

[Opensource Basic Project]

오픈소스 기초 프로젝트

- 8주차 : Window programming

2024. 5. 8(수).

우창우

Dr.woo@chungbuk.ac.kr

□ 주차별 강의계획

※ 4.10(국회의원선거날)

주차(변경 전)	주차(변경 후)	강의계획	과제	번호	제출 마감일
1주(03.06)	1주(03.06)	Python Started	VScode 설치화면, "Hello World" 출력화면 및 작성코드	1	03.06~03.09(토)
2주(03.13)	2주(03.13)	Variables and simple data types	다이아몬드 모양, 진수 변환 프로그램 및 동전 변환 프로그램의 출력화면/작성코드	2	03.13~03.16(토)
3주(03.20)	3주(03.20)	Control Statement	종합계산기, 구구단 출력 프로그램의 출력화면/작성코드	3	03.20~03.23(토)
4주(03.27)	4주(03.27)	Lists, Dictionaries, String	음식 궁합 출력, 문자열 거꾸로 출력 프로그램의 출력화면/작성코드	4	03.27~03.30(토)
5주(04.03)	5주(04.03)	Functions and module	GitHub Copilot Extension 설치화면, 로또번호 추첨 출력화면/작성코드	5	04.03~04.06(토)
6주(04.10)	6주(04.17)	Classes	팀프로젝트 과제 제출양식.hwp (조당 1개)	6	04.17~04.20(토)
7주(04.17)	7주(04.24)	Window programming	사진앨범 프로그램의 출력화면/작성코드	7	04.24~04.27(토)
8주(04.24)	8주(05.01)	Midterm (반영비율 20%)	-	-	-

□ 주차별 강의계획

※5.15(부처님오신날)

주차(변경 전)	주차(변경 후)	강의계획	과제	번호	제출 마감일
9주(05.01)	9주(05.08)	Files and Exceptions	사칙 연산문제지 10개 만들기 프로그램 출력화면/작성코드	8	05.08~05.11(토)
10주(05.08)	10주(05.22)	Database		9	05.22~05.25(토)
11주(05.15)	11주(05.29)	Data visualization		10	05.29~06.01(토)
12주(05.22)	12주(06.05)	Quiz (반영비율 10%), Team project implementation I	-	-	-
13주(05.29)	13주(06.17)	Team project implementation II	-	-	-
14주(06.05)	14주(06.18)	Team project implementation III	-	-	-
15주(06.12)	15주(06.20)	Project Presentation (반영비율 50%)	-	-	-

