

# Contents

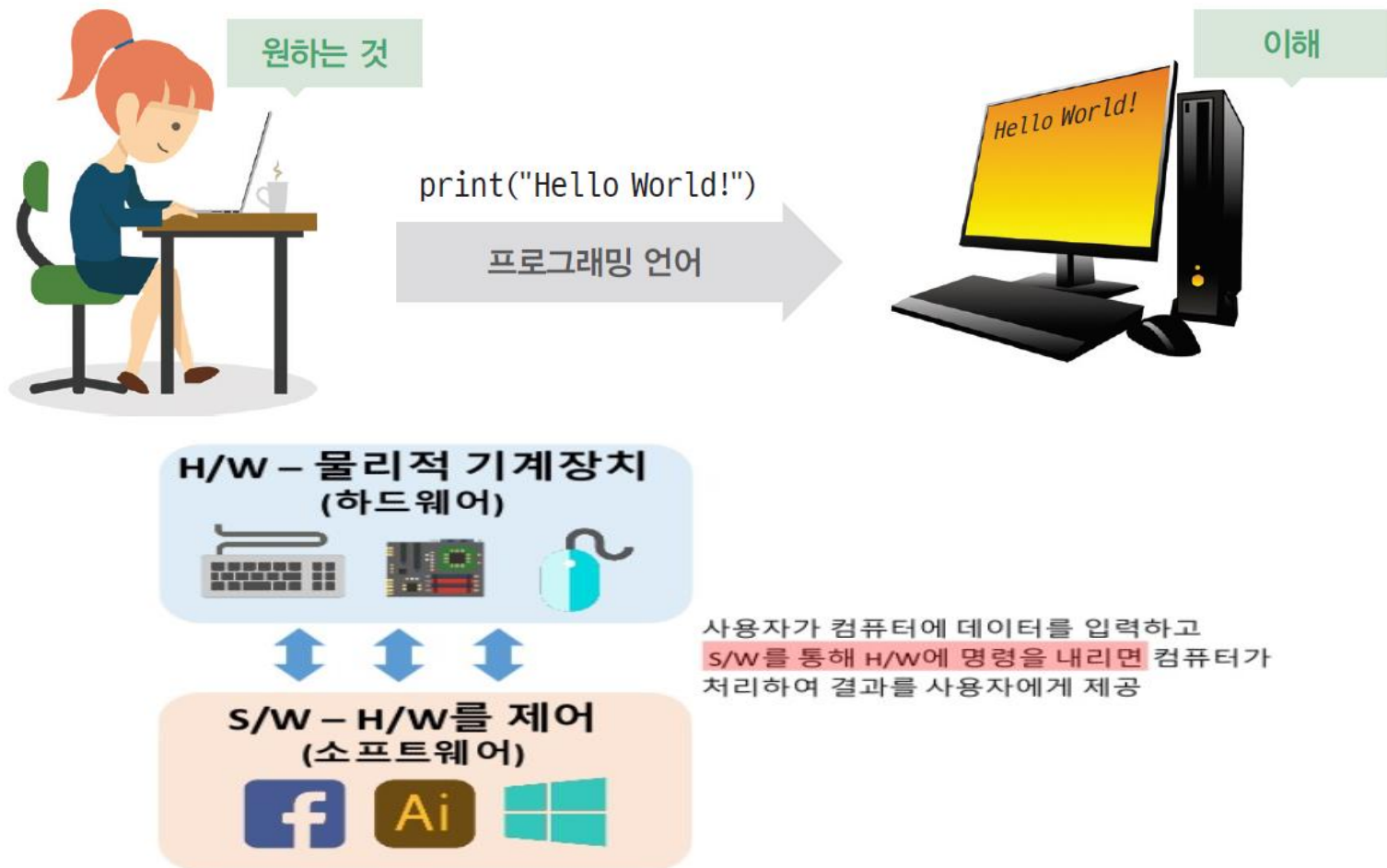
## ❖ 목차

- 1. 파이썬
- 2. 개발환경
- 3. 대화식과 스크립트

# 1. 파이썬

## ❖ 프로그래밍(programming)이란

프로그래밍 언어를 사용하여 프로그램을 개발하는 것



# 1. 파이썬

## ❖ 프로그래밍(programming)이란

프로그래밍 언어를 사용하여 프로그램을 개발하는 것



# 1. 파이썬

## ❖ 프로그래밍(programming)이란

2020년 가장 많이 사용되고 있는 프로그래밍 언어(<https://www.tiobe.com/tiobe-index/>)

Jul 2020	Jul 2019	Change	Programming Language	Ratings	Change
1	2	▲	C	16.45%	+2.24%
2	1	▼	Java	15.10%	+0.04%
3	3		Python	9.09%	-0.17%
4	4		C++	6.21%	-0.49%
5	5		C#	5.25%	+0.88%
6	6		Visual Basic	5.23%	+1.03%
7	7		JavaScript	2.48%	+0.18%
8	20	▲▲	R	2.41%	+1.57%
9	8	▼	PHP	1.90%	-0.27%
10	13	▲	Swift	1.43%	+0.31%
11	9	▼	SQL	1.40%	-0.58%
12	16	▲▲	Go	1.21%	+0.19%

# 1. 파이썬

## ❖ 프로그램의 다양한 응용

### ❖ 스마트폰 사용의 일상 변화

- SNS를 통해 손쉽게 주변 사람들의 소식을 주고 받기
- 몇 번의 터치만으로 은행 업무
- 스마트폰의 지도와 길 찾기 기능

### ❖ 자동차 분야

- 자동차 엔진은 ECU(Engine Control Unit, 엔진 제어 장치)라는 컴퓨터가 제어
- 차선 유지 기능
- 앞 차와의 충돌 방지 기능
- 자율 주행까지 가능한 컴퓨터가 내장
- 구글은 자율 주행 시스템을 위해 웨이모 개발

# 1. 파이썬

## ❖ 프로그램의 다양한 응용

### ❖ 영화 산업 분야

- 컴퓨터 그래픽스가 필수
- 컴퓨터와 3D 모델링 소프트웨어

### ❖ 금융 업계

- 오프라인 지점 없이 온라인으로만 영업하는 인터넷 은행은 나오자마자 큰 돌풍
- 개인 대출 시장도 인터넷을 통해 대출을 연결해주는 P2P 대출로 발전
- 국가 중앙은행의 통제를 받지 않는 비트코인 등 가상화폐까지 등장
- 이들 모두 금융과 소프트웨어가 결합한 핀테크(fintech)

### ❖ 유통 업계

- 미국의 아마존은 인터넷 쇼핑몰을 넘어서서 세계 최대의 클라우드 서비스 업체로 발전
- 국내도 유통 분야에서 인터넷 쇼핑몰이 보편화
- 빅데이터를 활용하여 소비자에게 최적화된 상품을 추천해주는 등 소프트웨어를 적극 활용
- 요즘은 이런 회사들을 유통 업체가 아닌 소프트웨어 업체로 분류

# 1. 파이썬

## ❖ 프로그램의 다양한 응용

### ❖ 생산 분야

- 3D 프린터가 도입되어 다품종 소량 생산 및 자동화가 가능
- 의료 업계 중에서도 이미 치과 보철 분야는 3D 프린터를 사용

### ❖ 의료 분야

- 빅데이터와 인공지능을 통해 최적화된 치료법을 제공
- 일상 생활에서도 스마트 워치로 심박수, 혈당 수치 측정, 칼로리 계산까지 가능
- 의료 정보는 모두 소프트웨어로 처리되며 스마트 헬스케어라는 분야로 자리를 잡았음

### ❖ 인공지능 발전

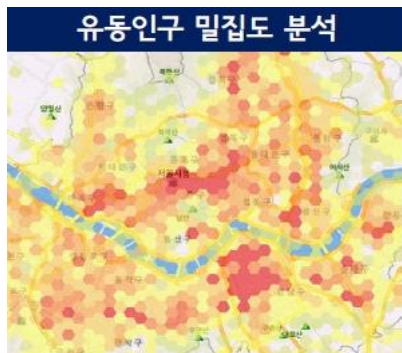
- 인공지능은 사람을 이길 수 없을 것이라 여겨졌던 바둑도 구글 알파고가 나오면서 사람을 압도
- 일상 생활에서는 스마트폰에 내장된 시리와 빅스비 같은 서비스가 활용

# 1. 파이썬

## ❖ 프로그램의 다양한 응용

### ❖ 빅데이터 분야

- 서울시 심야버스 노선 최적화
- 서울시와 KT는 사람들의 휴대전화 사용 위치
- 신용 카드와 교통카드 결제 데이터
- 택시 승하차 정보
- 휴대전화 청구지 주소 등을 분석
- 실제 유동인구를 파악한 뒤 노선을 최적화하여 심야버스 이용율을 크게 늘림

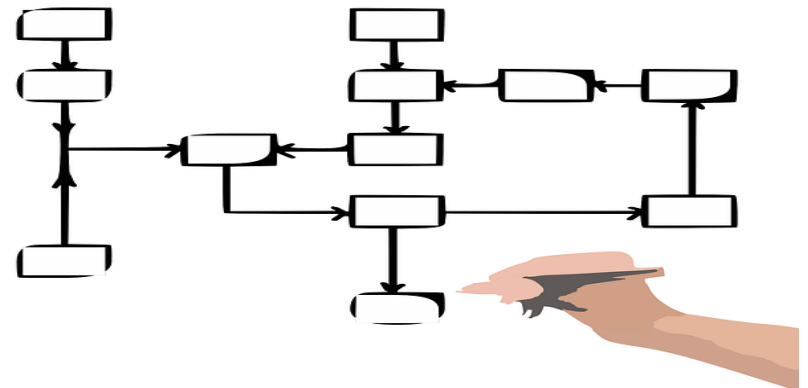
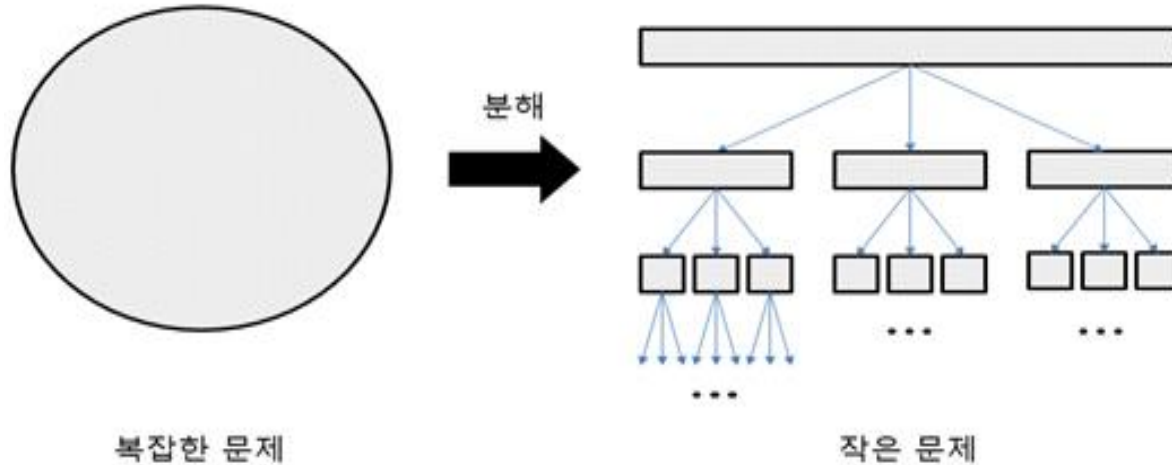


