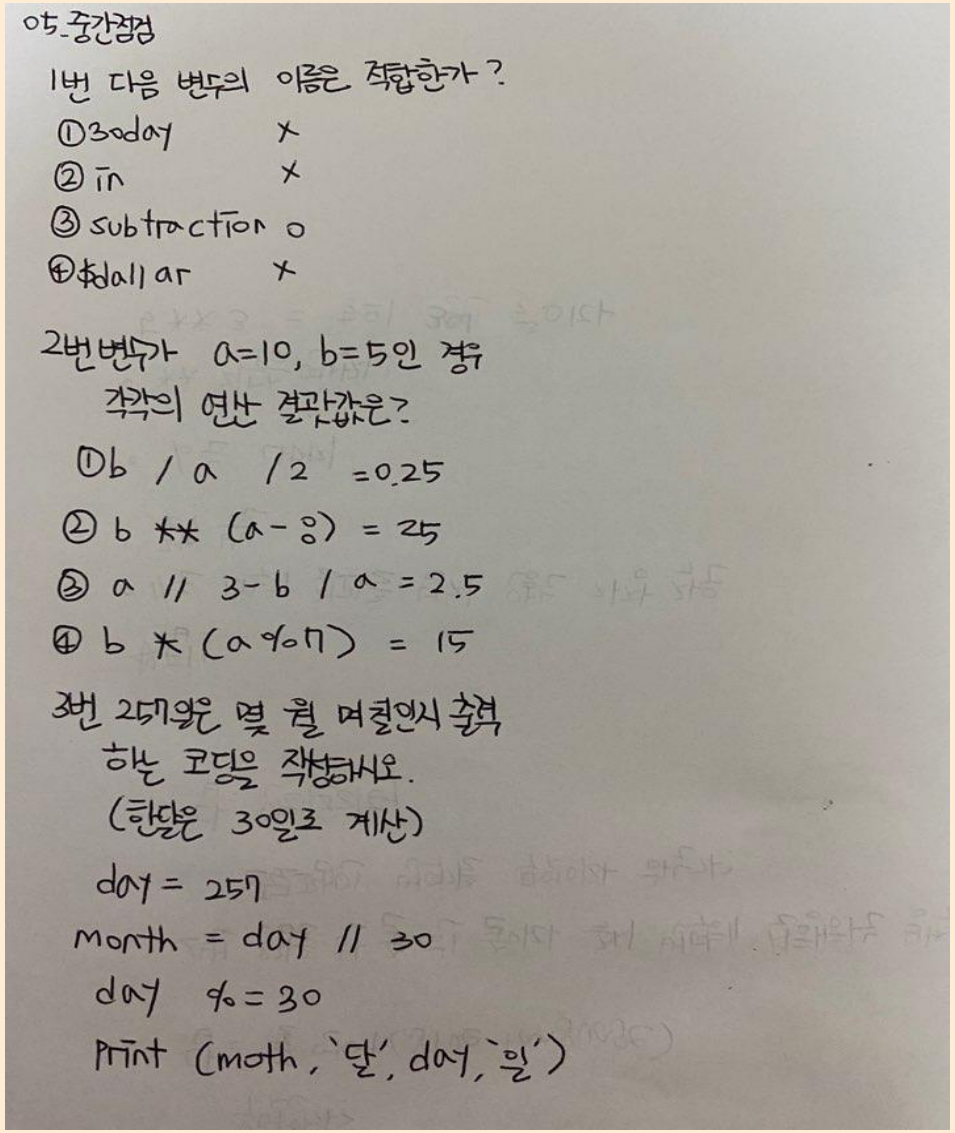
▶ 한 번에 여러 자료 대입

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> a, b = 5, 9
>>> print(a, b)
5 9
>>> a, b = 5, 9
>>> temp = a
>>> a = b
>>> b = temp
>>> print(a, b)
9 5
>>>
```

▶ divmod = 몫 연산과 나머지 연산을 함께 수행

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 28 // 3, 28 % 3
(9, 1)
>>> divmod(28, 3)
(9, 1)
>>>
```

Chapter02+02 – 중간점검



05_중간점검

1번 다음 변수의 이름은 적합한가?

① 3oday x

② in x

③ subtraction o

④ $dollar x

2번변수가 a=10, b=5인 경우
각각의 연산 결과값은?

① b / a / 2 =0.25

② b ** (a-3) = 25

③ a // 3-b / a = 2.5

④ b * (a%7) = 15

3번 25가 몇 월 며칠인지 출력
하는 코딩을 작성해보세요.
(한달은 30일로 계산)

```
day = 257
month = day // 30
day %= 30
print (month, '달', day, '일')
```

Chapter02+03 – 자료의 표준 입력과 자료 변환 함수

▶ 함수 input()으로 문자열 표준 입력

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> input()
java
'java'
>>> pl = input()
python
>>> print(pl)
python
>>>
```

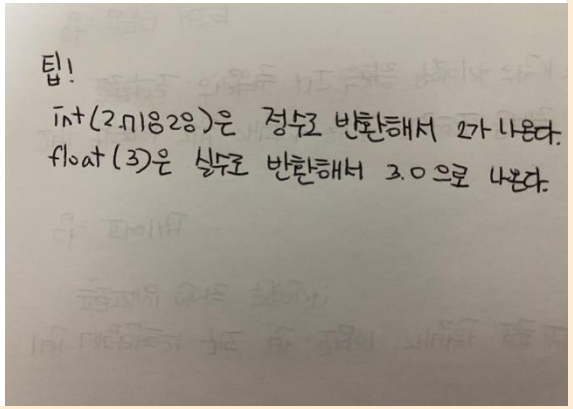
```
File  Edit  Format  Run  Options  Window  H
univ = input('대학은? ')
name = input('이름은? ')
print('대학 : ',univ, '이름 : ', name)
```

```
File  Edit  Shell  Debug  Options  Window  Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:/Users/joohe/OneDrive/Desktop/파이썬/1.8.py ==========
===
대학은? 동양미래
이름은? 김주희
대학 :  동양미래 이름 :  김주희
>>>
```

Chapter02+03 – 자료의 표준 입력과 자료 변환 함수

▶ 문자열과 정수, 실수 간의 자료 변환 함수 str(), int(), float()

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> str(235)
'235'
>>> str(2.71828)
'2.71828'
>>> int('6400')
6400
>>> float('3.141592')
3.141592
>>> int(2.71828)
2
>>> float(3)
3.0
>>>
```



▶ 숫자 형태의 문자열을 정수나 실수로 변환

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> rate = input('예금 만기 이율(%)은? ')
예금 만기 이율(%)은? 4.2
>>> float(rate) / 100
0.042
>>>
```

Chapter02+03 – 자료의 표준 입력과 자료 변환 함수

▶ 16진수, 10진수, 8진수, 2진수와 활용

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 10
10
>>> 0x1f, 0X1E, 0Xa #16진법 표현
(31, 30, 10)
>>> 0o17, 0O16, 0O12 #8진법 표현
(15, 14, 10)
>>> 0b11, 0B10, 0B1010 #2진법 표현
(3, 2, 10)
>>>
```

▶10진수의 변환 함수 bin(), oct(), hex()

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> bin(7), bin(16), bin(12)
('0b111', '0b10000', '0b1100')
>>> oct(8), oct(10), oct(12)
('0o10', '0o12', '0o14')
>>> hex(14), hex(15), hex(16)
('0xe', '0xf', '0x10')
>>>
```

Chapter02+03 – 자료의 표준 입력과 자료 변환 함수

▶ 16진수, 10진수, 8진수, 2진수와 활용

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 10
10
>>> 0x1f, 0X1E, 0Xa #16진법 표현
(31, 30, 10)
>>> 0o17, 0O16, 0O12 #8진법 표현
(15, 14, 10)
>>> 0b11, 0B10, 0B1010 #2진법 표현
(3, 2, 10)
>>>
```

▶10진수의 변환 함수 bin(), oct(), hex()

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> bin(7), bin(16), bin(12)
('0b111', '0b10000', '0b1100')
>>> oct(8), oct(10), oct(12)
('0o10', '0o12', '0o14')
>>> hex(14), hex(15), hex(16)
('0xe', '0xf', '0x10')
>>>
```

Chapter02+03 – 중간점검

06. 중간점검

1번 다음문자열을 10진수로 변환하는 문장을 작성해요.

① '156' = int('156')

② '0b101' = int('0b101',2)

③ '0011' = int('0011',8)

④ '0xf' = int('0xf',16)

2번 다음과 같이 표준입력으로 이름을 입력받아 출력하는 코드를 작성해요.

당신의 이름은? 홍길동

만나서 반가워요, 홍길동씨!

```
name = input("당신의 이름은?")
print("만나서 반가워요, " + name + "씨!")
```

Chapter03 – 문자열 다루기

▶ 함수 len()으로 문자열의 길이 참조

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> a = 'python'
>>> len(a)
6
>>> len('파이썬')
3
>>>
```

▶ 문자열의 문자 참조

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 'python'[0]
'p'
>>> 'python'[3]
'h'
>>> 'python'[-1]
'n'
>>> 'python'[len('python')-1]
'n'
>>> 'python'[-4]
't'
>>>
```

오름차순 첨자 : 0~ [len('python')-1]
내림차순 첨자 : [-len('python')]~-1

0	1	2	3	4	5
p	y	t	h	o	n
-6	-5	-4	-3	-2	-1

Chapter03 – 문자열 다루기

▶ 0과 양수를 이용한 문자열 슬라이싱

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 'python'[1:5]
'ytho'
>>> 'python'[2:4]
'th'
>>> 'python'[0:3]
'pyt'
>>> 'python'[0:6]
'python'
>>> 'python'[0:len('python')]
'python'
>>>
```

▶ 음수를 이용한 문자열 슬라이싱

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 'python' [-5:-1]
'ytho'
>>> 'python'[-4:-1]
'tho'
>>> 'python'[-6:-3]
'pyt'
>>> 'python'[-6:-1]
'pytho'
>>> 'python'[-len('python'):-1]
'pytho'
>>> 'python'[1:-1]
'ytho'
>>> 'python'[0:-2]
'pyth'
>>> 'python'[5:-1]
''
>>>
```

음수와 양수를 사용한 것과,
문자열이 없어 빈 문자열을 반환 하는 것

Chapter03 – 문자열 다루기

▶ start와 end를 비우면 '처음부터' 와 '끝까지'를 의미

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 'python'[:3]
'pyt'
>>> 'python'[1:]
'ython'
>>> 'python'[:] #전체 반환
'python'
>>>
```

▶ str[start:end:step]으로 문자 사이의 간격을 step으로 조절 가능

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 'python'[::1]
'python'
>>> 'python'[1:5:2]
'yh'
>>> 'python'[::-1] #역순으로 반환
'nohtyp'
>>> 'python'[5:0:-1]
'nohty'
>>>
```

Chapter03 – 문자열 다루기

▶ 문자 함수 ord()와 chr()

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ord('가')
44032
>>> chr(44032)
'가'
>>> hex(ord('가'))
'0xac00'
>>> hex(ord('힣'))
'0xd7a3'
>>>
```

▶ 이스케이프 시퀀스 문자

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> '₩₩'
'₩₩'
>>> print('₩₩')
₩
>>> '₩''
"'"
>>> '₩"'
'"'

