본 컨텐츠는 과학기술정보통신부정보통신기획평가원의
SW중심대학지원사업단의 연구결과로 수행되었습니다.
(2019 - 0 - 01838)

과학기술정보통신부 IITP 정보통신기획평가원 배재대학교 PAI CHAI UNIVERSITY AI·SW중심대학사업단

AI·SW중심대학사업단

파이썬으로 AI게임 만들기

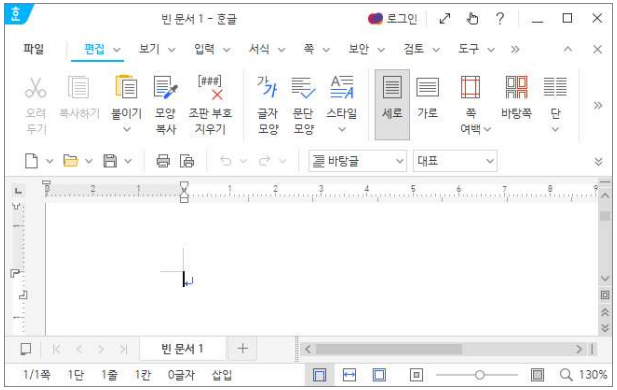
1주차. 프로그래밍과 파이썬

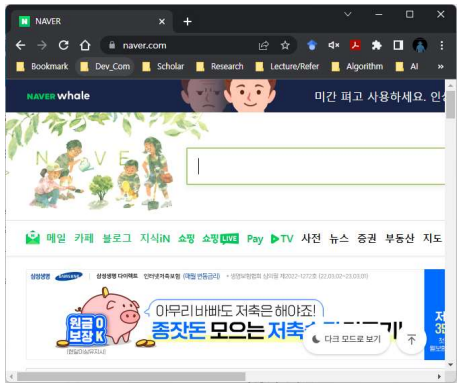
이태준
배재대학교 컴퓨터공학과 석사과정

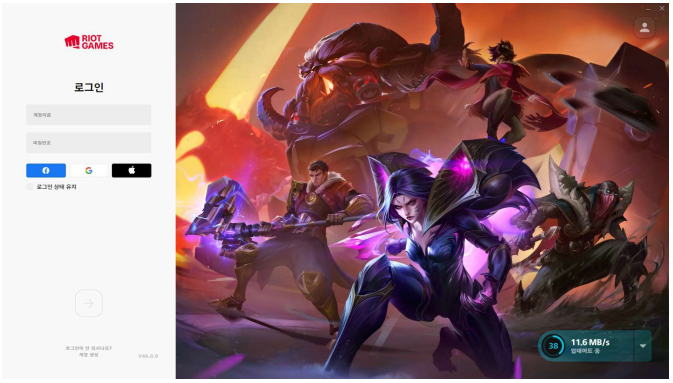
목차

1. 프로그래밍과 파이썬

- 프로그램(Program): 사용자에게 편리한 기능을 제공하기 위해 컴퓨터에게 일을 시키고(**명령**) 이와 관련된 **데이터**를 저장
  - ➢ 사용자에게 제공하는 기능의 예



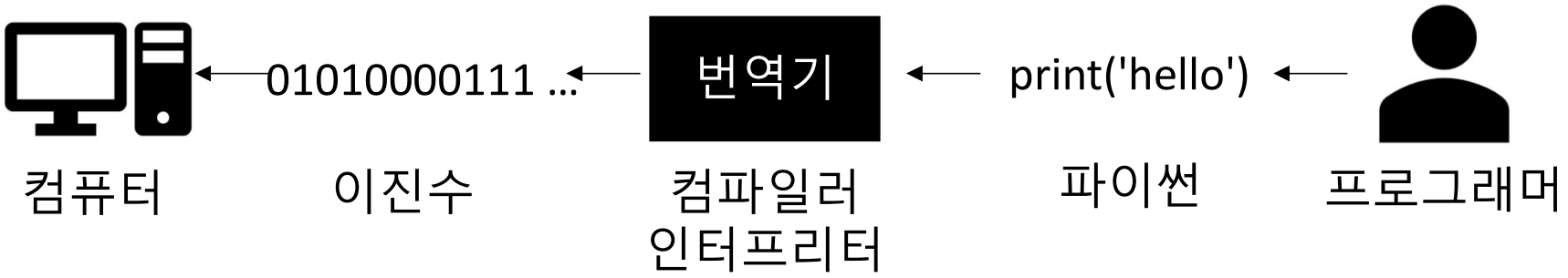




  - ➢ 컴퓨터에게 내리는 명령: 컴퓨터는 이진수(0, 1)를 이해함
    → 사람은 이러한 이진수 명령을 모두 이해하기는 매우 어려움
    → 컴퓨터에게 내리는 명령을 사람도 이해할 수는 없을까?

1. 프로그래밍과 파이썬

- 프로그램(Program): 사용자에게 편리한 기능을 제공하기 위해 컴퓨터에게 일을 시키고(**명령**) 이와 관련된 **데이터**를 저장
  - ➢ 절충안: 사람이 이해하는 명령어를 중간에 번역기를 두어 컴퓨터가 이해할 수 있는 명령(이진수)으로 바꾸자!



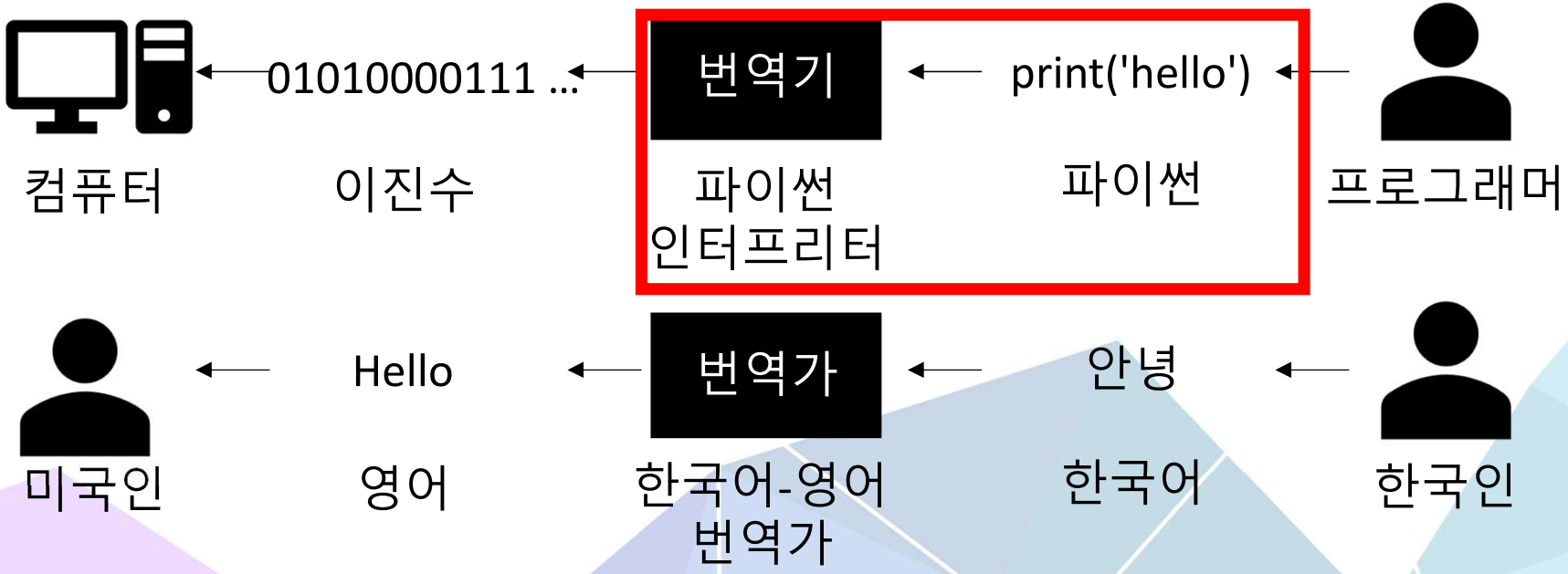
  - ➢ 프로그래밍(programming): 프로그래머(programmer)가 프로그램을 제작하기 위한 일련의 과정
    - 대체적인 과정: 기획 > 설계 > 코딩 > 테스트 > 릴리즈 > 유지/보수

1. 프로그래밍과 파이썬

- 프로그램(Program): 사용자에게 편리한 기능을 제공하기 위해 컴퓨터에게 일을 시키고(**명령**) 이와 관련된 **데이터**를 저장
  - ➢ 프로그램의 종류
    - 시스템: 하드웨어와 밀접한 관련 (Windows, Mac OS, Android, iOS 등)
    - 응용: 사용자와 밀접한 관련 (워드프로세서, 게임, 인터넷 관련 등)
  - ➢ 프로그래머(programmer):
    프로그램을 설계하여 코딩(소스 코드 제작)하는 직업군
    → SW 개발자(developer)로고도 함
    - 시스템/디바이스 관련 프로그래머: 시스템 프로그램 개발
    - 웹 프로그래머: 웹 페이지, 웹 서버 개발
    - 모바일 프로그래머: 모바일 애플리케이션 개발
    - 게임 프로그래머: 게임 개발
    - 이 외에 여러 가지 프로그램 개발이 있으며 분야 안에 세부적 분야 존재

1. 프로그래밍과 파이썬

- 프로그램(Program): 사용자에게 편리한 기능을 제공하기 위해 컴퓨터에게 일을 시키고(**명령**) 이와 관련된 **데이터**를 저장
  - ➢ 프로그래밍 언어(programming language)
    : 프로그램을 만들기 위해 번역기에게 제공해 줄 언어이자 문법
    → 번역기는 프로그래머가 작성한 프로그래밍 언어 문법을 이해하고 이를 실행할 수 있도록 함



1. 프로그래밍과 파이썬

- 파이썬(Python): 최근 많은 분야에서 사용하고 있는 고급 언어
  - ➢ 머신러닝/딥러닝/통계(빅데이터) 분야 등에서 많이 사용

Programming Language Hall of Fame

The hall of fame listing all "Programming Language of the Year" award winners is shown below. The award is given to the programming language that has the highest rise in ratings in a year.