Chapter 11. 파일 입출력

Selection 01. 이장에서 만들 프로그램

Selection 02. 파일 입출력의 기본

Selection 03. 텍스트 파일 입출력

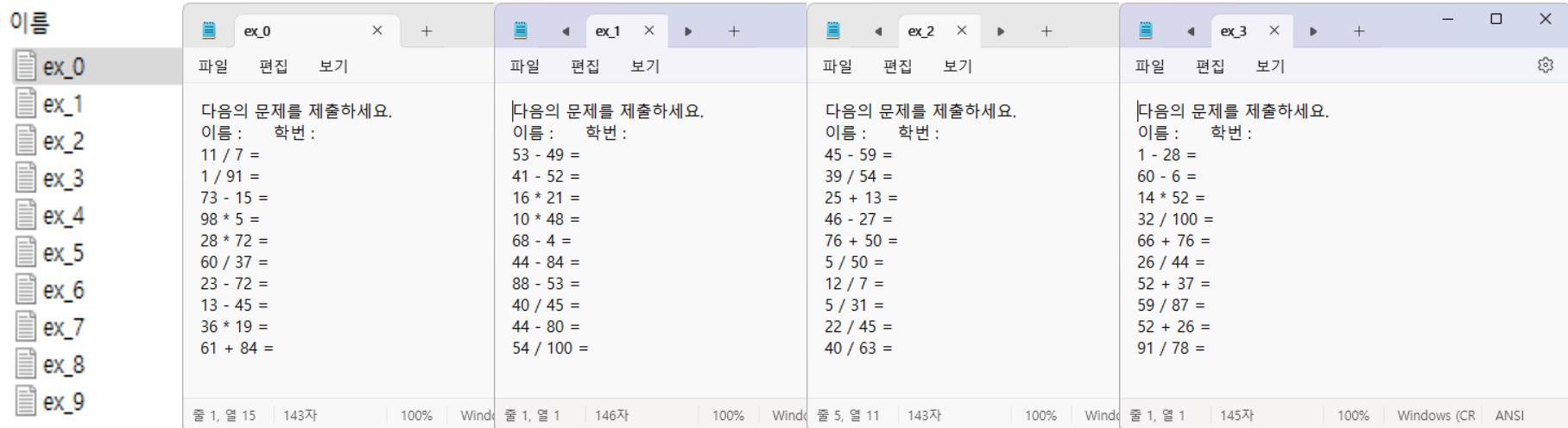
Selection 04. 이진 파일 입출력

Selection 05. 파일 입출력의 심화 내용

Selection 01. 이 장에서 만들 프로그램

1. 사칙 연산 문제지 10개 만들기

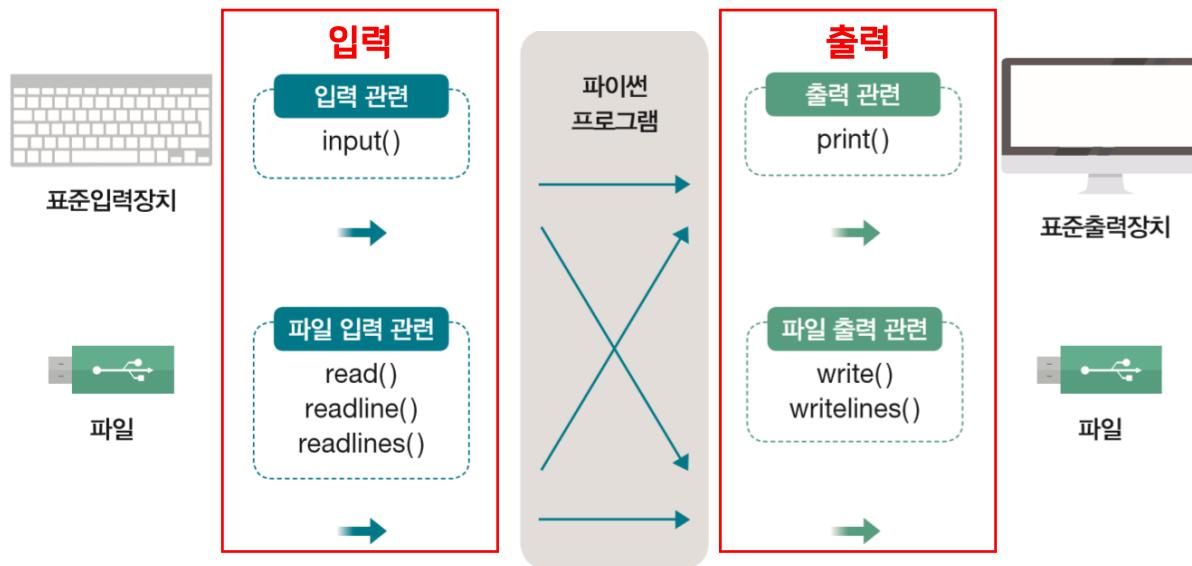
- 실행할 때마다 쓰기용 파일을 10개 만들고 각각 파일에 사칙 연산 문제를 랜덤하게 출제하는 프로그램



Selection 02. 파일 입출력의 기본

1. 파일 입출력의 개념

- 파일 입출력은 키보드에서 입력되는 것을 표준 입력, 출력되는 것을 표준 출력이라고 하며 키보드와 화면을 합쳐서 콘솔(Console)이라고 함



〈 표준 입출력과 파일 입출력 함수 〉

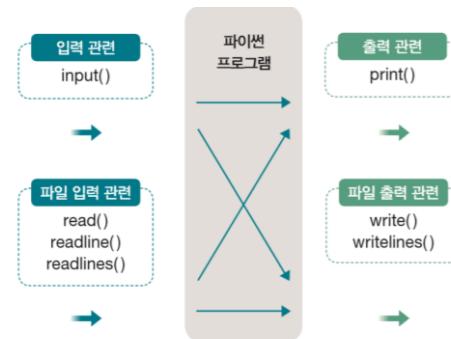
Selection 02. 파일 입출력의 기본

2. 파일 입출력의 기본 과정

- 파일 입출력을 하기 위해서는 표준 입출력과 달리 두 가지 작업을 추가로 해야함
- 파일을 사용하기 전의 '파일 열기' 작업과 파일 사용이 끝난 후의 '파일 닫기' 작업



읽기용 : 변수명 = open("파일명", "r")
쓰기용 : 변수명 = open("파일명", "w")



Selection 02. 파일 입출력의 기본

2. 파일 입출력의 기본 과정

- 파일을 입력할 때 사용하는 `open()` 함수의 마지막 매개변수는 '모드 (Mode)'라고 하며, 파일을 열 때 어떤 용도로 여는지 결정

읽기용 : 변수명 = `open("파일명", "r")`
쓰기용 : 변수명 = `open("파일명", "w")`

종류	설명
생략	r과 동일하다.
r	읽기 모드. 기본값이다.
w	쓰기 모드. 기존에 파일이 있으면 덮어쓴다.
r+	읽기/쓰기 겸용 모드이다.
a	쓰기 모드. 기존에 파일이 있으면 이어서 쓴다. append의 약어이다.
t	텍스트 모드. 텍스트 파일을 처리한다. 기본값이다.
b	이진 모드. 이진 파일을 처리한다.

- ✓ r : Read w : Write
- a : Append t : Text Mode
- b : Binary Mode

- ✓ 일반 텍스트 파일의 읽기 모드는 r 또는 rt를 사용하고, 이진 파일을 출력할 때는 wb를 사용

Selection 03. 텍스트 파일 입출력

1. 파일을 이용한 입력과 출력

- 파일을 입력할 때는 `read()`, `readline()`, `readlines()` 함수를 사용



- 출력 결과를 파일에 저장할 때는 `write()`나 `writelines()` 함수를 사용



Selection 03. 텍스트 파일 입출력

1. 파일을 이용한 입력과 출력

- `read()` : 파일의 내용을 문자열로 반환
- `readline()` : 파일의 한 줄을 읽어서 문자열로 반환
- `readlines()` : 파일의 모든 줄을 읽어서 리스트로 반환
- ✓ 전체 파일을 한 번에 처리해야 하는 경우 `read()`를 사용하고, 줄 단위로 처리해야 하는 경우 `readline()` 또는 `readlines()`를 사용

- `write()` : 주어진 하나의 문자열을 파일에 작성
- `writelines()` : 주어진 문자열 리스트를 파일에 작성
- ✓ `write()`를 사용할 때 줄바꿈 문자(`\n`)을 포함하면 여러 줄 입력이 가능하고, 문자열 리스트를 쓰려면 `writelines()`를 사용

Selection 03. 텍스트 파일 입출력

2. 한 행씩 읽어들이기

- (예시) 파일로 데이터를 입력한 후 이를 화면에 출력하는 코드

The image shows two side-by-side instances of the Visual Studio Code (VS Code) code editor. Both instances have the same file open: `Untitled-1.py`. The left instance contains the following code:

```
C: > Users > rryan > Downloads > Untitled-1.py > ...
1  inFp = None
2  inStr = ""
3
4  inFp = open("sample.txt", "r", encoding="UTF8")
5
6  inStr = inFp.readline()
7  print(inStr, end = "")
8
9  inStr = inFp.readline()
10 print(inStr, end = "")
11
12 inStr = inFp.readline()
13 print(inStr, end = "")
14
15 inFp.close()
```

The code between lines 6 and 13 is highlighted with a red rectangular box. The right instance contains the following code:

```
C: > Users > rryan > Downloads > Untitled-1.py > ...
1  inFp = None
2  inStr = ""
3
4  inFp = open("sample.txt", "r", encoding="UTF8")
5
6  while True :
7      inStr = inFp.readline()
8      if inStr == "" :
9          break
10     print(inStr, end = "")
11
12 inFp.close()
```

The code between lines 6 and 10 is highlighted with a red rectangular box. A red arrow points from the highlighted code in the left window to the corresponding code in the right window. To the right of the right window, the text "반복문으로 변경한 코드" (Changed code using a loop) is displayed.