〈핸드폰 데이터 사용량을 이용한 0~5시 사이의 유동인구 밀집지역〉 - 서울시 제공



# 1. 파이썬

## ❖ 문제해결을 위한 과학적이고 논리적인 사고



# 1. 파이썬

## ❖ 프로그래밍 언어

### ■ 프로그램 작성하는 도구의 일종

### ■ 컴파일 언어

- 모든 명령을 일괄 번역, 실행
- 속도 빠른 반면 구조 복잡함

### ■ 인터프리터 언어

- 명령어 만날 때마다 즉시 번역하여 실행
- 속도 느리지만 단순하고 쉬움

구분	컴파일러	인터프리터
작동 방식	소스코드를 기계어로 먼저 번역하고, 해당 플랫폼에 최적화되어 프로그램을 실행함	별도의 번역 과정 없이 소스코드를 실행 시점에 해석하여 컴퓨터가 처리할 수 있도록 함
장점	실행 속도가 빠름	간단히 작성, 메모리가 적게 필요
단점	한 번에 많은 기억 장소가 필요함	실행 속도가 느림
주요 언어	C, 자바(Java), C++, C#	파이썬, 스칼라



# 1. 파이썬

## ❖ 파이썬 (Python)

- 1991년 귀도 반 로섬(Guido van Rossum)이 개발
- 계속해서 개선
  - 2.x 세대와 3.x 세대 공존
  - 2020년부터 2.x는 (End Of Life, 수명 종료)
  - 여러 가지 변형 존재함
- 특징
  - 배우거나 사용하기 쉬움
  - 어느 운영체제에서나 사용 가능함
  - 다운로드 비용이 없음
  - 기본 패키지만으로도 각종 작업 처리 가능함
  - 객체지향적이며 클래스 지원함
  - C 언어와의 접착성 좋아 혼합 프로그래밍 가능함

### 느린 속도

파이썬은 컴파일러 언어가 아닌 스크립트 언어이기 때문에 컴파일러 언어보다 느림

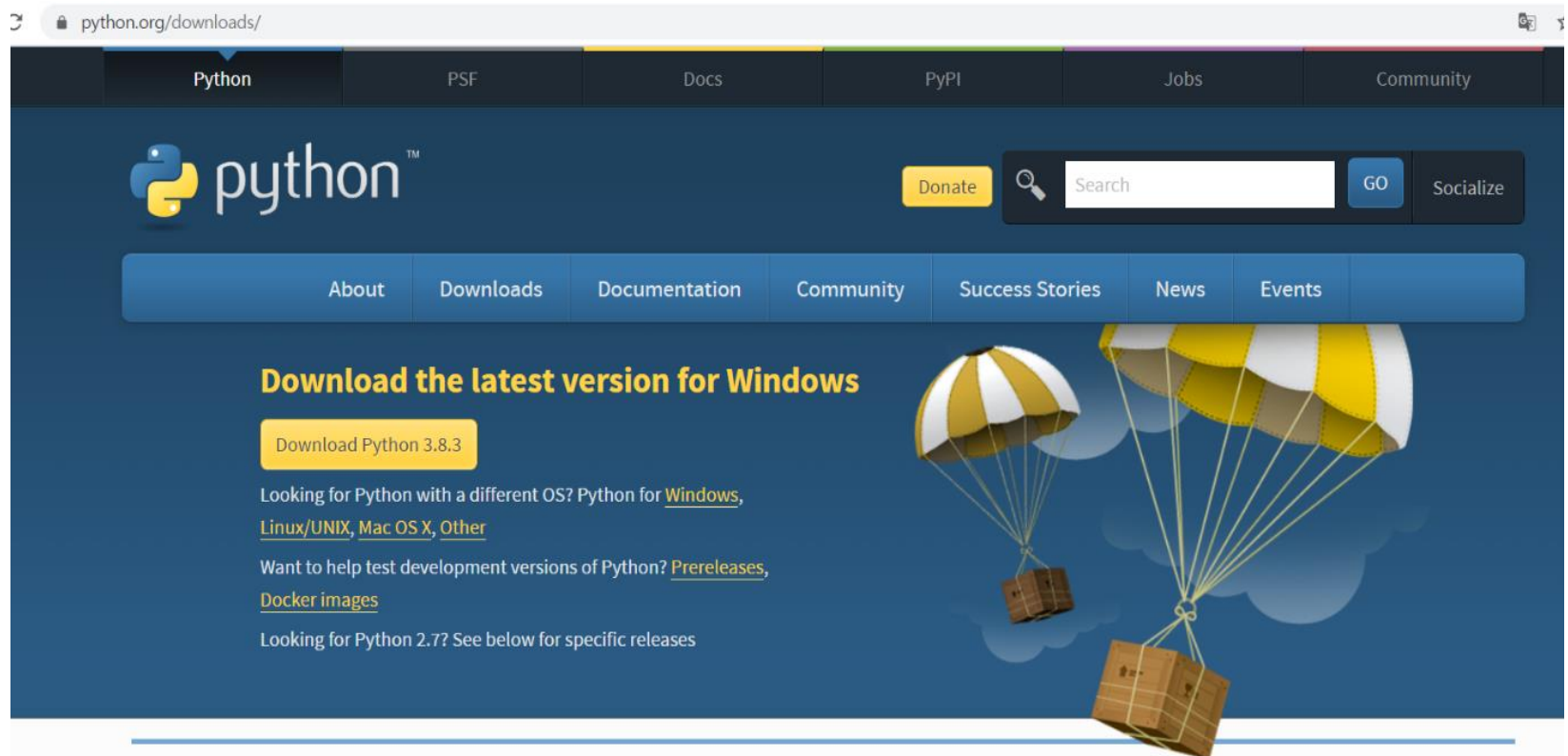
→ 이를 보완하려고 많은 파이썬 패키지를 최적화시키고 있음

모바일 컴퓨팅 분야에 지원이 약하고 하드웨어 제어 등과 관련된 부분 사용이 어려움

## 2. 개발환경

### ❖ 파이썬 설치

- <https://www.python.org/downloads/> 에서 다운로드



Active Python Releases

For more information visit the Python Developer's Guide.

## 2. 개발환경

### ❖ 파이썬 설치

- <https://www.python.org/downloads/> 에서 다운로드

#### Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
<a href="#">Gzipped source tarball</a>	Source release		4d5b16e8c15be38eb0f4b8f04eb68cd0	23276116	<a href="#">SIG</a>
<a href="#">XZ compressed source tarball</a>	Source release		a224ef2249a18824f48fba9812f4006f	17399552	<a href="#">SIG</a>
<a href="#">macOS 64-bit installer</a>	Mac OS X	for OS X 10.9 and later	2819435f3144fd973d3dea4ae6969f6d	29303677	<a href="#">SIG</a>
<a href="#">Windows help file</a>	Windows		65bb54986e5a921413e179d2211b9bfb	8186659	<a href="#">SIG</a>
<a href="#">Windows x86-64 embeddable zip file</a>	Windows	for AMD64/EM64T/x64	5ae191973e00ec490cf2a93126ce4d89	7536190	<a href="#">SIG</a>
<a href="#">Windows x86-64 executable installer</a>	Windows	for AMD64/EM64T/x64	70b08ab8e75941da7f5bf2b9be58b945	26993432	<a href="#">SIG</a>
<a href="#">Windows x86-64 web-based installer</a>	Windows	for AMD64/EM64T/x64	b07dbb998a4a0372f6923185ebb6bf3e	1363056	<a href="#">SIG</a>
<a href="#">Windows x86 embeddable zip file</a>	Windows		5f0f83433bd57fa55182cb8ea42d43d6	6765162	<a href="#">SIG</a>
<a href="#">Windows x86 executable installer</a>	Windows		4a9244c57f61e3ad2803e900a2f75d77	25974352	<a href="#">SIG</a>
<a href="#">Windows x86 web-based installer</a>	Windows		642e566f4817f118abc38578f3cc4e69	1324944	<a href="#">SIG</a>

## 2. 개발환경

### ❖ 파이썬 설치

- <https://www.python.org/downloads/> 에서 다운로드

For more information visit the Python Developer's Guide.

Python version	Maintenance status	First released	End of support	Release schedule
3.8	bugfix	2019-10-14	2024-10	PEP 569
3.7	bugfix	2018-06-27	2023-06-27	PEP 537
3.6	security	2016-12-23	2021-12-23	PEP 494
3.5	security	2015-09-13	2020-09-13	PEP 478
2.7	end-of-life	2010-07-03	2020-01-01	PEP 373

Looking for a specific release?

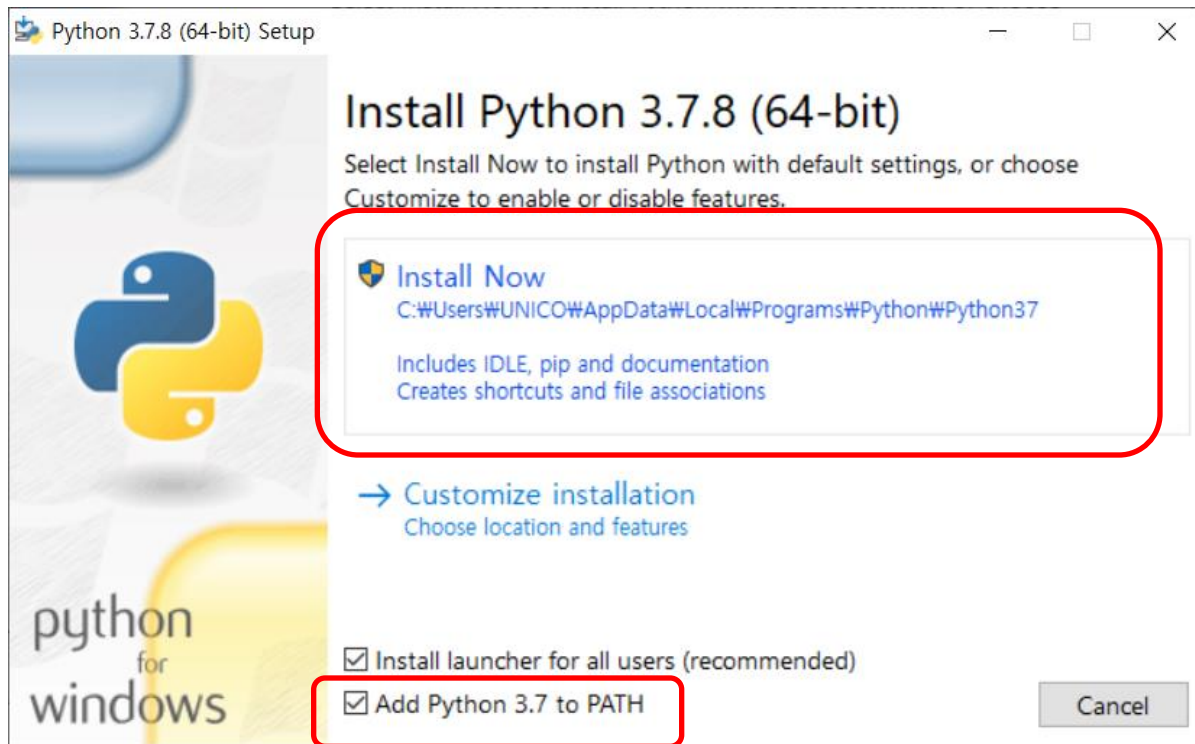
Python releases by version number:

