

파일입출력 디버깅 및 예외처리

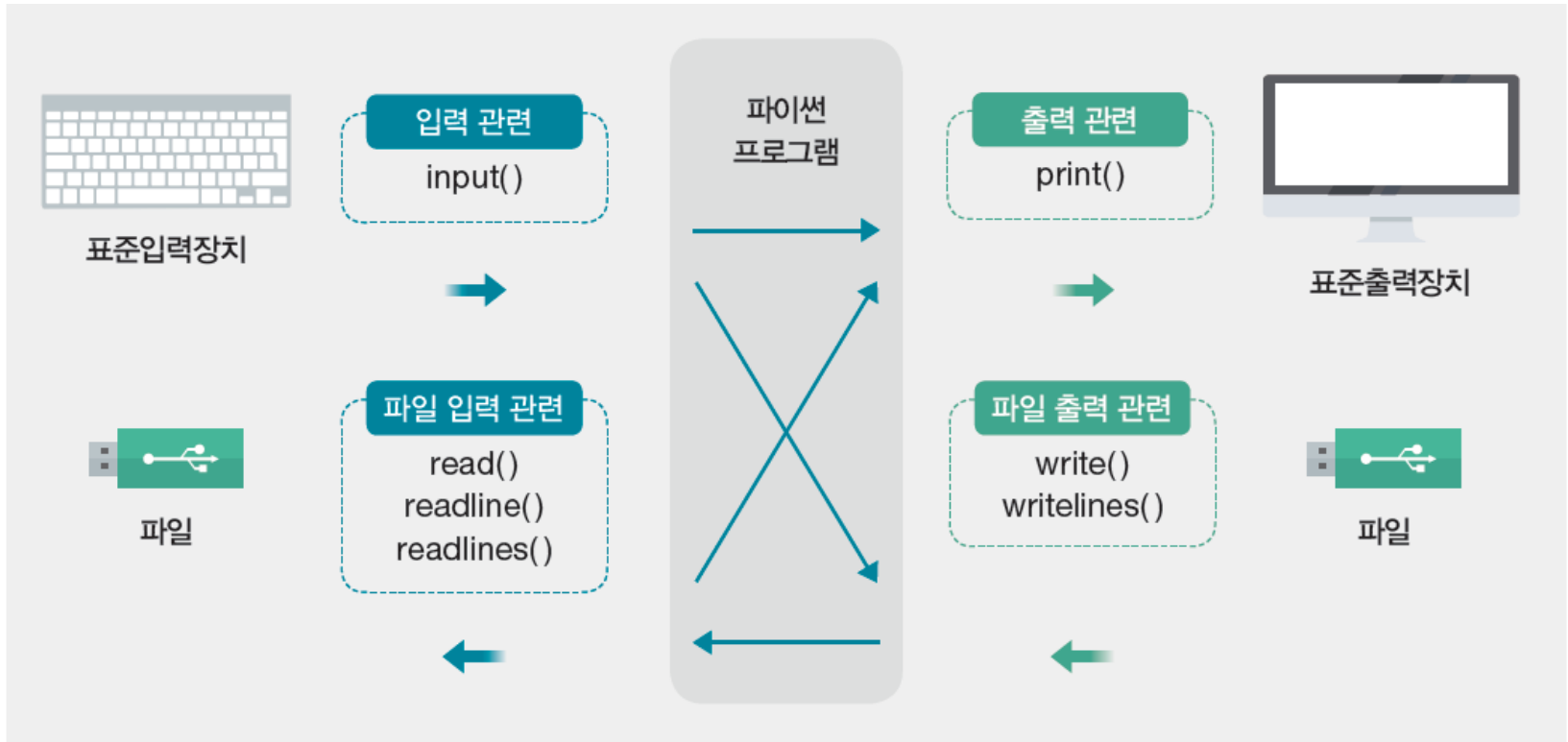
Seolyoung Jeong, Ph.D.

경북대학교 IT대학 컴퓨터학부

파일 입출력

표준 입출력 vs. 파일 입출력

◆ 입출력 과정



- 표준입출력 : 키보드로 입력하는 것을 표준 입력, 모니터로 출력하는 것을 표준 출력 (콘솔(Console) : 키보드 + 모니터)
- 키보드 + 모니터의 입출력 함수 : `input()` / `print()`
- 파일 읽기 함수 : `read()`, `readline()`, `readlines()`
- 파일 쓰기 함수 : `write()`, `writelines()`

파일 입출력 기본 과정



- **1단계 : 파일 열기**

- `open()` 함수 : 읽을 파일명, 읽기/쓰기(모드) 지정

읽기용 : 변수명 = `open("파일명", "r")`

쓰기용 : 변수명 = `open("파일명", "w")`

파일 입출력 기본 과정

– 파일 열기 모드

파일 열기 모드	의미
생략	r과 동일
r	읽기 모드, 기본값
w	쓰기 모드, 기존에 파일이 있으면 덮어씀.
r+	읽기/쓰기 겸용 모드
a	쓰기 모드, 기존에 파일이 있으면 이어서 씀. Append의 약자
t	텍스트 모드, 텍스트 파일을 처리. 기본값
b	바이너리 모드, 바이너리 파일(=이진 파일)을 처리

- 2단계 : 파일 처리
 - 파일에 데이터를 쓰거나 파일로부터 데이터를 읽어옴
- 3단계 : 파일 닫기

```
변수명.close()
```

텍스트 파일 입력 (읽기)

파일을 이용한 입력

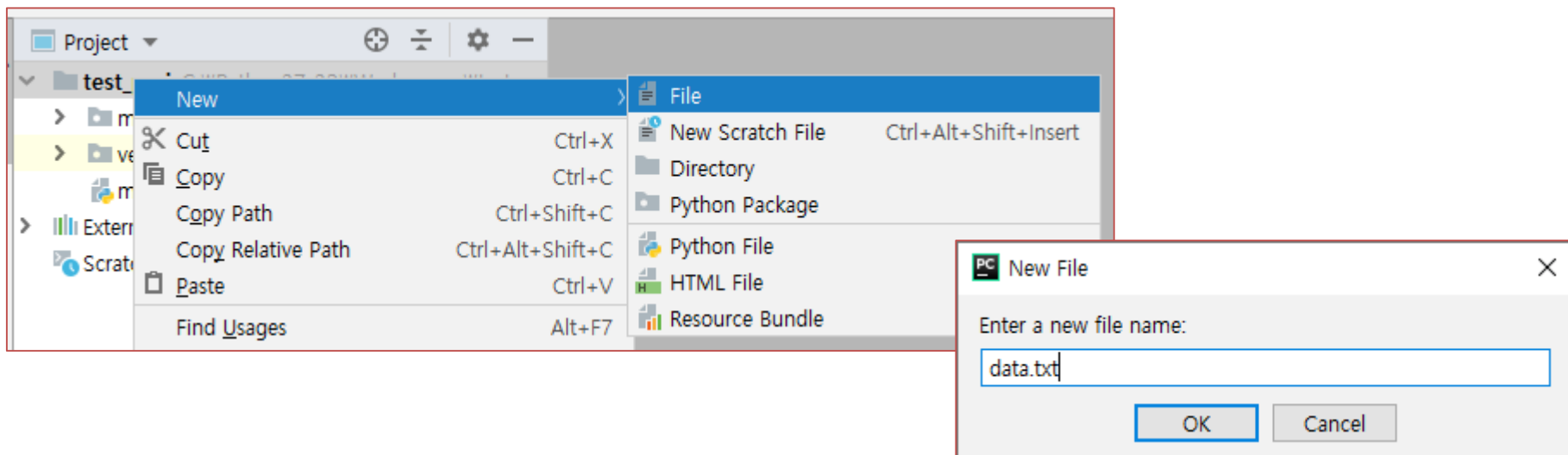
◆ 파일 입력과 표준 출력



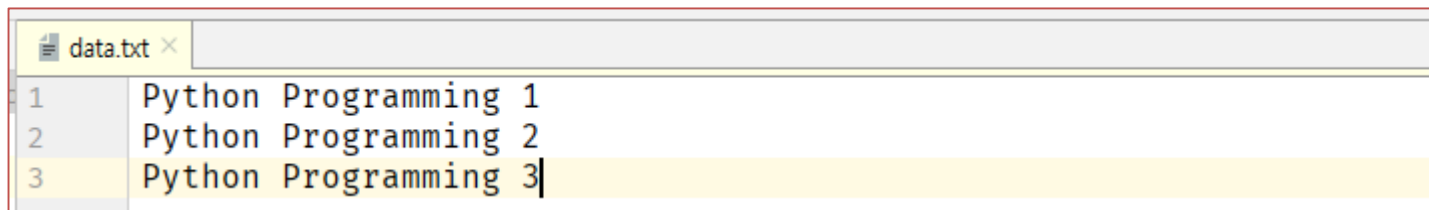
- `read()` : 파일 전체를 읽어 문자열로 return
- `readline()` : 한 줄을 읽어 return
- `readlines()` : 파일 전체를 읽어 list 형태로 return