Selection 03. 텍스트 파일 입출력

3. 한번에 모두 읽어들이기

- (예시) 파일로 데이터를 한번에 읽어들인 후 이를 출력하는 코드

The image shows two identical code editors side-by-side, displaying Python code for reading a file. Both editors have a dark theme and show the same code in the editor area.

```
C: > Users > rryan > Downloads > Untitled-1.py > ...
1  inFp = None
2  inList = ""
3
4  inFp = open("sample.txt", "r", encoding="UTF8")
5
6  inList = inFp.readlines()
7  print(inList)
8
9  inFp.close()
10
```

The code reads a file named "sample.txt" and prints its contents to the terminal. The terminal output for both instances is:

```
PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python
['cookbook 파일을 공부합니다. [1]\n', 'cookbook 파일을 공부합니다. [2] \n', 'cookbook 파일을 공부합니다. [3]
PS C:\Users\rryan>
```

The terminal output is highlighted with a red box in both code editors.

Selection 03. 텍스트 파일 입출력

4. 파일을 열 때 오류 처리

- 없는 파일명을 열려고 하면 오류가 발생하며 이때 오류가 발생하지 않게 하려면
'os.path.exists(파일명)' 형식을 사용하여 오류를 처리

```
File Edit Selection View Go Run ... ← →
Untitled-1.py ×
C: > Users > rryan > Downloads > Untitled-1.py > ...
1 inFp = None
2 fName, inList, inStr = "", [ ], ""
3
4 fName = input("파일명을 입력하세요 : ")
5 inFp = open(fName, "r")
6
7 inList = inFp.readlines()
8 for inStr in inList :
9     print(inStr, end = "")
10
11 inFp.close()
12
```

```
File Edit Selection View Go Run ... ← →
Untitled-1.py ×
C: > Users > rryan > Downloads > Untitled-1.py > ...
5
6 fName = input("파일명을 입력하세요 : ")
7
8 if os.path.exists(fName) :
9     inFp = open(fName, "r")
10
11 inList = inFp.readlines()
12 for inStr in inList :
13     print(inStr, end = "")
14
15 inFp.close()
16 else :
17     print("%s 파일이 없습니다" %fName)
18
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python
파일명을 입력하세요 : c:\\Windows\\win2.ini
Traceback (most recent call last):
  File "c:\\Users\\rryan\\Downloads\\Untitled-1.py", line 5, in <module>
    inFp = open(fName, "r")
    ~~~~~~
FileNotFoundException: [Errno 2] No such file or directory: 'c:\\Windows\\win2.ini'
PS C:\Users\rryan>
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python
파일명을 입력하세요 : c:\\Windows\\win2.ini
c:\\Windows\\win2.ini 파일이 없습니다
PS C:\Users\rryan>
```

Selection 03. 텍스트 파일 입출력

5. 한 행씩 파일에 쓰기

- (예시) 키보드에서 입력한 내용을 한 행씩 파일에 쓰는 코드

The screenshot shows a Python code editor interface with the following details:

- Code Editor Area:** Displays the script `Untitled-1.py`.

```
C: > Users > rryan > Downloads > Untitled-1.py > ...
1  outFp = None
2  outStr = ""
3
4  outFp = open("sample.txt", "w")
5
6  while True:
7      outStr = input("내용 입력 : ")
8      if outStr != "":
9          outFp.writelines(outStr + "\n")
10     else:
11         break
12
13 outFp.close()
14 print("--- 파일에 정상적으로 써졌음 ---")
15
```
- Terminal Area:** Shows the command line output of running the script.

```
PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python.exe c:/Users/rryan/Downloads/Untitled-1.py
내용 입력 : 파일을
내용 입력 : 배우고
내용 입력 : 있습니다.
내용 입력 :
--- 파일에 정상적으로 써졌음 ---
PS C:\Users\rryan>
```
- File Preview:** A separate window titled "sample" shows the contents of the file: "파이썬을 배우고 있습니다."
- Annotations:** A red box highlights the terminal output, and a red arrow points from the terminal output to the file preview window.

Selection 03. 텍스트 파일 입출력

6. 도스 명령어 type의 구현(유닉스/리눅스 계열의 cat 명령어)