Release version	Release date	Click for more	
<a href="#">Python 3.7.8</a>	June 27, 2020	 Download	<a href="#">Release Notes</a>
<a href="#">Python 3.6.11</a>	June 27, 2020	 Download	<a href="#">Release Notes</a>
<a href="#">Python 3.8.3</a>	May 13, 2020	 Download	<a href="#">Release Notes</a>
<a href="#">Python 2.7.18</a>	April 20, 2020	 Download	<a href="#">Release Notes</a>
<a href="#">Python 3.7.7</a>	March 10, 2020	 Download	<a href="#">Release Notes</a>
<a href="#">Python 3.8.2</a>	Feb. 24, 2020	 Download	<a href="#">Release Notes</a>

## 2. 개발환경

### ❖ 파이썬 설치

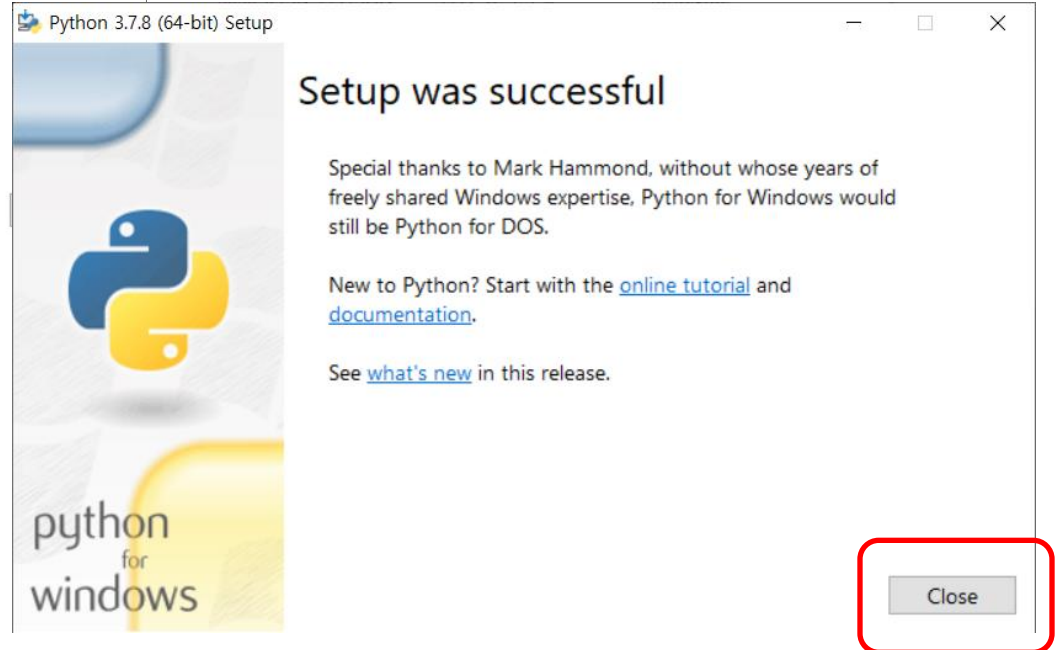
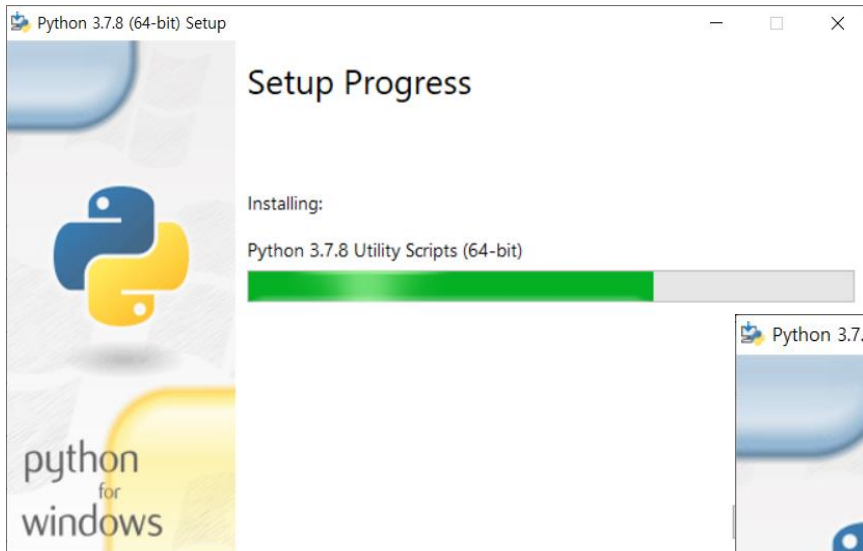
- python-3.7.8-amd64.exe 파일을 더블 클릭하여 설치시작



## 2. 개발환경

### ❖ 파이썬 설치

- python-3.7.8-amd64.exe 파일을 더블 클릭하여 설치시작

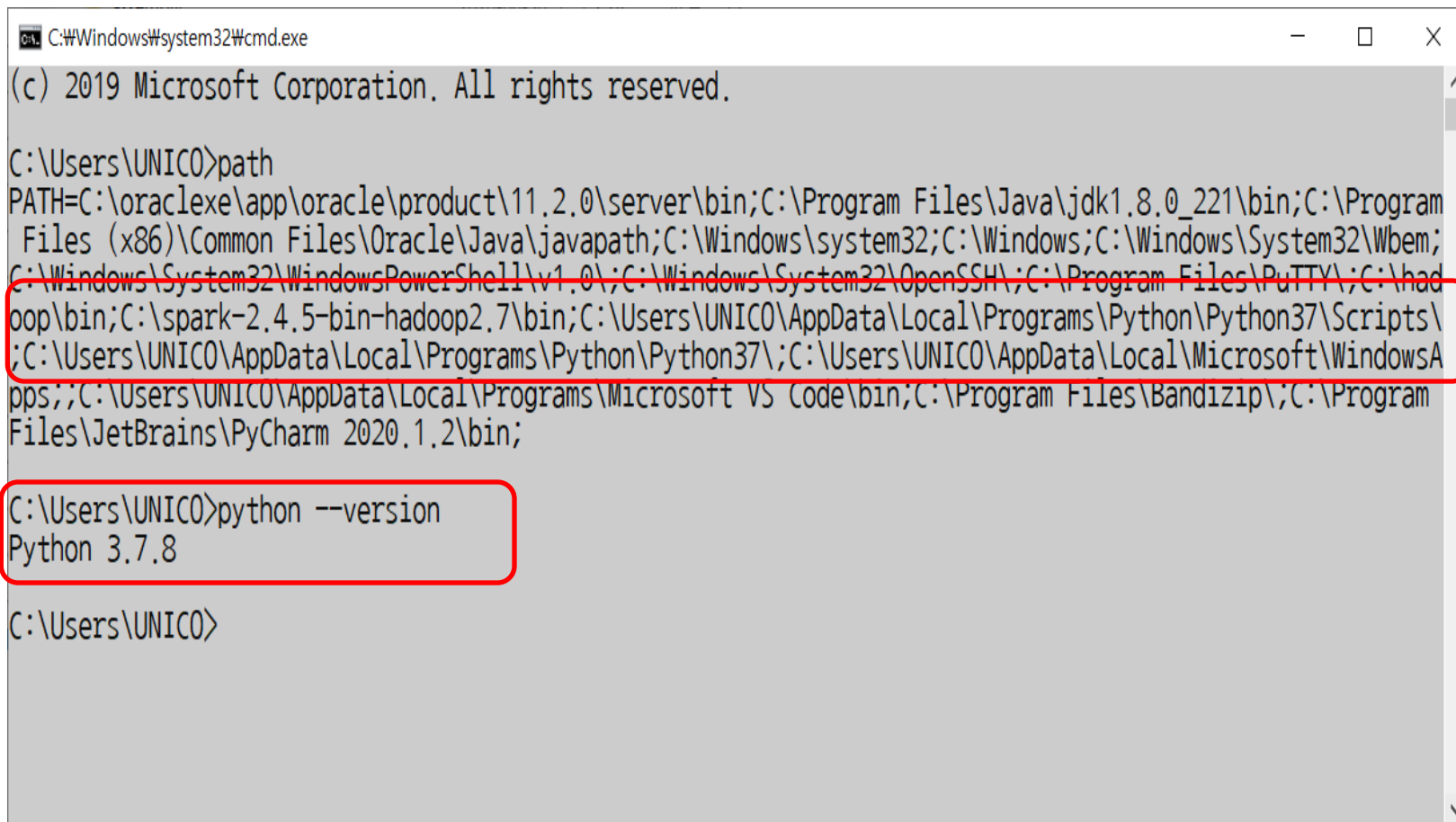




## 2. 개발환경

### ❖ 파이썬 설치

- 패스 설정이 잘 되었는지 확인



```
C:\Windows\system32\cmd.exe
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\UNICO>path
PATH=C:\oracle\app\oracle\product\11.2.0\server\bin;C:\Program Files\Java\jdk1.8.0_221\bin;C:\Program Files (x86)\Common Files\Oracle\Java\javapath;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Windows\System32\OpenSSH\;C:\Program Files\PuTTY\;C:\hadoop\bin;C:\spark-2.4.5-bin-hadoop2.7\bin;C:\Users\UNICO\AppData\Local\Programs\Python\Python37\Scripts\;C:\Users\UNICO\AppData\Local\Programs\Python\Python37\;C:\Users\UNICO\AppData\Local\Microsoft\WindowsApps\;C:\Users\UNICO\AppData\Local\Programs\Microsoft VS Code\bin;C:\Program Files\Bandizip\;C:\Program Files\JetBrains\PyCharm 2020.1.2\bin;

C:\Users\UNICO>python --version
Python 3.7.8

C:\Users\UNICO>
```