>>>
```

Chapter03 – 문자열 다루기

▶ 이스케이프 시퀀스 문자 표

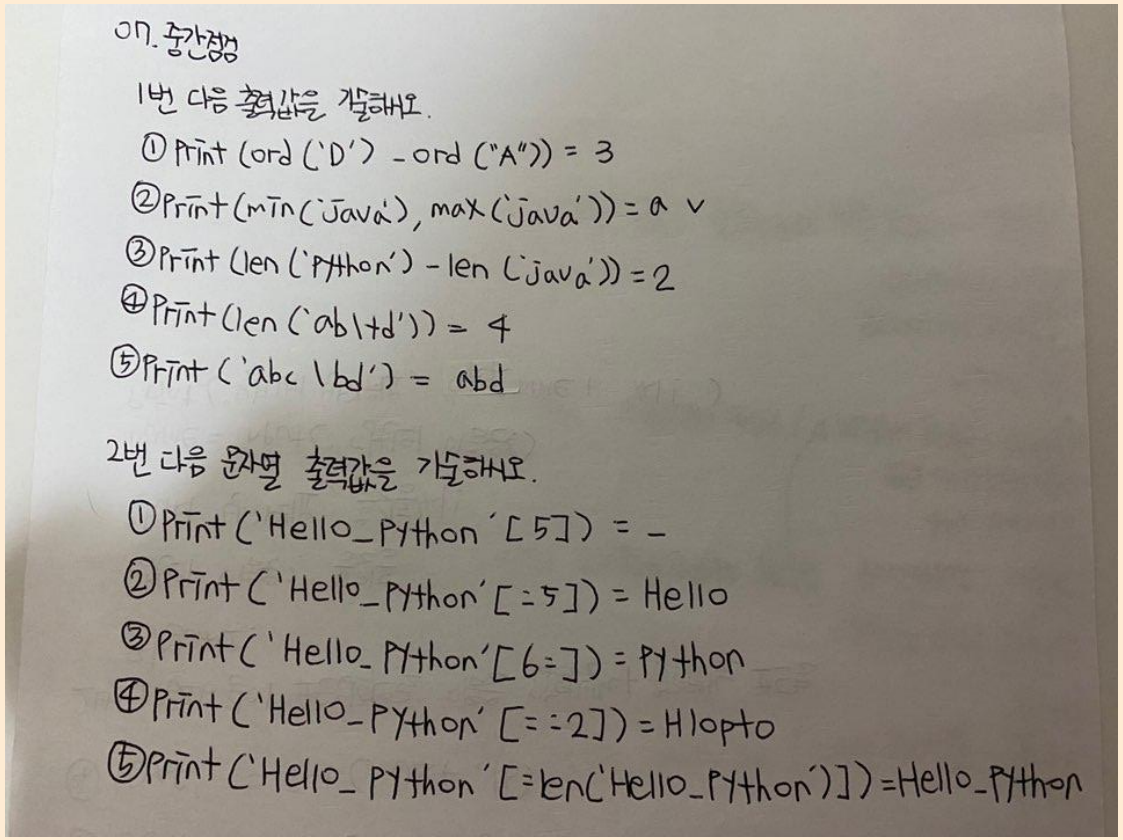
\\	역슬래시	\N{name}	유니코드 이름
*	작은따옴표	\r	동일한 줄의 맨 앞으로 이동
\"	큰따옴표	\f	폼피드
\a	벨소리(알람)	\t	수평 탭
\b	백스페이스	\v	수직 탭
\n	새 줄	\uxxxx	16비트 16진수 코드

\Uxxxxxxxx	32비트 16진수 코드
\ooo	8진수의 코드 문자
\xhh	16진수의 코드 문자

▶ 문자열의 최대와 최소

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> min('ipython')
'h'
>>> max('ipython')
'y'
>>> min('3259')
'2'
>>> max('3259')
'9'
>>> min('ipython', 'java')
'ipython'
>>> max('ipython', 'java')
'java'
>>>
```

Chapter03 – 중간점검



0기. 중간점검

1번 다음 출력값을 예측해보.

① Print (ord ('D') - ord ("A")) = 3

② Print (min('Java'), max('Java')) = a v

③ Print (len ('Python') - len ('Java')) = 2

④ Print (len ('ab\td')) = 4

⑤ Print ('abc \bd') = abd

2번 다음 문자열 출력값을 예측해보.

① Print ('Hello_Python'[5]) = _

② Print ('Hello_Python'[:5]) = Hello

③ Print ('Hello_Python'[6:]) = Python

④ Print ('Hello_Python'[::2]) = Hlopto

⑤ Print ('Hello_Python'[:len('Hello_Python')]) = Hello_Python

Chapter03+02 – 문자열 관련 메소드

▶ 문자열 바꿔 반환하는 메소드 replace()

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> str = '파이썬 파이썬 파이썬'
>>> str.replace('파이썬', 'Python!')
'Python! Python! Python!'
>>> str.replace('파이썬', 'Python!', 1)
'Python! 파이썬 파이썬'
>>> str.replace('파이썬', 'Python!', 2)
'Python! Python! 파이썬'
>>>
```

▶ 부분 문자열 출현 횟수를 반환하는 메소드 count()

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> str = '단순한 것이 복잡한 것보다 낫다.'
>>> str.count('복잡')
1
>>> str.count('것')
2
>>>
```

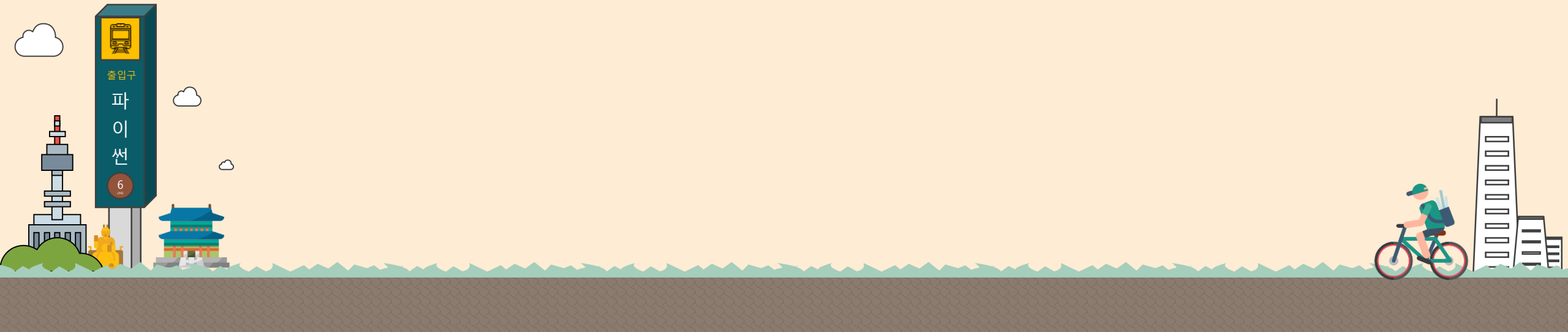
Chapter03+02 – 문자열 관련 메소드

▶ 문자와 문자 사이에 문자열을 삽입하는 메소드 join()

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> num = '12345'
>>> '->'.join(num)
'1->2->3->4->5'
>>>
```

▶ 문자열을 찾는 메소드 find()와 index()

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> str = '자바 C 파이썬 코틀린'
>>> str.find('자바')
0
>>> str.index('자바')
0
>>> str.find('파이')
5
>>> str.index('파이')
5
>>> str.index('파이썬')
5
>>>
```



Chapter03+02 – 문자열 관련 다루기

▶ 문자열을 여러 문자열로 나누는 split()메소드

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> '사과 배 복숭아 딸기 포도'.split()
['사과', '배', '복숭아', '딸기', '포도']
>>> '데스크톱 1000 , 노트북 18000, 스마트폰 120000'.split(',')
['데스크톱 1000 ', ' 노트북 18000', ' 스마트폰 120000']
>>> m,n = '100 200'.split()
>>> m,n
('100', '200')
>>>
```

▶ 영문자 알파벳의 다양한 변환 메소드

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (I
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 'python'.upper() # 모두 대문자로 변환해 변환
'PYTHON'
>>> 'PYTHON'.lower() # 모두 소문자로 변환해 변환
'python'
>>> 'python lecture'.capitalize() # 첫 문자만 대문자로 변환해 변환
'Python lecture'
>>> 'python lecture'.title() # 단어마다 첫 문자를 대문자로 변환해 반환
'Python Lecture'
```

Chapter03+02 – 문자열 관련 다루기

▶ 폭을 지정하고 중앙에 문자열 배치하는 메소드 center()

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ' 파이썬 강좌 '.center(30, '*')
'********** 파이썬 강좌 **********'
>>> ' 파이썬 강좌'.center(30)
'            파이썬 강좌            '
>>> ' 파이썬 강좌'.center(30,'=')
'========== 파이썬 강좌============'
>>>
```

▶ 폭을 지정하고 왼쪽 또는 오른쪽 정렬하는 메소드 Ljust()와 rjust()

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 'python'.ljust(10)
'python    '
>>> 'python'.ljust(10,'*')
'python****'
>>> 'python'.rjust(10)
'    python'
>>> 'python'.rjust(10,'*')
'****python'
>>>
```

Chapter03+02 – 문자열 관련 다루기

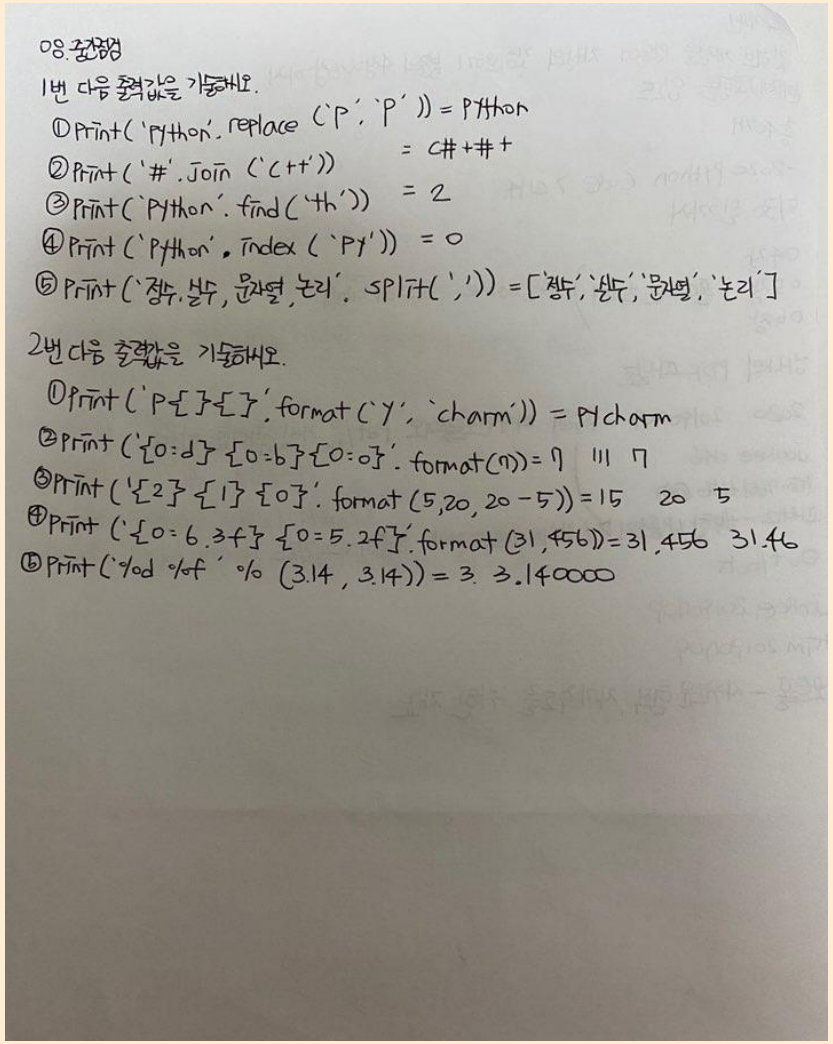
▶ 문자열 앞뒤의 특정 문자들을 제거하는 strip()메소드

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ' python  '.lstrip()
'python  '
>>> '  python  ' .rstrip()
'  python'
>>> ' python '.strip()
'python'
>>> '***python---'.strip('* -')
'python'
>>>
```

▶ 폭을 지정하고 왼쪽 또는 오른쪽 정렬하는 메소드 Ljust()와 rjust()

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> '234'.zfill(5)
'00234'
>>> 'abc'.zfill(5)
'00abc'
>>> '-345'.zfill(8)
'-0000345'
>>>
```

Chapter03+02 – 중간점검



ㅇㅇ.중간점검

1번 다음 출력값을 기술하세요.