- (예시) type 명령어의 기능처럼 지정한 파일의 내용을 화면에 출력하는 코드

The image shows two windows side-by-side. On the left is a Windows Command Prompt window titled 'C:\WINDOWS\system32\cmd'. It displays the output of the command 'type c:\windows\win.ini', which prints the contents of the 'win.ini' file. The output includes sections like [fonts], [extensions], [mci extensions], [files], [Mail], and MAPI=1. On the right is a screenshot of the Visual Studio Code (VS Code) interface. The left sidebar shows a file tree with 'Untitled-1.py'. The main editor area contains a Python script that reads the contents of a file specified by user input. The terminal tab at the bottom of VS Code shows the execution of this script and its output, which matches the output from the Command Prompt.

```
Microsoft Windows [Version 10.0.22631.3447]
(c) Microsoft Corporation. All rights reserved.

C:\Users\rryan>type c:\windows\win.ini
; for 16-bit app support
[fonts]
[extensions]
[mci extensions]
[files]
[Mail]
MAPI=1

C:\Users\rryan>
```

```
Untitled-1.py
C: > Users > rryan > Downloads > Untitled-1.py > ...
1  inFp = None
2  fName, inList, inStr = "", [ ], ""
3
4  fName = input("파일명을 입력하세요 : ")
5  inFp = open(fName, "r")
6
7  inList = inFp.readlines()
8  for inStr in inList :
9      print(inStr, end = "")
10
11 inFp.close()

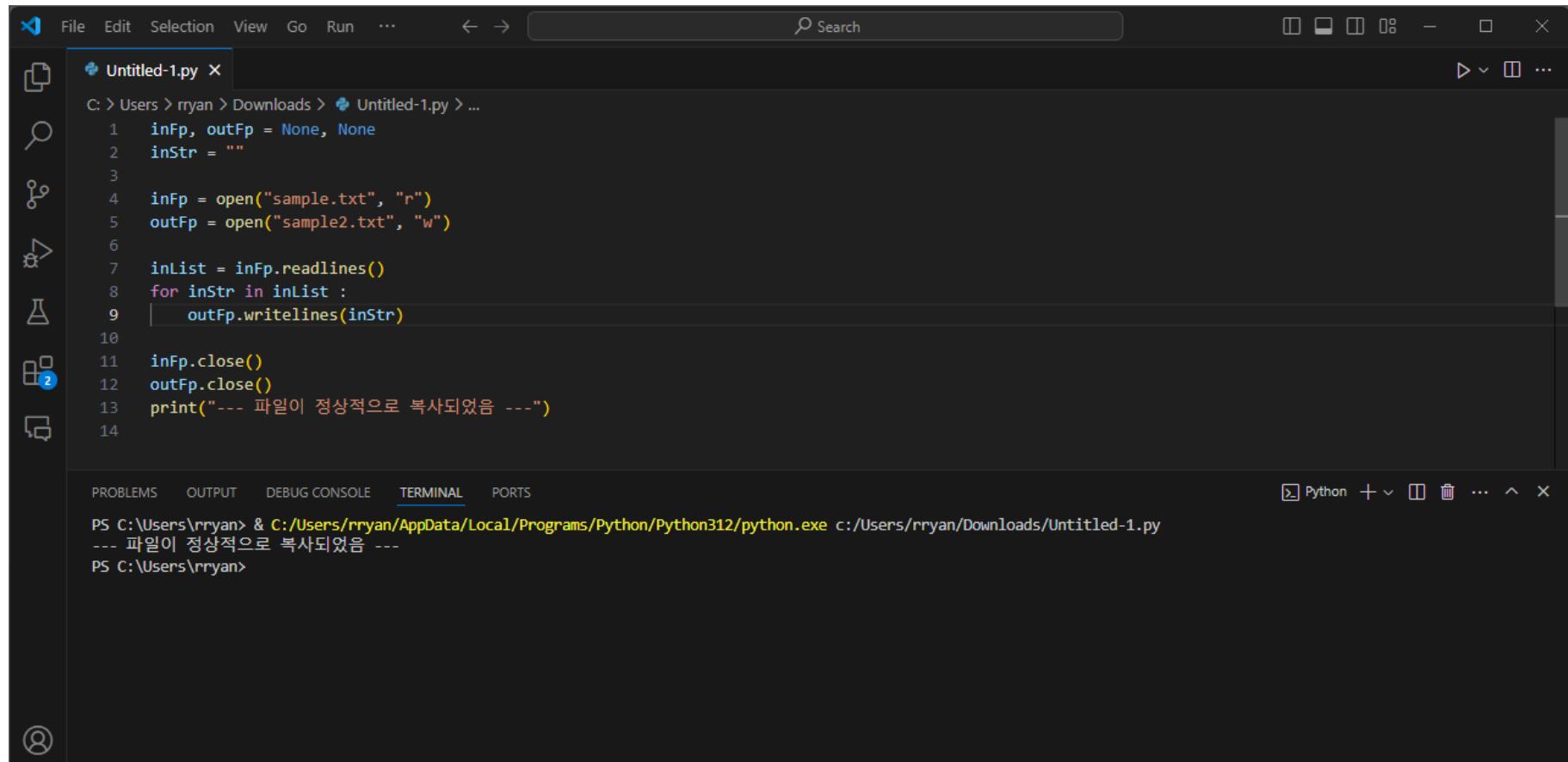
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/pyt
파일명을 입력하세요 : C:\Windows\win.ini
; for 16-bit app support
[fonts]
[extensions]
[mci extensions]
[files]
[Mail]
MAPI=1
PS C:\Users\rryan> ]
```

Selection 03. 텍스트 파일 입출력

7. 도스 명령어 copy의 구현(유닉스/리눅스 계열의 cp 명령어)

- (예시) copy 명령어의 기능처럼 지정한 파일의 내용을 복사하는 코드



The screenshot shows a dark-themed code editor interface. On the left is a sidebar with various icons: file, search, file list, navigation, and others. The main area displays a Python script named 'Untitled-1.py'. The code reads the contents of 'sample.txt' and writes them to 'sample2.txt'. The terminal tab at the bottom shows the command 'python Untitled-1.py' being run, followed by the message '--- 파일이 정상적으로 복사되었음 ---'.

```
C: > Users > rryan > Downloads > Untitled-1.py > ...
1  inFp, outFp = None, None
2  inStr = ""
3
4  inFp = open("sample.txt", "r")
5  outFp = open("sample2.txt", "w")
6
7  inList = inFp.readlines()
8  for inStr in inList :
9      |     outFp.writelines(inStr)
10
11 inFp.close()
12 outFp.close()
13 print("--- 파일이 정상적으로 복사되었음 ---")
14
```

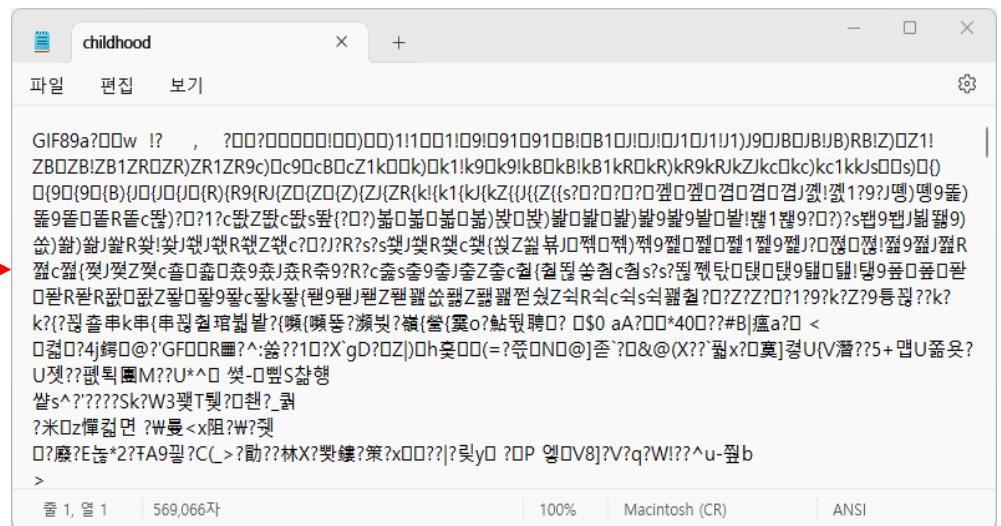
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python.exe c:/Users/rryan/Downloads/Untitled-1.py
--- 파일이 정상적으로 복사되었음 ---
PS C:\Users\rryan>