## 2. 개발환경

### ❖ 파이썬 설치

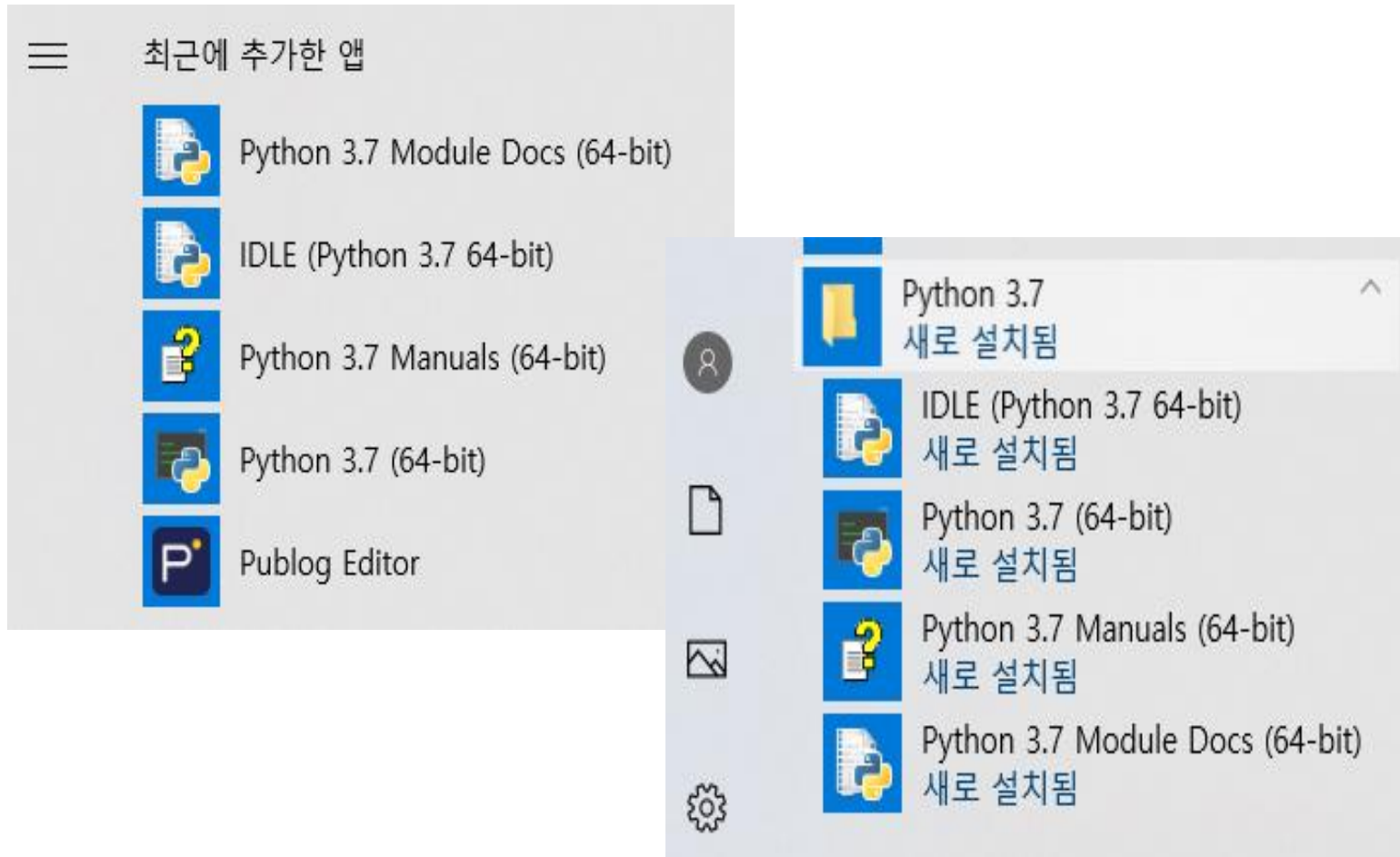
#### ■ 파이썬 설치 디렉토리

> 로컬 디스크 (C:) > 사용자 > UNICO > AppData > Local > Programs > Python > Python37 >				
이름	수정한 날짜	유형	크기	
DLLs	2020-07-01 오전 8:05	파일 폴더		
Doc	2020-07-01 오전 8:05	파일 폴더		
include	2020-07-01 오전 8:05	파일 폴더		
Lib	2020-07-01 오전 8:05	파일 폴더		
libs	2020-07-01 오전 8:05	파일 폴더		
Scripts	2020-07-01 오전 8:05	파일 폴더		
tcl	2020-07-01 오전 8:05	파일 폴더		
Tools	2020-07-01 오전 8:05	파일 폴더		
LICENSE.txt	2020-06-28 오전 8:57	텍스트 문서	30KB	
NEWS.txt	2020-06-28 오전 8:57	텍스트 문서	728KB	
python.exe	2020-06-28 오전 8:55	응용 프로그램	98KB	
python3.dll	2020-06-28 오전 8:54	응용 프로그램 확장	58KB	
python37.dll	2020-06-28 오전 8:54	응용 프로그램 확장	3,667KB	
pythonw.exe	2020-06-28 오전 8:55	응용 프로그램	97KB	
vcruntime140.dll	2019-05-22 오후 9:32	응용 프로그램 확장	84KB	

## 2. 개발환경

### ❖ 파이썬 프로그램 개발 환경

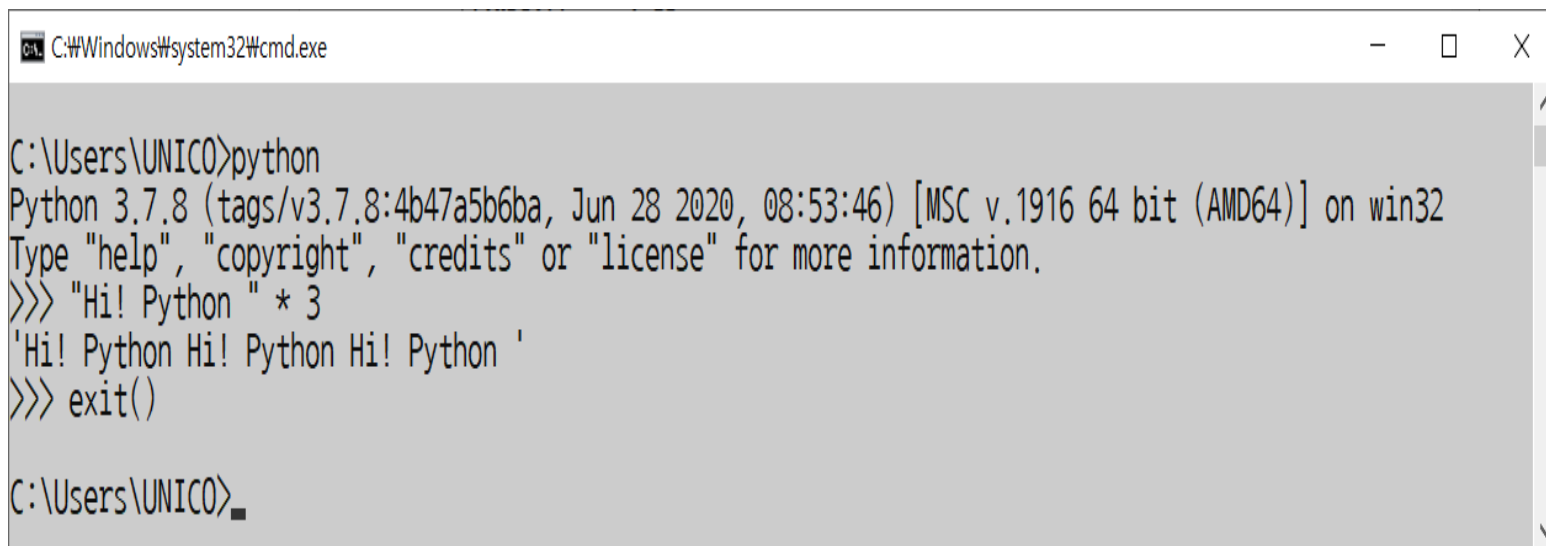
#### ■ 시작 메뉴



## 2. 개발환경

### ❖ 파이썬 실행

- python.exe 파일
- 명령행에서 실행할 수도 있음
- 대화식 모드
  - 명령 내리면 결과 즉시 보여주고 다음 명령 대기



```
C:\Windows\system32\cmd.exe

C:\Users\UNICO>python
Python 3.7.8 (tags/v3.7.8:4b47a5b6ba, Jun 28 2020, 08:53:46) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> "Hi! Python " * 3
'Hi! Python Hi! Python Hi! Python '
>>> exit()

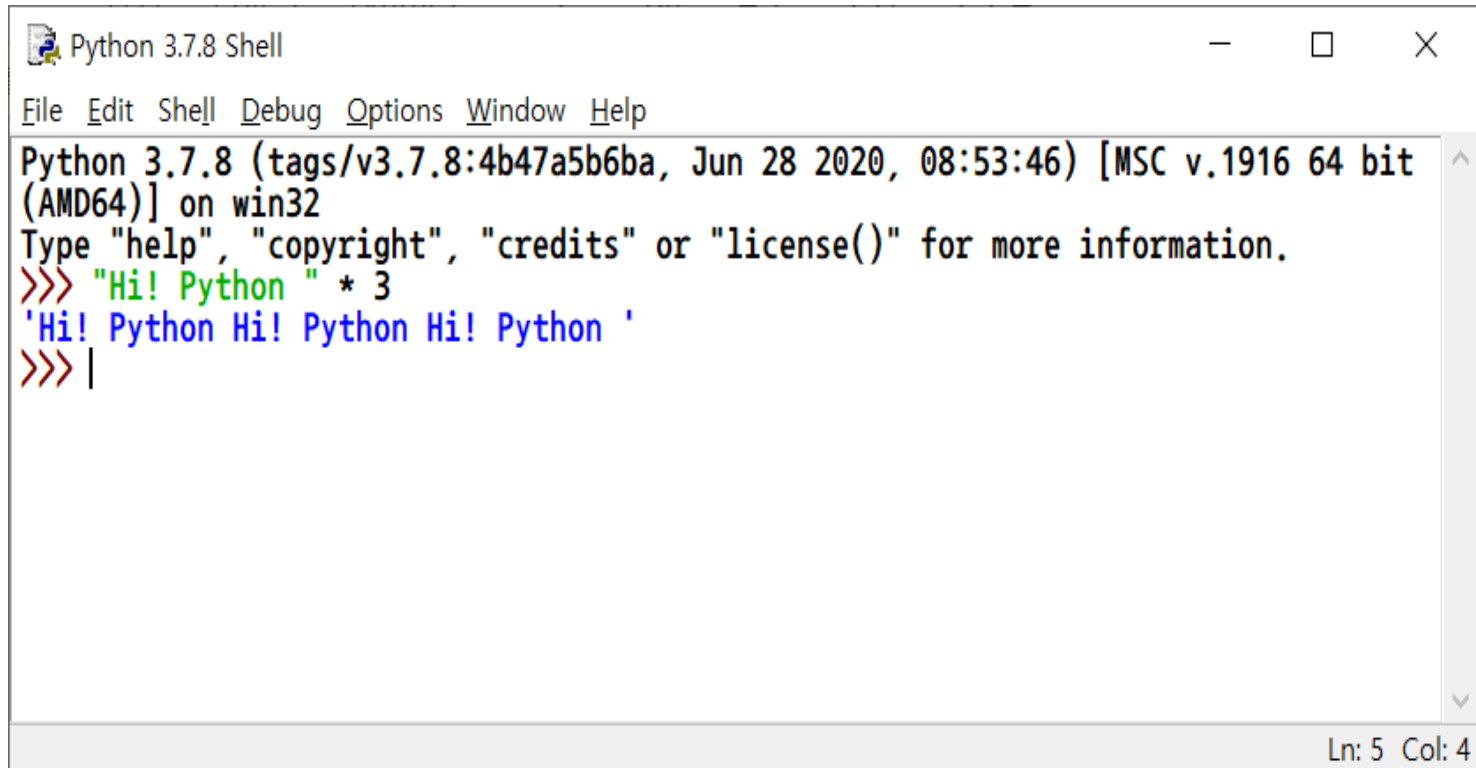
C:\Users\UNICO>
```