① Print('python'.replace('P', 'P')) = Python

② Print('#'.join('C++')) = C#+#+

③ Print('Python'.find('th')) = 2

④ Print('Python'.index('Py')) = 0

⑤ Print('정두,빌두,문제열,논리'.split(',')) = ['정두','빌두','문제열','논리']

2번 다음 출력값을 기술하세요.

① Print('P{}{}'.format('y','charm')) = Pycharm

② Print('{0:d}{0:b}{0:o}'.format(7)) = 7 111 7

③ Print('{2}{1}{0}'.format(5,20,20-5)) = 15 20 5

④ Print('{0:6.3f}{0:5.2f}'.format(31.456)) = 31.456 31.46

⑤ Print('%d %f' % (3.14, 3.14)) = 3 3.140000

Chapter03+03 – 논리 자료와 다양한 연산

▶ 논리 유형 bool 과 함수 bool()

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> bool(0), bool(0.0), bool('')
(False, False, False)
>>> bool(10), bool(3.14), bool('python')
(True, True, True)
>>> int (True), int(False)
(1, 0)
>>>
```

▶ 논리곱과 논리합 연산자 and와 or
▶ 배타적 논리합 연산자 ^와 not 연산자

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (Ir
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> True & True, True and True
(True, True)
>>> True & False, True and False
(False, False)
>>> True | True, True or True
(True, True)
>>> False | True, False or True
(True, True)
>>> False | False, False or False
(False, False)
>>>
```

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> True ^ True
False
>>> True ^ False
True
>>> False ^ True
True
>>> not True, not False
(False, True)
>>>
```

Chapter03+03 – 논리 자료와 다양한 연산

▶ 논리 연산 우선순위 not and or

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> not False and True
True
>>> (not False) and True
True
>>> not True or True and False
False
>>>
```

▶ 관계 연산자

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ord('a'), ord('A'), ord('₩0'), ord('B')
(97, 65, 0, 66)
>>> 8 > 2, 'a' >= 'A'
(True, True)
>>> 8 < 2, 'a' < 'aB'
(False, True)
>>> 8 == 2, 'a' == 'aB'
(False, False)
>>> 8 != 2, 'a' != 'aB'
(True, True)
>>>
```

Chapter03+03 – 논리 자료와 다양한 연산

▶ 논리 값 True와 False의 산술 연산

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 40 + True, 30 + False
(40, 0)
>>>
```

▶ 비트 논리곱 &, 비트 논리합 |, 비트 배타적 논리합 ^

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> m, n = 0,0
>>> '{} {} {} {} {}'.format(m, n, m&n, m|n, m^n)
'0 0 0 0 0'
>>> m, n = 0, 1
>>> '{} {} {} {} {}'.format(m, n, m&n, m|n, m^n)
'0 1 0 1 1'
>>>
```

Chapter03+03 – 논리 자료와 다양한 연산

▶ 논리 값 True와 False의 산술 연산

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 40 + True, 30 + False
(40, 0)
>>>
```

▶ 비트 논리곱 &, 비트 논리합 |, 비트 배타적 논리합 ^

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> m, n = 0,0
>>> '{} {} {} {} {}'.format(m, n, m&n, m|n, m^n)
'0 0 0 0 0'
>>> m, n = 0, 1
>>> '{} {} {} {} {}'.format(m, n, m&n, m|n, m^n)
'0 1 0 1 1'
>>>
```

Chapter03+03 – 논리 자료와 다양한 연산

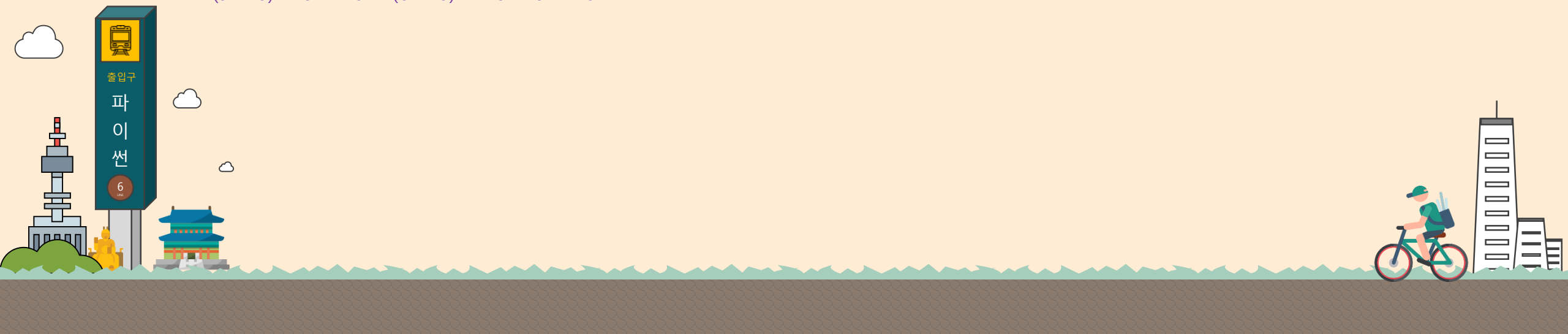
▶ 비트 논리곱 &로 특정 비트의 값 알아 내기

- 정수의 특정 비트 값 알아 내는데 사용된다.
- 원하는 특정 비트를 모두 1로 지정한 마스크 값을 정수 a 와 논리곱으로 연산한 a & mask 결과는 a의 특정 비트 값만을 뽑아 낸다.

a :	X	X	X	X	X	X	X
mask :	0	0	0	0	0	0	0
a & mask :	0	0	0	X	X	X	X

▶ 비트 배타적 논리합 ^ 을 사용해 암호화

a ^ a == 0, a ^ 0 == a, a ^ 1 == -a
a ^ b == b ^ a, (a ^ b) ^ c == a ^ (b ^ c)
(a ^ b) ^ b == a ^ (b ^ b) == a ^ 0 == a

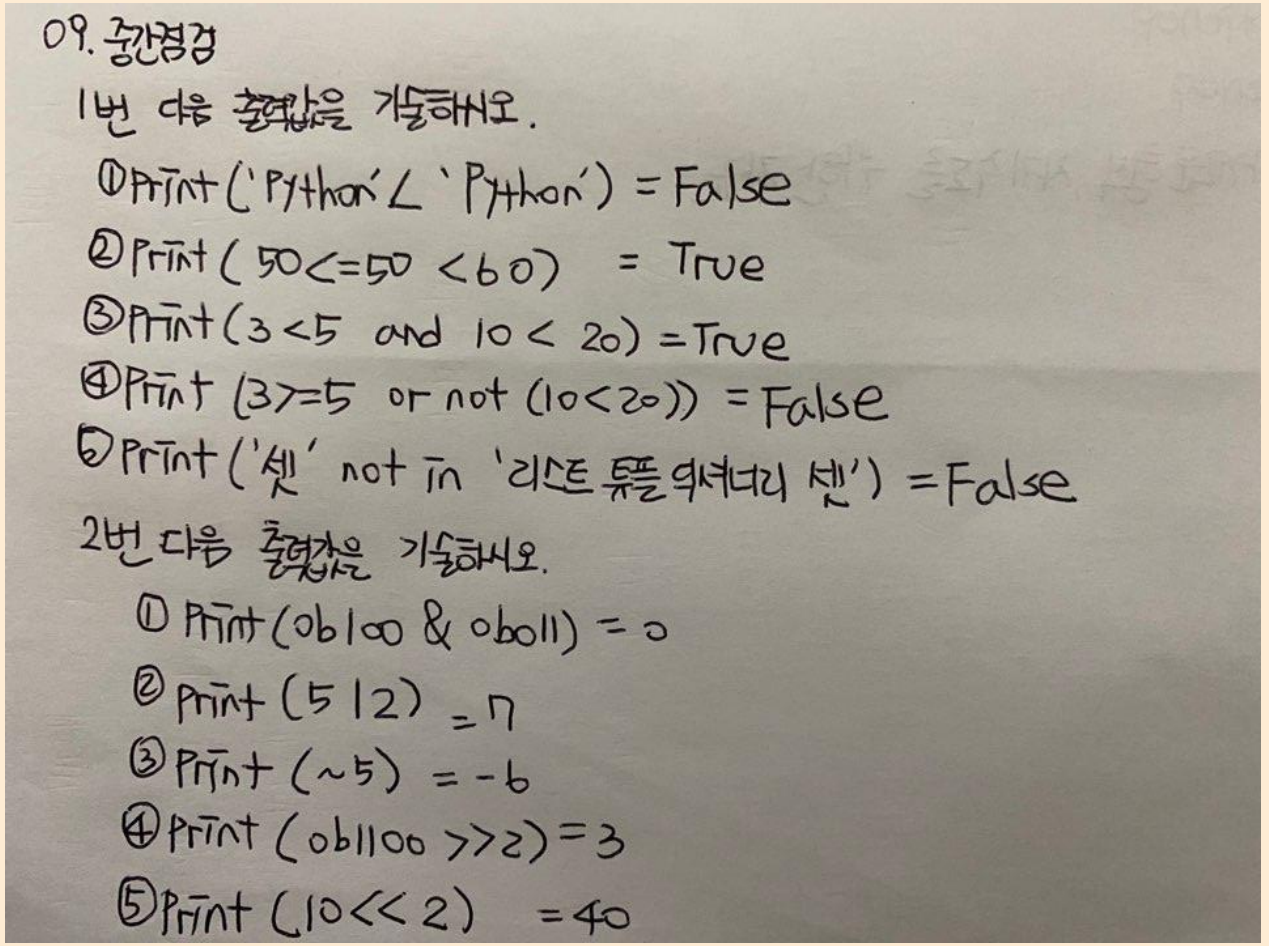


Chapter03+03 – 논리 자료와 다양한 연산

▶ 비트 이동 연산자 >> 와 <<

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> a = 0b00010111
>>> print('10진수 {0:3d}, 2진수 {0:08b}'.format(a))
10진수  23, 2진수 00010111
>>> print('10진수 {0:3d}, 2진수 {0:08b}'.format(a >> 1))
10진수  11, 2진수 00001011
>>> print('10진수 {0:3d}, 2진수 {0:08b}'.format(a << 2))
10진수  92, 2진수 01011100
>>>
```

Chapter03+03– 중간점검



Chapter04 – 조건에 따른 선택 if … else

▶ 조건에 따른 선택을 결정하는 if문

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> weather = '화창'
>>> if weather == '화창' :
        print('어제 산 신발을 신고 가야지!')