Year	Winner
2021	Python
2020	Python
2019	C
2018	Python
2017	C

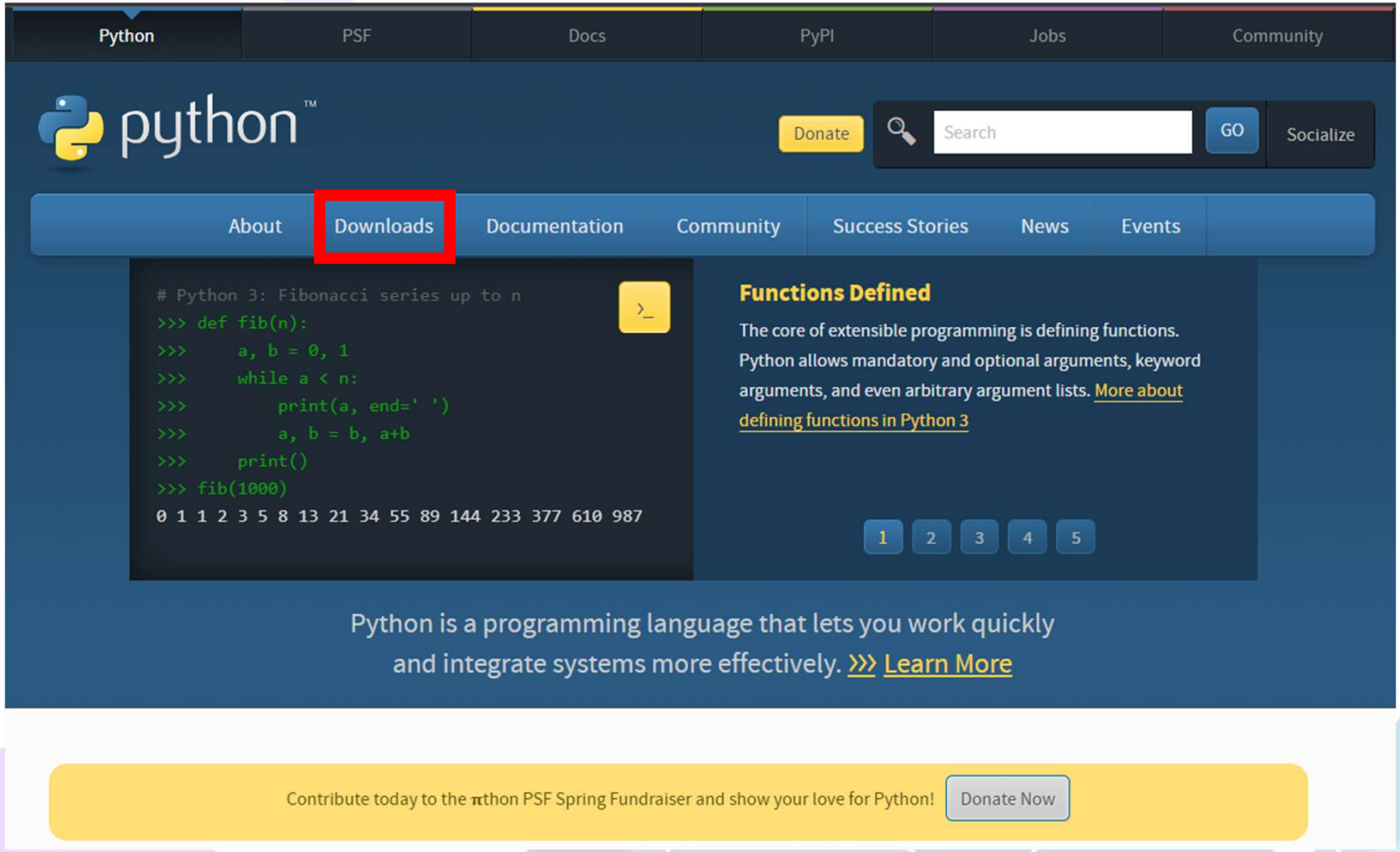
* https://www.tiobe.com/tiobe-index/

  - ➢ 귀도 반 로섬(Guido van Rossum)이 발표한 언어
  - ➢ 초보자도 쉽게 배울 수 있는 언어로 문법이 단순함
    → 초/중/고에서도 배우는 추세이며,
    대학의 이공 계열 뿐만 아니라
    사회/경제 분야 등 다양한 계열에서 배우고/사용되고 있음