- >>> 프롬프트에 파이썬 명령을 입력

## 2. 개발환경

### ■ IDLE 이용하여 실행

- 명령행은 각종 부가기능 등 없어 비효율적

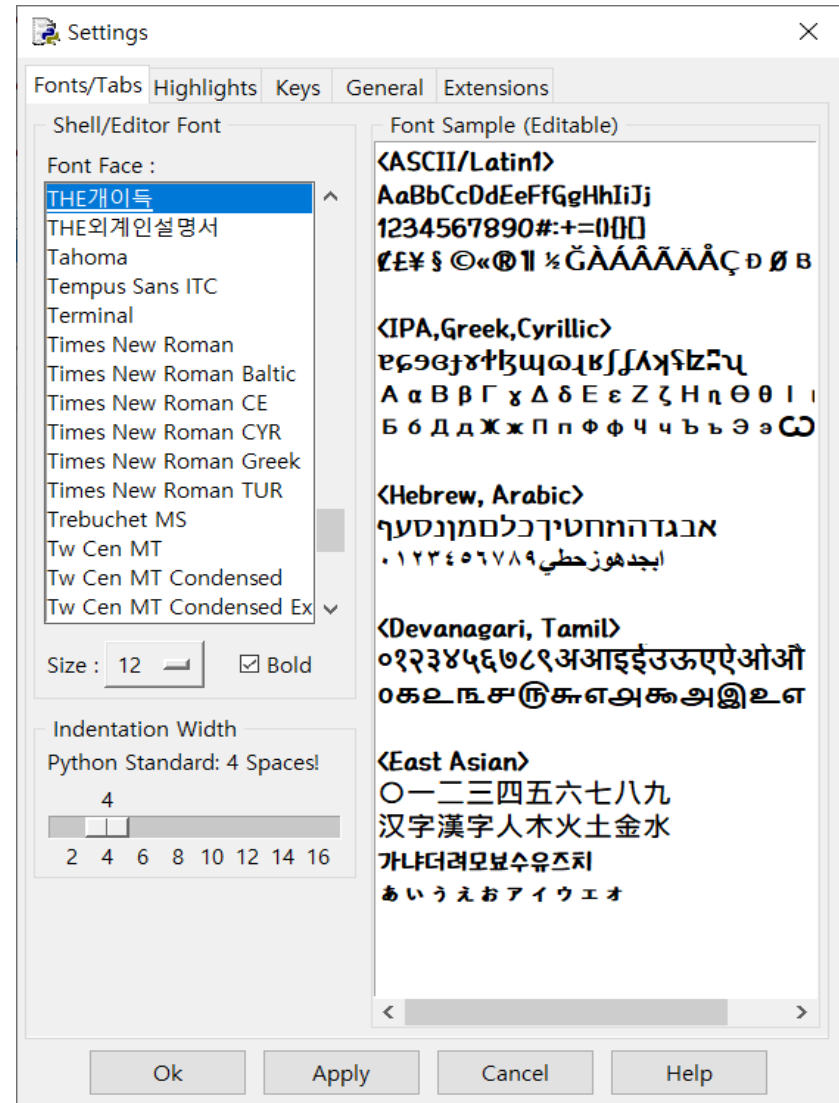
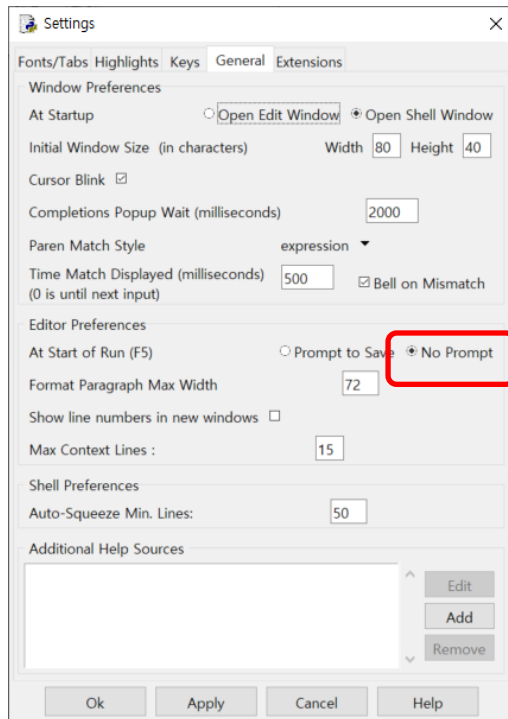
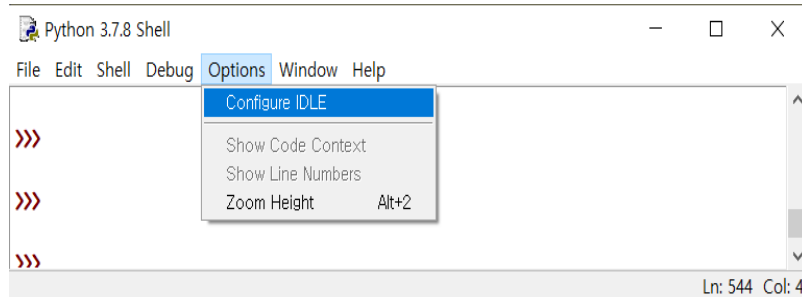


```
Python 3.7.8 Shell
File Edit Shell Debug Options Window Help
Python 3.7.8 (tags/v3.7.8:4b47a5b6ba, Jun 28 2020, 08:53:46) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> "Hi! Python " * 3
'Hi! Python Hi! Python Hi! Python '
>>> |
```

Ln: 5 Col: 4

## 2. 개발환경

- 메뉴에서 [Options/Configure IDLE] 클릭



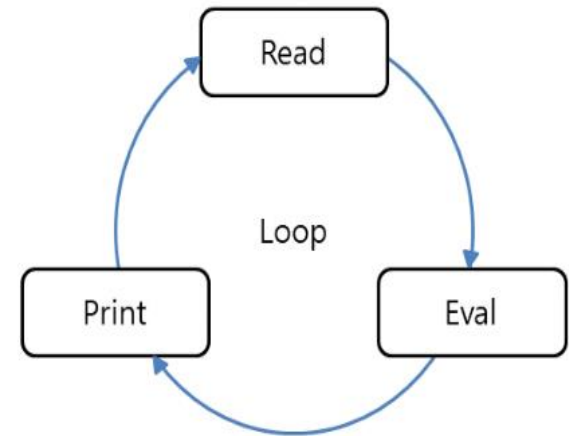
### 3. 파이썬 실행 모드

#### ❖ 대화식(Interactive) 모드-REPL(Read-Eval-Print Loop)

- 각 명령 입력 즉시 응답

- >>> 프롬프트 뒤에 명령 입력하면 바로 결과 나오고 다시 프롬프트 표시

```
>>> 2 * 3
6
>>>
```



- 잘못된 명령일 경우 해석기가 오류메시지로 알림

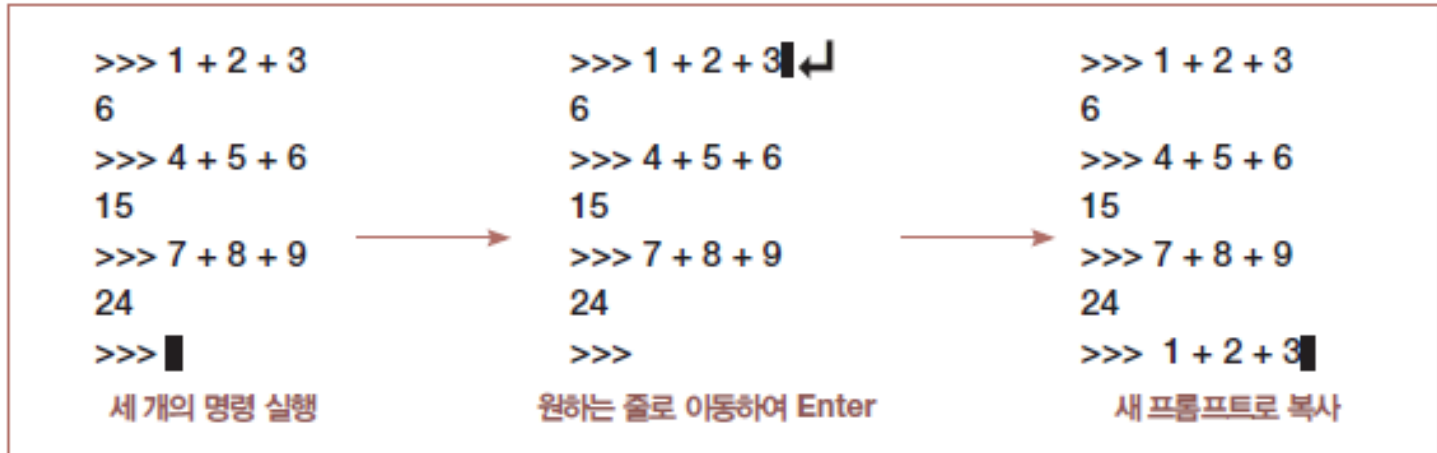
```
>>> 2 = 3
SyntaxError: can't assign to literal
>>>
```

- 각종 복잡한 수식과 조건문, 반복문 명령도 실행할 수 있음

### 3. 파이썬 실행 모드

#### ■ 앞서 내린 명령 재활용 가능

- 다시 수행할 명령으로 이동하여 [Enter] 키 누르기



#### ■ 단점

- 길고 복잡한 프로그램 짜기에 불편함
- 명령 저장 불가
- 에러 수정도 번거로움



### 3. 파이썬 실행 모드

#### ❖ 스크립트(Script) 모드

- 텍스트 파일에 일련의 명령 작성하여 한꺼번에 순차적으로 실행

- 스크립트

- 명령어들을 작성해 놓은 텍스트 파일(소스파일)
- IDLE 내장 편집기 사용
  - 입력 후 [File] – [Save]
  - [Run] – [Run Module]
  - » 실행 결과 출력

```
for y in range(1, 10) :  
    for x in range(y) :  
        print('*', end = ' ')  
    print()
```

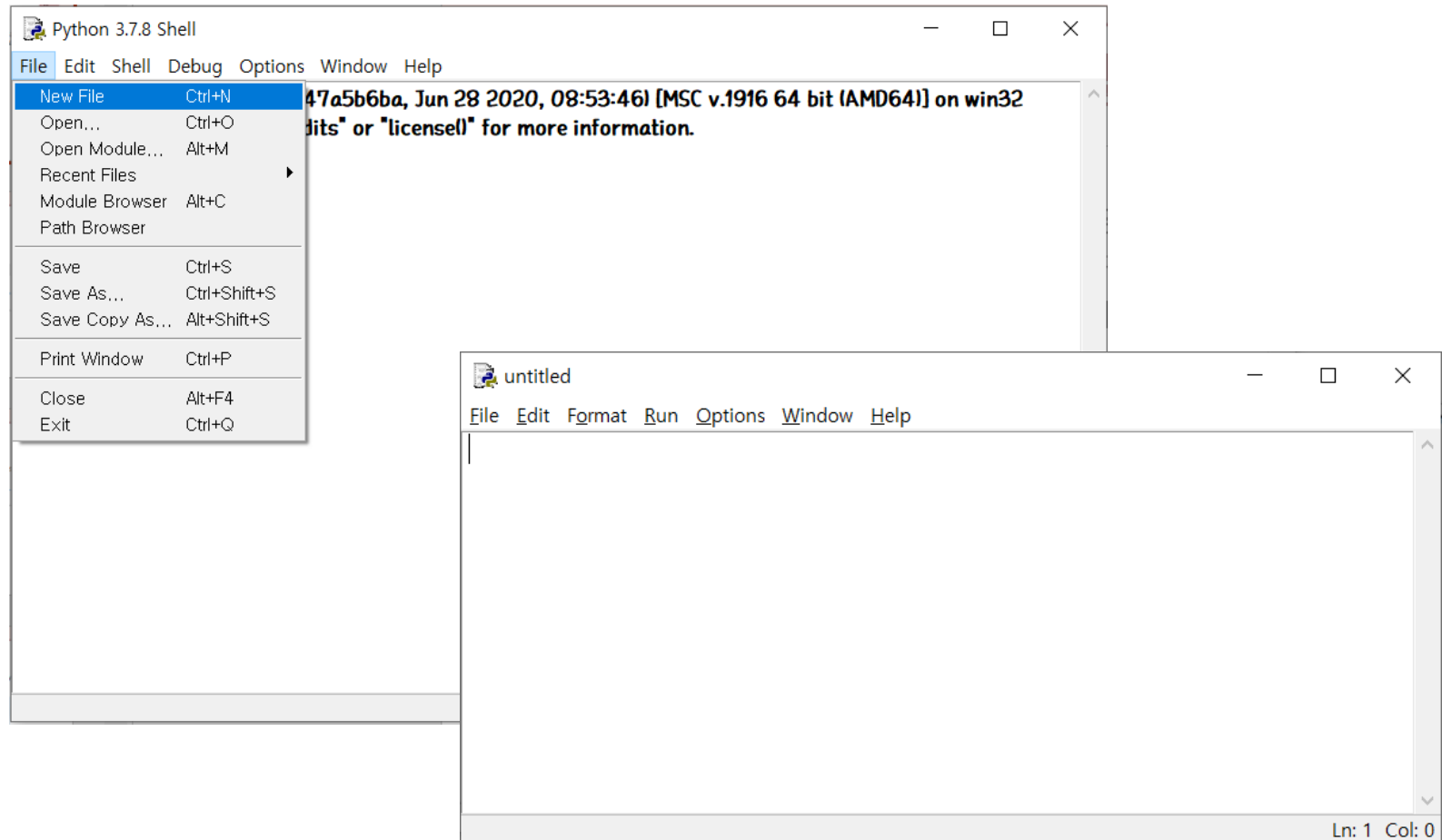


```
Python 3.6.4 Shell  
File Edit Shell Debug Options Window Help  
>>>  
===== RESTART: C:/PyStudy/First.py =====  
*  
**  
***  
****  
*****  
*****  
*****  
*****  
*****  
*****  
*****  
>>> |  
Ln: 41 Col: 4
```