어제 산 신발을 신고 가야지!
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> pm = 90
>>> if 81 < pm:
        print('마스크를 착용합시다!')


마스크를 착용합시다!
>>>
```

▶ 논리 표현식 결과인 True 와 False에 따라 나뉘는 if … else문

```
n = int(input('정수 입력 >>'))
if n%2 == 0 :
    print('%d은 짝수다.' % n)
else :
    print('%d은 홀수다.' % n)
```

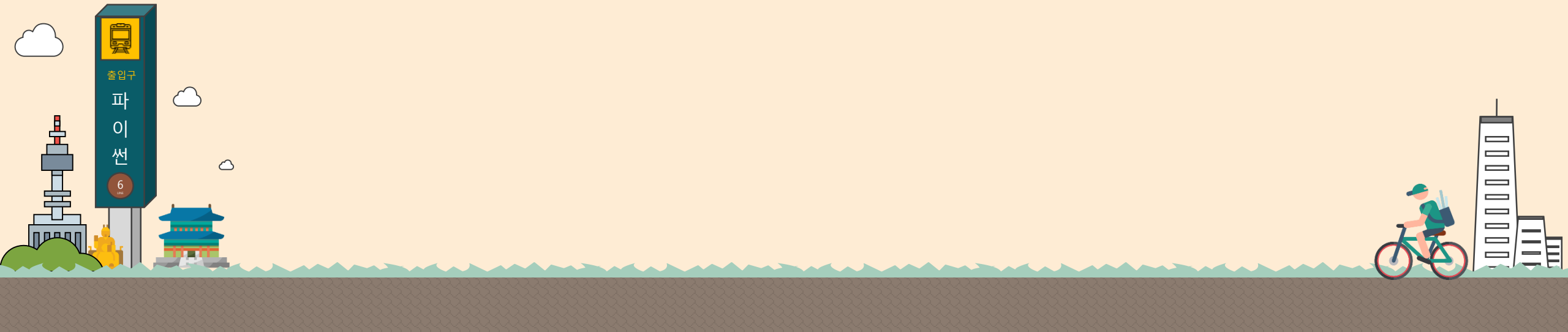
```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============== RESTART: C:/Users/joohe/OneDrive/Desktop/파이썬/1.9.py ==========
===
정수 입력 >>11
11은 홀수다.
>>>
============== RESTART: C:/Users/joohe/OneDrive/Desktop/파이썬/1.9.py ==========
===
정수 입력 >> 10
10은 짝수다.
>>>
```

Chapter04 – 조건에 따른 선택 if ... else

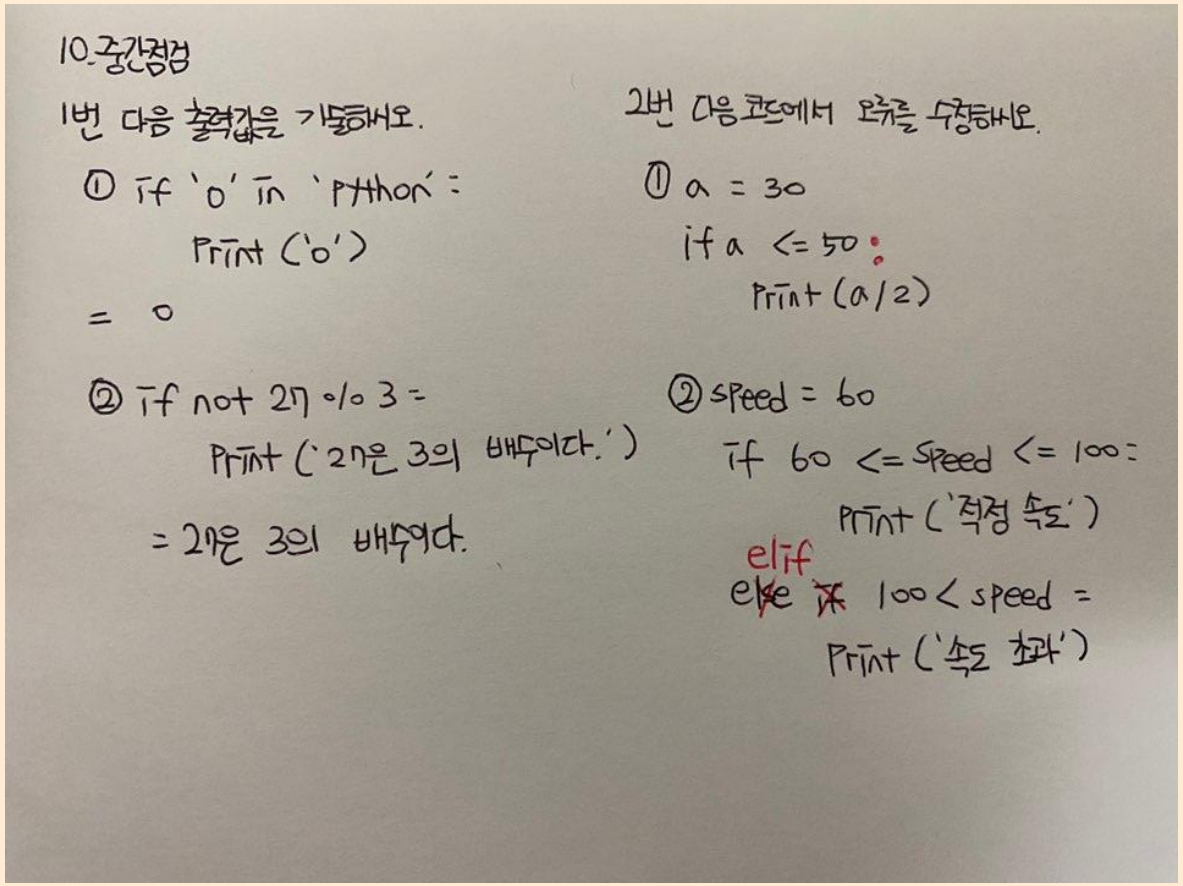
▶ 여러 조건 중에서 하나를 선택하는 구문 If ... elif

```
point = 82
if 90 <= point:
   print('점수 {}, 성적 {}'.format(point, 'A'))
elif 80 <= point:
    print('점수 {}, 성적 {}'.format(point, 'B'))
elif 70 <=point:
    print('점수 {}, 성적 {}'.format(point, 'C'))
elif 0 <=point:
    print('점수 {}, 성적 {}'.format(point, 'D'))
else:
    print('점수 {}, 성적 {}'.format(point, 'F'))

...
...
점수 82, 성적 B
```



Chapter04– 중간점검



Chapter04+02 – 반복을 제어하는 for문과 while문

▶ 일상생활과 같은 반복의 실행

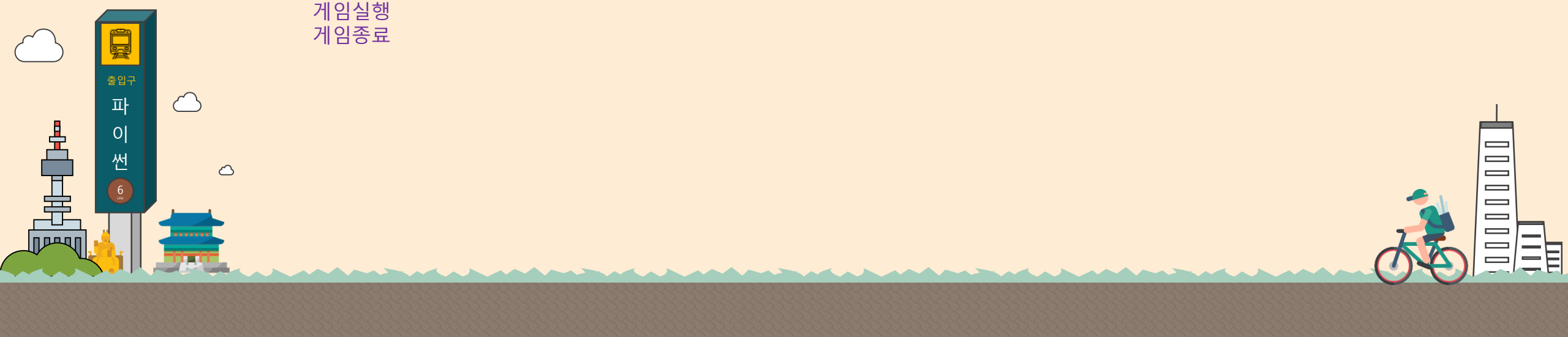
```
For i in 1, 2, 3:
    print('로그인')
    print('캐릭터 및 아이템 등 여러 옵션 선택')
    print('게임 실행')
    print('게임 종료')
```

결과 :

```
로그인
캐릭터 및 아이템 등 여러 옵션 선택
게임 실행
게임 종료
로그인
캐릭터 및 아이템 등 여러 옵션 선택
게임실행
게임종료
로그인
캐릭터 및 아이템 등 여러 옵션 선택
게임실행
게임종료
```

▶ 정해져 있는 시퀀스의 항목 값으로 반복을 실행하는 for 문

1) 여러 개의 값을 갖는 시퀀스에서 변수에 하나의 값을 순서대로 할당한다.
2) 할당된 변숫값을 갖고, 블록의 문장들(문장1, 문장2)을 순차적으로 실행한다.
3) 반복 몸체인 문장1, 문장2 에서 변수를 사용할 수 있다.
4) 시퀀스의 그 다음 값을 변수에 할당해 다시 반복 블록을 실행한다.
5) 이러한 과정을 시퀀스의 마지막 항목까지 수행한다.
6) 시퀀스의 마지막 항목까지 실행한 후 선택 사항인 else : 블록을 실행하고 반복을 종료한다.



Chapter04+02 – 반복을 제어하는 for문과 while문

▶ 내장 함수 range()를 사용한 for문

```
for i in range(5):
    print(i, end = ' ')
...
...
0 1 2 3 4
```

결과 :
```
>>>list(range(5))
[0, 1, 2, 3, 4]
```