목차

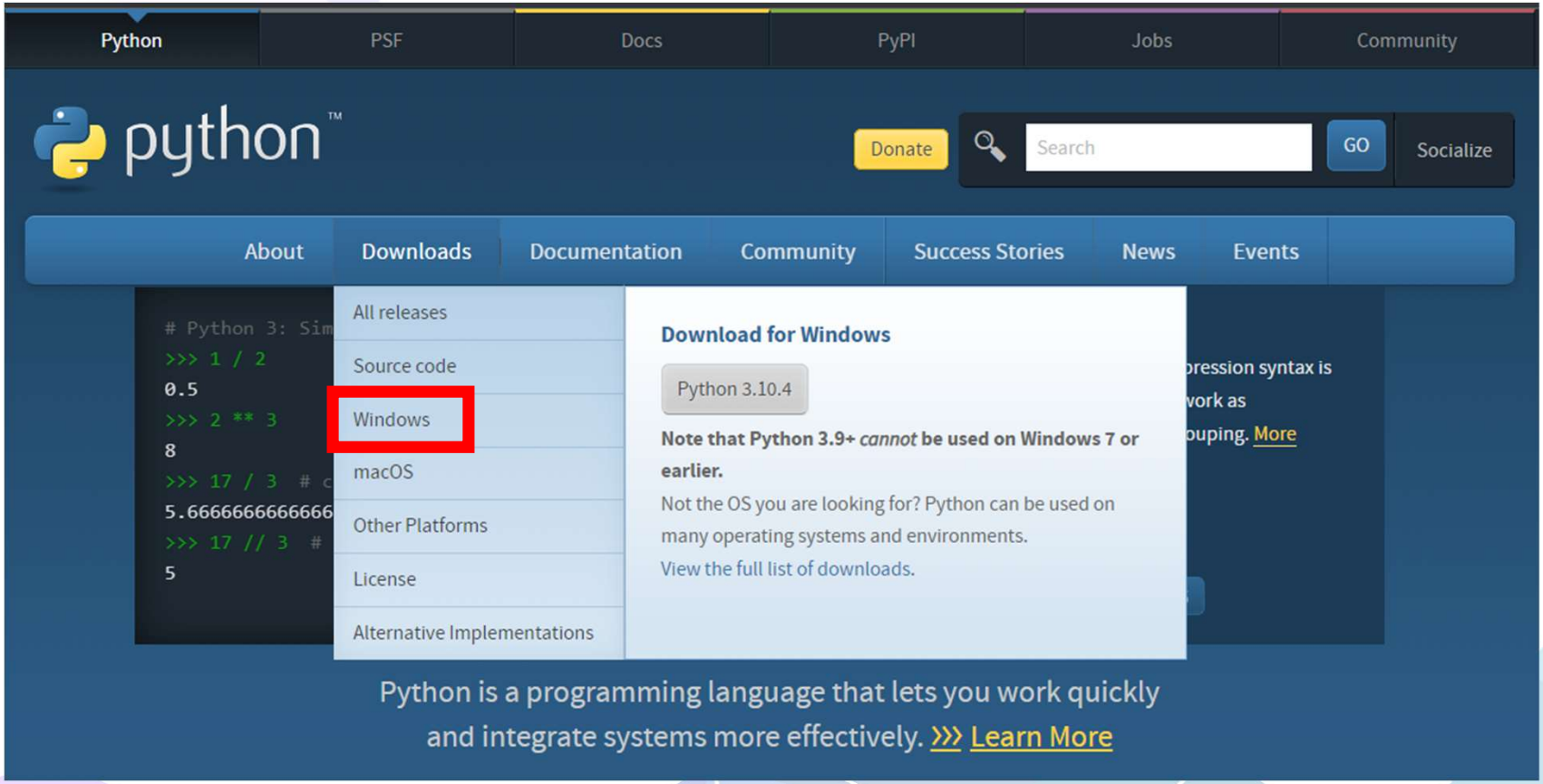
2. 파이썬 설치

- 파이썬(Python) 다운로드: https://www.python.org



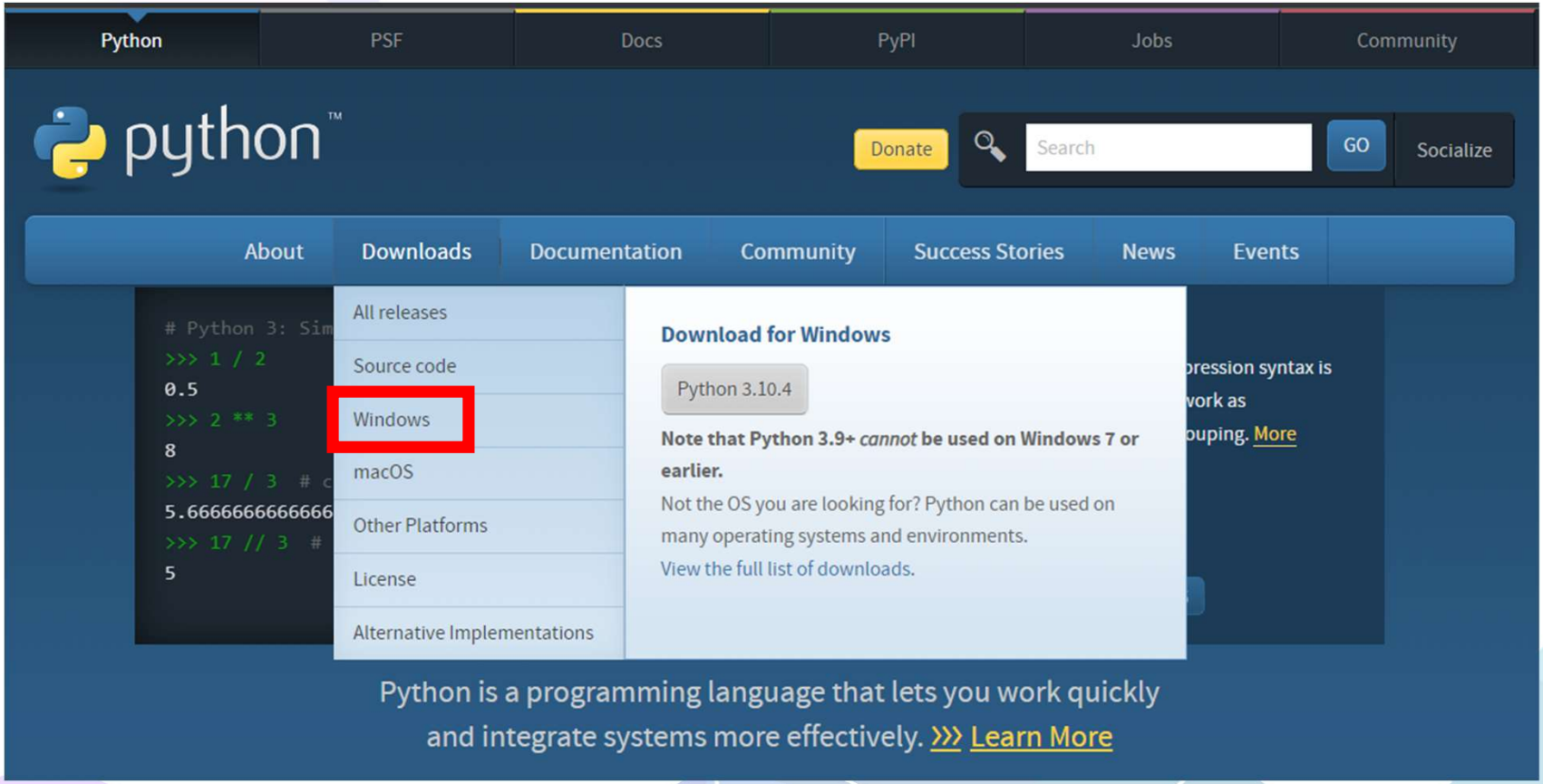
2. 파이썬 설치

- 운영체제 환경과 버전에 맞게 다운로드



2. 파이썬 설치

- 운영체제 환경과 버전에 맞게 다운로드



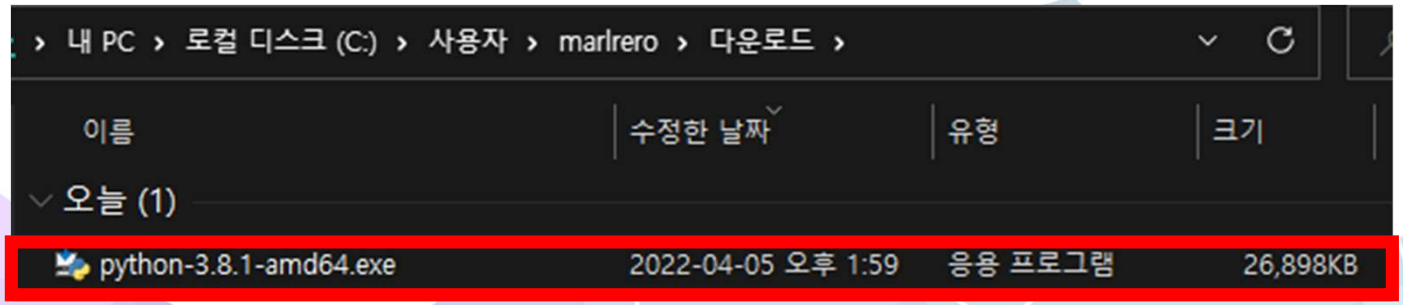
2. 파이썬 설치

- 운영체제 환경(64비트인 경우)과 버전에 맞게 다운로드
  → 3.8.1 – Dec. 18. 2019를 찾아서 설치 (Ctrl + F 이용)

- Python 3.8.1 - Dec. 18, 2019

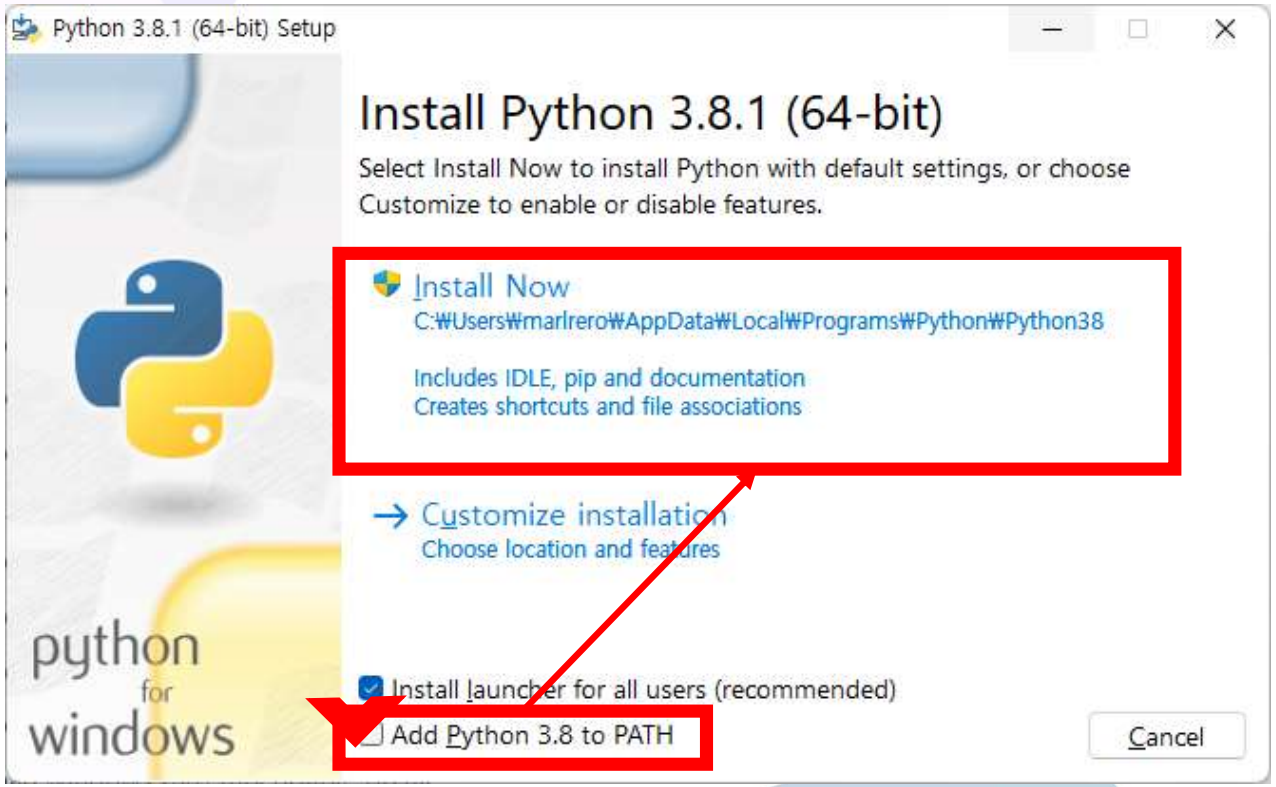
  Note that Python 3.8.1 *cannot* be used on Windows XP or earlier.

  - Download Windows help file
  - Download Windows x86-64 embeddable zip file
  - Download Windows x86-64 executable installer
  - Download Windows x86-64 web-based installer
  - Download Windows x86 embeddable zip file
  - Download Windows x86 executable installer
  - Download Windows x86 web-based installer



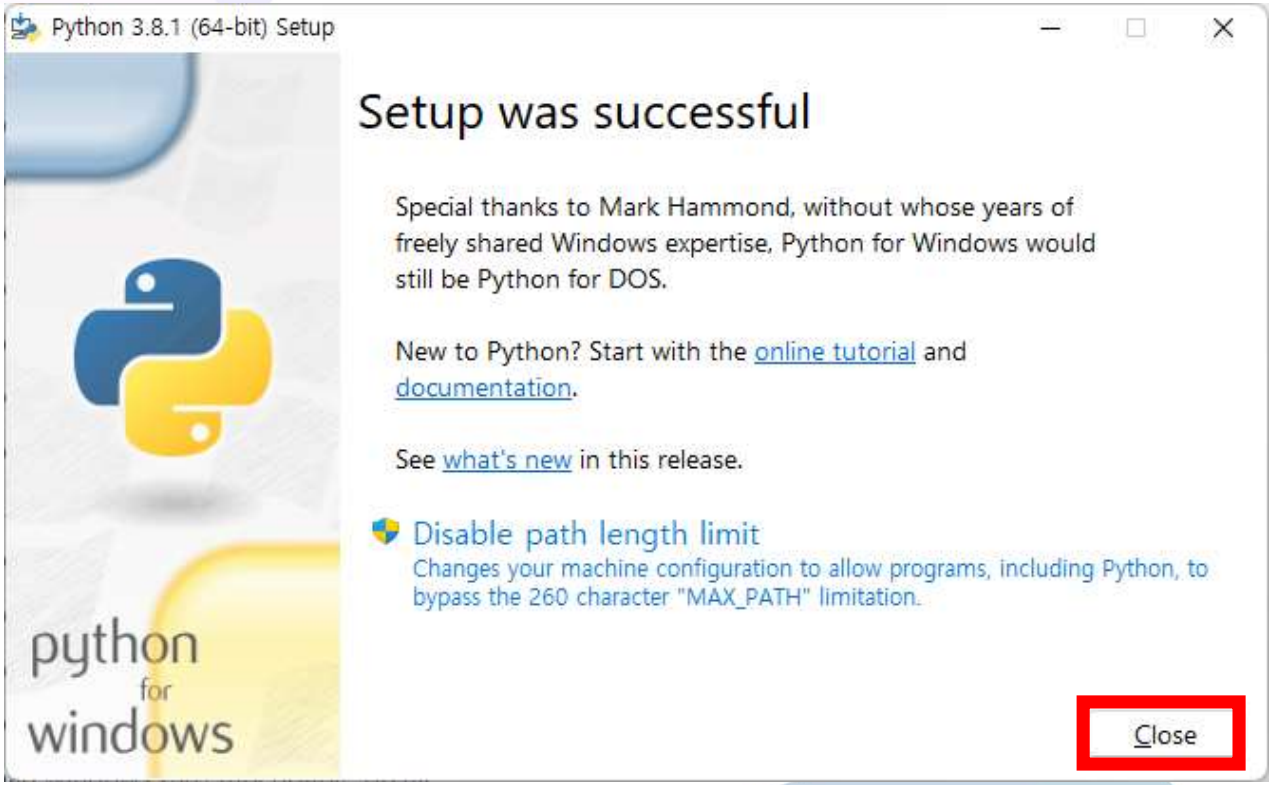
2. 파이썬 설치

- 설치 과정 (Add Python 3.x to PATH 체크 선택 반드시 해야 함!)



2. 파이썬 설치

- 설치 과정 (Add Python 3.x to PATH 선택)



2. 파이썬 설치

- 파이썬 설치 확인 및 IDLE 실행 (시작 버튼에서 앱에서 확인)