### 3. 파이썬 실행 모드

#### ❖ 스크립트(Script) 모드

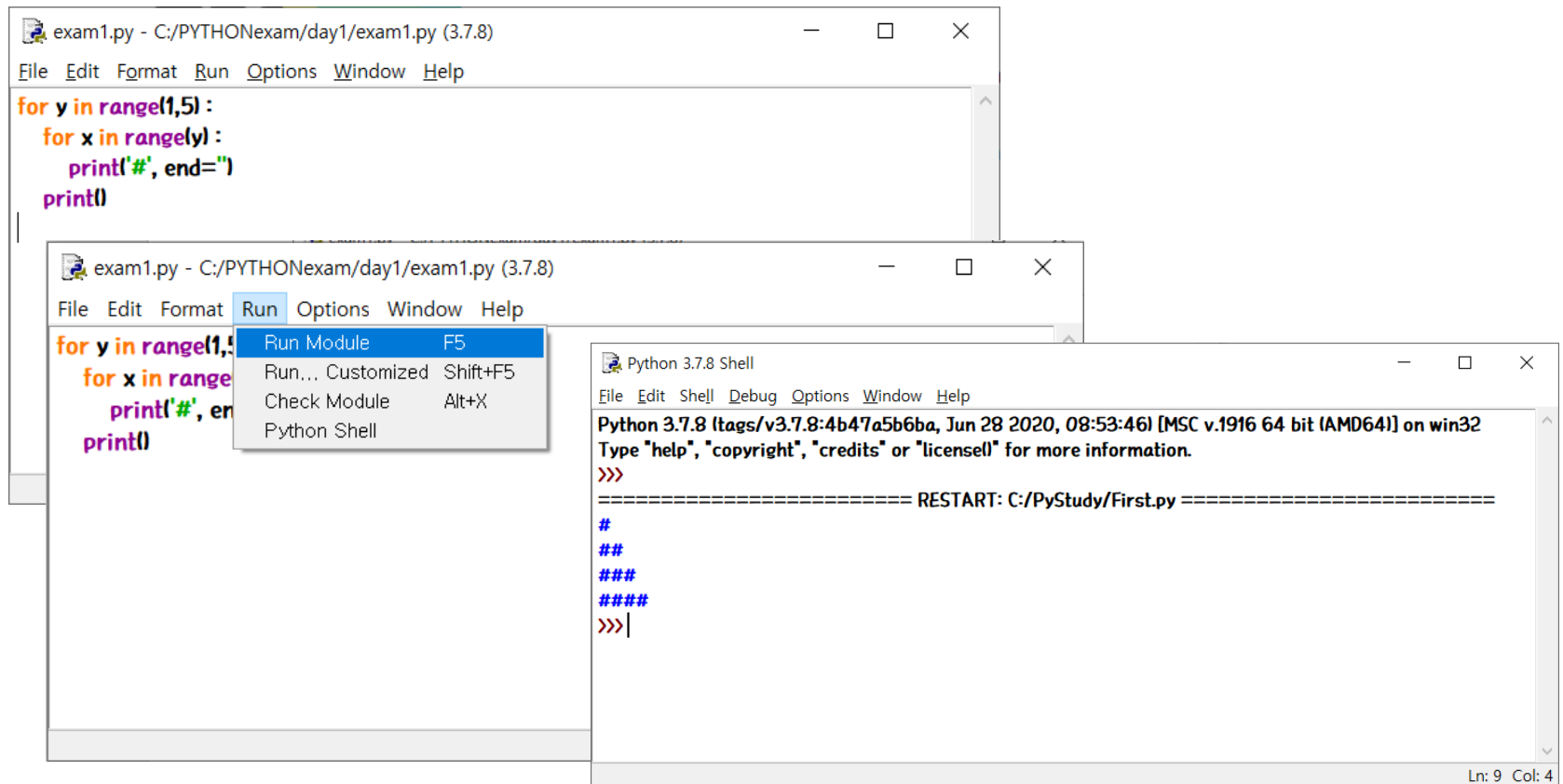
- IDLE을 실행하며 IDL Shell 창이 출력됨. 메뉴에서 File > New File을 클릭



### 3. 파이썬 실행 모드

#### ❖ 스크립트(Script) 모드

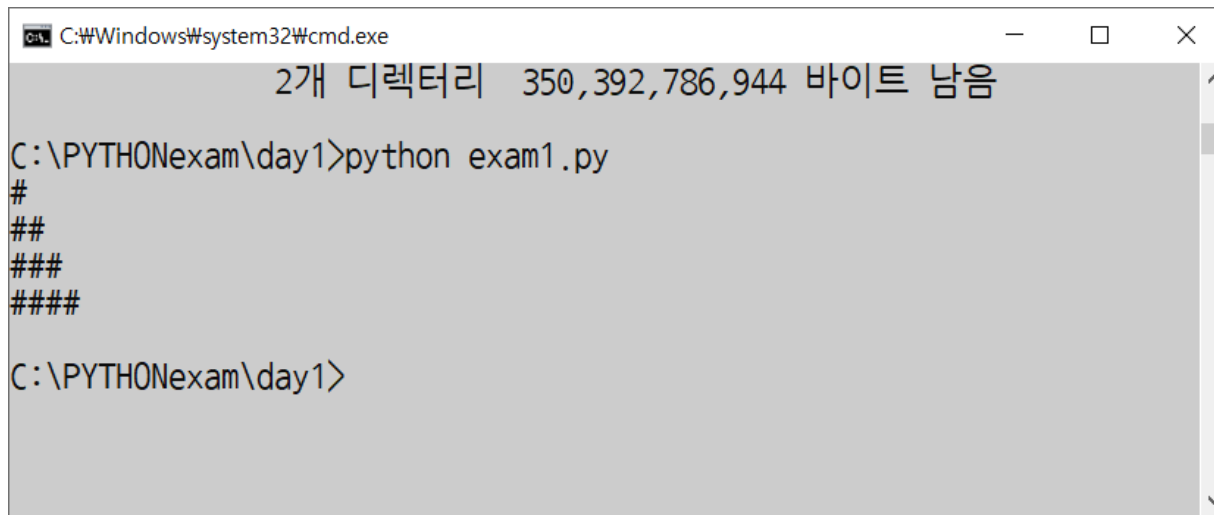
- IDLE의 편집기에서 Run > Run Module 을 클릭하거나 F5 키를 입력하여 실행하면 IDL Shell 창에서 실행 결과가 출력됨



### 3. 파이썬 실행 모드

#### ❖ 스크립트(Script) 모드

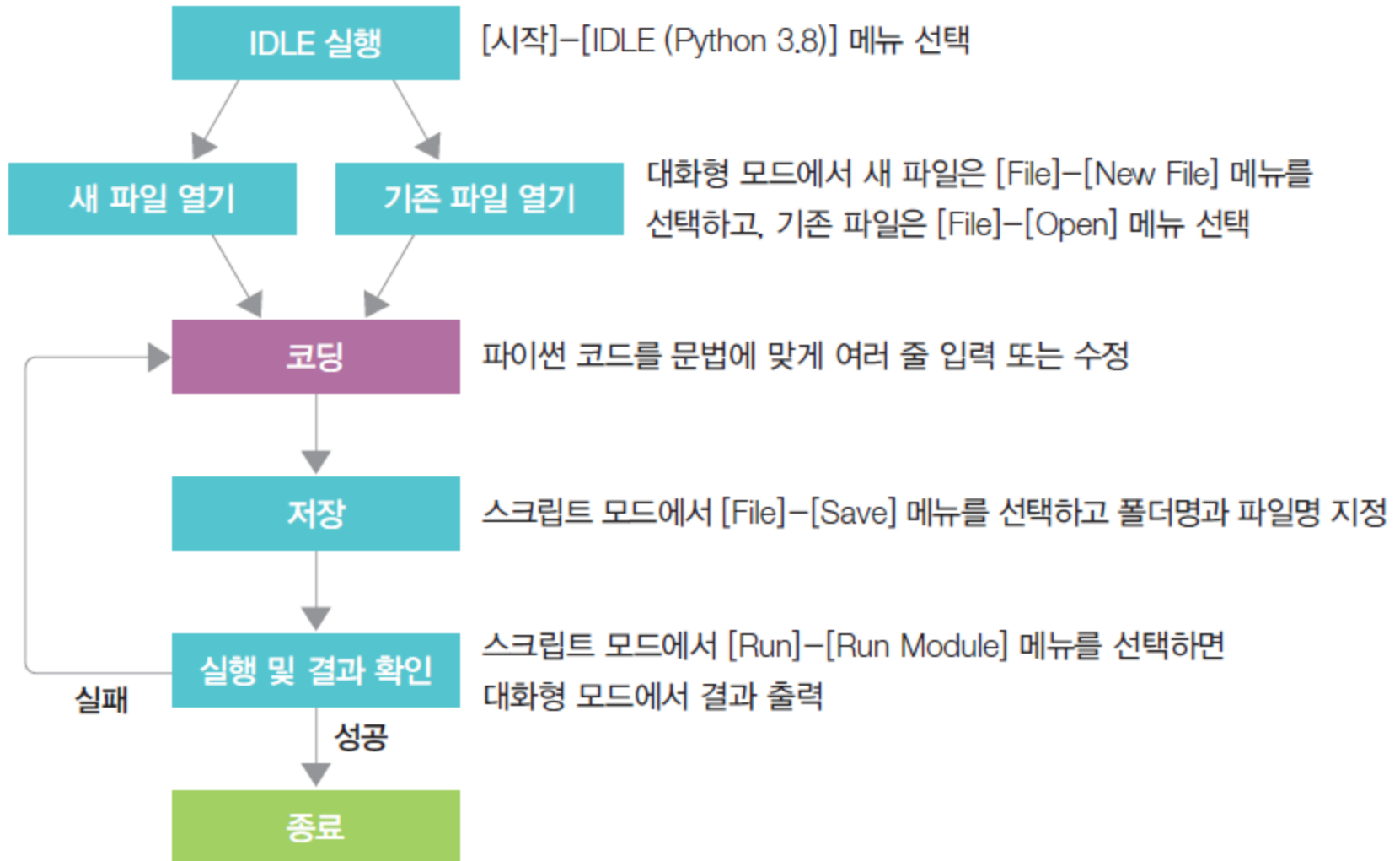
- 생성된 파이썬 파일을 cmd(명령 프롬프트) 창에서 명령으로 실행



```
C:\Windows\system32\cmd.exe
2개 디렉터리 350,392,786,944 바이트 남음
C:\PYTHONexam\day1>python exam1.py
#
##
###
####
C:\PYTHONexam\day1>
```