```
for i in range(10):
    print(i, end = ' ')
...
...
0 1 2 3 4 5 6 7 8 9
```

결과 :
```
>>>list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
for i in range(1, 10):
    print(i, end = ' ')
...
...
1 2 3 4 5 6 7 8 9
```

결과 :
```
>>>list(range(1, 10))
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
For i in range(1, 10, 2):
    print(i, end = ' ')
....
....
1 3 5 7 9
```

Step : 증가 값

Stop : 시퀀스의 끝(미만),
실질적으로 stop-1까지

Start: 시퀀스의 시작

Chapter04+02 – 반복을 제어하는 for문과 while문

▶ 반복 구조가 간단한 while 반복

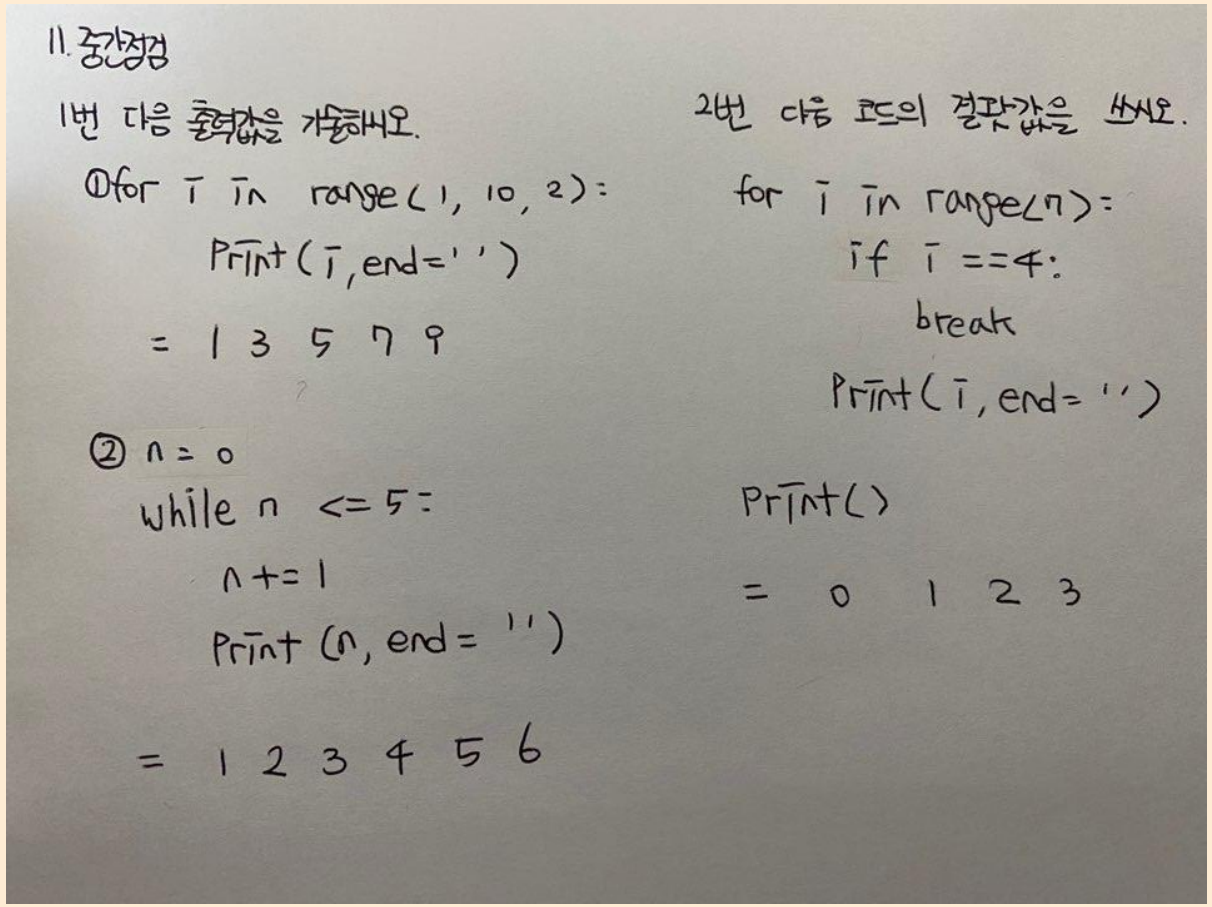
```
>>> n = 1
>>> while n <= 5:
    print(n, end = ' ')
    n += 1
Else:
     print('₩n 반복 while 종료 : n => %d" % n)
...
1 2 3 4 5
반복 while 종료 : n => 6
```

▶ 일상코딩 : 반복 for의 중첩 표준 구구단

```
i = 6
for j in range(1, 10):
     print('%d * %d = %2d' % (i, j, i * j), end = ' ')
print()
```

결과 :
6 * 1 =6
6 * 2 = 12
6 * 3 = 18
6 * 4 = 24
.....중략 .....
6 * 8 = 48
6 * 9 = 54

Chapter04+02– 중간점검



Chapter04+03 – 임의의 수인 난수와 반복을 제어하는 break문, continue문

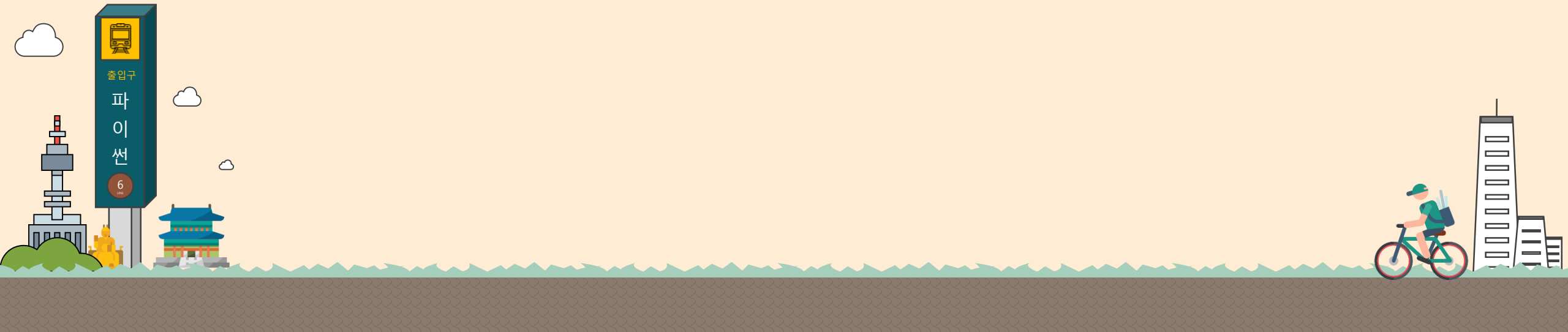
▶ 임의의 수를 발생하는 난수

```
>>>import random
>>>random.randint(1, 3)
3
>>>random.randint(1, 3)
```

▶ 반복을 종료하는 break문

```
for 변수 in 시퀀스 :
    break_앞_문장들
    if 조건 :
        break
    break_뒤_문장들
else :
    반복이_정상적으로_종료되면_실행되는_문장들
for문_종료_이후_코드
```

```
While 논리 표현식 :
    break_앞_문장들
    if 조건 :
        break
    break_뒤_문장들
else :
    반복이_정상적으로_종료되면_실행되는_문장들
While문_종료_이후_코드
```



Chapter04+03 – 임의의 수인 난수와 반복을 제어하는 break문, continue문

▶ **continue 이후의 반복 몸체를 실행하지 않고 다음 반복을 계속 실행**

```
for s in 'python' :
        if s in 'aeiou' :
                continue
        print(s, end = ' ')
else :
        print()
...
p y t h n
```

Chapte05– 여러 자료 값을 편리하게 처리하는 리스트

▶ 관련된 나열 항목을 관리하는 리스트

```
>>>menu = ['COFFEE', 'EEVERAGE', 'ADE']
>>>menu
>>>print(menu)
['COFFEE', 'EEVERAGE', 'ADE']
```

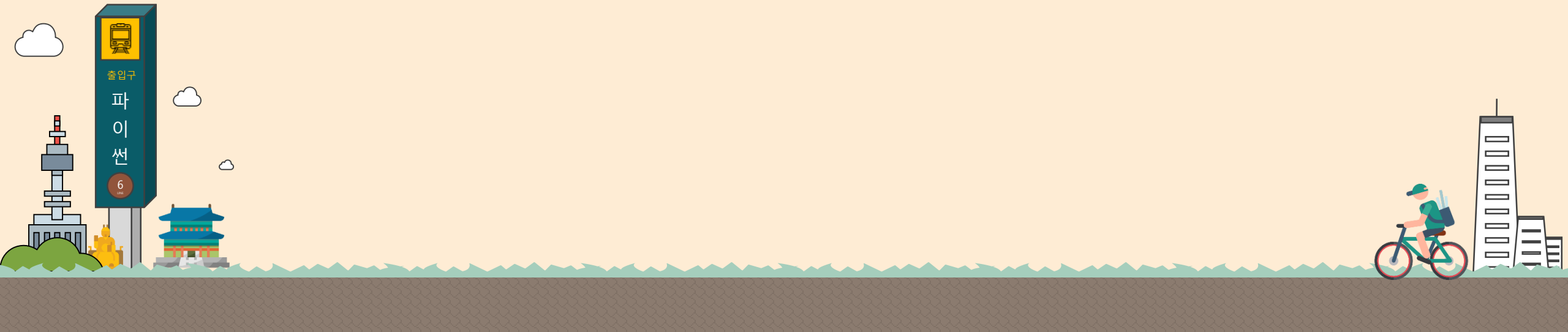
▶ 빈 리스트의 생성과 항목 추가

```
>>>pl = list()
>>>pl.append('C++')
>>>pl.append('java')
>>>print(pl)
['C++', 'java']
```

▶ 문자열의 문자로 구성되는 리스트와 리스트의 항목 수를 반환하는 함수 len()

```
>>>py = ['p', 'y', 't', 'h', 'o', 'n']
>>>py
['p', 'y', 't', 'h', 'o', 'n']
```

```
>>>py = list('python')
>>>py
['p', 'y', 't', 'h', 'o', 'n']
>>>len(py)
6
```



Chapte05– 여러 자료 값을 편리하게 처리하는 리스트

▶ 문자열 시퀀스와 같이 첨자로 리스트의 항목 참조

```
>>>py = list('python')
>>>print(py[0], py[5])
p n
>>>ptin(py[-3], py[-1])
h n
>>>print(py[-len[(py)], py[len(py)-1])
p   n
```

▶ 리스트의 첨자로 항목 수정

```
>>>top = ['BTS', '볼빨간사춘기', 'BTS', '블랙핑크', ' 태연', 'BTS']
>>>top[1] = '장범준'
>>>top[3] = '잔나비'
>>>print(top)
['BTS', '장범준', 'BTS', '잔나비',  '태연', 'BTS']
```

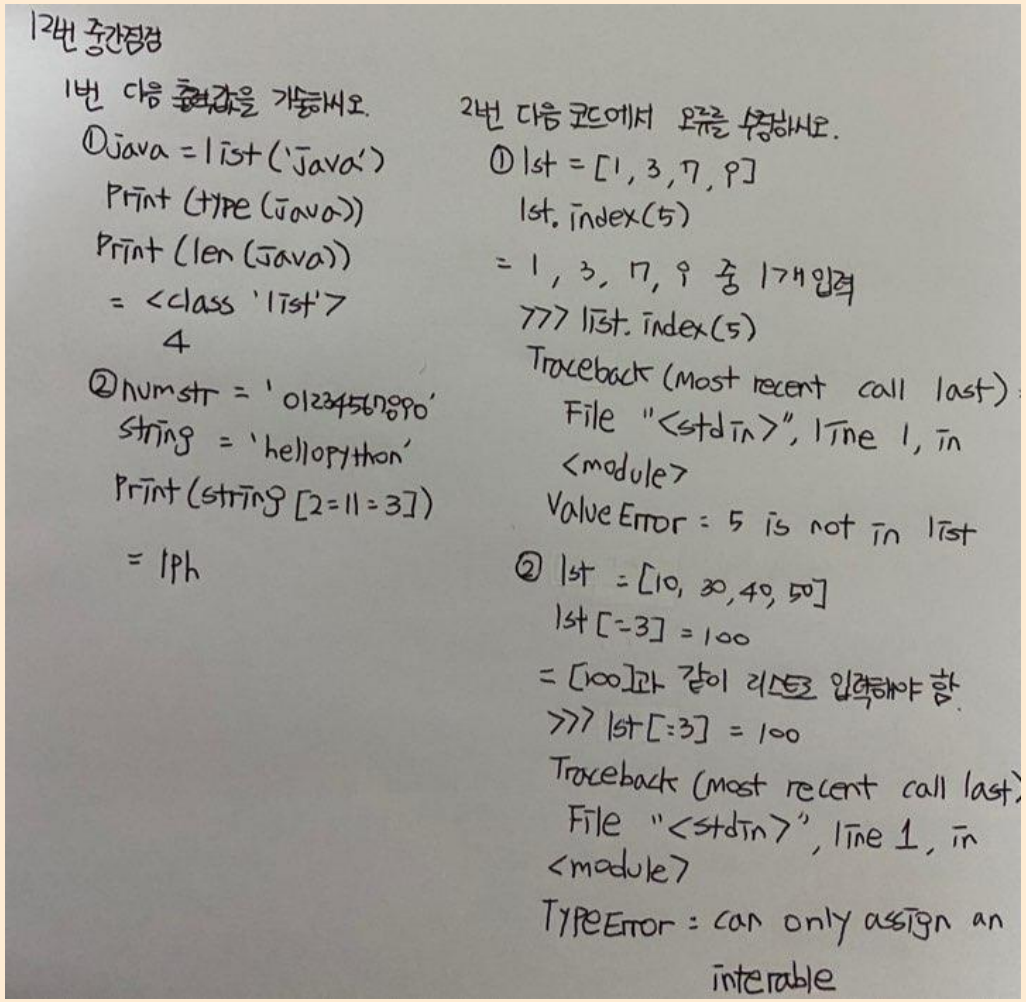
▶ 리스트의 메소드 count()와 index()