Selection 04. 이진 파일 입출력

1. 이진 파일의 개념

- 이진(Binary : 바이너리) 파일 : 글자가 아닌 비트(Bit) 단위로 의미가 있는 파일
 - 텍스트 파일을 제외한 나머지 파일로 그림 파일, 음악 파일, 동영상 파일 등
 - 텍스트 파일과 이진 파일을 구분하는 간단한 방법은, 파일을 메모장에서 열었을 때
글자처럼 보이면 텍스트 파일이고 이상하게 보이면 이진 파일에 해당



〈이미지 파일(이진파일)을 메모장에서 열었을 때〉

Selection 04. 이진 파일 입출력

2. 이진 파일의 복사

- (예시) 이진 파일을 복사해서 새로운 이진 파일로 만드는 코드

The screenshot shows a VS Code interface with a dark theme. On the left is a sidebar with icons for file operations like open, save, and search. The main area has a title bar with 'File', 'Edit', 'Selection', 'View', 'Go', 'Run', and a search bar. Below the title bar is a status bar showing the current directory: 'C: > Users > rryan > Downloads > Untitled-1.py > ...'. The code editor contains the following Python script:

```
C: > Users > rryan > Downloads > Untitled-1.py > ...
1 inFp, outFp = None, None
2 inStr = ""
3
4 inFp = open("childhood.gif", "rb")
5 outFp = open("childhood2.gif", "wb")
6
7 while True :
8     inStr = inFp.read(1)
9     if not inStr :
10         break
11     outFp.write(inStr)
12
13 inFp.close()
14 outFp.close()
15 print("--- 이진 파일이 정상적으로 복사되었음 ---")
16
```

Below the code editor are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is selected), and 'PORTS'. The terminal window shows the command 'python' followed by the path to the script, and the message '--- 이진 파일이 정상적으로 복사되었음 ---'.

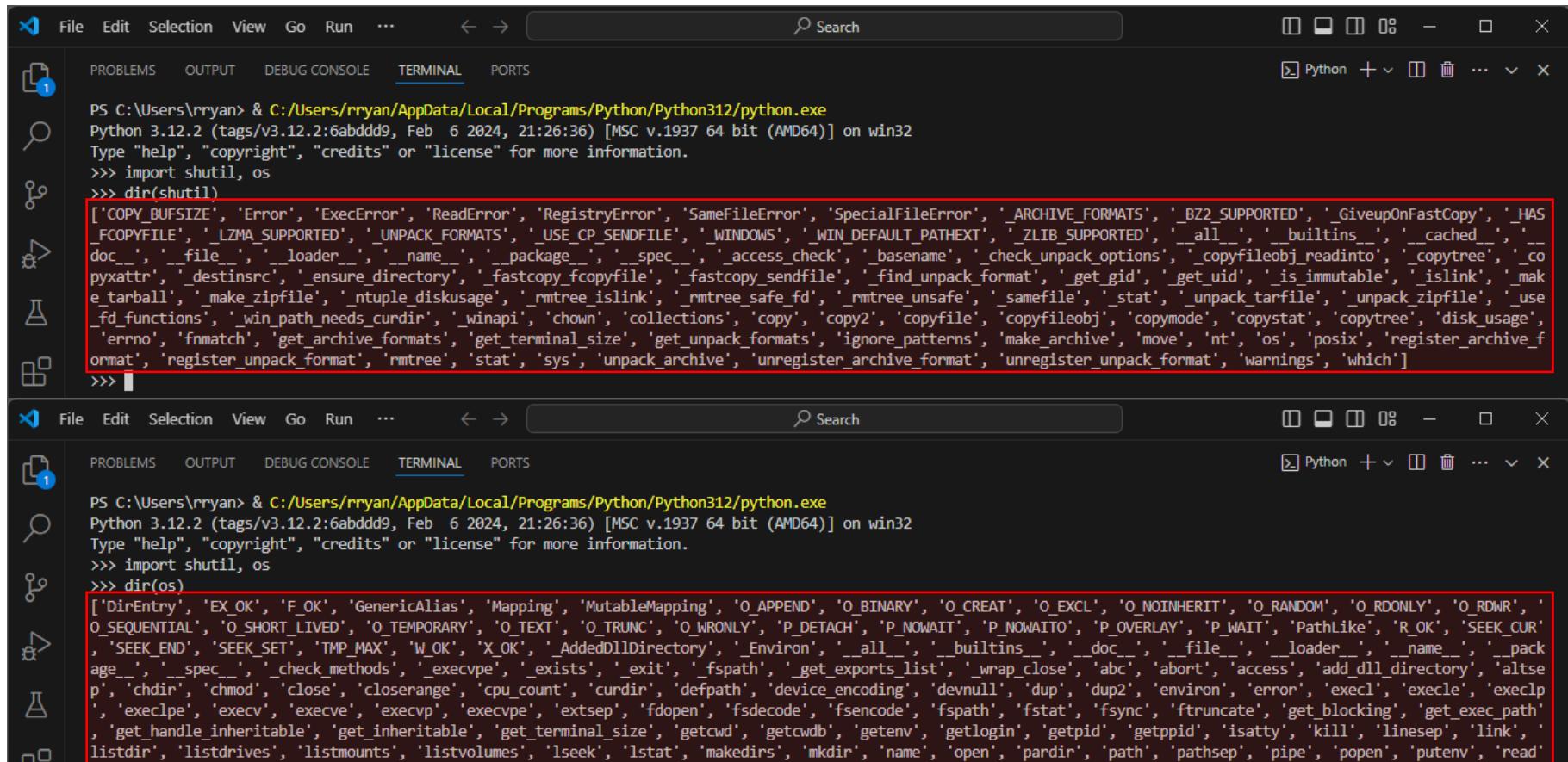
텍스트 파일을 복사하지 않기 때문에 encoding이 없고,
파일모드를 rb 및 wb로 설정하여 작성

종류	설명
생략	r과 동일하다.
r	읽기 모드. 기본값이다.
w	쓰기 모드. 기존에 파일이 있으면 덮어쓴다.
r+	읽기/쓰기 겸용 모드이다.
a	쓰기 모드. 기존에 파일이 있으면 이어서 쓴다. append의 약어이다.
t	텍스트 모드. 텍스트 파일을 처리한다. 기본값이다.
b	이진 모드. 이진 파일을 처리한다.