### 3. 파이썬 실행 모드

#### ❖ 스크립트(Script) 모드



# Contents

## ❖ 목차

- 1. 기본 구조
- 2. 변수

# 1. 기본 구조

## ❖ 소스의 형식

- 일정한 규칙에 따라 스크립트 파일 코드를 작성

### ■ 들여쓰기 (Indent)

- [Tab] 혹은 4개 공백
- 일반적으로 >>> 프롬프트의 첫 칸부터 명령을 입력
- 텍스트 파일에 스크립트 작성 시 역시 첫 칸부터 입력
- 조건문, 반복문 등 여러 개 문장이 블록 구성하는 경우
  - 들여쓰기 통해 아래쪽 문장이 조건, 반복 대상임을 표시

```
if age > 19:  
    print("성인입니다")  
4칸 들여쓰다
```

```
for a in range(5):  
    print(a)  
4칸 들여쓰다
```

# 1. 기본 구조

- 파이썬은 개행과 들여쓰기를 규칙에 맞도록 해야 함
  - 연산자와 피연산자 사이의 공백은 무관

```
>>> 1+2
>>> 1 + 2
```

## ■ 한 줄에 하나의 명령 작성

- 세미콜론으로 한 줄에 모두 작성할 수 있으나 지양하는 것을 권장함

## ■ 대문자와 소문자의 구분

- 명령어나 함수 등은 대부분 소문자로 되어 있음

```
>>> print(3 + 4) # 맞음
>>> Print(3 + 4) # 틀림
>>> PRINT(3 + 4) # 틀림
```



# 1. 기본 구조

## ■ # 문자로 주석 달 수 있음

- 명령어가 아닌, 사용자 위한 설명 문장
- 해석기는 주석 부분을 무시

```
# 별을 출력하는 코드. 2018년 3월 1일 권성직이 만들
for y in range(10) :          # 0~10까지 반복한다.
    for x in range(y) :       # 문종민 과장, 이 코드 좀 검토해 주세요.
        print('*', end = '')  # * 문자를 출력한다.
    print()                   # 개행한다.
```

# 1. 기본 구조

## ❖ 출력

### ■ print 명령

- `print(출력 내용 [, sep=구분자] [, end=끝 문자])`
- 괄호 안에 상수, 변수, 수식 등 출력 내용 입력

```
>>> print(3 + 4)
7
```

- 스크립트 모드에서 값 출력 시에는 반드시 print 명령 사용
- 출력할 내용 여러 개일 경우 콤마로 나열

```
>>> a = 12
>>> b = 34
>>> print(a, b)
12 34
```

# 1. 기본 구조

- 여러 개 출력 결과를 공백이나 구분자(Separator) 사용하여 구별

```
>>> print(a, b, sep = ',')  
12,34
```

printsep

```
s = '서울'  
d = '대전'  
g = '대구'  
b = '부산'  
print(s, d, g, b, sep = ' 찍고 ')
```

실행결과

서울 찍고 대전 찍고 대구 찍고 부산

- print는 출력한 후 다음 줄로 자동 개행

printtwo

```
a = '강아지'  
b = '고양이'  
print(a)  
print(b)
```

실행결과

강아지  
고양이

# 1. 기본 구조

## ❖ 입력

- 사용자에게 질문하여 값 입력받기
- **input 명령**
  - 변수 = input('질문 내용')
  - 괄호 안에 질문 내용 입력
  - 사용자가 질문에 대해 입력한 값 돌려줌
- **int() 함수**
  - 입력받은 문자열을 정수로 바꿈

```
>>> age = input('몇 살이세요? ')
몇 살이세요? 29
>>> print(age)
29
```

### intinput

```
price = input('가격을 입력하세요 : ')
num = input('개수를 입력하세요 : ')
sum = int(price) * int(num)
print('총액은', sum, '원입니다')
```

#### 실행결과

```
가격을 입력하세요 : 100
개수를 입력하세요 : 5
총액은 500 원입니다
```

# 1. 기본 구조

- 처음부터 정수를 입력 받으려면 input() 호출문을 int()로 감싼다
  - 변수 = int(input('질문 내용'))

intinput2

```
price = int(input('가격을 입력하세요 : '))  
num = int(input('개수를 입력하세요 : '))  
sum = price * num  
print('총액은', sum, '원입니다')
```

## 2. 변수

### ❖ 변수명

#### ■ 변수

- 메모리에 이름 붙이고 값을 저장하는 것

#### ■ 명칭 (Identifier)

- 변수가 다른 것과 구분되도록 이름을 붙인 것

#### ■ 규칙

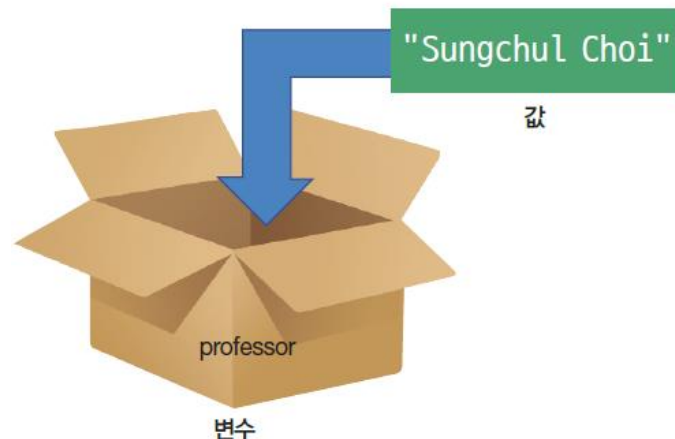
- 키워드나 내장 함수, 표준 모듈명은 사용할 수 없음

```
>>> import keyword
```

```
>>> keyword.kwlist
```

```
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', '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']
```

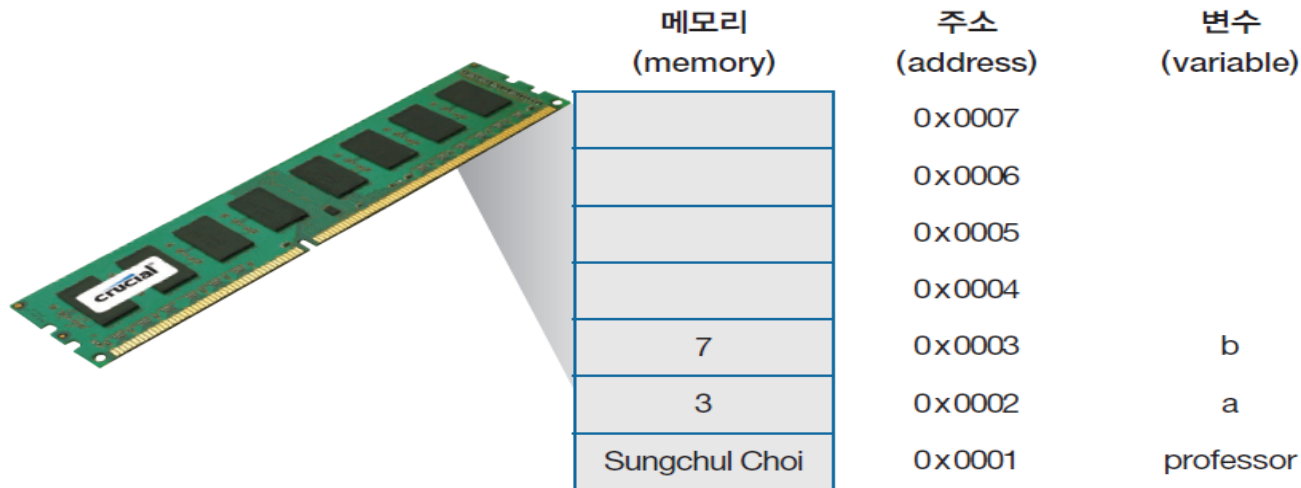
- 모든 명칭은 대소문자를 구분함
- 알파벳, 밑줄 문자, 숫자 등으로 구성되며 그 외의 공백, +, -, \*, # 등 특수문자들은 사용 불가
- 첫 글자로 숫자 사용 불가
- 한글이나 한자 사용 가능(코드 가독성을 위해서 가급적 지양하는 것이 좋음)



## 2. 변수

### ❖ 변수와 메모리

- professor = "Sungchul Choi", a = 3, b = 7과 같은 변수를 선언하면, 아래 그림과 같이 메모리 어딘가에 주소 값을 할당받아 저장한다.



- 프로그래밍에서 변수는 **어떠한 값을 저장하는 메모리상의 장소**라는 뜻으로 사용된다.
- 변수에 값을 넣으라고 선언하는 순간, 물리적으로 메모리 어딘가에 공간을 확보할 수 있게 운영체제와 파이썬 인터프리터가 협력하여 메모리 저장 위치를 할당한다. 이 위치를 메모리 주소라고 한다.

## 2. 변수

### ❖ 변수 사용

- 파이썬 변수는 타입을 지정하지 않음
  - 대입하는 값에 의해 결정

```
>>> score = 98
>>> print(score)
98
```

```
>>> score='high'
>>> print(score)
high
```

- 동적 타입 (Dynamic Type)
  - 실행 중에 변수 타입 바꿀 수 있음
  - 현재 할당하고 있는 값에 의해 변수의 타입이 정해 짐
- 변수는 일단 만들어지면 계속 존재하게 되며 값을 유지함
  - del 명령으로 삭제

```
>>> score = 98
>>> print(score)
98
>>> print('score')
score
>>> print(score1)
Traceback (most recent call last):
  File "<pyshell#254>", line 1, in <module>
    print(score1)
NameError: name 'score1' is not defined
```



# Contents

## ❖ 목차

- 1. 기본 자료형
- 2. 수치형
- 3. 문자열

# 1. 기본 자료형

## ❖ 기본 자료형

- 정수형(integer type) : 자연수를 포함해 값의 영역이 정수로 한정된 값.
- 실수형(floating-point type) : 소수점이 포함된 값.
- 문자형(string type) : 값이 문자로 출력되는 자료형.
- 불린형(boolean type) : 논리형으로, 참(True) 또는 거짓(False)을 표현할 때 사용.