```
>>>top = ['BTS', '볼빨간사춘기', 'BTS', '블랙핑크', ' 태연', 'BTS']
>>>top.count('BTS')
3
>>>top.index('볼빨간사춘기')
1
>>>top.index('BTS')
```

▶ 리스트의 항목으로 리스트 구성

```
>>>animal = [['사자', '코끼리', '호랑이'], '조류', '어류']
>>>print(animal)
[['사자', '코끼리', '호랑이'], '조류', '어류']
>>>print(animal[0])
['사자', '코끼리', '호랑이']
>>>print(animal[0][1])
코끼리
```

Chapter05– 중간점검



12번 중간점검

1번 다음 출력값을 기술하세요.

① java = list('java')
print(type(java))
print(len(java))
= <class 'list'>
4

② numstr = '01234567890'
string = 'hellopython'
print(string[2:11:3])
= lph

2번 다음 코드에서 오류를 수정하세요.

① lst = [1, 3, 17, 9]
lst.index(5)
= 1, 3, 17, 9 중 1개입력
>>> list.index(5)
Traceback (most recent call last)
File "<stdin>", line 1, in <module>
ValueError: 5 is not in list

② lst = [10, 30, 40, 50]
lst[-3] = 100
= [100]과 같이 리스트 입력해야 함.
>>> lst[:3] = 100
Traceback (most recent call last)
File "<stdin>", line 1, in <module>
TypeError: can only assign an iterable

Chapte05+02– 리스트의 부분 참조와 항목의 삽입과 삭제

▶ 첨자 3개로 리스트 일부분을 참조하는 슬라이싱

```
>>>alp = list('abcdefghij')
>>>print(alp)
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i',  'j']
>>>len(alp)
10
>>>alp = list('abcdefghij')
>>>print(alp[1:5])
['b', 'c', 'd', 'e']
>>>alp = list('abcdefghij')
>>>print(alp[-2:-9:-1])
['i', 'h', 'g', 'f', 'e', 'd', 'c']
```

▶ 리스트 슬라이스에 슬라이스 대입

```
>>>sports = ['풋살', '족구', '비치사커', ' 야구', '농구', '배구']
>>>sports[1:3] = sports[4:6]
>>>print(sports)
['축구', '농구', '배구', '야구', '농구', '배구 ' ]
```

▶ 리스트의 슬라이스로 리스트의 일부분을 수정

```
>>>sports = ['풋살', '족구', '비치사커', ' 야구', '농구', '배구']
>>>sports[0:3] = ['축구']
>>>print(sports)
['축구', '야구', '농구', '배구']
```

▶ 리스트 메소드 insert(첨자, 항목)으로 삽입

```
>>>kpop = []
>>>kpop.insert(0, '블랙핑크')
>>>kpop.insert(0, 'BTS')
>>>kpop
['BTS', '블랙핑크']
```

+첨자 1에 '장범준'을 삽입하면 두 번째 항목으로 삽입이 된다.

```
>>>kpop.insert(1,  '장범준')
>>>kpop
['BTS', '장범준', '블랙핑크']
```

Chapte05+02– 리스트의 부분 참조와 항목의 삽입과 삭제

▶ 리스트 메소드 remove(항목)과 pop(첨자), pop()로 항목 삭제

```
>>>kpop = ['BTS', '장범준', '블랙핑크', '잔나비']
>>>kpop.remove('장범준')
>>>print(kpop)
['BTS', '블랙핑크', '잔나비']
>>>print(kpop.pop(1))
장범준
++첨자 1인 장범준이 삭제가 된다.
>>>print(kpop)
['BTS', '블랙핑크', '잔나비']
```

▶ 문장 del()에 의한 항목이나 변수의 삭제

```
>>>kpop = ['BTS', '장범준', '블랙핑크', '잔나비']
>>>del kpop[0]
++del은 [0]이 가능하다.
>>>print(kpop)
['장범준', '블랙핑크', '잔나비']
```

▶ 리스트의 모든 항목을 제거해 빈 리스트로 만드는 메소드 clear()

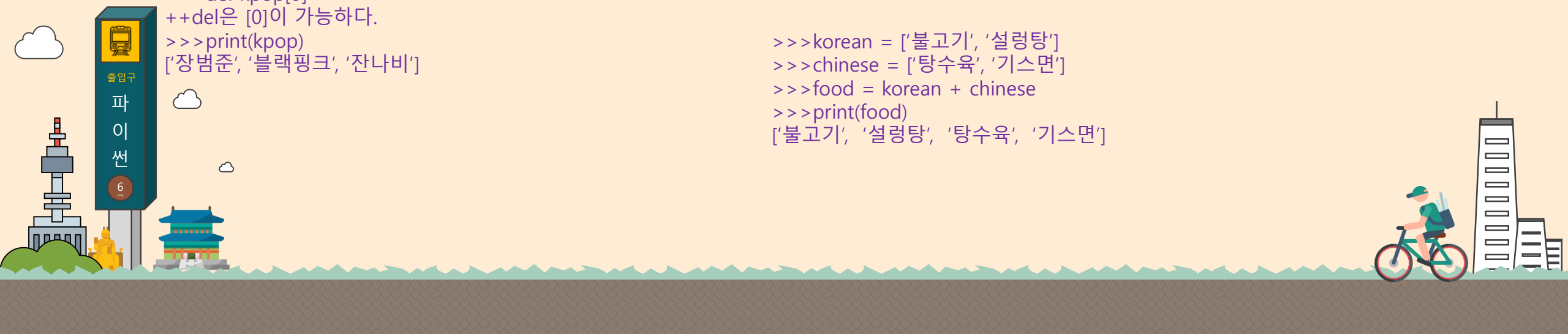
```
>>>kpop = ['BTS', '장범준', '블랙핑크', '잔나비']
>>>kpop.clear()
>>>print(kpop)
[ ]
```

▶ 리스트에 리스트를 추가하는 메소드 extend()

```
>>>day = ['월', '화', '수']
>>>day2 = ['목', '금', '토', '일']
>>>day.extend(day2)
>>>print(day)
['월', '화', '수', '목', '금', '토', '일']
```

▶ 리스트를 연결하는 + 연산자

```
>>>korean = ['불고기', '설렁탕']
>>>chinese = ['탕수육', '기스면']
>>>food = korean + chinese
>>>print(food)
['불고기', '설렁탕', '탕수육', '기스면']
```



Chapte05+02– 리스트의 부분 참조와 항목의 삽입과 삭제

▶ 리스트 항목 순서를 뒤집는 메소드 reverse()

```
>>>one = '잣밤배귤감'
>>>wlist = list(one)
>>>print(wlist)
['잣', '밤', '배', '귤', '감']

>>>wlist.reverse()
>>>print(wlist)
['감', '귤', '배', '밤', '잣']
```

▶ 리스트 항목 순서를 정렬하는 메소드 sort()

```
>>>one = '잣밤배귤감'
>>>wlist = list(one)
>>>wlist.sort()
>>>print(wlist)
['감', '귤', '배', '밤', '잣']
>>>wlist..sort(reverse=True)
>>>print(wlist)
['잣', '밤', '배', '귤', '감']
```

▶ 리스트 항목의 순서를 정렬한 리스트를 반환하는 내장 함수 sorted()

```
>>>fruit = ['사과', '귤', '복숭아', '파인애플']
>>>s_fruit = sorted(fruit)
>>>print(s_fruit)
['귤', '복숭아', '사과', '파인애플']
>>>print(fruit)
['사과', '귤', '복숭아', '파인애플']
>>>rs_fruit = sorted(fruit, reverse = True)
>>>print(rs_fruit)
['파인애플', '사과', '복숭아', '귤']
```

++역순인 내림차순으로 정렬된 리스트를 반환한다. 없으면 0을 반환

▶ 조건을 만족하는 항목으로 리스트를 간결히 생성하는 컴프리헨션

```
>>>even = [ ]
>>>for i in range(2, 11, 2):
        even.append(i)

>>>print(even)
[2, 4, 6, 8, 10]
```

조건이 있는 컴프리헨션

```
>>>odd = [ ]
>>>for i in range(10):
        if i%2 == 1:
            odd.append(i)

>>>print(odd)
[1, 3, 5, 7, 9]
```

Chapte05+02– 리스트의 부분 참조와 항목의 삽입과 삭제

▶ 리스트 대입에 의한 동일 리스트의 공유