파일 생성

◆ data.txt 파일 생성



◆ data.txt 파일 입력



read() 함수

```
data.txt x 01_read.py x
1 in_file = None # txt file
2 in_str = "" # read string
3
4 # 1. open file
5 in_file = open("data.txt", "r")
6
7 # 2. read
8 in_str = in_file.read()
9 print(in_str, end="")
10
11 # 3. close file
12 in_file.close()
```

```
Python Programming 1
Python Programming 2
Python Programming 3
Process finished with exit code 0
```

➔ 파일 전체 읽은 후, 출력

readline() 함수

```
1 in_file = None # txt file
2 in_str = ""    # read string
3
4 # 1. open file
5 in_file = open("data.txt", "r")
6
7 # 2. read
8 in_str = in_file.readline() # read line
9 print(in_str, end="")      # print string
10
11 # 3. close file
12 in_file.close()
```



A screenshot of a terminal window. The title bar says "Python Programming 1". The terminal output shows "Process finished with exit code 0". There are some small icons on the left side of the terminal window.

- 한 줄만 출력
- 모든 줄을 읽고 싶으면???

readline() 함수

```
1 in_file = None # txt file
2 in_str = "" # read string
3
4 # 1. open file
5 in_file = open("data.txt", "r")
6
7 # 2. read
8 while True:
9     in_str = in_file.readline() # read line
10    if in_str == "": break # end
11    print(in_str, end="") # print string
12
13 # 3. close file
14 in_file.close()
```

```
Python Programming 1
Python Programming 2
Python Programming 3
Process finished with exit code 0
```

➔ 한 줄씩 읽으며, 빈 문자이면 종료

readlines() 함수

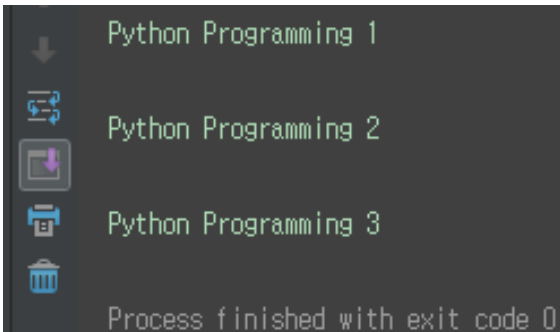
```
1 in_file = None # txt file
2 in_list = []   # list or string
3
4 # 1. open file
5 in_file = open("data.txt", "r")
6
7 # 2. read
8 in_list = in_file.readlines() # read
9 print(in_list)                # print string
10
11 # 3. close file
12 in_file.close()
```

```
['Python Programming 1\n', 'Python Programming 2\n', 'Python Programming 3']
Process finished with exit code 0
```

- ➔ 읽은 라인들을 리스트로 생성
- ➔ 한 라인씩 출력되도록 수정

readlines() 함수

```
1 in_file = None # txt file
2 in_list = []   # list or string
3 in_str = ""    # read string
4
5 # 1. open file
6 in_file = open("data.txt", "r")
7
8 # 2. read
9 in_list = in_file.readlines() # read
10 for in_str in in_list:
11     print(in_str)             # print string
12
13 # 3. close file
14 in_file.close()
```



Python Programming 1

Python Programming 2

Python Programming 3

Process finished with exit code 0

readlines() 함수

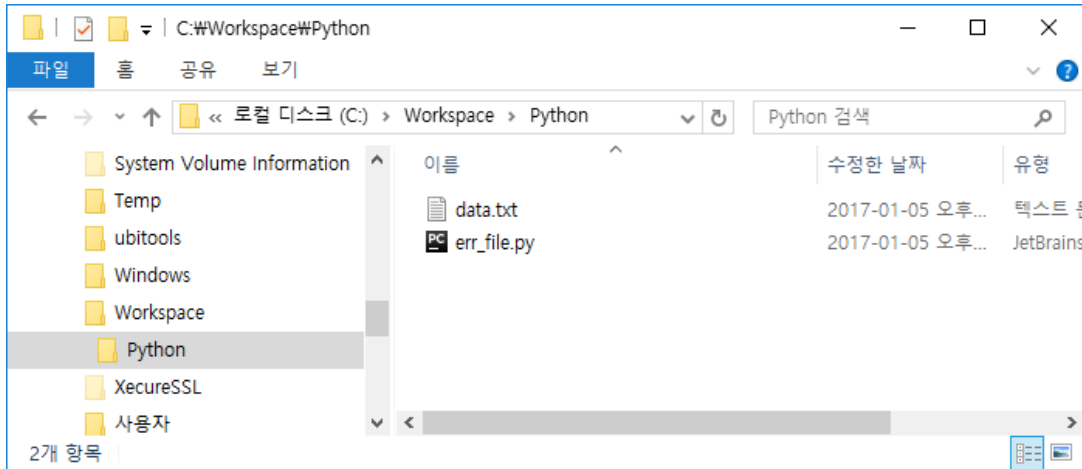
```
1 in_file = None # txt file
2 in_list = []   # list or string
3 in_str = ""    # read string
4
5 # 1. open file
6 in_file = open("data.txt", "r")
7
8 # 2. read
9 in_list = in_file.readlines() # read
10 for in_str in in_list:
11     print(in_str.strip('\n')) # print string
12
13 # 3. close file
14 in_file.close()
```

```
Python Programming 1
Python Programming 2
Python Programming 3
```

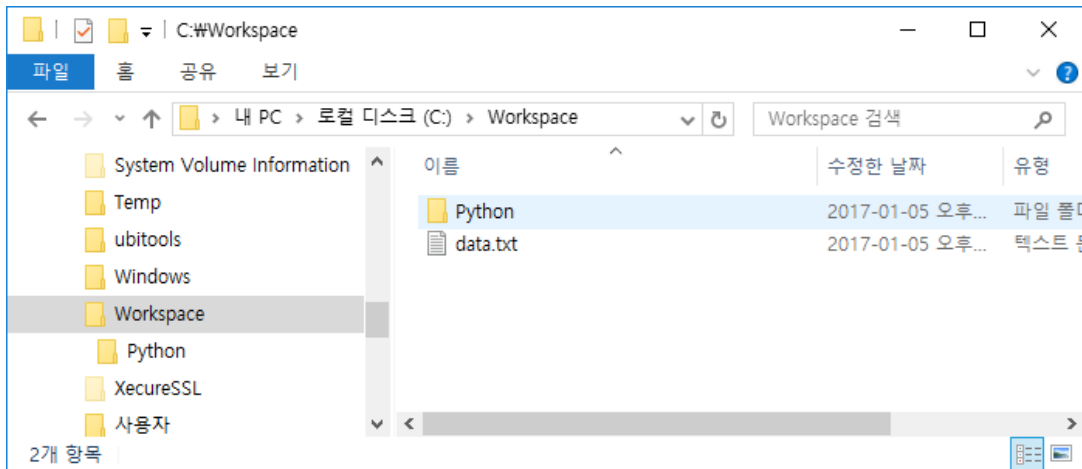
```
Process finished with exit code 0
```

open file 경로

- ◆ “data.txt” 파일 위치를 옮김 → 디렉터리 한 단계 위



예) `C:\Workspace\Python\data.txt`



예) `C:\Workspace\data.txt`

open file 경로

◆ 상대적 경로

```
1 in_file = None # txt file
2 in_list = [] # list or string
3 in_str = "" # read string
4
5 # 1. open file
6 in_file = open("../data.txt", "r")
7
8 # 2. read
9 in_list = in_file.readlines() # read
10 for in_str in in_list:
11     print(in_str.strip('\n')) # print string
12
13 # 3. close file
14 in_file.close()
```