목차

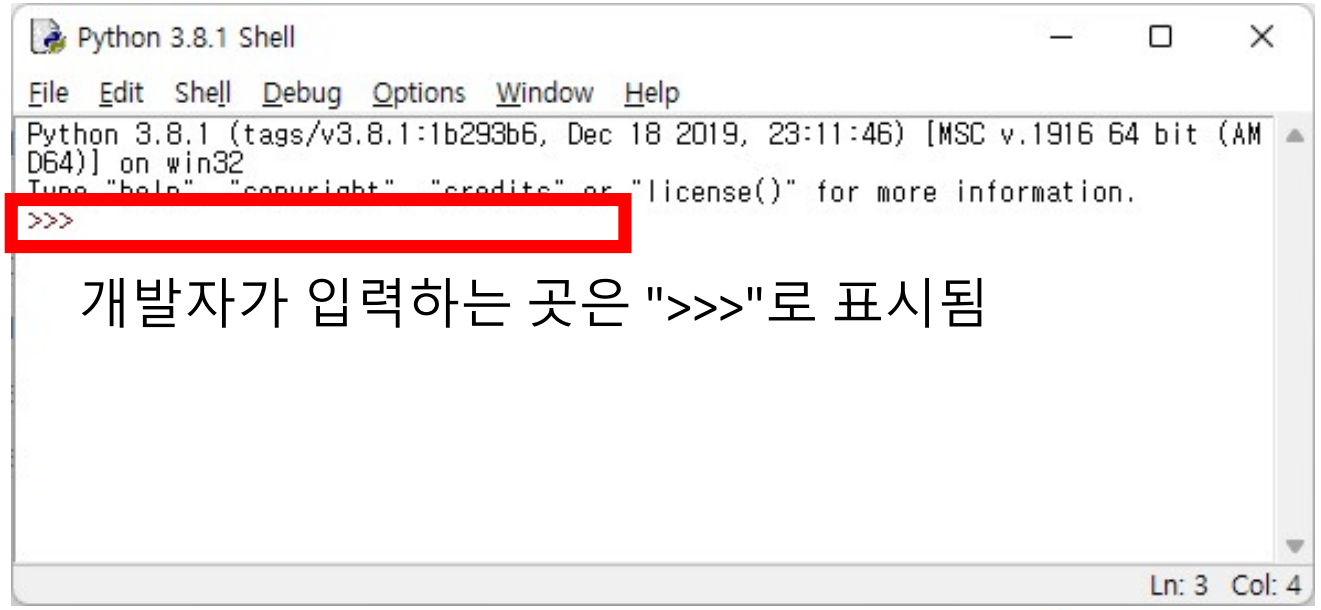
3. IDLE와 사칙연산

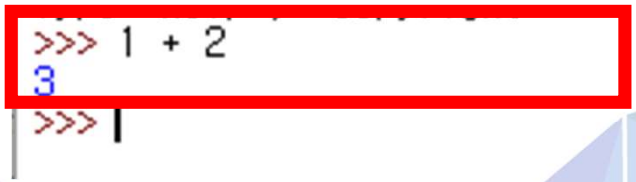
- 파이썬 통합 개발 환경(Integrated Development Environment)
  - 통합 개발 환경: 소프트웨어(프로그램) 개발을 지원하는 도구
    → 소스 코드 작성 + 디버깅(debugging)
    + 프로그램에 필요한 데이터 관리 등
  - 버그(bug): 프로그램 실행 시 발생하는 에러
    → 사용자가 발견할 수 있고,
    개발자가 발견할 수도 있음
    → 이 에러를 고치는 행위를 디버깅이라고 함
  - 통합 개발 환경은 다양하게 있음
    - 파이썬 설치 시 기본 제공하는 IDLE
    - PyCharm
    - Visual Studio Code 등…
  - 통합 개발 환경은 협업 시 고려해야 할 사항
    → 협업하지 않고 혼자 개발한다면 자신이 편한 것을 쓰면 됨

스미소니언 박물관에 보관중인
위 사진은 최초의 버그.
컴퓨터 회로 안에 벌레가 들어가
합선된 것으로 추정.

3. IDLE와 사칙연산

- 파이썬 통합 개발 환경(Integrated Development Environment)
  - ➢ 파이썬 설치 시 기본 제공하는 IDLE





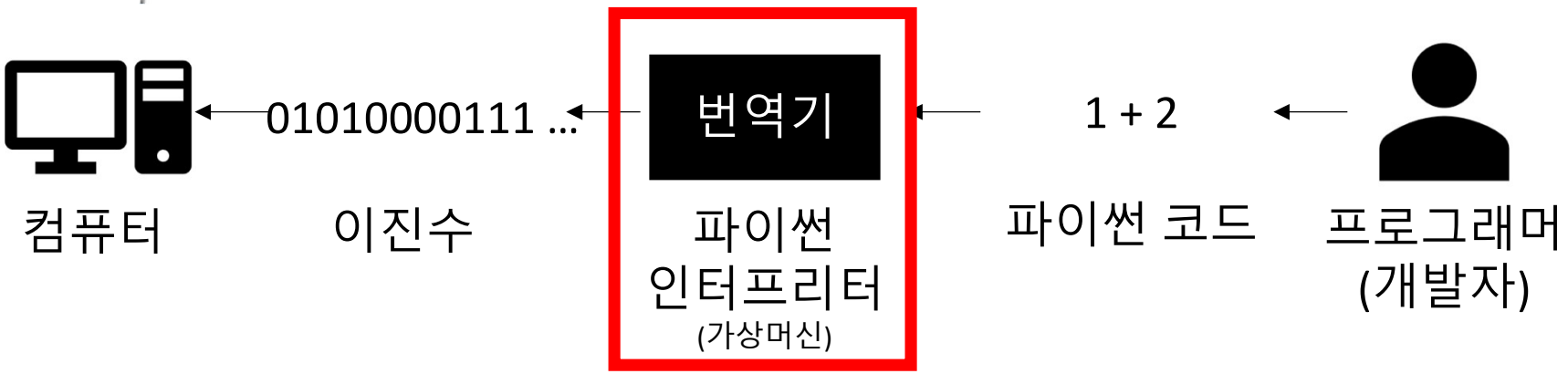
1 + 2를 입력하고 엔터를 누르면
3(파란색)이 출력됨

3. IDLE와 사칙연산

- 파이썬 통합 개발 환경(Integrated Development Environment)
  - ➢ 파이썬 IDLE로 컴퓨터와 대화해보기!

```
>>> 1 + 2
3
>>>
```

1 + 2를 입력하고 엔터를 누르면
3(파란색)이 출력됨



우리가 방금 설치한 파이썬

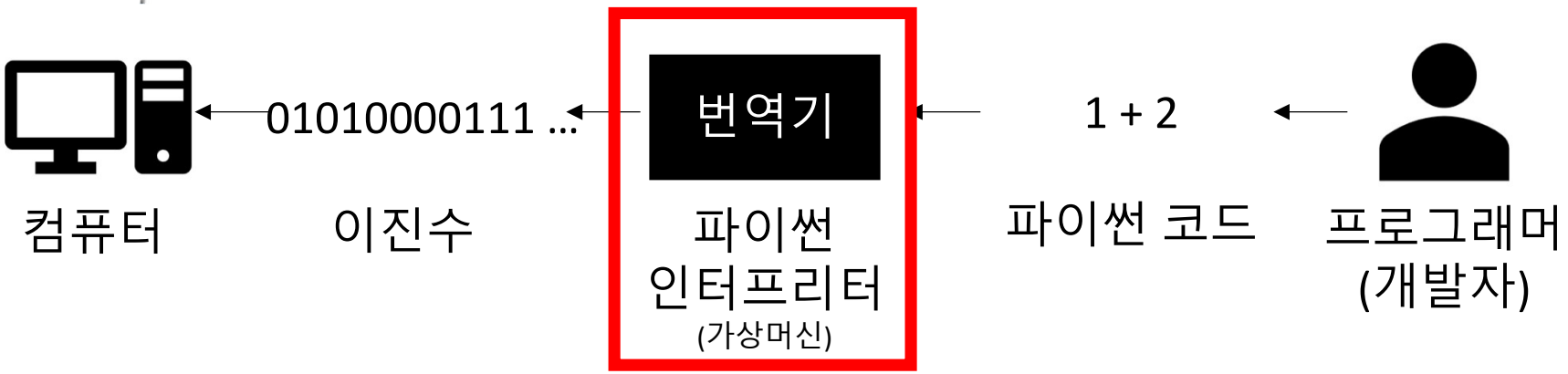
  - ➢ 파이썬 인터프리터(Interpreter): 프로그래머가 입력한 파이썬 코드를 파이썬 가상머신이 이해할 수 있도록 한 줄 단위로 번역해주는 프로그램

3. IDLE와 사칙연산

- 파이썬 통합 개발 환경(Integrated Development Environment)
  - ➢ 파이썬 IDLE로 컴퓨터와 대화해보기!

```
>>> 1 + 2
3
>>>
```

1 + 2를 입력하고 엔터를 누르면
3(파란색)이 출력됨



우리가 방금 설치한 파이썬

  - ➢ 파이썬 가상머신(Virtual Machine): 파이썬 인터프리터가 해석해 준 코드를 바탕으로 파이썬 프로그램을 실행시키는 프로그램

3. IDLE와 사칙연산

- 사칙 연산자: 사칙 연산(덧셈, 뺄셈, 곱셈, 나눗셈) 실행
  - 연산자(operator): 연산을 수행하기 위해 파이썬 인터프리터와 약속한 기호
  - 피연산자(operand): 연산을 수행하기 위한 데이터
  - 예시) 3 + 5에서 연산자는 +이고, 피연산자는 3과 5이다.
    → 파이썬 인터프리터는 덧셈 연산을 3과 5에 대해서 수행할 것임을 이해할 수 있음
  - 사칙 연산은 수학 기호와는 조금 다름

사칙 연산	수학 기호	컴퓨터 기호(연산자)
덧셈	+	+
뺄셈	-	-
곱셈	×	*
나눗셈	÷	/

3. IDLE와 사칙연산

- 사칙 연산자: 사칙 연산(덧셈, 뺄셈, 곱셈, 나눗셈) 실행
  - ➢ 사칙 연산 해보기

```
>>> 7 * 7
49
>>> 10 / 5
2.0
>>> 10 / 0
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    10 / 0
ZeroDivisionError: division by zero
```

이렇게 개발 중에 발생하는 에러를 버그라고 하지는 않음

버그는 이러한 에러를 개발자가 무시하거나 예상하지 못해서 발생함

파이썬 인터프리터가 이해하지 못함 → 에러(error)

ZeroDivisionError: division by zero

→ 0으로 나눴습니다!

```
>>> 7 +
SyntaxError: invalid syntax
```

문법 에러

→ 덧셈(+) 연산은 피연산자가 2개!

3. IDLE와 사칙연산

- 사칙 연산자: 사칙 연산(덧셈, 뺄셈, 곱셈, 나눗셈) 실행
  - 사칙 연산과 수학에서 소괄호 사용
    → 소괄호 연산자: 먼저 연산을 수행하라는 명령

```
>>> (1 + 2) * (4 / 1)
12.0
```

  - 나눗셈 연산자: 실수로 나눠지게 됨
  - 몫 연산자(//): 나눗셈 이후 나오는 몫(정수 값) 출력
  - 나머지 연산자(%): 나눗셈 이후 떨어지는 나머지(정수 값) 출력

```
>>> 5 / 2
2.5
>>> 5 // 2
2
>>> 5 % 2
1
```

목차

4. 문자열

- 문자열(string): 문자의 배열 → 사람이 사용하는 말
  - ➢ 파이썬에서 문자열을 다루는 방법

```
>>> hello world
SyntaxError: invalid syntax
>>> 안녕하세요
Traceback (most recent call last):
  File "<pyshell#10>", line 1, in <module>
    안녕하세요
NameError: name '안녕하세요' is not defined
```

문자열은
특별한 처리를 해줘야 함

```
>>> 'hello world'
'hello world'
>>> "hello world"
'hello world'
>>> '안녕하세요'
'안녕하세요'
>>> "안녕하세요"
'안녕하세요'
```

문자열은 작은따옴표('')로 묶어주거나
큰따옴표("")로 묶어줘야 함

4. 문자열

- 문자열(string): 문자의 배열 → 사람이 사용하는 말
  - ➢ 파이썬에서 문자열을 다루는 방법

```
>>> '안녕하세요"
SyntaxError: EOL while scanning string literal
```

작은따옴표와 큰따옴표를 혼용해서 사용하면 에러 발생
반드시, 작은따옴표로 시작했으면 작은따옴표로 끝나야 하고,
큰따옴표로 시작했으면 큰따옴표로 끝나야 함

```
>>> 'My name is "taejun"'
'My name is "taejun"'
```