Selection 05. 파일 입출력의 심화 내용

1. 파일 및 디렉터리 다루기

- shutil 모듈과 os 모듈 등을 통해 파일과 디렉터리를 다룰 수 있는 다양한 함수를 제공



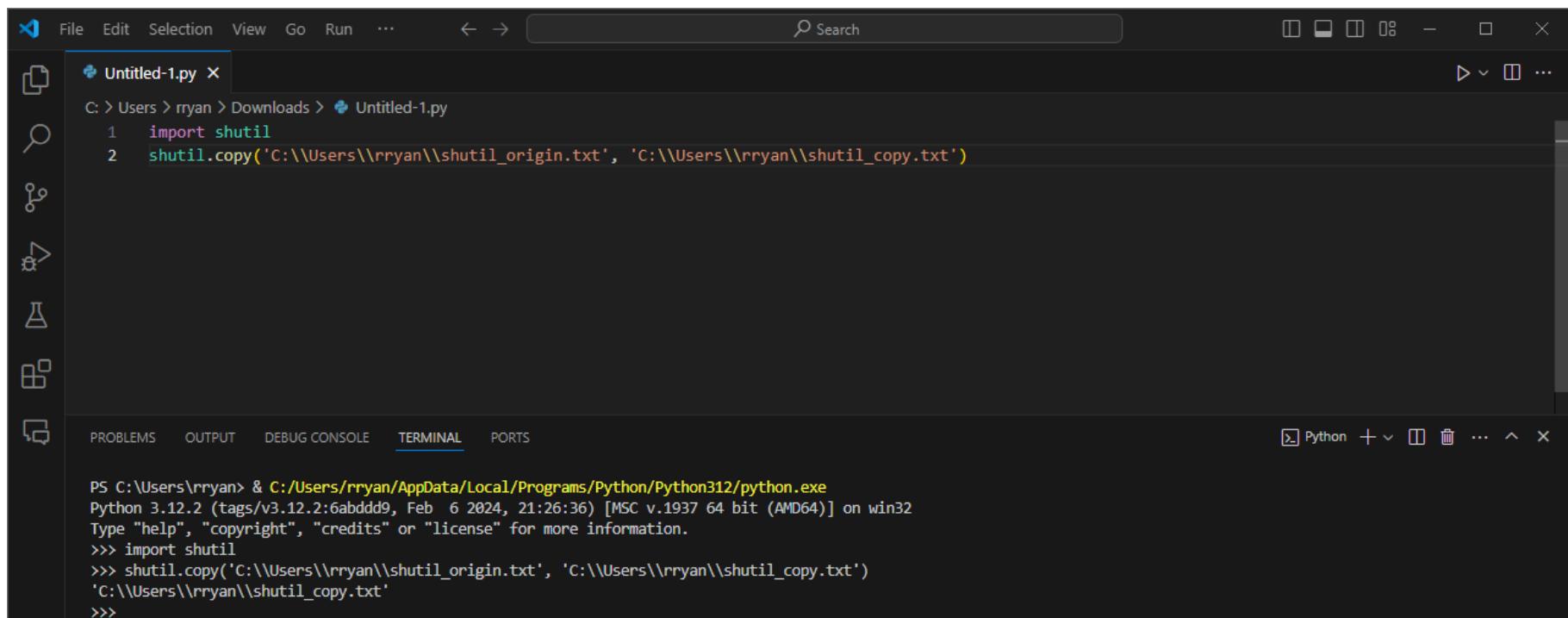
```
PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python.exe
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import shutil, os
>>> dir(shutil)
['COPY_BUFSIZE', 'Error', 'ExecError', 'ReadError', 'RegistryError', 'SameFileError', 'SpecialFileError', '_ARCHIVE_FORMATS', '_BZ2_SUPPORTED', '_GiveupOnFastCopy', '_HAS_FCOPYFILE', '_LZMA_SUPPORTED', '_UNPACK_FORMATS', '_USE_CP_SENDFILE', '_WINDOWS', '_WIN_DEFAULT_PATHEXT', '_ZLIB_SUPPORTED', '__all__', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', '__access_check', '__basename', '__check_unpack_options', '__copyfileobj_readinto', '__copytree', '__co_pyxattr', '__destinsrc', '__ensure_directory', '__fastcopy_fcopyfile', '__fastcopy_sendfile', '__find_unpack_format', '__get_gid', '__get_uid', '__is_imutable', '__islink', '__make_tarball', '__make_zipfile', '__ntuple_diskusage', '__rmtree_islink', '__rmtree_safe_fd', '__rmtree_unsafe', '__samefile', '__stat', '__unpack_tarfile', '__unpack_zipfile', '__use_fd_functions', '_win_path_needs_curdir', '_winapi', 'chown', 'collections', 'copy', 'copy2', 'copyfile', 'copyfileobj', 'copymode', 'copystat', 'copytree', 'disk_usage', 'errno', 'fnmatch', 'get_archive_formats', 'get_terminal_size', 'get_unpack_formats', 'ignore_patterns', 'make_archive', 'move', 'nt', 'os', 'posix', 'register_archive_f
ormat', 'register_unpack_format', 'rmtree', 'stat', 'sys', 'unpack_archive', 'unregister_archive_format', 'unregister_unpack_format', 'warnings', 'which']
>>>

PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python.exe
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import shutil, os
>>> dir(os)
['DirEntry', 'EX_OK', 'F_OK', 'GenericAlias', 'Mapping', 'MutableMapping', 'O_APPEND', 'O_BINARY', 'O_CREAT', 'O_EXCL', 'O_NOINHERIT', 'O_RANDOM', 'O_RDONLY', 'O_RDWR', 'O_SHORT_LIVED', 'O_TEMPORARY', 'O_TEXT', 'O_WRONLY', 'P_DETACH', 'P_NOWAIT', 'P_OVERLAY', 'P_WAIT', 'PathLike', 'R_OK', 'SEEK_CUR', 'SEEK_END', 'SEEK_SET', 'TMP_MAX', 'W_OK', 'X_OK', '_AddedDllDirectory', '_Environ', '__all__', '__builtins__', '__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', '__check_methods', '_execve', '_exists', '_exit', '_fspath', '_get_exports_list', '_wrap_close', 'abc', 'abort', 'access', 'add_dll_directory', 'altsep', 'chdir', 'chmod', 'close', 'closerange', 'cpu_count', 'curdir', 'defpath', 'device_encoding', 'devnull', 'dup', 'dup2', 'environ', 'erron', 'exec', 'execle', 'execvp', 'execpe', 'execv', 'execve', 'execvpe', 'execvpe', 'extsep', 'fdopen', 'fsdecode', 'fsencode', 'fspath', 'fstat', 'fsync', 'ftruncate', 'get_blocking', 'get_exec_path', 'get_handle_inheritable', 'get_inheritable', 'get_terminal_size', 'getcwd', 'getcwdb', 'getenv', 'getlogin', 'getpid', 'getppid', 'isatty', 'kill', 'linesep', 'link', 'listdir', 'listdrives', 'listmounts', 'listvolumes', 'lseek', 'lstat', 'makedirs', 'mkdir', 'name', 'open', 'pardir', 'path', 'pathsep', 'pipe', 'popen', 'putenv', 'read'
```