◆ 절대적 경로

```
1 in_file = None # txt file
2 in_list = [] # list or string
3 in_str = "" # read string
4
5 # 1. open file
6 in_file = open("C:\\Workspace\\data.txt", "r")
7
8 # 2. read
9 in_list = in_file.readlines() # read
10 for in_str in in_list:
11     print(in_str.strip('\n')) # print string
12
13 # 3. close file
14 in_file.close()
```


file open 시 오류 처리

- ◆ 만약 존재하지 않은 파일을 읽고자 한다면?? → 에러 발생

```
1  in_file = None    # txt file
2  in_list = []      # list or string
3  in_str = ""       # read string
4
5  # 1. open file
6  in_file = open("data_1.txt", "r")
7
8  # 2. read
9  in_list = in_file.readlines()  # read
10 for in_str in in_list:
11     print(in_str.strip('\n'))    # print string
12
13 # 3. close file
14 in_file.close()
```

```
in_file = open("data_1.txt", "r")
FileNotFoundError: [Errno 2] No such file or directory: 'data_1.txt'

Process finished with exit code 1
```

file open 시 오류 처리

- ◆ 사용자로부터 파일명을 입력 받아 open 하는 프로그램
- ◆ open 시 OS 모듈을 사용하여 파일 존재 여부 확인

```
1  import os    # os module for searching file path
2
3  file_name = ""    # file name
4  in_file = None    # txt file
5  in_str = ""    # read string
6
7  while True:
8      file_name = input("\ninput file name : ")
9
10     # check file path
11     if os.path.exists(file_name):
12         # 1. open file
13         in_file = open(file_name, "r")
14
15         # 2. read
16         in_list = in_file.readlines() # read
17         for in_str in in_list:
18             print(in_str.strip('\n')) # print string
19
20         # 3. close file
21         in_file.close()
22
23     else:
24         print("error - '%s' file no exist..." % file_name)
```

file open 시 오류 처리



```
↓  
input file name : date.txt  
error - 'date.txt' file no exist...  
input file name : data.txt  
Python Programming 1  
Python Programming 2  
Python Programming 3  
input file name :
```

◆ `os.path.exists(path)`

입력 받은 경로가 존재하면 `True`를 반환

존재하지 않는 경우는 `False`를 반환

(주의. 리눅스에서는 파일이나 디렉터리가 존재하지만,
읽기 권한이 없는 경우에도, `False`를 반환할 수 있음)

File Encoding

- ◆ 문자집합 (character set) : 텍스트를 기호로 표현
- ◆ ASCII (American Standard Code for Information Interchange)
 - 미국 정보 교환 표준 부호
 - 1960 제정된 문자 집합, 이후 개발된 문자 집합들의 토대가 됨
 - ASCII는 7bits를 이용하여 음이 아닌 수(0~127)에 문자를 할당
예) '=' [67], 'A' [65], 'a' [97]
 - 52개의 알파벳 대소문자(A~Z, a~z), 10개의 숫자(0~9), 32개의 특수문자(!@#\$ 등), 1개의 공백 문자, 33개의 출력 불가능 제어문자 → 128개 문자 표현

File Encoding

◆ ASCII Table

Hex	Dec	Char	Hex	Dec	Char	Hex	Dec	Char	Hex	Dec	Char
0x00	0	NULL null	0x20	32	Space	0x40	64	@	0x60	96	`
0x01	1	SOH Start of heading	0x21	33	!	0x41	65	A	0x61	97	a
0x02	2	STX Start of text	0x22	34	"	0x42	66	B	0x62	98	b
0x03	3	ETX End of text	0x23	35	#	0x43	67	C	0x63	99	c
0x04	4	EOT End of transmission	0x24	36	\$	0x44	68	D	0x64	100	d
0x05	5	ENQ Enquiry	0x25	37	%	0x45	69	E	0x65	101	e
0x06	6	ACK Acknowledge	0x26	38	&	0x46	70	F	0x66	102	f
0x07	7	BELL Bell	0x27	39	'	0x47	71	G	0x67	103	g
0x08	8	BS Backspace	0x28	40	(0x48	72	H	0x68	104	h
0x09	9	TAB Horizontal tab	0x29	41)	0x49	73	I	0x69	105	i
0x0A	10	LF New line	0x2A	42	*	0x4A	74	J	0x6A	106	j
0x0B	11	VT Vertical tab	0x2B	43	+	0x4B	75	K	0x6B	107	k
0x0C	12	FF Form Feed	0x2C	44	,	0x4C	76	L	0x6C	108	l
0x0D	13	CR Carriage return	0x2D	45	-	0x4D	77	M	0x6D	109	m
0x0E	14	SO Shift out	0x2E	46	.	0x4E	78	N	0x6E	110	n
0x0F	15	SI Shift in	0x2F	47	/	0x4F	79	O	0x6F	111	o
0x10	16	DLE Data link escape	0x30	48	0	0x50	80	P	0x70	112	p
0x11	17	DC1 Device control 1	0x31	49	1	0x51	81	Q	0x71	113	q
0x12	18	DC2 Device control 2	0x32	50	2	0x52	82	R	0x72	114	r
0x13	19	DC3 Device control 3	0x33	51	3	0x53	83	S	0x73	115	s
0x14	20	DC4 Device control 4	0x34	52	4	0x54	84	T	0x74	116	t
0x15	21	NAK Negative ack	0x35	53	5	0x55	85	U	0x75	117	u
0x16	22	SYN Synchronous idle	0x36	54	6	0x56	86	V	0x76	118	v
0x17	23	ETB End transmission block	0x37	55	7	0x57	87	W	0x77	119	w
0x18	24	CAN Cancel	0x38	56	8	0x58	88	X	0x78	120	x
0x19	25	EM End of medium	0x39	57	9	0x59	89	Y	0x79	121	y
0x1A	26	SUB Substitute	0x3A	58	:	0x5A	90	Z	0x7A	122	z
0x1B	27	FSC Escape	0x3B	59	;	0x5B	91	[0x7B	123	{
0x1C	28	FS File separator	0x3C	60	<	0x5C	92	\	0x7C	124	
0x1D	29	GS Group separator	0x3D	61	=	0x5D	93]	0x7D	125	}
0x1E	30	RS Record separator	0x3E	62	>	0x5E	94	^	0x7E	126	~
0x1F	31	US Unit separator	0x3F	63	?	0x5F	95	_	0x7F	127	DEL

File Encoding

◆ 유니코드 (Unicode)