작은따옴표 안에 큰따옴표 사용 가능
(물론, 큰따옴표 안에 작은따옴표 사용 가능)

4. 문자열

- 문자열(string): 문자의 배열 → 사람이 사용하는 말
  - ➢ 파이썬에서 문자열을 다루는 방법

```
>>> "My name is "taejun""
SyntaxError: invalid syntax
```

큰따옴표 안에 큰따옴표를 넣을 수 없음
(물론, 작은따옴표 안에 작은따옴표를 넣을 수 없음)
→ 파이썬 인터프리터가 문자열의 끝을 파악할 수 없음

```
>>> "My name is ₩"taejun₩""
'My name is "taejun"'
```

큰따옴표 안에 큰따옴표를 넣으려면
\" (한글 폰트인 경우 ₩")를 넣어줘야 함
작은따옴표의 경우 \'(₩')를 넣어주면 됨

4. 문자열

- print 함수: 오른쪽에 오는 데이터를 출력하는 함수
  (함수에 관한 자세한 설명은 다음 시간에)

```
>>> print("Hello, world!")
Hello, world!
```

```
>>> print(5)
5
```

```
>>> print("5")
5
```

```
>>> print(3 + 5)
8
```

```
>>> print("3 + 5")
3 + 5
```

큰따옴표나 작은따옴표로 묶인 것은 문자열로 처리됨

4. 문자열

- print 함수: 오른쪽에 오는 데이터를 출력하는 함수
  (함수에 관한 자세한 설명은 다음 시간에)

```
>>> print("3 + 5 =", 8)
3 + 5 = 8
```

```
>>> print(1, 2, 3, "Hello", "World!")
1 2 3 Hello World!
```

```
>>> print("3 + 5 =", 3 + 5)
3 + 5 = 8
```

콤마(,)를 사용해서 띄어쓰기 되어 출력됨

목차

5. 소스 코드

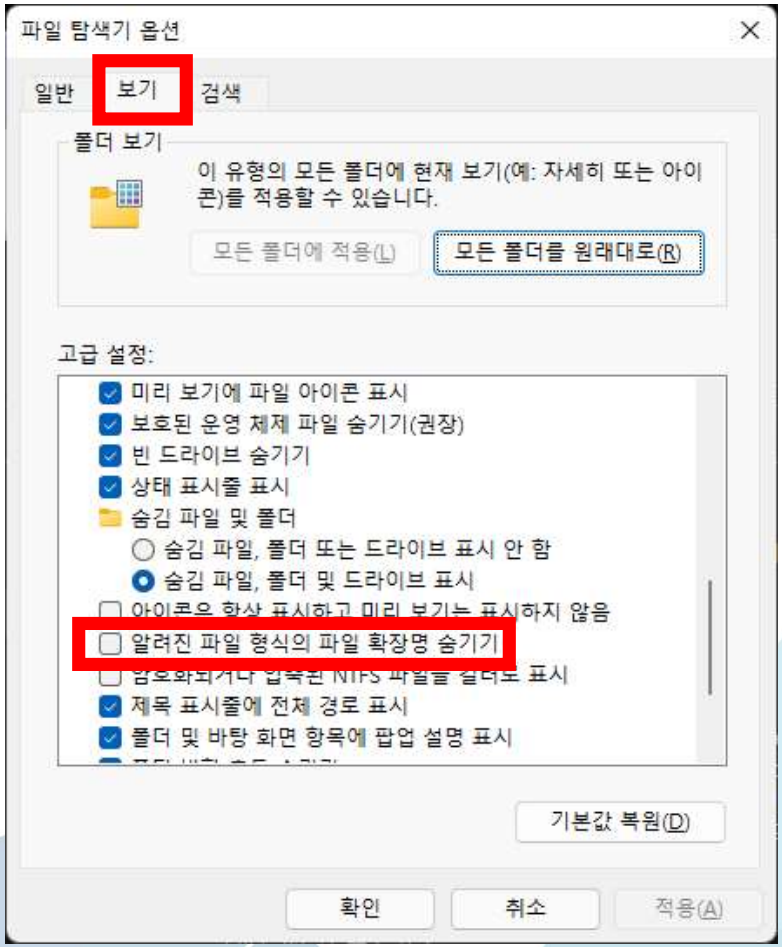
- 지금까지는, IDLE를 이용해 한 줄 씩 실행해 본 것임
- 파일에 작성해서 한꺼번에 돌리는 방법이 있음
  - ➢ 파일의 확장자: 파일의 종류를 표시하기 위해 파일명 끝에 붙이는 문자열

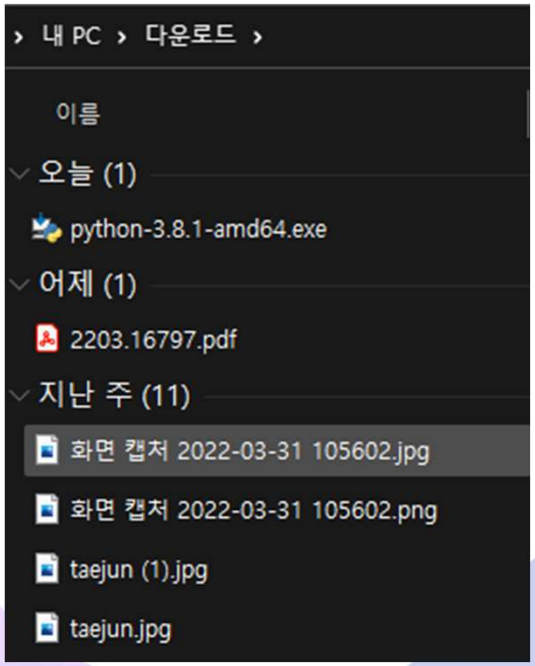
aaaaa.py
파일명.확장자

.hwp: 한글 파일
.xlsx: 엑셀 파일
.exe: 실행 파일
.py: 파이썬 소스코드 파일

5. 소스 코드

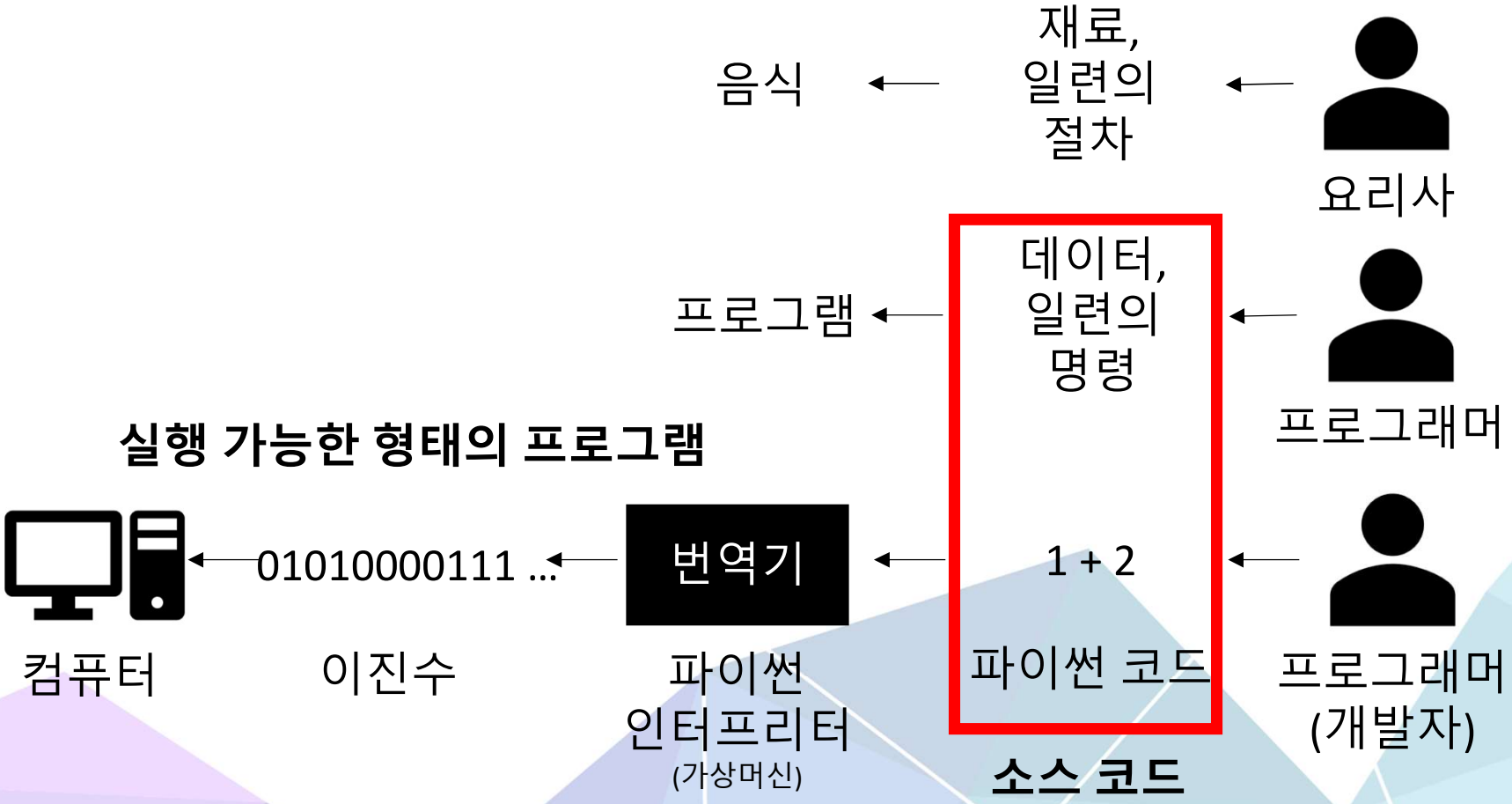
- 파일 확장자 보는 방법
  - ➢ 윈도우 버튼 + S 를 누르고 "파일 탐색기 옵션" 혹은 "폴더 옵션" 찾기
  - ➢ 폴더 옵션 상자에서 보기 탭을 클릭 후 "알려진 파일 형식의 파일 확장명 숨기기"를 체크 해제