```
>>>f1 = ['사과', '귤', '복숭아', '파인애플']
>>>f2 = f1
>>>f2.pop()
'파인애플'
>>>print(f1)
['사과', '귤', '복숭아']
>>>print(f2)
['사과', '귤', '복숭아']
```

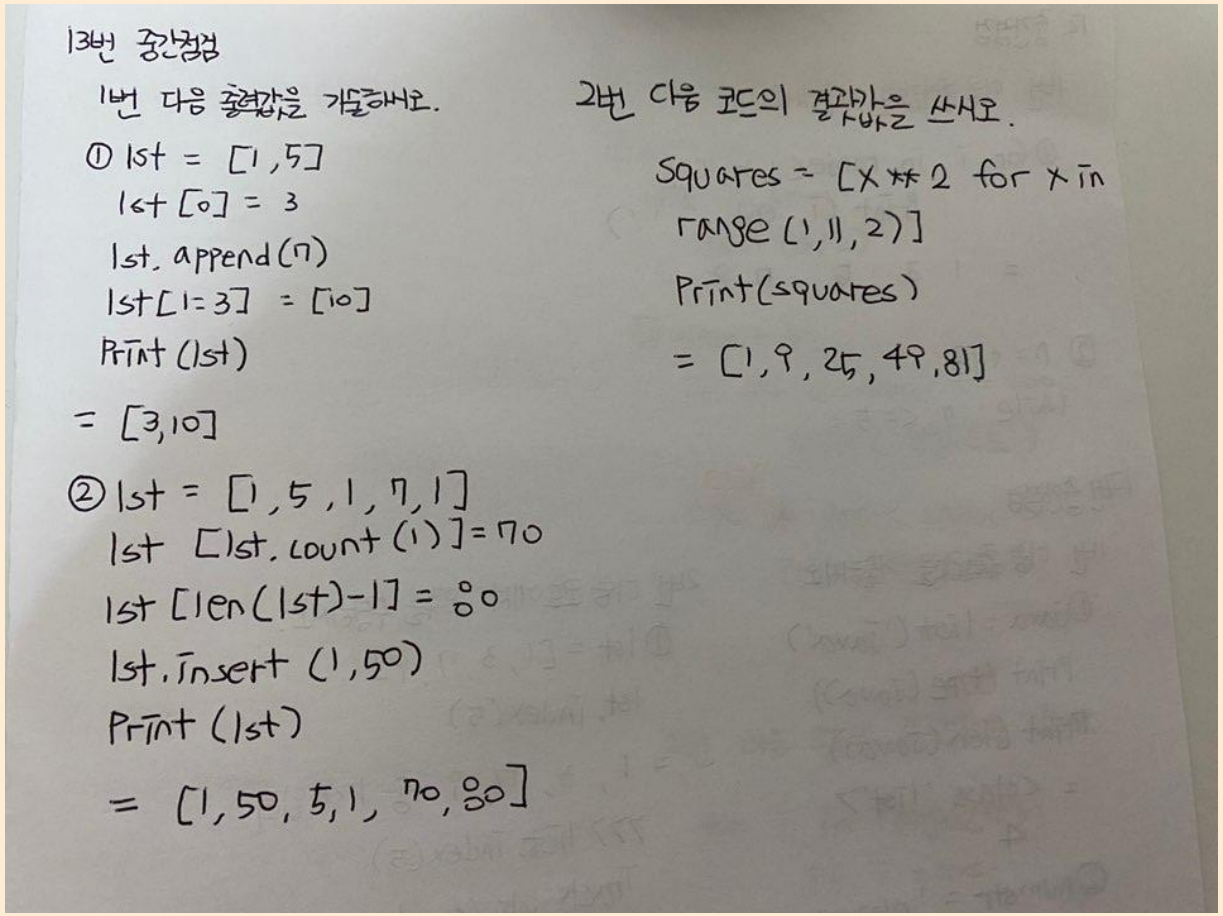
▶ 변수의 동일 객체 여부를 검사하는 is

```
>>>f1 = ['사과', '귤', '복숭아', '파인애플']
>>>f2 = f1
>>>print(f1 is f2)
True
>>>f3 = f1[ : ]
>>>print( f1 is f3)
False
```

▶ 리스트의 깊은 복사에 의한 대입으로 새로운 리스트의 생성

```
>>>f1 = ['사과', '귤', '복숭아', '파인애플']
>>>f2 = f1[ : ]
>>>f2.pop(1)
'귤'
>>>print(f1, f2)
['사과', '귤', '복숭아'] ['사과', '귤', '복숭아']
```

Chapter05+02– 중간점검



13번 중간점검

1번 다음 출력값을 기술하세요.

① lst = [1,5]
lst[0] = 3
lst.append(7)
lst[1:3] = [10]
Print(lst)

= [3,10]

② lst = [1,5,1,7,1]
lst [lst.count(1)] = 70
lst [len(lst)-1] = 80
lst.insert(1,50)
Print(lst)

= [1,50,5,1,70,80]

2번 다음 코드의 결과값을 쓰세요.

squares = [x**2 for x in range(1,11,2)]
Print(squares)

= [1,9,25,49,81]

Chapte05+03– 항목의 순서나 내용을 수정할 수 없는 튜플

▶튜플

생성

```
>>>empty1 = ()
>>type(empty1)
<class 'tuple'>
>>>print(empty1)
( )
>>>inta = 1
>>>tupa = 1,
>>>print(tupa)
(1, )
```

++항목이 하나인 튜플을 나타낸다.

튜플 항목 참조와 출력

```
>>>nation = '대한민국', '뉴질랜드', '캐나다']
>>>city = ('부산', '웰링톤', '몬트리올')
>>>nation[1]
'뉴질랜드'
>>>city[1:3]
('웰링톤', '몬트리올')
```

튜플 연결 + 연산자와 반복 * 연산자

```
>>>kpop('BTS', '블랙핑크')
>>>num = (7, 4)
>>>print(kpop + num)
('BTS', '블랙핑크', 7, 4)
>>>days = ('1학기', '2학기')
>>>print(days * 4)
('1학기', '2학기', '1학기', '2학기', '1학기',
'2학기', '1학기', '2학기')
```

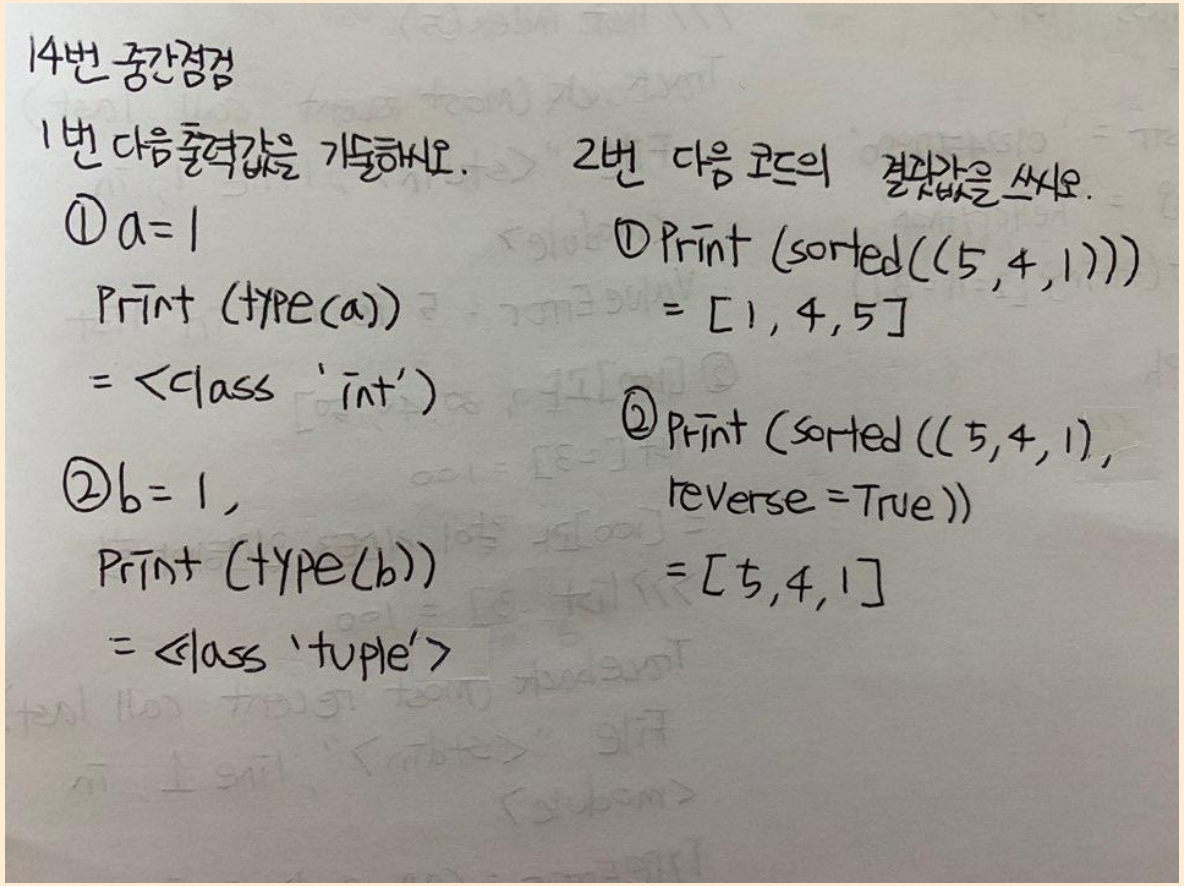
튜플 항목의 순서를 정렬한 리스트를 반환하는 내장 함수 sorted()

```
>>>fruit = ('사과', '귤', '복숭아', '파인애플')
>>>tup = sorted(fruit)
>>>print(type(tup), tup)
<class 'list'> ['귤', '복숭아', '사과', '파인애플']
>>>print(sorted(fruid, reverse = True))
['파인애플', '사과', '복숭아', '귤']
>>>print(fruit)
('사과', '귤', '복숭아', '파인애플')
```

문장 del에 의한 튜플 변수 자체의 제거

- 문장 del에서 변수 kpop을 지정하면 변수 자체가 사라진다.
- 삭제 이후에는 변수 자체가 메모리에서 제거되므로 참조하면 오류난다.

Chapter05+03– 중간점검



Chapte06– 키와 값인 쌍의 나열인 딕셔너리

▶ 키와 값의 쌍을 항목으로 관리하는 딕셔너리

```
groupnumber = {'엑소' : 9, '트와이스' : 9, '블랙핑크' : 4, '방탄소년단': 7}
coffeeprice = {'에스프레소': 2500, '아메리카노': 2800, '카페라테': 3200}
mycar = {"brand": "현대", "model": "제네시스", "year": 2016}

>>>mycar = {"brand": "현대", "model": "제네시스", "year": 2016}
>>>print(mycar)
{"brand": "현대", "model": "제네시스", "year": 2016}
>>>print(type(mycar))
<class 'dict'>
```

▶ 빈 딕셔너리의 생성과 항목 추가

```
>>>lect = { }
>>>print(lect)
{ }
>>>lect = dict()
>>>lect['강좌명'] = '파이썬 기초';
>>>lect['개설년도'] = [2020, 1];
>>>print(lect)
{'강좌명' : '파이썬 기초', '개설년도 '  : [2020, 1]}
```

▶ 리스트 또는 튜플로 구성된 키 – 값을 인자로 사용

```
>>>day = dict[['월', 'Monday'], ['화', 'tuesday'], ['수', 'Wednesday']])
>>>day = dict[['월', 'Monday'], ['화', 'tuesday'], ['수', 'Wednesday']])
>>>day = dict[['월', 'Monday'], ['화', 'tuesday'], ['수', 'Wednesday']])
>>>day = dict[['월', 'Monday'], ['화', 'tuesday'], ['수', 'Wednesday']])
>>>print (day)
{'월': 'Monday', '화', 'tuesday', '수', 'Wednesday'}
```

▶ 키가 문자열이면 키 = 값 항목의 나열로도 딕셔너리 생성

```
>>>day  = dict(월 = 'monaday', 화 = 'tuesday', 수 = 'wednesday')
>>>print(day)
{'월' : 'monday', '화' : 'tuesday', '수', 'wednesday'}
```

Chapte06– 키와 값인 쌍의 나열인 딕셔너리

▶ 딕셔너리의 키로 정수, 실수 등 사용 가능

```
>>>real = {3.14: '원주율'}
>>>real[2.71] = '자연수'
>>>print(real)
{3.14: '원주율', 2.17 : '자연수'}
>>>real[2.72] = '자연수'
>>>print(real)
{3.14: '원주율', 2.17 : '오일러수', 2.72 : '자연수'}
```

▶ 튜플은 키로 가능하지만 리스트는 키로 사용 불가능

- 수정 불가능한 튜플은 딕셔너리의 키로 사용될 수 있다.
- 그러나 수정 가능한 리스트는 키로 사용할 수 없다.

▶딕셔너리 메소드 keys()