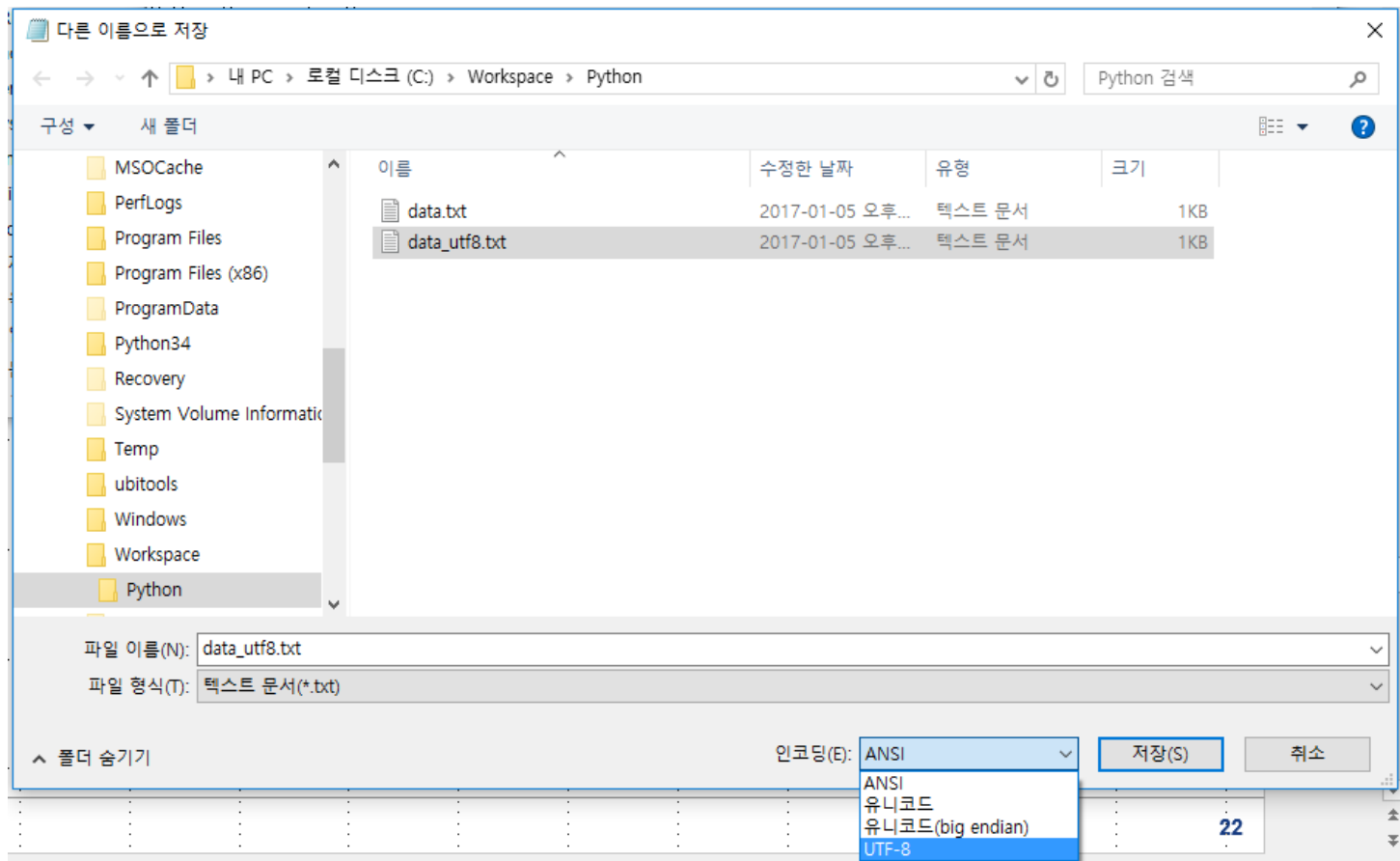
- 문자 집합 하나로 모든 문자를 표현하고자 하는 것이 목적
- 초기에는 전 세계의 언어별 문자들을 2bytes 안에서 영역을 나눠 할당
- 유니코드에서 문자에 부여되는 번호를 코드 포인트(code point)라고 함
 - 코드포인트는 “U+” 뒤에 2바이트의 수를 16진수로 표현하여 붙여 표시
예) ‘A’의 코드 포인트는 U+0041, ‘한’의 코드 포인트는 U+D55C, ‘글’의 코드 포인트는 U+AE00
 - U+0000부터 U+007F까지를 ASCII와 동일하게 맞춰두고 그 뒤 번호부터 각국의 문자를 할당
- 누락된 현대 문자와 기호를 추가적으로 할당하고, 고대문자와 음악기호 등을 추가하자 코드포인트가 2bytes를 넘어서게 됨

◆ UTF(Unicode Transformation Format)

- 유니코드 변환 인코딩 형식
- UTF-8은 코드포인트의 크기에 따라 1byte에서부터 4bytes까지 가변폭으로 인코딩하므로 1byte로 표현 가능한 U+0000(십진수 0)부터 U+007F(십진수 127)까지는 ASCII와 완벽하게 호환
 - UTF-8 인코딩 방식으로 저장된 문서는 유니코드를 알지 못하는 시스템에서도 사용 가능
- UTF-8 외에도 UTF-7, UTF-16, UTF-32 인코딩 등이 있음

File Encoding

- ◆ 유니코드 형식 파일 생성
- ◆ “data.txt” 파일을 메모장에서 열기
- ◆ 다른 이름으로 저장 “data_utf8.txt” (인코딩 : UTF-8)



File Encoding

- ◆ “data_utf8.txt” 파일 열기 및 읽기 → 깨져 보임

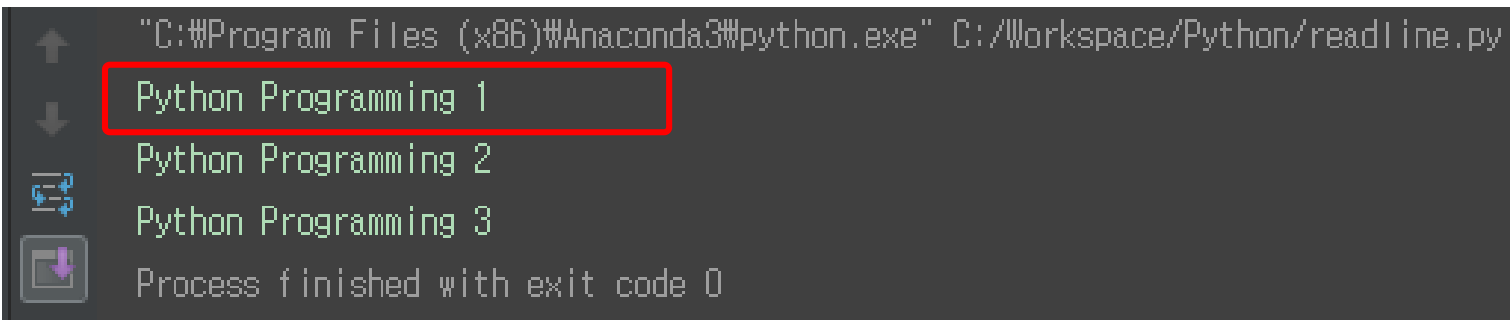
```
1 in_file = None # txt file
2 in_list = [] # list or string
3 in_str = "" # read string
4
5 # 1. open file
6 in_file = open("data_utf8.txt", "r")
7
8 # 2. read
9 in_list = in_file.readlines() # read
10 for in_str in in_list:
11     print(in_str.strip('\n')) # print string
12
13 # 3. close file
14 in_file.close()
```

```
↑ "C:\Program Files (x86)\Anaconda3\python.exe" C:/Workspace/Python/readline.py
↓ 癡型python Programming 1
Python Programming 2
Python Programming 3
Process finished with exit code 0
```


File Encoding

- ◆ “data_utf8.txt” 파일 열기 (인코딩 파라미터 전달) → 정상 출력

```
1 in_file = None # txt file
2 in_list = [] # list or string
3 in_str = "" # read string
4
5 # 1. open file
6 in_file = open("data_utf8.txt", "r", encoding='utf-8')
7
8 # 2. read
9 in_list = in_file.readlines() # read
10 for in_str in in_list:
11     print(in_str.strip('\n')) # print string
12
13 # 3. close file
14 in_file.close()
```



```
"C:\Program Files (x86)\Anaconda3\python.exe" C:/Workspace/Python/readline.py
Python Programming 1
Python Programming 2
Python Programming 3
Process finished with exit code 0
```

자동으로 파일 객체 닫기

- ◆ 1. Open File
 - ◆ 2. Read/Write File
 - ◆ 3. Close File
- ◆ → with ~ as ~ : 파일 사용 뒤 자동으로 파일 객체 닫음

```
1 in_file = None # txt file
2 in_list = [] # list or string
3 in_str = "" # read string
4
5 # 1.open & 3.close file
6 with open("data.txt", "r") as in_file:
7     # 2.read
8     in_list = in_file.readlines() # read
9     for in_str in in_list:
10        print(in_str.strip('\n')) # print string
```

텍스트 파일 출력 (쓰기)