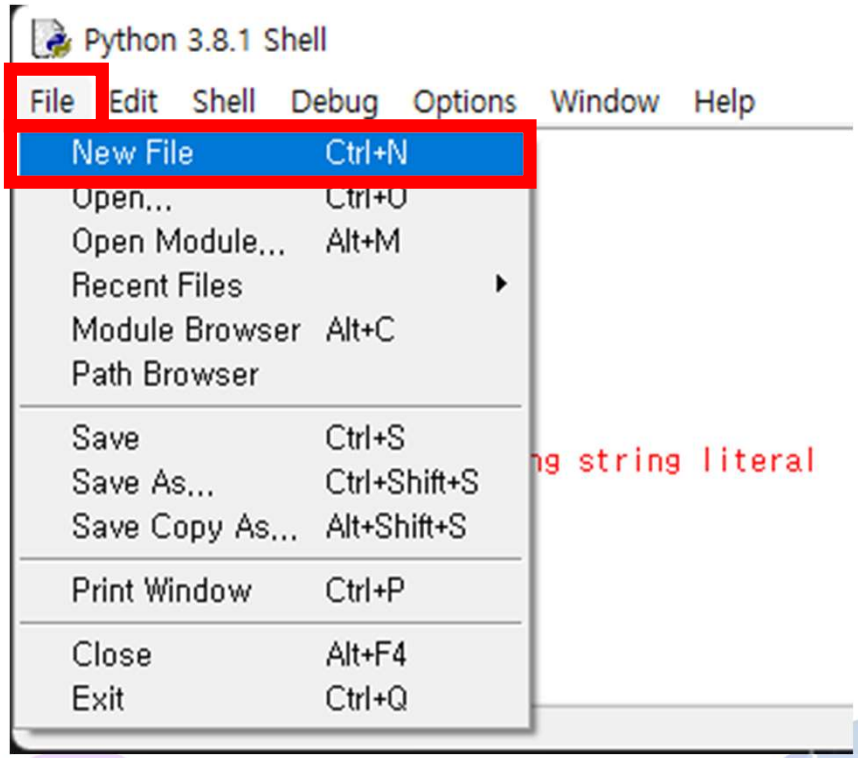
5. 소스 코드

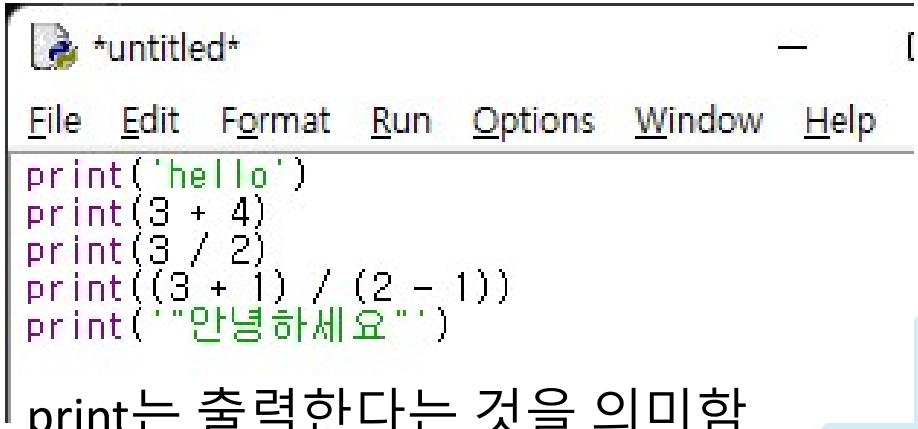
- 소스 코드(source code): 프로그래머가 프로그램을 만들기 위해 프로그래밍 언어로 작성한 코드 집합



5. 소스 코드

- 파이썬 소스 코드 파일: 확장자가 py임
- IDLE로 소스 코드 파일 생성과 저장 방법

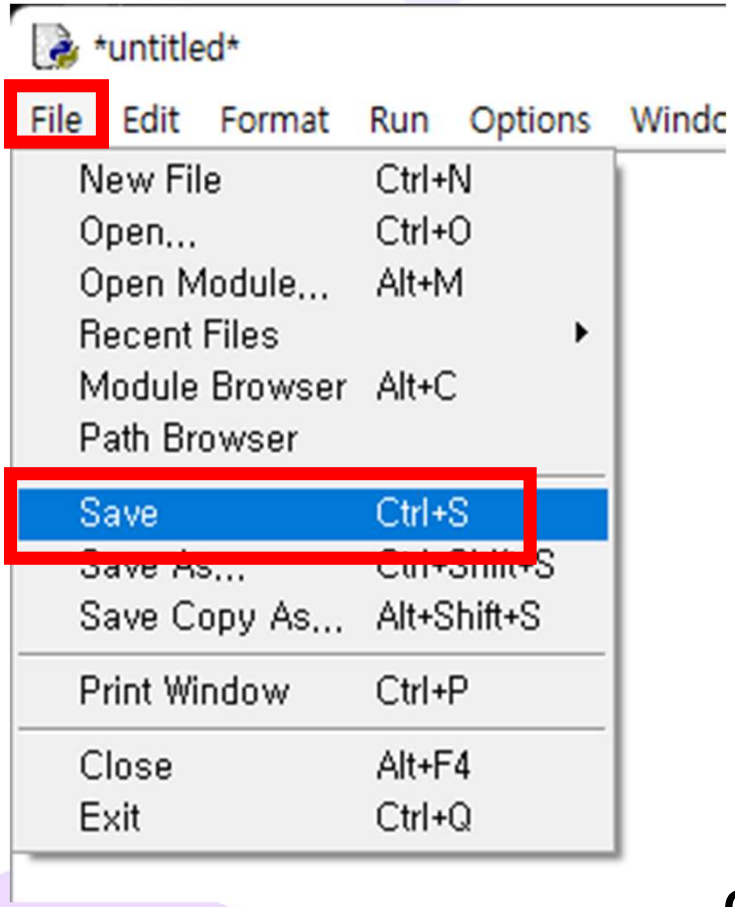


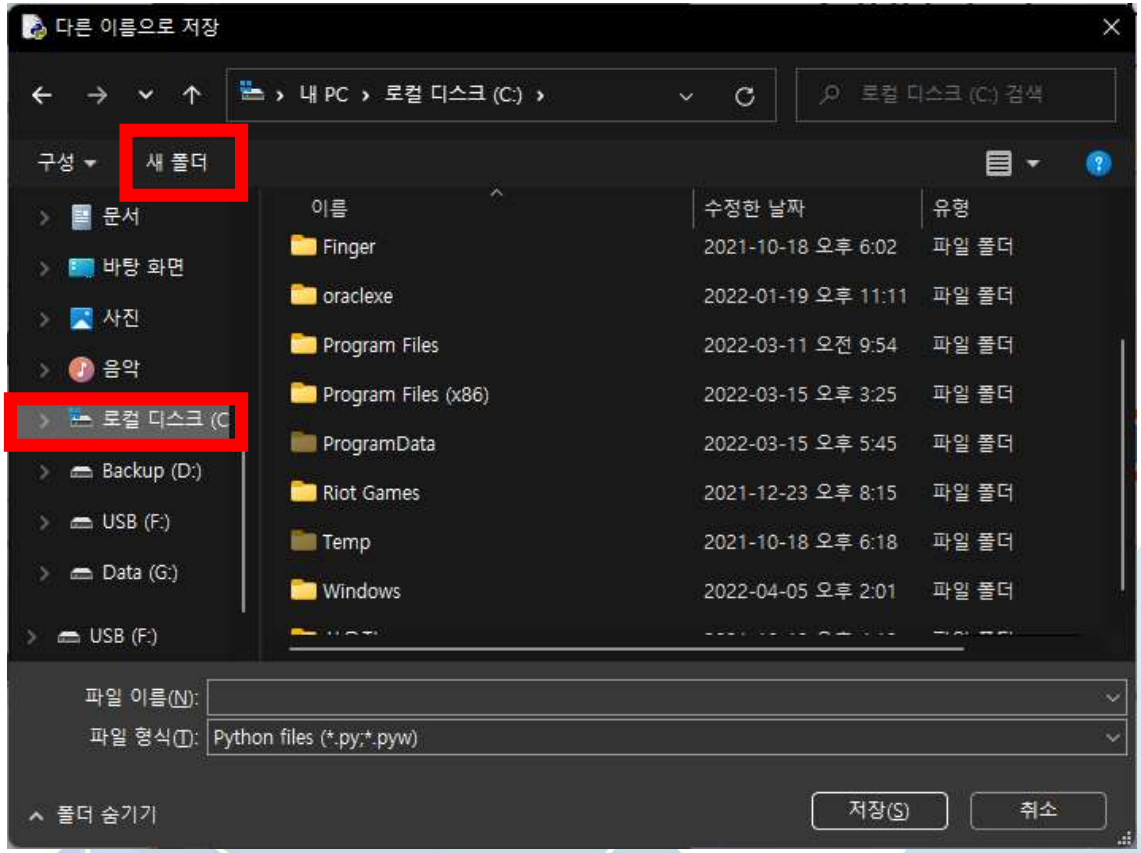


print는 출력한다는 것을 의미함
(나중에 더 자세히 배울 예정)