유형	자료형	설명	예	선언 형태
수치형	정수형	양수와 정수	1, 2, 3, 100, -9	data = 1
	실수형	소수점이 포함된 실수	10.2, -9.3, 9.0	data = 9.0
문자형	문자형	따옴표에 들어가 있는 문자형	abc, a20abc	data = 'abc'
논리형	불린형	참 또는 거짓	True, False	data = True

# 1. 기본 자료형

## ❖ 기본 자료형

```
>>> a = 1                # 정수형
>>> b = 1                # 정수형
>>> print(a, b)
1 1
>>> a = 1.5              # 실수형
>>> b = 3.5              # 실수형
>>> print(a, b)
1.5 3.5
>>> a = "ABC"            # 문자형
>>> b = "101010"         # 문자형
>>> print(a, b)
ABC 101010
>>> a = True             # 불린형
>>> b = False            # 불린형
>>> print(a, b)
True False
```

# 1. 기본 자료형

## ❖ 기본 자료형

```
>>> total = 10
>>> print(total)
10
>>> print(type(total))
<class 'int'>
>>> total = True
>>> print(total)
True
>>> print(type(total))
<class 'bool'>
```

```
>>> total = '가나다'
>>> print(total)
가나다
>>> print(type(total))
<class 'str'>
>>> total = 3.14
>>> print(total)
3.14
>>> print(type(total))
<class 'float'>
```

## 2. 수치형

### ❖ 정수형

- 가장 간단한 수치형
- 소수점 이하 값은 표현할 수 없음
- 10진수 아닌 정수는 앞에 접두어를 붙여 진법 지정, 표시

```
>>> a = 1234567890
>>> print(a)
1234567890
```

진법	접두	사용 가능한 숫자	예
16진법(hexadecimal)	0x	0~9, a~f	0x2f
8진법(octal)	0o	0~7	0o17
2진법(binary)	0b	0, 1	0b1101

```
>>> a = 0x1a
>>> print(a)
26
```

## 2. 수치형

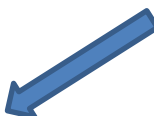
### ❖ 정수형(진법에 따른 숫자 인식 : 교재 63페이지)

$0x1a \rightarrow 16^1 * 1 + 16^0 * 10$

$0x1a \rightarrow 0b00011010 \rightarrow 2^4 * 1 + 2^3 * 1 + 2^1 * 1 \rightarrow 16 + 8 + 2 \rightarrow 26$

$0b1101 \rightarrow 2^3 + 2^2 + 2^0 \rightarrow 8 + 4 + 1 \rightarrow 13$

```
>>> letter = 'A'
>>> print(letter)
A
>>> hexaLetterCode = hex(letterCode)
>>> print(hexaLetterCode)
0x41
>>> octalLetterCode = oct(letterCode)
>>> print(octalLetterCode)
0o101
>>> binaryLetterCode = bin(letterCode)
>>> print(binaryLetterCode)
0b1000001
```



```
>>> letterCode = ord(letter)
>>> print(letterCode)
65
```

## 2. 수치형

### ❖ 실수형

- 소수점 이하 정밀한 값 표현
- 아주 크거나 작은 값은 부동 소수점 방식으로 표기
  - 가수+지수
    - 9조 4600억 =  $9.46e12$
  - 숫자가 짧아지고 비교에도 용이함

### ❖ 복소수형

- 실수부+허수부j
  - 알파벳 j 접미가 복소수임을 나타냄

```
>>> a = 1+2j
>>> b = 3+4j
>>> print (a + b)
(4+6j)
```

### 3. 문자열

#### ❖ 문자열 (String)

- 일련의 문자를 따옴표로 감싸 나열한 것
- 각종 문자, 기호, 숫자 등 포함 가능
- 따옴표는 따옴표 안에 적을 수 없음
  - 큰따옴표 포함하려면 문자열을 작은따옴표로 감싸야
- 한 문자열에 두 따옴표 섞어 사용할 수 없음

```
>>> a = "Korea 서울 1234"  
>>> print(a)  
Korea 서울 1234
```

```
'I Say "Help" to you'
```

"string"  
'string'

맞음

'string"  
"string'

틀림



### 3. 문자열

#### ❖ 확장열 (Escape Sequence)

- 따옴표(인용부호)안에 작성한 특정 기능의 문자들
- \ 문자 뒤에 문자나 기호로 표기
  - 큰따옴표 내 큰따옴표

여는 따옴표                      닫는 따옴표

↓                                      ↓

"I Say \"Help\" to you"

                                 ↙      ↘

                                 따옴표 문자

- 개행

```
>>> a = "first\nsecond"
>>> print(a)
first
second
```

확장열	설명
\n	개행
\t	탭
\"	큰따옴표
'	작은따옴표
\\	\ 문자

### 3. 문자열

#### ❖ 긴 문자열

#### ❖ 다음표 3개 연속으로 사용하여 긴 문자열 정의

longstring

```
s = """강나루 건너서 밭갈 길을 구름에 달 가듯이 가는 나그네  
길은 외줄기 남도 삼백리 술 익는 마을마다 타는 저녁놀  
구름에 달 가듯이 가는 나그네"""  
print(s)
```

실행결과

강나루 건너서 밭갈 길을 구름에 달 가듯이 가는 나그네  
길은 외줄기 남도 삼백리 술 익는 마을마다 타는 저녁놀  
구름에 달 가듯이 가는 나그네

#### ■ 계속문자 \ 사용하는 것도 가능

longstring2

```
s = "강나루 건너서 밭갈 길을 구름에 달 가듯이 가는 나그네 \  
길은 외줄기 남도 삼백리 술 익는 마을마다 타는 저녁놀 \  
구름에 달 가듯이 가는 나그네"  
print(s)
```

실행결과

강나루 건너서 밭갈 길을 구름에 달 가듯이 가는 나그네 길은 외줄기 남도 삼백리  
술 익는 마을마다 타는 저녁놀 구름에 달 가듯이 가는 나그네

### 3. 문자열

- 코드에서도 활용 가능
  - 들여쓰기는 무관함

linecontinue

```
totalsec = 365 * 24 * \  
           60 * 60  
print(totalsec)
```

실행결과     31536000

- 개행하려면 `\n` 확장열 사용
- 전체를 괄호로 묶어서, 여러 개 문자열 개행한 것을 긴 문자열로 만들

multiline

```
s = ("korea"  
     "japan"  
     "2002")  
print(s)
```

실행결과     koreajapan2002

# Contents

## ❖ 목차

- 1. 대입 및 산술
- 2. 타입 변화
- 3. 타입 확인

# 1. 대입 및 산술

## ❖ 대입 연산자

- 변수에 값을 저장하는 연산자
- 변수 = 수식
  - 대입되는 값에 따라 변수 타입이 결정됨

```
a = 3
```

```
s = 'korea'  
f = 3.1415
```

```
a = (1 + 2) * 3  
b = c * d + e
```

# 1. 대입 및 산술

## ❖ 산술 연산자

### ■ 사칙연산 수행하는 연산자

연산자	설명
+	더하기
-	빼기
*	곱하기
/	나누기

연산자	설명
**	거듭제곱
//	정수 나누기
%	나머지

```
>>> print(5+2)
```

```
7
```

```
>>> print(5-2)
```

```
3
```

```
>>> print(5*2)
```

```
10
```

```
>>> print(5/2)
```

```
2.5
```

```
>>> print(5//2)
```

```
2
```

```
>>> print(5%2)
```

```
1
```

```
>>> print(5**2)
```

```
25
```

```
>>> print(5+2*10)
```

```
25
```

```
>>> print((5+2)*10)
```

```
70
```

```
>>> print(10%2, 5%2, 3%2, 2%2)
```

```
0 1 1 0
```

```
>>> print(1%3, 2%3, 3%3, 4%3, 5%3, 6%3, 7%3, 8%3, 9%3, 10%3)
```

```
1 2 0 1 2 0 1 2 0 1
```

```
>>> print('가나다'+ 'ABC')
```

```
가나다ABC
```

```
>>> print('가나다'*3)
```

```
가나다가나다가나다
```

# 1. 대입 및 산술

## ❖ 나눗셈 연산(/)

```
>>> 5 / 2  
2.5
```

- 5 나누기 2는 2.5가 나옴
- 파이썬 3은 나눗셈이 완전히 나누어 떨어져도 실수가 나옴

```
>>> 4 / 2  
2.0
```

## ❖ 나눗셈 후 소수점 이하를 버리는 // 연산자

- 몫만 구하는 연산자이며 나머지는 버림

```
>>> 5 // 2  
2  
>>> 4 // 2  
2
```

# 1. 대입 및 산술

- //은 버림 나눗셈(floor division)이라고 부르며 나눗셈의 결과에서 소수점 이하는 버림
- 실수에 // 연산자를 사용하면 결과는 실수가 나오며 소수점 이하는 버림
- 결과는 항상 .0으로 끝남

```
>>> 5.5 // 2
2.0
>>> 4 // 2.0
2.0
>>> 4.1 // 2.1
1.0
```

## ❖ 나눗셈 후 나머지를 구하는 % 연산자

- 나눗셈 후 나머지 값을 구하고자 할 때 사용

```
>>> 5 % 2
1
```

- 5를 2로 나누면 두 번 나눌 수 있고 1이 남음
- %는 두 수를 나누었을 때 나머지만 구하며 모듈로(modulo) 연산자라고 부름



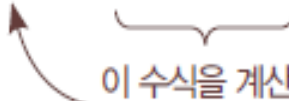
# 1. 대입 및 산술

## ❖ 복합 대입 연산자

- 우변의 값이나 수식 계산하여 좌변에 대입

```
>>> a = 5
>>> a = a + 1
>>> print(a)
6
```

$a = a + 1$



이 수식을 계산하여  
왼쪽 변수에 대입한다.

- **+=** 연산자
  - 좌변의 값에 우변의 값을 더함
- **-=** 연산자
  - 원래 값에서 일정 값을 뺌
- **\*=** 연산자
  - 원래 값의 일정 배수 만들

```
>>> a = 5
>>> a += 1
>>> print(a)
6
```

## 2. 타입 변환

### ❖ 문자열 연산

■ **+** 와 **\*** 연산자는 문자열에 대해서도 사용 가능

- **+**: 문자열을 연결
- **\***: 문자열을 정수 횟수만큼 반복

```
>>> s1 = "대한민국"  
>>> s2 = "만세"  
>>> print(s1 + s2)  
대한민국만세
```

```
>>> print("싫어 " * 5)      # 싫어 싫어 싫어 싫어 싫어
```

## 2. 타입 변환

### ❖ 정수와 문자열

- + 연산자는 피연산자 타입 판별하여 숫자는 덧셈, 문자열은 연결함
  - 문자열과 숫자 섞어 더할 수는 없음

```
print("korea" + 2002) # 에러
```

```
print("korea" + str(2002)) # korea2002
```

### ■ int 함수

- 문자열을 숫자로 바꿀 때

```
print(10 + int("22")) # 32
```

## 2. 타입 변환

### ❖ 실수의 변환

#### ■ float 함수

- 실수가 저장된 문자열을 실수로 변경

```
print(10 + float(("22.5"))) # 32.5  
print(10 + float(("314e-2"))) # 13.14
```

#### ■ 문자열에 저장된 실수를 정수로 바꾸는 경우

- float 함수로 문자열을 실수로 변경
- 다시 int 함수 사용하여 정수로 변경

#### ■ round 함수

- 실수를 소수점 첫째 자리에서 반올림하여 정수 반환

### 3. 타입 확인

#### ❖ 자료형 확인하기

- `type()` 함수 : 자료형을 확인할 수 있는 함수.

```
>>> a = int(10.3)           # a는 정수형으로 10.3을 할당
>>> b = float(10.3)         # b는 실수형으로 10.3을 할당
>>> c = str(10.3)           # c는 문자형으로 10.3을 할당
>>>
>>> type(a)                  # a의 타입을 출력
<class 'int'>
>>> type(b)                  # b의 타입을 출력
<class 'float'>
>>> type(c)                  # c의 타입을 출력
<class 'str'>
```