파일을 이용한 출력

◆ 표준 입력과 파일 출력



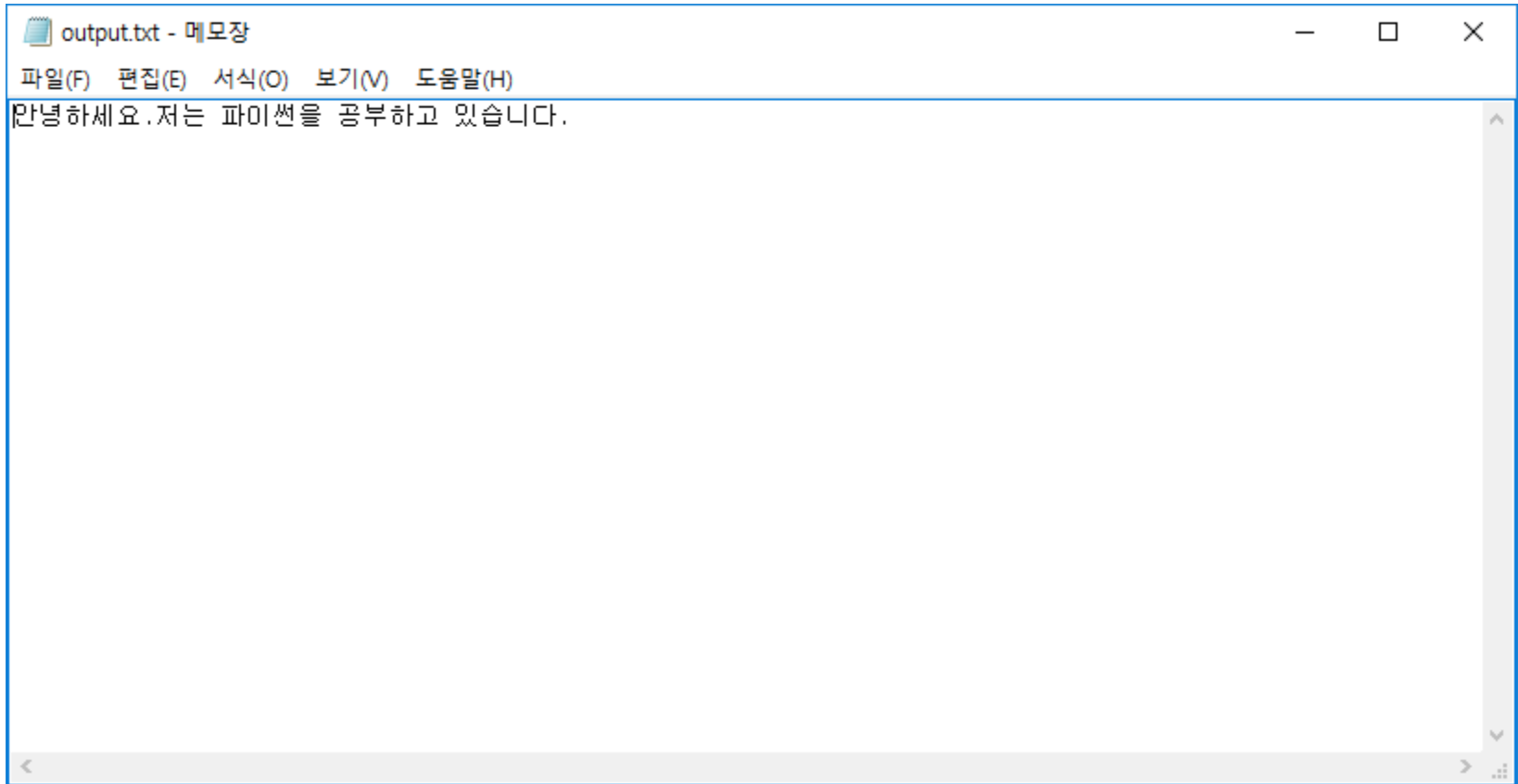
write() 함수

```
1 out_file = None
2 out_str = ""
3
4 # 1. open file
5 out_file = open("output.txt", "w")
6
7 while True:
8     out_str = input("input string : ")
9
10    if out_str == "exit": break
11
12    # 2. write string
13    out_file.write(out_str)
14
15    # e. close file
16    out_file.close()
```

```
↑ "C:\Program Files (x86)\Anaconda3\python.exe" "D:/수업관련/2016_2학기 겨울계절(Python)/sources/PY12/06_write.py"
↓ input string : 안녕하세요.
↕ input string : 저는 파이썬을 공부하고 있습니다.
↕ input string : exit
⏏ Process finished with exit code 0
```

write() 함수

- ◆ 생성된 “output.txt” 파일 직접 열어보기



write() 함수

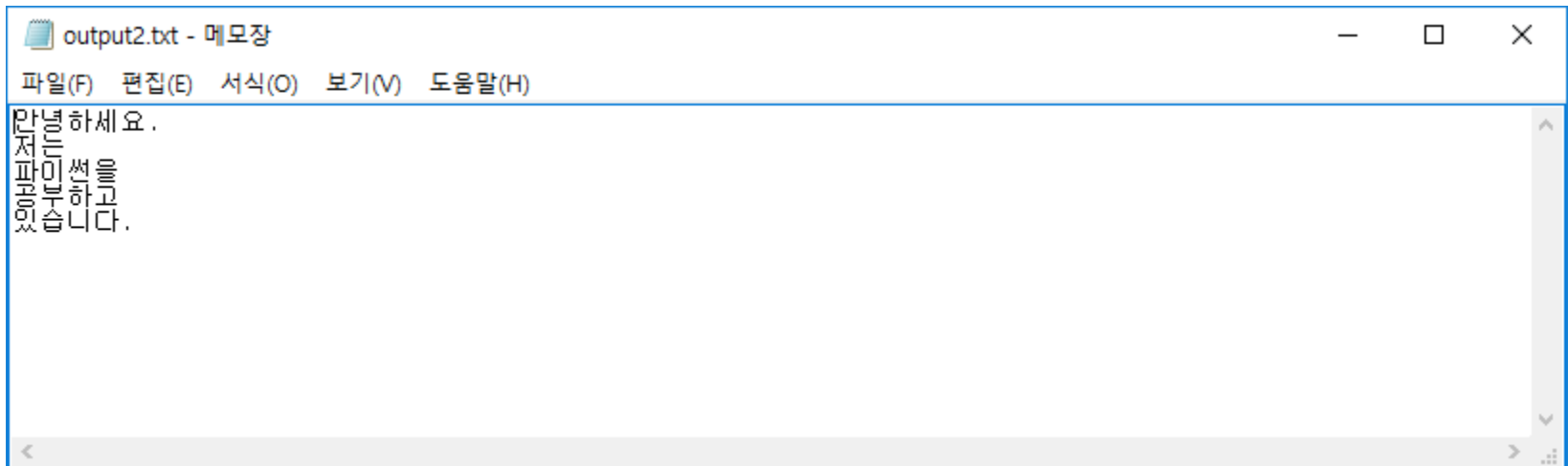
◆ 줄 띄우기

```
1 out_file = None
2 out_str = ""
3
4 # 1. open file
5 out_file = open("output.txt", "w")
6
7 while True:
8     out_str = input("input string : ")
9
10    if out_str == "exit": break
11
12    # 2. write string
13    out_file.write(out_str)
14    out_file.write('\n')
15
16 # e. close file
17 out_file.close()
```

writelines() 함수

◆ 리스트로부터 파일 쓰기

```
1 out_file = None
2 str_list = ["안녕하세요.\n", "저는\n", "파이썬을\n", "공부하고\n", "있습니다."]
3
4 # 1. open file
5 out_file = open("output2.txt", "w")
6
7 # 2. write string
8 out_file.writelines(str_list)
9
10 # e. close file
11 out_file.close()
```



output2.txt - 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

안녕하세요.
저는
파이썬을
공부하고
있습니다.

write() + join()

◆ 리스트로부터 파일 쓰기 + '\n' 추가 (join 함수 사용)

```
1 out_file = None
2 str_list = ["안녕하세요.", "저는", "파이썬을", "공부하고", "있습니다."]
3
4 # 1. open file
5 out_file = open("output2.txt", "w")
6
7 # 2. write string
8 out_file.write('\n'.join(str_list))
9
10 # 3. close file
11 out_file.flush()
12 out_file.close()
```