Selection 05. 파일 입출력의 심화 내용

1. 파일 및 디렉터리 다루기

- `shutil.copy`(소스파일, 타겟파일) 함수를 통해 파일 및 디렉터리를 복사할 수 있음
※ 복사할 때는 원본 파일이 있어야하고 복사하려는 폴더도 존재해야하며,
`shutil.copytree()` 함수를 통해 디렉터리를 통으로 복사 할 수도 있음



The screenshot shows a code editor interface with a dark theme. On the left, there's a sidebar with various icons for file operations like copy, search, and refresh. The main area displays a Python script named `Untitled-1.py`. The code is as follows:

```
C: > Users > rryan > Downloads > Untitled-1.py
1 import shutil
2 shutil.copy('C:\\\\Users\\\\rryan\\\\shutil_origin.txt', 'C:\\\\Users\\\\rryan\\\\shutil_copy.txt')
```

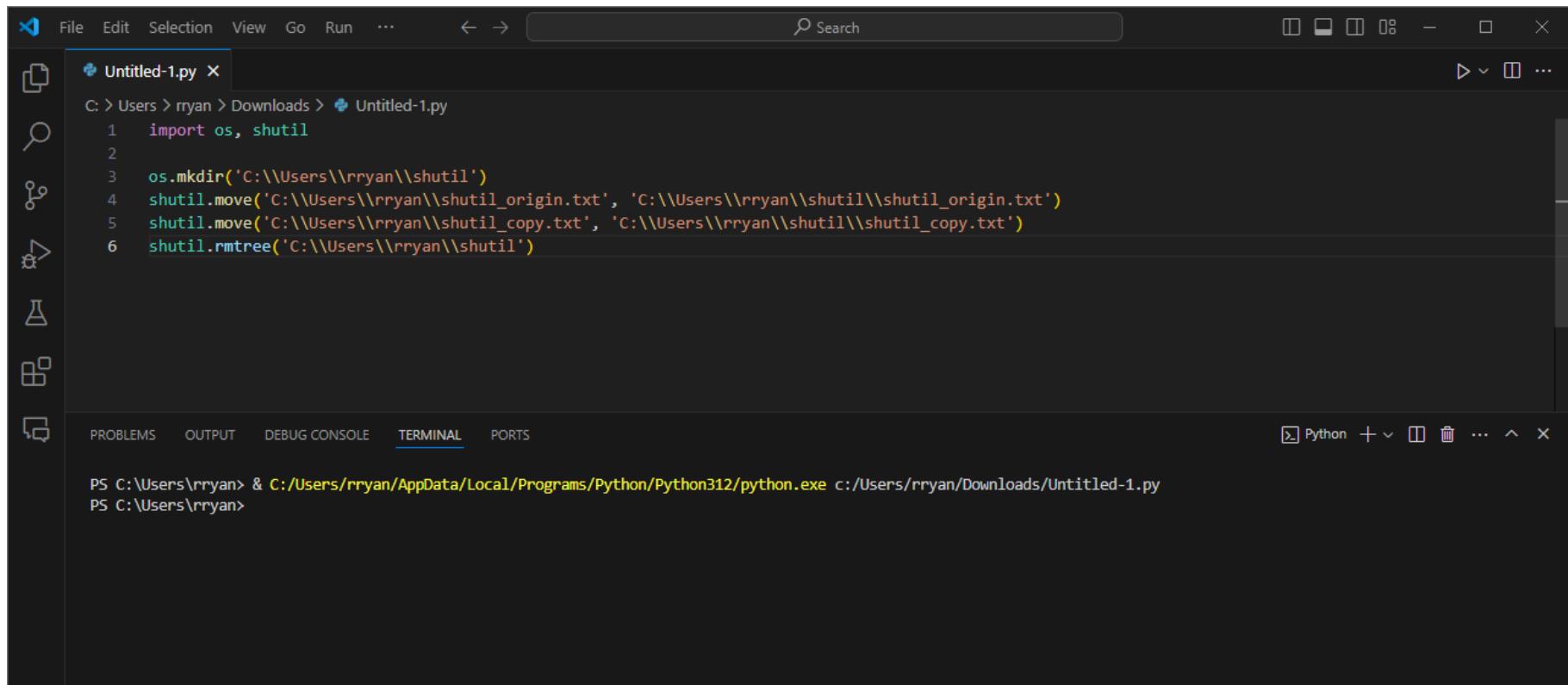
Below the code editor is a terminal window showing the execution of the script. The terminal output is:

```
PS C:\\\\Users\\\\rryan & C:\\\\Users\\\\rryan\\\\AppData\\\\Local\\\\Programs\\\\Python\\\\Python312\\\\python.exe
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import shutil
>>> shutil.copy('C:\\\\Users\\\\rryan\\\\shutil_origin.txt', 'C:\\\\Users\\\\rryan\\\\shutil_copy.txt')
'C:\\\\Users\\\\rryan\\\\shutil_copy.txt'
>>>
```