5. 소스 코드

- IDLE로 소스 코드 파일 생성과 저장 방법

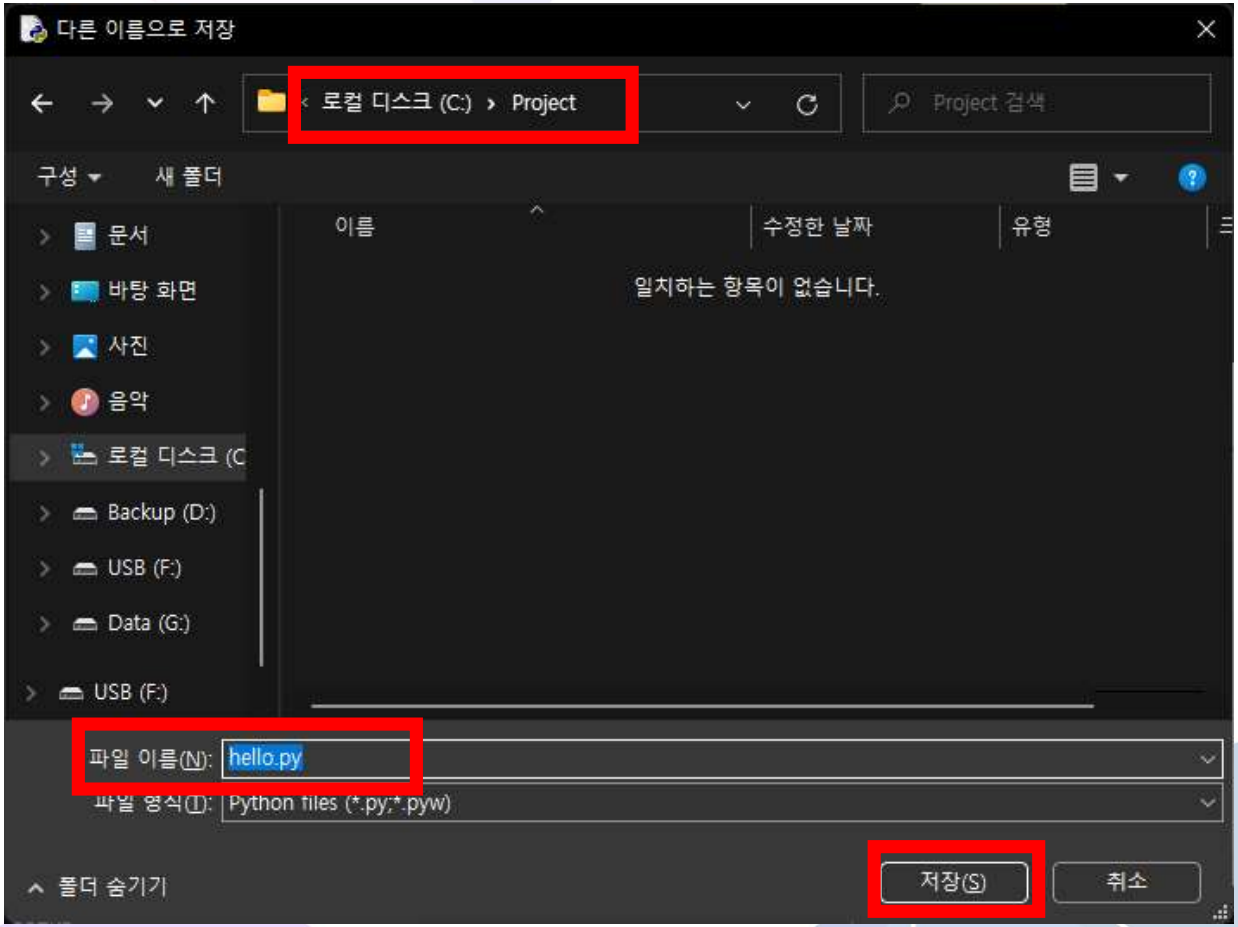




c드라이브에 폴더 생성 → 작업 환경을 만든다고 함

5. 소스 코드

- IDLE로 소스 코드 파일 생성과 저장 방법

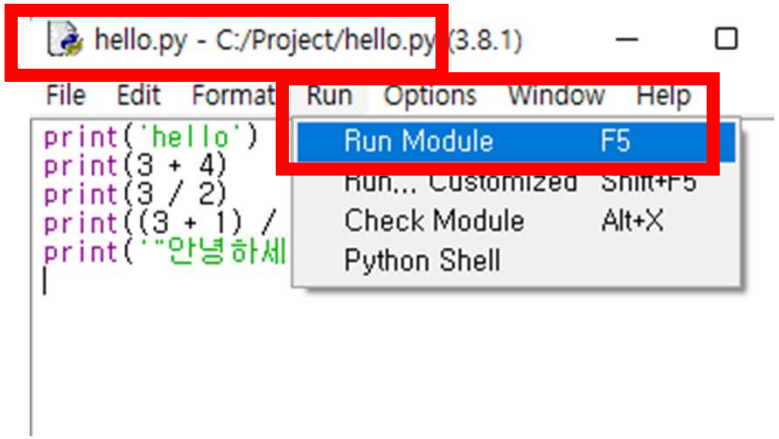


Project라는 작업 환경(폴더)을 만들고

hello.py라는 이름으로 저장

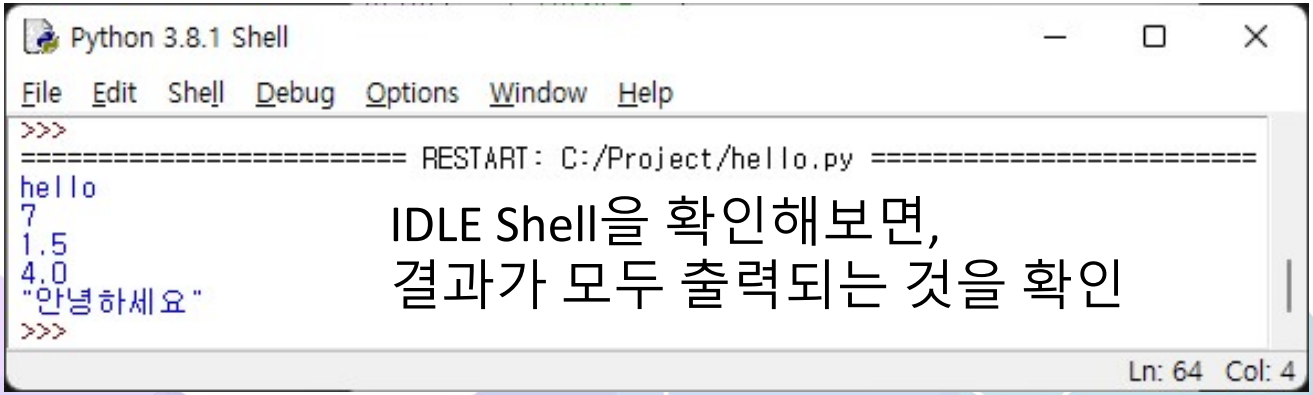
5. 소스 코드

- IDLE로 소스 코드 파일 실행
  → 파이썬 인터프리터한테 파일 통째로 번역하라는 의미
  → 번역 완료 후 파이썬 가상 머신이 이를 실행함



C:/Project/hello.py 임을 확인
→ C드라이브 밑에 Project 폴더 안에 hello.py라는 파일이 존재함

Run 메뉴에서 Run Module 메뉴 클릭
(키보드에서 F5 버튼 클릭해도 됨)



목차

6. 입출력 명령

- 출력을 수행하는 print()

변수는 일단 지금 데이터를 저장하는 상자라고 생각
변수 단원에서 자세히...

test1.py

```
a = 10
print(a)
```

a라는 변수(상자)에 10을 저장(=)하라
변수 a를 출력(print)하라

- 입력을 수행하는 input()

test2.py

```
print('이름 입력: ')
name = input()
print('이름: ' + name)
```

'이름 입력'을 출력하라
사용자가 입력한 이름을 name이라는 변수에 저장하라
'이름: name' 형태로 출력하라

문자열끼리 덧셈(+) 연산은 문자열끼리 연결시킨다는 의미!

위 코드 대신에 name = input('이름 입력: ')도 해보자.

목차

7. 파이썬 프로그램 작성 규칙

- 대소문자를 구별함

```
>>> print('Hello')
Hello
>>> Print('Hello')
Traceback (most recent call last):
  File "<pyshell#30>", line 1, in <module>
    Print('Hello')
NameError: name 'Print' is not defined
```

- 변수를 만들 때(선언할 때)나 명령의 () 내의 공백은 상관 없음

```
>>> a=10
>>> print(a)
10
>>> a = 20
>>> print(a)
20
>>> print( '파이선' )
파이선
>>> print('파이선' )
파이선
```

7. 파이썬 프로그램 작성 규칙

- 주석(comment)을 추가할 수 있음
  - 주석: 프로그램 도중 작성하는 메모
    → 설명하기 어려운 코드에 대한 설명을 적을 때 사용
    → 코드를 남겨둘 때 사용하기도 함
    (코드를 지우지 않는 게 더 효율적일 때)
  - 주석은 한 줄인 경우 '#'을 사용
  - 주석은 파이썬 인터프리터가 번역하지 않음

```
>>> print('Python') # 파이썬을 출력한다는 의미
Python
>>> #print('Python')
```

7. 파이썬 프로그램 작성 규칙

- 주석(comment)을 추가할 수 있음
  - 주석이 여러 줄일 경우 작은따옴표 3번이나 큰따옴표 3번을 사용

```
comment.py - C:/Project/comment.py (3.8.1)
File  Edit  Format  Run  Options  Window  Help
'''
여러 줄
일때
주석
'''
print('hello') # hello를 출력하라
"""
여러
줄
"""
```

```
========== RESTART: C:/Project/comment.py ==========
hello
>>>
```

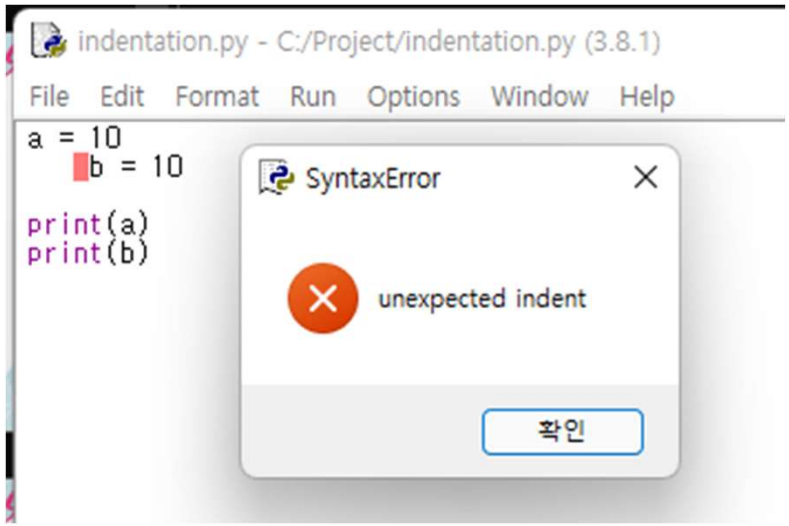
  - 주의: IDLE 셸(>>>)에서는 문자열로 다뤄짐

```
>>> '''
여러
줄
일때
'''
'\n여러\n줄\n일 때\n'
>>>
여러 줄
일때
"""
'\n여러 줄\n일 때\n'
```

\n (한글 폰트인 경우 ₩n)은
개행(new line), 엔터 키의 의미

7. 파이썬 프로그램 작성 규칙

- 들여쓰기(indentation)가 매우 중요함
  - ➢ 들여쓰기는 탭(tab) 키를 통해서 할 수 있음



  - ➢ 나중에 배울 조건문, 반복문, 함수 등에서 매우 중요하게 다뤄짐
  - ➢ 들여쓰기가 코드의 묶음, 한 덩어리(code block)를 의미하는 역할

목차

8. 변수

- "아빠"의 암산 능력을 알고 "아들"이 질문함
  - 아빠 3이랑 50을 곱해
  - 거기에 120을 더해
  - 그리고 그걸 다시 3으로 나누면 얼마야?

```
>>> x = 3 * 50
>>> y = x + 120
>>> z = y / 3
>>> print(z)
90.0
```

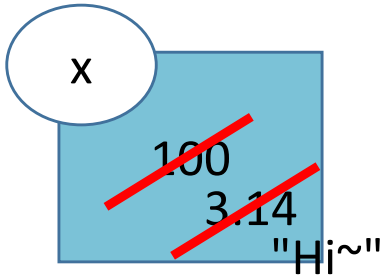
프로그램에서 데이터와 명령은 모두 메모리에 있어야 함
(우리 머리 속이라고 생각하자!)