◆ flush 함수

- 내부 출력버퍼 비움
- 버퍼에 다 채워지지 않았어도, 출력 파일에 씀

바이너리 파일

바이너리 파일

- ◆ 텍스트 파일은 사람이 읽을 수 있는 글자로 구성된 파일
- ◆ 바이너리(Binary, 이진) 파일은 시스템이 읽을 수 있는 bit 단위로 의미가 있는 파일
예) 그림, 음악, 동영상, exe 파일 등
- ◆ 읽을 때 'rb'
- ◆ 쓸 때 'wb'

JPG 파일 읽기

```
1 copy_file = open("copy.jpg", "wb")
2
3 # open binary file
4 bin_file = open("cafe.jpg", "rb")
5 #print(bin_file.read())
6
7 # read and write binary file
8 copy_file.write(bin_file.read())
9
10 # close file
11 bin_file.close()
12 copy_file.close()
```



```
"C:\Program Files (x86)\Anaconda3\python.exe" "D:/수업관련/2016
b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x01\x02\x00\x0d\x00\x00'
Process finished with exit code 0
```

<바이너리 내용 화면 출력>

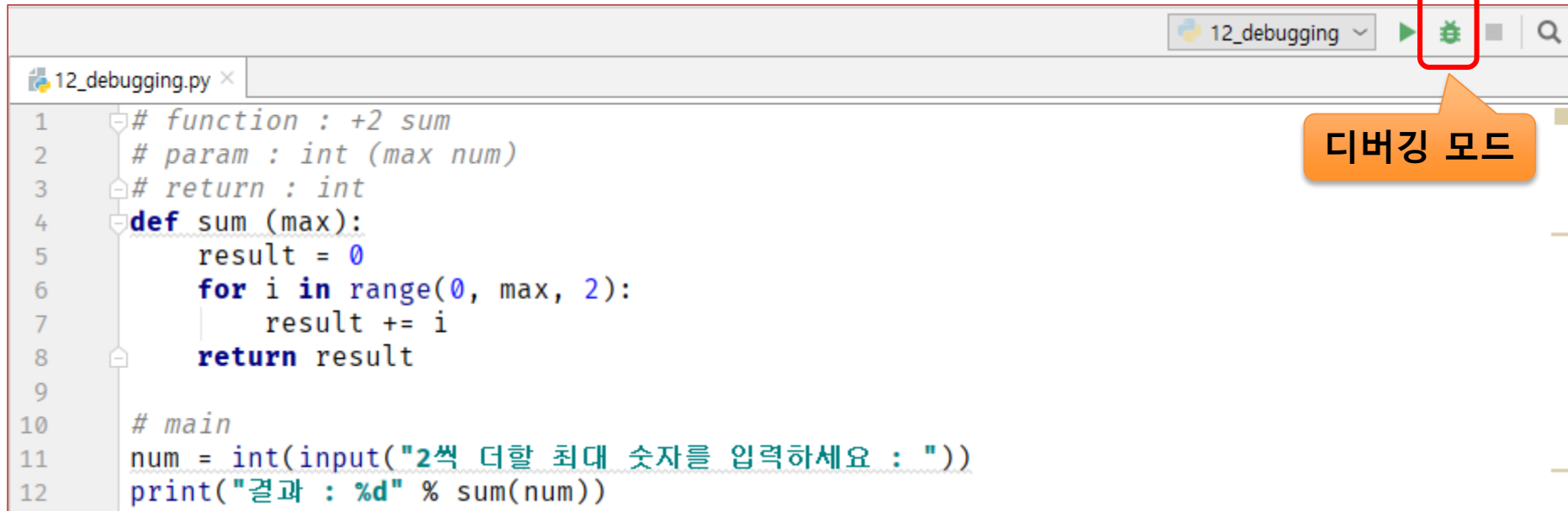
디버깅

디버깅

- ◆ 오류 처리 (Debugging) : 프로그램 오동작의 원인이 되는 버그(Bug)를 제거하는 것
- ◆ 오류 종류
 - 컴파일 오류 : Syntax Error (문법에 맞지 않게 작성한 오류, 쉽게 찾을 수 있음!!!)
 - Run-time 오류 : 컴파일 시에는 오류가 없었으나, 프로그램 실행 시 잘못된 연산으로 인해 에러 발생
 - Logic 오류 : 프로그램은 정상적으로 실행되나, 연산 결과가 잘못되어 의도하지 않은 결과를 출력. 수정하기가 가장 어려움!!!

디버깅 in Pycharm

- ◆ 사용자로부터 max 값을 입력받은 후, 0~max값까지 2씩 더한 결과 출력



```
1  # function : +2 sum
2  # param : int (max num)
3  # return : int
4  def sum (max):
5      result = 0
6      for i in range(0, max, 2):
7          result += i
8      return result
9
10 # main
11 num = int(input("2씩 더할 최대 숫자를 입력하세요 : "))
12 print("결과 : %d" % sum(num))
```

디버깅 in Pycharm

◆ Debugging Mode

The screenshot shows the PyCharm IDE interface with a Python file named `12_debugging.py` open. The code is as follows:

```
1 # function : +2 sum
2 # param : int (max num)
3 # return : int
4 def sum(max): max: 5
5     result = 0 result: 0
6     for i in range(0, max, 2): i: 2
7         result += i
8     return result
9
10 # main
11 num = int(input("2씩 더할 최대 숫자를 입력하세요 : "))
12 print("결과 : %d" % sum(num))
13
14
```

A breakpoint is set at line 6. The bottom panel shows the Debug console with the current frame `sum()` and the following variables:

- `i` = {int} 2
- `max` = {int} 5
- `result` = {int} 0

The bottom status bar shows the current configuration: 7:1 CRLF UTF-8 4 spaces Python 3.7.

디버깅 in Pycharm

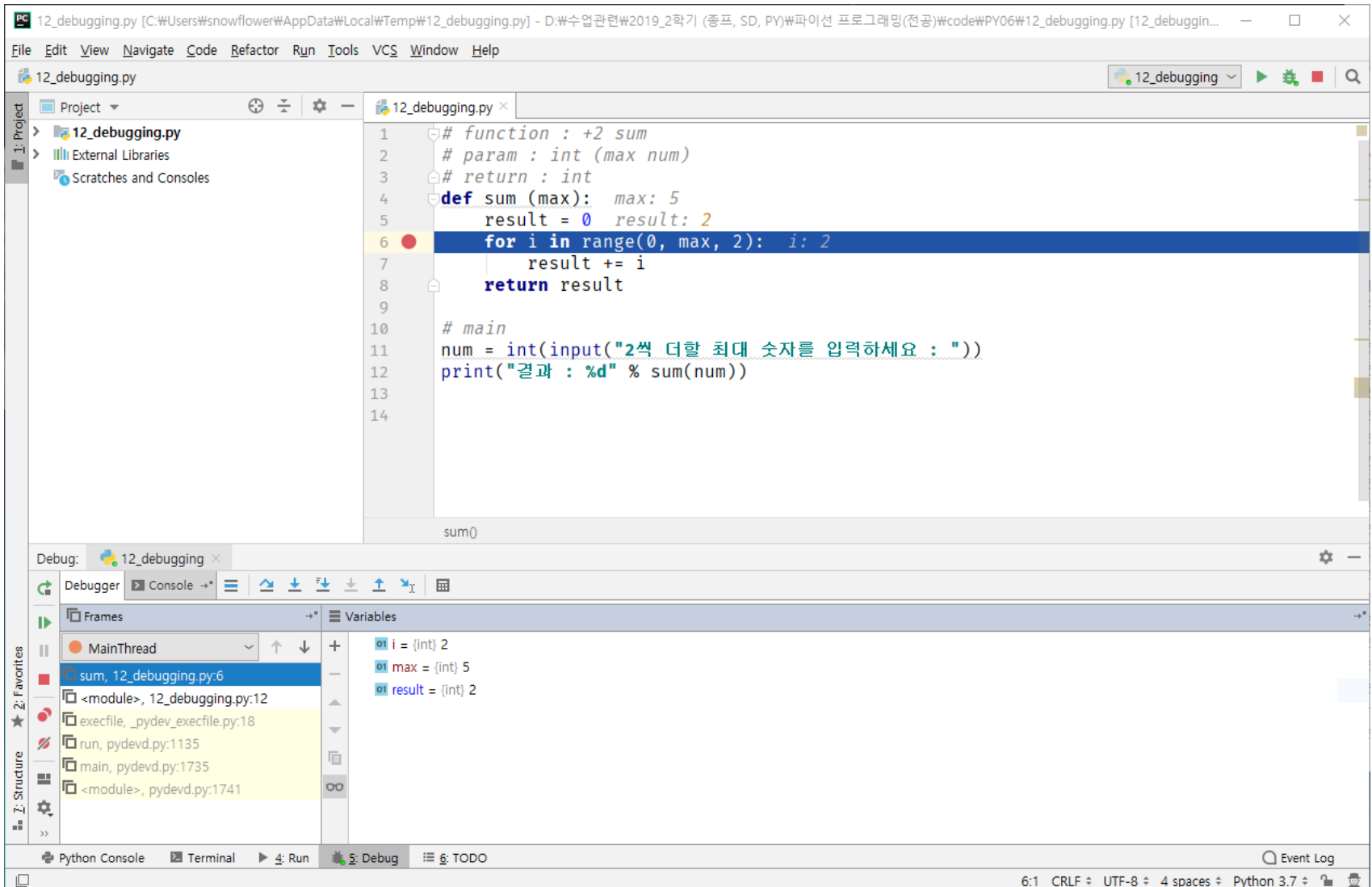
◆ Set “Breakpoint”