```
>>>day  = dict(월 = 'monaday', 화 = 'thuesday', 수 = 'wednesday')
>>>print(day)
{'월' : 'monday', '화' : 'thuesday', '수', 'wednesday'}
>>>print(day.keys())
dict_keys(['월', '화', '수'])
```

▶딕셔너리 메소드 items()

```
>>>day  = dict(월 = 'monaday', 화 = 'tuesday', 수 = 'wednesday')
>>>print(day.items())
day. items( [ ('월', 'monaday',), ('화', 'tuesday'), ('수', 'wednesday') ] )
>>>for key, value in day.items():
        print('%s요일 %s' %(key, value))
월요일  monday
화요일 tuesday
수요일 wednesday
```

Chapte06– 키와 값인 쌍의 나열인 딕셔너리

▶ 딕셔너리 메소드 values()

```
>>>day  = dict(월 = 'monaday', 화 = 'thuesday', 수 = 'wednesday')
>>>print(day.values())
dict_values(['monday', 'tuesday', 'wednesday'])
```

▶ 반복 for문에서 딕셔너리 변수로만 모든 키 순회

```
>>>game = dict(일월 = '소나무', 이월 = '매화', 삼월 = '벚꽃')
>>>for key in game:
    print('%s: %s' % (key, game[key]))
일월 : 소나무
이월 : 매화
삼월 : 벚꽃
```

▶ 임의의 항목을 삭제하는 딕셔너리 메소드 popitem()

```
>>>city = {'대한민국' : '부산', '뉴질랜드' : '웰링톤'}
>>>print(city.popitem( ))
('뉴질랜드', '웰링톤')
>>>city
('대한민국' : '부산')
```

▶ 키로 조회하는 딕셔너리 메소드 get()

```
>>>city = {'대한민국' : '부산', '뉴질랜드' : '웰링톤'}
>>>city.get ('대한민국')
'부산'
>>>city.get('미국')
>>>city.get('미국', '없네요')
++해당하는 키가 없으면 없네요 라고 반환
```

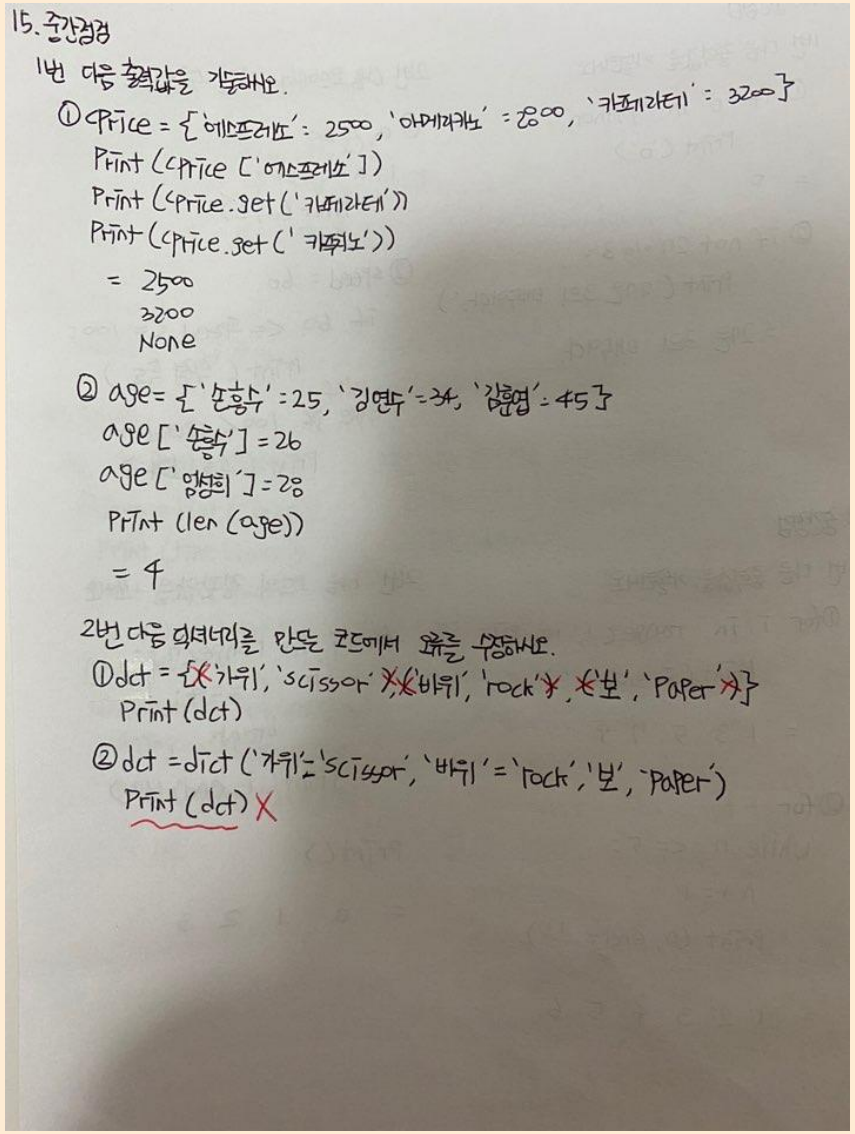
▶ 키로 삭제하는 딕셔너리 메소드 pop()

```
>>>city = {'대한민국' : '부산', '뉴질랜드' : '웰링톤'}
>>>print(city.pop ('대한민국')
부산
>>>city
{'대한민국' : '부산', '뉴질랜드' : '웰링톤'}
```

▶ 문장 del()로 딕셔너리 항목 삭제

```
>>>city = {'대한민국' : '부산', '뉴질랜드' : '웰링톤'}
>>> del['대한민국']
>>>city
{'뉴질랜드' : '웰링톤'}
```

Chapter06– 중간점검



15. 중간점검

1번 다음 출력값을 예측해보.

① cPrice = {'에스프레소': 2500, '아메리카노': 2000, '카페라테': 3200}

```
Print (cPrice ['에스프레소'])
Print (cPrice.get('카페라테'))
Print (cPrice.get('카푸치노'))
```

= 2500
3200
None

② age = {'손흥민': 25, '김연아': 34, '김희열': 45}

```
age['손흥민'] = 26
age['임정희'] = 28
Print (len (age))
```

= 4

2번 다음 딕셔너리를 만드는 코드에서 오류를 수정해보.

① dct = {'가위', 'scissor', '바위', 'rock', '보', 'paper'}

```
Print (dct)
```

② dct = dict ('가위'='scissor', '바위'='rock', '보', 'paper')

```
Print (dct)
```

Chapte06+02– 중복과 순서가 없는 집합

▶ 원소는 유일하고 순서는 의미 없는 집합

S1 = {1, 2, 3, 4, 5}
S2 = {'py', 'java', 'go'}
S3 = {(1, 1), (2, 4), (3, 9)}

▶ 집합을 만드는 내장 함수 set()

Set(원소로 구성된 리스트_or_튜플_or_문자열)

▶ 함수 set()호출로 공집합 만들기

```
>>>s = set()
>>>type(s)
<class 'set'>
>>>s
Set()
d = { }
>>>type(d)
<class 'dict'>
```

▶ 리스트나 튜플을 인자로 사용하는 함수 set()

```
>>>set([1, 2, 3])
{1, 2, 3}
>>>set((1, 2, 2))
{1, 2}
>>>set(['a', 'b'])
{'b', 'a'}
```

+순서는 의미 없다.

▶ 문자열을 인자로 사용하는 함수 set()

```
>>>set('abc')
{'c', 'b', 'a'}
```

▶ {원소 1, 원소 2, ...}로 생성

```
>>>{1, 2, 3}
{1, 2, 3}
>>>{1, 'seoul', 'a', (1.2, 3.4)}
{1, (1.2, 3.4), 'seoul', 'a'}
```

Chapte06+02– 중복과 순서가 없는 집합

▶ 집합 메소드 add(원소)로 추가

```
>>>odd = {1, 3, 5}
>>>odd.add(7)
>>>odd.add(9)
>>>print(odd)
{1, 3, 5, 7, 9}
```

▶ 메소드 clear()로 집합의 모든 원소 삭제

```
>>>odd = {1, 3, 5}
>>>odd.clear()
>>>print(odd)
set()
```

▶ 집합 메소드 remove(원소)와 discard(원소), pop()으로 항목 삭제

```
>>>odd = {1, 3, 5}
>>>odd.remove(3)
>>>print(odd)
{1, 5}
>>>odd = {1, 3, 5}
>>>odd.discard(3)
>>>print(odd)
{1, 5}
>>>odd.discard(9)
```

```
>>>odd = {1, 3, 5}
>>>print(odd.pop())
1
>>>print(odd)
{3, 5}
```

▶ 합집합 연산자 | 와 메소드 union(), update()

```
>>>a = {4, 6, 8, 10, 12}
>>>b = {3, 6, 9, 12}
>>>a | b
{3, 4, 6, 8, 9, 10, 12}
>>>a.union(b)
{3, 4, 6, 8, 9, 10, 12}
>>>a
{4, 6, 8, 10, 12}
>>>a.update(b)
>>>a
{3, 4, 6, 8, 9, 10, 12}
```

Chapte06+02– 중복과 순서가 없는 집합

▶ 교집합 연산자 &와 메소드 intersection()

```
>>>a = {4, 6, 8, 10, 12}
>>>b = {3, 6, 9, 12}
>>>a & b
{12, 6}
>>>a.intersection(b)
{12, 6}
```

▶ 차집합 연산자 –와 메소드 difference()

```
>>>a = {4, 6, 8, 10, 12}
>>>b = {3, 6, 9, 12}
>>>a - b
{8, 10, 4}
>>>a.difference(b)
{8, 10, 4}
>>>b – a
{9, 3}
```

▶ 여집합 연산자 ^와 메소드 symmetric_difference()

```
>>>a ^ b
{3, 4, 8, 9, 10}
>>>a.symmetric_difference(b)
{3, 4, 8, 9, 10}
```

▶ 집합 연산의 축약 대입 연산자 |=, &=, -=, ^=와 메소드 intersection_update()등

```
>>>A = set('abcd'); B = set('cde')
>>>A |= B
>>>A
{'b', 'a', 'c', 'e', 'd'}
>>>A = set('abcd'); B = set('cde')
>>>A &= B
>>>A
{'c', 'd',}
>>>A = set('abcd'); B = set('cde')
>>>A.intersection_update(B)
>>>A
{'c', 'd'}
>>>A = set('abcd'); B = set('cde')
>>>A -= B
>>>A
{'b', 'a'}
```

```
>>>A = set('abcd'); B = set('cde')
>>>A ^= B
>>>A
{'b', 'a', 'e'}
```

마무리

이렇게 06-09 예제문제까지 배운 내용 포트폴리오로 준비했습니다.
이런 과제를 통해 파이썬 이라는 과목을 처음부터 끝까지 다시 복습을 하고
 제대로 공부한 것 같았습니다.
온라인 강의로도 수업을 했지만 몰랐던 부분들도 있었고,
대충 넘긴 것들도 발견할 수 있었습니다.
비록 시간은 많이 걸렸지만 다 끝낸 것에
또한 다시 한번 공부할 수 있었던 것에 뜻 깊은 시간으로 남은 것 같습니다.
이 포트폴리오는 기말고사 준비하면서 사용할 것입니다.
책에 있는 내용들을 통해 자기주도학습 용으로 정리하였습니다.
책에 있는 중간점검은 손으로 통해 직접 풀고 답을 맞춰보며 마무리 했습니다.
끝까지 봐주셔서 감사합니다.