- 변수(variable): 데이터(값)를 저장하는 공간 (값이 변할 수 있음)
  - 변수는 메모리(memory) 공간에 저장됨
  - 대입(assignment) 연산자 '=': 수학 기호에서 같음(equal)이 아닌, **대입 연산자 오른쪽에 있는 데이터를 왼쪽에 저장**하라는 의미

8. 변수

- 변수(variable): 데이터(값)를 저장하는 공간 (값이 변할 수 있음)
  - ➢ 변수는 메모리(memory) 공간에 저장됨
  - ➢ 대입(assignment) 연산자 '=': 수학 기호에서 같음(equal)이 아닌, 대입 연산자 오른쪽에 있는 데이터를 왼쪽에 저장하라는 의미
    → 왼쪽은 메모리 공간에 붙여진 이름 (상자에 붙인 이름)
    → x라는 이름의 상자 안에 여러 값을 바꿔서 넣을 수 있음

```
>>> x = 100
>>> print(x)
100
>>> x = 3.14
>>> print(x)
3.14
>>> x = "Hi~"
>>> print(x)
Hi~
```

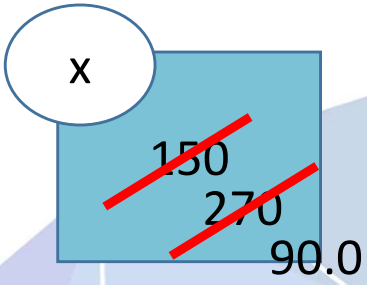


8. 변수

- 변수(variable): 데이터(값)를 저장하는 공간 (값이 변할 수 있음)
  - ➢ 변수는 메모리(memory) 공간에 저장됨
  - ➢ 대입(assignment) 연산자 '=': 수학 기호에서 같음(equal)이 아닌, 대입 연산자 오른쪽에 있는 데이터를 왼쪽에 저장하라는 의미
    → 왼쪽은 메모리 공간에 붙여진 이름 (상자에 붙인 이름)
    → x라는 이름의 상자 안에 여러 값을 바꿔서 넣을 수 있음

```
>>> x = 3 * 50
>>> y = x + 120
>>> z = y / 3
>>> print(z)
90.0
```


```
>>> x = 3 * 50
>>> x = x + 120
>>> x = x / 3
>>> print(x)
90.0
```

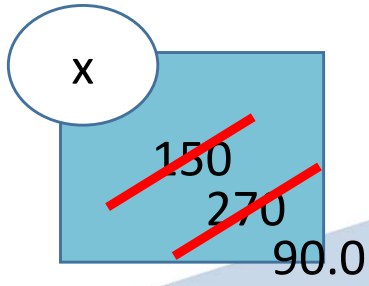


8. 변수

- 변수(variable): 데이터(값)를 저장하는 공간 (값이 변할 수 있음)
  - ➢ 대입 연산자의 왼쪽과 오른쪽에서 변수의 이름은 다른 의미
    → 대입 연산자의 왼쪽 변수 이름(x)는 값을 저장함으로써 의미
    → 대입 연산자의 오른쪽 변수 이름(x)는 값을 가져오라는 의미

```
>>> x = 3 * 50
>>> y = x + 120
>>> z = y / 3
>>> print(z)
90.0
```


```
>>> x = 3 * 50
>>> x = x + 120
>>> x = x / 3
>>> print(x)
90.0
```



8. 변수

상자 박스에
큰 것을 담을지, 작은 것을 담을지...

- 데이터 타입: 파이썬 인터프리터가
  변수를 관리하기 위한 방법 중 하나

→ 데이터를 어떻게 저장하고 처리할 것인가를 의미
→ 정수(int), 실수(float), 문자열(str) 등이 존재함

➢ 타입을 알아보기 위한 type()

```
>>> type(3)
<class 'int'>
>>> type('3')
<class 'str'>
>>> type(3.0)
<class 'float'>
```

8. 변수

- input 함수와 변수는 찰떡궁합
  - x = input()은 사용자가 입력한 값을 x에 저장함
  - 주의: x는 문자열(string, str)임

```
>>> x = input()
3

>>> type(x)
<class 'str'>
```

변수 x의 타입이 무엇이냐?
str입니다! → 문자열 입니다!

  - 이걸로 덧셈 연산을 한다면?

```
>>> x + 5
Traceback (most recent call last):
  File "<pyshell#54>", line 1, in <module>
    x + 5
TypeError: can only concatenate str (not "int") to str
```

문자열(str)을 정수(integer, int)와 결합(concatenate, +)할 수 없어요
→ 타입이 다른 피연산자를 연산할 수 없음에 유의

8. 변수

- input 함수와 변수는 찰떡궁합
  - 문자열과 문자열 덧셈은 연결(결합)을 의미하므로 아래와 같이 가능함

```
>>> x + '5'
'35'
```

  - 만약 문자열 3을 숫자(정수, int)로 바꾸고 싶다면 아래와 같이 수행

```
>>> x = int(x)
>>> x
3
>>> type(x)
<class 'int'>
>>> x + 5
8
```

x 값을 int(정수)로 바꿔서 다시 x에 저장하라
x 값은?

x의 타입은?
int(정수)
x + 5는?
8

파이썬에서는 int 3과 str '3'이 엄연히 다릅니다!

8. 변수

- input 함수와 변수는 찰떡궁합
  - ➢ input 함수와 타입을 한꺼번에 사용하는 방법

```
>>> x = int(input())
3
>>> x
3
>>> type(x)
<class 'int'>
```

맨 위의 문장 설명

1. input()이 먼저 실행됨 → 문자열로 '3'이 입력
2. int('3')이 실행되어 정수(int)로 3이 됨
3. x에는 최종 정수(int) 3이 저장됨

수학에서 함수와 비슷한 논리임

f(x) = 3x

g(y) = 2y가 있을 때,

f(g(2))는?

→ g(2) = 2 * 2 = 4

f(4) = 3 * 4 = 12

  - ➢ 잠깐: 일단, 파이썬 함수에 대해 자세히 모르지만,
    어떤 이름 옆에 ()가 붙어 있다면 함수라고 생각
    예) print(), input(), type(), int(), float(), str(), …

8. 변수

- int(): 오른쪽 괄호 안에 있는 문자열이나 실수를 정수로 변환
  → 실수의 경우 소수점 이하는 버림
  → 만약 문자열이 정수 형태가 아니면 에러 발생

```
>>> x = int(3.14)
>>> x
3
>>> type(x)
<class 'int'>
```

```
>>> x = int('3.14')
Traceback (most recent call last):
  File "<pyshell#66>", line 1, in <module>
    x = int('3.14')
ValueError: invalid literal for int() with base 10: '3.14'
>>> x = int('안녕')
Traceback (most recent call last):
  File "<pyshell#67>", line 1, in <module>
    x = int('안녕')
ValueError: invalid literal for int() with base 10: '안녕'
```

10진 정수의 형태로 입력하세요!

8. 변수

- float(): 오른쪽 괄호 안에 있는 문자열이나 정수를 실수로 변환
  → 만약 문자열이 실수 형태가 아니면 에러 발생

```
>>> x = float(3)
>>> x
3.0
>>> type(x)
<class 'float'>
```

```
>>> x = float('3.14')
>>> x
3.14
>>> type(x)
<class 'float'>
>>> x = float('안녕!')
Traceback (most recent call last):
  File "<pyshell#79>", line 1, in <module>
    x = float('안녕!')
ValueError: could not convert string to float: '안녕!'
```

- str(): 오른쪽 괄호 안에 있는 정수나 실수를 문자열로 변환

```
>>> x = str(3)
>>> x
'3'
>>> type(x)
<class 'str'>
```

```
>>> x = str(3.0)
>>> x
'3.0'
>>> type(x)
<class 'str'>
```

8. 변수

- 아래 예제 모두 각각 소스 코드(pig.py, student.py)로 저장한다.
- 예제 1. 돼지 저금통(pig)이 있다고 가정할 때,
  100원, 200원, 300원을 넣어 최종 합계를 출력

- 예제 2. 학생 3명(student1, student2, student3)이
  있다고 가정할 때, 학생 3명의 키 합계와 평균을 출력
  (학생 3명의 키는 아래와 같이 입력(input)받는다.)

학생 1의 키: *170*
학생 2의 키: *173*
학생 3의 키: *167*
학생 3명의 키 합계: 510
학생 3명의 키 평균: 170.0

기울임 꼴은 사용자가 입력한 값!

힌트: student1 = int(input("학생 1의 키: "))

힌트: 사칙 연산자를 이용해보자.

8. 변수

- 아래 예제 모두 각각 소스 코드(job.py, number.py)로 저장한다.
- 예제 3. 이름(name)과 직업(job)을 입력 받아
  아래와 같이 출력하세요.

이름 입력: *Han su*
직업 입력: *학생*
Han su 학생 반가워요~

힌트: 문자열의 덧셈(+) 연산은 결합!

- 예제 4. 아래와 같은 프로그램을 작성하라.

실수 입력: *150.7*
정수로 변환한 값: 150
정수로 변환한 값의 파이썬 데이터 타입: <class: 'int'>

힌트: int()와 float(), type()을 잘 사용해보세요!

8. 변수

- 변수 이름 규칙
  - ➢ 대소문자를 구분함

```
>>> num = 0
>>> Num = 10      # 파이썬은 대소문자 구분
>>> print(num, Num)
0 10
```

  - ➢ 숫자로 시작할 수 없음

```
>>> 2num = 0      # 숫자로 시작하는 것은 불가
SyntaxError: invalid syntax
```

  - ➢ 파이썬 인터프리터와 약속한 단어(IDLE에서 색깔이 변경됨)는 사용 불가

```
>>> return = 2          # 키워드는 이름으로 쓸 수 없음
SyntaxError: invalid syntax
```

8. 변수

- 변수 이름 규칙
  - ➢ 파이썬 인터프리터와 약속한 단어 보기
    → 프로그래머와 파이썬 인터프리터가 서로 약속한 특정 명령의 단어

```
>>> 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']
>>>
```

문의사항 및 질문

- 혹시 질문있나요?

- 모르는 문제나 더 알고 싶은 사항이 있으면 언제든 연락 가능
  - 배재대학교 컴퓨터공학과 석사과정 이태준
  - Tel: 010-5223-2912
  - Email: marlrero@kakao.com

연습문제

- 각 문제를 소스 코드 형태로
  파일명(solution1.py, solution2.py, ...)로 저장하라.
- 문제 1. 예제에서 못 풀었던 문제 풀기
- 문제 2. 두 정수를 입력 받아 몫과 나머지를 구하시오.

  정수 1 입력: *5*
  정수 2 입력: *2*
  몫 2이고 나머지 1 입니다.

힌트: 문자열과 문자열끼리 덧셈으로 연결이 가능합니다.
문자열과 정수, 문자열과 실수는 연결이 불가능합니다.
따라서, str() 함수를 이용해 변환해서 +를 하거나,
print('몫 ', x, '이고', ...)와 같이 print 함수에서 콜론(,)을 사용해야 합니다.

몫과 나머지 연산은 앞서 사칙 연산에서 배웠습니다!

연습문제

- 문제 3. 아래의 코드를 실행해보고,
  이번 시간을 통해서 배운 것에서 어떤 것에 위배되는지
  확인해 보시오. (에러가 발생하지 않음)

```
>>> '안녕' * 3
'안녕안녕안녕'
```

- 문제 4. 변수 이름 규칙에 위배되는 2가지 사항을 테스트하여
  에러를 출력하시오.

  처음 공부할 때 에러를 많이 보는 것은 나중에 큰 도움이 됩니다.

- 문제 5. 정수, 실수, 문자열을 여러 가지 조합으로
  사칙 연산자를 사용해 결과가 어떻게 되는지
  분석하시오. (에러가 있을 수 있음)

다음 시간에 배울 내용

1. 리스트와 문자열
2. 참과 거짓, 조건문
3. 반복문 for과 while
4. 튜플과 레인지