- 프로그램 실행을 중간에 멈추고, 데이터를 확인하고 싶은 라인에 포인트를 만들어 둬

```
1  # function : +2 sum
2  # param : int (max num)
3  # return : int
4  def sum (max):  max: 5
5      result = 0  result: 0
6  ●  for i in range(0, max, 2):  i: 2
7  ←  result += i
8      return result
9
10 # main
11 num = int(input("2씩 더할 최대 숫자를 입력하세요 : "))
12 print("결과 : %d" % sum(num))
```

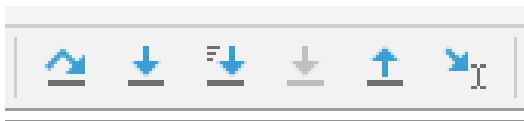
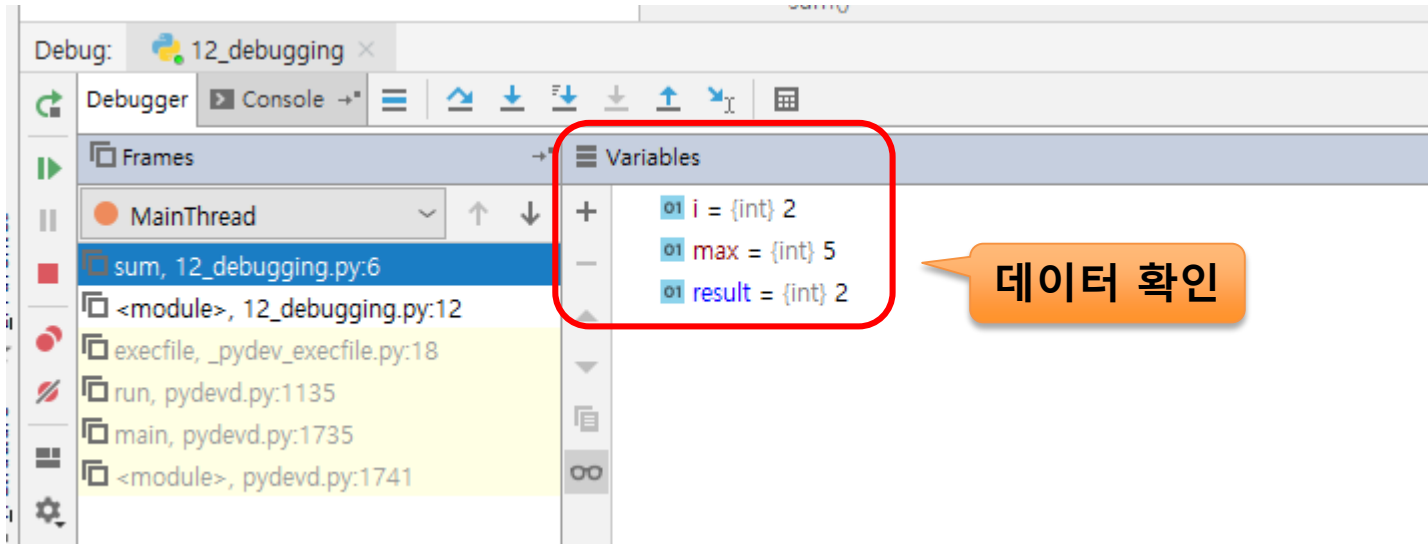
디버깅 in Pycharm

- ◆ 디버깅 모드로 실행 시 breakpoint가 있는 지점에서 일시 정지됨



디버깅 in Pycharm

◆ 디버깅 관련 메뉴



메뉴	기능
Step Over	method 단위의 debugging
Step Into	라인 단위의 debugging
Step into My Code	라이브러리 소스는 단계적으로 건너 뛰고 자신의 코드 안에서만 한 라인씩 진행
Step Out	디버깅 중인 메서드 종료
Run to cursor	Cursor 위치 까지 실행

예외처리

예외처리

- ◆ 프로그램 실행 중 발생한 오류를 처리하는 방법
- ◆ 문법적으로는 문제가 없으나, 실행 중 오류 발생

```
1 while True:
2     num = int(input("숫자 입력 : "))
3     print("결과값 : %d" % (100/num))
```

```
num = int(input("숫자 입력 : "))
ValueError: invalid literal for int() with base 10: '3.5'

Process finished with exit code 1
```

예외처리 코드

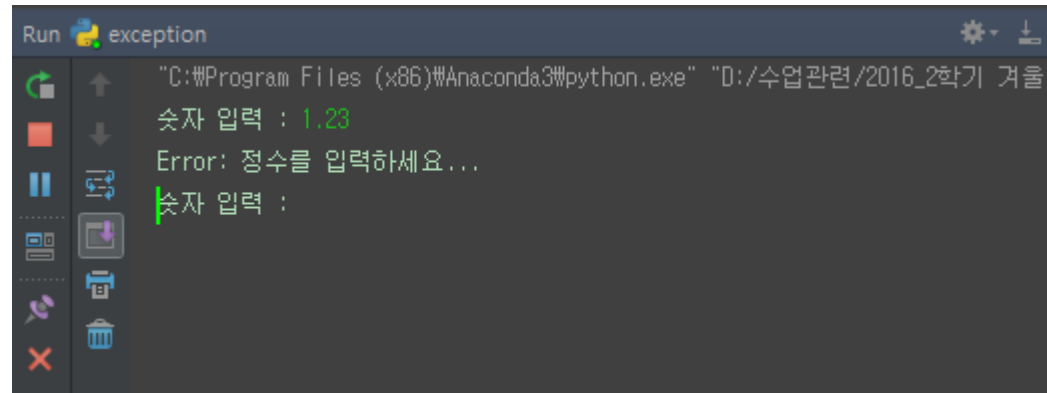
try:

문제가 없을 시 실행할 코드

except:

문제 발생 시 실행할 코드

```
1 while True:
2     try:
3         num = int(input("숫자 입력 : "))
4         print("결과값 : %d" % (100/num))
5
6     except:
7         print("Error: 정수를 입력하세요...")
```



```
Run exception
"C:\Program Files (x86)\Anaconda3\python.exe" "D:/수업관련/2016_2학기 겨울
숫자 입력 : 1.23
Error: 정수를 입력하세요...
숫자 입력 :
```

예외형식 예외처리

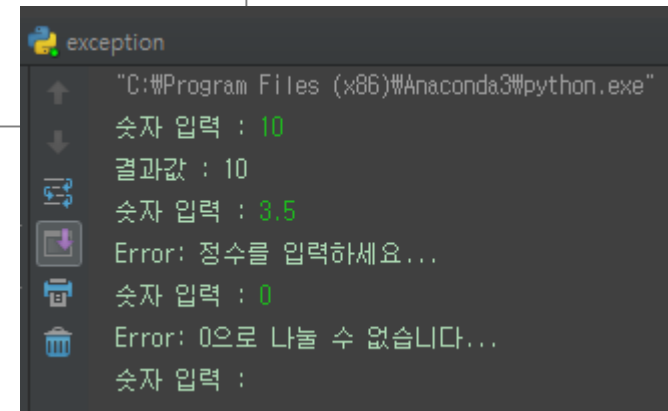
`try:`
문제가 없을 시 실행할 코드
`except 예외형식:`
문제 발생 시 실행할 코드

```
1 while True:
2     try:
3         num = int(input("숫자 입력 : "))
4         print("결과값 : %d" % (100/num))
5
6     except ValueError:
7         print("Error: 정수를 입력하세요...")
```

여러 개의 예외처리

```
try:
    문제가 없을 시 실행할 코드
except 예외형식 1:
    문제 발생 시 실행할 코드
except 예외형식 2:
    문제 발생 시 실행할 코드
```

```
1 while True:
2     try:
3         num = int(input("숫자 입력 : "))
4         print("결과값 : %d" % (100/num))
5
6     except ValueError:
7         print("Error: 정수를 입력하세요...")
8
9     except ZeroDivisionError:
10        print("Error: 0으로 나눌 수 없습니다...")
```



```
exception
"C:\Program Files (x86)\Anaconda3\python.exe"
숫자 입력 : 10
결과값 : 10
숫자 입력 : 3.5
Error: 정수를 입력하세요...
숫자 입력 : 0
Error: 0으로 나눌 수 없습니다...
숫자 입력 :
```


예외 형식 종류

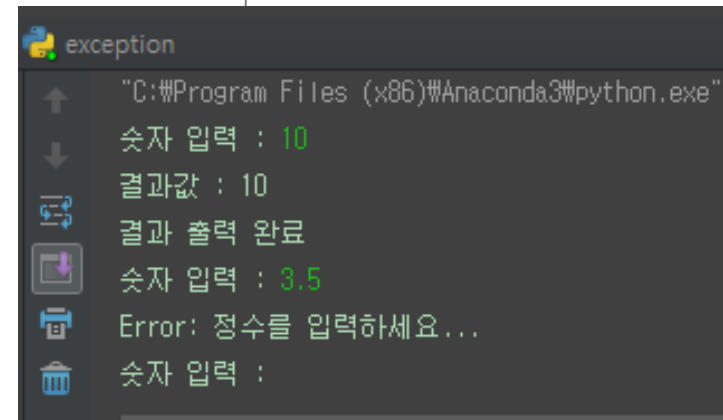
<https://docs.python.org/3.7/library/exceptions.html#builtin-exceptions>

BaseException	시스템 종료	
+-- SystemExit		
+-- KeyboardInterrupt	키보드로 인한 종료	
+-- GeneratorExit		
+-- Exception		
+-- StopIteration		
+-- StandardError		
+-- BufferError		
+-- ArithmeticError	계산 오류	
+-- FloatingPointError		
+-- OverflowError		
+-- ZeroDivisionError		
+-- AssertionError		
+-- AttributeError		
+-- EnvironmentError		
+-- IOError		
+-- OSError		
+-- WindowsError (Windows)		
+-- VMSError (VMS)		
+-- EOFError		
+-- ImportError	잘못된 import	
+-- LookupError		
+-- IndexError		
+-- KeyError		
		+-- MemoryError
		+-- NameError
		+-- UnboundLocalError
		+-- ReferenceError
		+-- RuntimeError
		+-- NotImplementedError
		+-- SyntaxError
		+-- IndentationError
		+-- TabError
		+-- SystemError
		+-- TypeError
		+-- ValueError
		+-- UnicodeError
		+-- UnicodeDecodeError
		+-- UnicodeEncodeError
		+-- UnicodeTranslateError
		+-- Warning
		+-- DeprecationWarning
		+-- PendingDeprecationWarning
		+-- RuntimeWarning
		+-- SyntaxWarning
		+-- UserWarning
		+-- FutureWarning
		+-- ImportWarning
		+-- UnicodeWarning
		+-- BytesWarning

예외처리 후 나머지 처리

```
try:
    문제가 없을 시 실행할 코드
except 예외형식 1:
    문제 발생 시 실행할 코드
except 예외형식 2:
    문제 발생 시 실행할 코드
else:
    except 절을 만나지 않을 시 실행할 코드
```

```
1 while True:
2     try:
3         num = int(input("숫자 입력 : "))
4         print("결과값 : %d" % (100/num))
5
6     except ValueError:
7         print("Error: 정수를 입력하세요...")
8
9     except ZeroDivisionError:
10        print("Error: 0으로 나눌 수 없습니다...")
11
12    else:
13        print("결과 출력 완료")
```



exception

"C:\Program Files (x86)\Anaconda3\python.exe"

숫자 입력 : 10

결과값 : 10

결과 출력 완료

숫자 입력 : 3.5

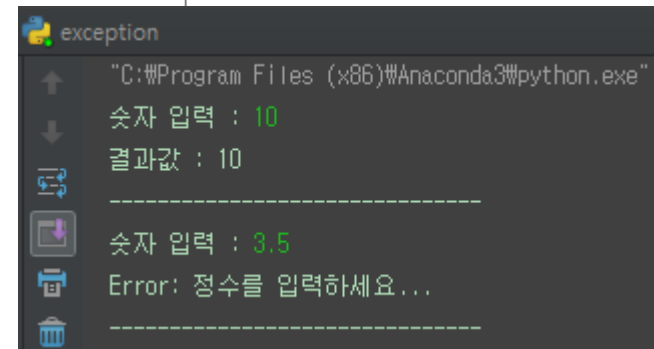
Error: 정수를 입력하세요...

숫자 입력 :

예외처리 후 나머지 처리

```
try:  
    문제가 없을 시 실행할 코드  
except 예외형식 1:  
    문제 발생 시 실행할 코드  
except 예외형식 2:  
    문제 발생 시 실행할 코드  
finally:  
    항상 실행되는 코드
```

```
1 while True:  
2     try:  
3         num = int(input("숫자 입력 : "))  
4         print("결과값 : %d" % (100/num))  
5  
6     except ValueError:  
7         print("Error: 정수를 입력하세요...")  
8  
9     except ZeroDivisionError:  
10        print("Error: 0으로 나눌 수 없습니다...")  
11  
12    finally:  
13        print("-----")
```



exception

"C:\Program Files (x86)\Anaconda3\python.exe"

숫자 입력 : 10
결과값 : 10

숫자 입력 : 3.5
Error: 정수를 입력하세요...

파이썬 에러 메시지

파이썬 에러 메시지

구문에러 (SyntaxError) : 명령의 조건 중 따옴표 오류 등, 구문오류

1) 따옴표나 괄호 닫기 오류

SyntaxError : EOL while scanning string literal

SyntaxError : unexpected EOF while parsing

2) 철자나 따옴표를 빼먹은 경우

SyntaxError : invalid syntax

3) 반복 블록의 들여쓰기 오류

SyntaxError : expected as indented block

SyntaxError : unindent does not match any outer indentation level

SyntaxError : unexpected indent (들여쓰지 말아야 할 곳을 들여쓴 경우)

파이썬 에러 메시지

이름에러 (NameError) : 명령의 철자 오류

```
TraceBack (most recent call last):  
  File "<pyshell#7>", line 1 ,in <module>  
    PRINT("Hello")  
NameError: name "PRINT" in not defined
```

외부모듈 호출오류 (Import Error) : Import 로 호출 모듈이름 오류

```
TraceBack (most recent call last):  
  File "<pyshell#10>", line 11 ,in <module>  
    import turtle as t  
ImportError: No module named 'turtle'
```

파이썬 에러 메시지

속성 오류 (AttributeError) : 호출 모듈의 함수, 변수를 잘못 입력

TraceBack (most recent call last):

File "<pyshell#18>", line 21 ,in <module>

t.forward(50)

AttributeError: 'module' object has no attribute 'forward'

타입 에러 (TypeError) : 함수에 전달할 인자가 빠진 경우

TypeError : ... **missing**... required positional argument :

값 에러 (ValueError) : 정수, 문자 간 값 변환이 불가능 오류

ValueError : **invalid literal** for ... ():

Any Questions...
Just Ask!