Selection 05. 파일 입출력의 심화 내용

1. 파일 및 디렉터리 다루기

- `os.mkdir`(폴더명) 과 `os.rmdir`(폴더명) 함수를 통해 디렉터리를 생성/삭제 할 수 있고,
디렉터리 내에 파일이 있는 경우 `shutil.rmtree`(폴더명) 함수를 통해 삭제 할 수 있음



The screenshot shows a Python code editor interface with the following details:

- File Explorer:** Shows a single file named "Untitled-1.py".
- Code Editor:** Displays the following Python code:

```
C: > Users > rryan > Downloads > Untitled-1.py
1 import os, shutil
2
3 os.mkdir('C:\\\\Users\\\\rryan\\\\shutil')
4 shutil.move('C:\\\\Users\\\\rryan\\\\shutil_origin.txt', 'C:\\\\Users\\\\rryan\\\\shutil\\\\shutil_origin.txt')
5 shutil.move('C:\\\\Users\\\\rryan\\\\shutil_copy.txt', 'C:\\\\Users\\\\rryan\\\\shutil\\\\shutil_copy.txt')
6 shutil.rmtree('C:\\\\Users\\\\rryan\\\\shutil')
```
- Terminal:** Shows the command being run in a terminal window:

```
PS C:\\\\Users\\\\rryan> & C:\\\\Users\\\\rryan\\\\AppData\\\\Local\\\\Programs\\\\Python\\\\Python312\\\\python.exe c:\\\\Users\\\\rryan\\\\Downloads\\\\Untitled-1.py
PS C:\\\\Users\\\\rryan>
```

Selection 05. 파일 입출력의 심화 내용

1. 파일 및 디렉터리 다루기

- `os.walk(폴더)` 함수는 디렉터리의 모든 파일 및 하위 폴더를 탐색하고, 이 함수의 반복을 통해 디렉터리의 목록을 조회 할 수 있음

The screenshot shows a Python code editor with a dark theme. A file named `Untitled-1.py` is open, containing the following code:

```
C: > Users > rryan > Downloads > Untitled-1.py > ...
1 import os
2
3 for dirname, subdirname, filenames in os.walk('C:\\Windows\\debug'):
4     for filename in filenames:
5         print(os.path.join(dirname, filename))
6
```

A red box highlights the line `for dirname, subdirname, filenames in os.walk('C:\\Windows\\debug'):`. Another red box highlights the line `print(os.path.join(dirname, filename))`. A callout box with a red border points from the highlighted code to the following list of parameters:

- `dirname`: 현재 폴더명
- `subdirname`: 하위 폴더 목록
- `filenames`: 파일명 목록

Below the code editor, a terminal window shows the execution of the script:

```
PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python.exe c:/Users/rryan/Downloads/Untitled-1.py
C:\Windows\debug\mrt.log
C:\Windows\debug\NetSetup (1).LOG
C:\Windows\debug\NetSetup.LOG
C:\Windows\debug\PASSWD.LOG
C:\Windows\debug\sammui.log
PS C:\Users\rryan>
```

Selection 05. 파일 입출력의 심화 내용

1. 파일 및 디렉터리 다루기

- os.path.exists() 함수는 파일 또는 폴더가 이미 존재하는지 확인할 때 사용하는 함수로, 존재하면 True, 그렇지 않으면 False를 반환하여 디렉터리의 목록을 조회 할 수 있음

The screenshot shows a Python code editor in VS Code with a dark theme. A file named 'Untitled-1.py' is open, containing the following code:

```
C: > Users > rryan > Downloads > Untitled-1.py
1 import os.path
2
3 os.path.exists('C:\\Windows\\debug\\PASSWD.LOG')
4 os.path.isfile('C:\\Windows\\debug\\PASSWD.LOG')
5 os.path.isdir('C:\\Windows\\debug')
```

Three lines of code are highlighted with red boxes: 'os.path.exists...', 'os.path.isfile...', and 'os.path.isdir...'. A red arrow points from these highlighted lines to a callout box on the right side of the screen. The callout box contains two items:

- ✓ os.path.isfile 함수는 파일이 존재하면 True
- ✓ os.path.isdir 함수는 폴더가 존재하면 True

Below the code editor, the terminal window shows the execution of the script:

```
PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python.exe
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import os.path
>>> os.path.exists('C:\\Windows\\debug\\PASSWD.LOG')
True
>>> os.path.isfile('C:\\Windows\\debug\\PASSWD.LOG')
True
>>> os.path.isdir('C:\\Windows\\debug')
True
>>>
```

Selection 05. 파일 입출력의 심화 내용

1. 파일 및 디렉터리 다루기

- os.remove() 함수는 파일을 삭제할 때 사용되는 함수로, 존재하지 않는 파일을 삭제하고자 하는 경우 기본적으로 FileNotFoundError가 발생

The image shows two side-by-side code editors. Both have dark themes and feature a sidebar with various icons.

Left Editor:

- File: Untitled-1.py
- Content:

```
C: > Users > rryan > Downloads > Untitled-1.py
1 import os
2
3 os.remove('sample_remove.txt')
```
- Terminal Output:

```
PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python
Traceback (most recent call last):
  File "c:\Users\rryan\Downloads\Untitled-1.py", line 3, in <module>
    os.remove('sample_remove.txt')
[Errno 2] 지정된 파일을 찾을 수 없습니다: 'sample_remove.txt'
PS C:\Users\rryan>
```

Right Editor:

- File: Untitled-1.py
- Content:

```
C: > Users > rryan > Downloads > Untitled-1.py
1 import os
2
3 if os.path.exists('sample_remove.txt'):
4     os.remove('sample_remove.txt')
5 else:
6     print('지정된 파일이 없습니다.')
```
- Terminal Output:

```
PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python
지정된 파일이 없습니다.
PS C:\Users\rryan>
```

Selection 05. 파일 입출력의 심화 내용

2. 예외처리 (try, except 문)

- 오류가 발생할 때 파일이 처리하지 않고 프로그래머가 작성한 코드를 실행하는 방식
- try, except 문을 활용하면 직접 오류 메시지를 작성하여 프로그램을 계속 실행하게 함

The image shows two side-by-side code editors, likely from VS Code, demonstrating Python file operations and exception handling.

Left Editor:

- File: Untitled-1.py
- Code:

```
C: > Users > rryan > Downloads > Untitled-1.py
1 import os
2
3 os.remove('C:\\\\Users\\\\rryan\\\\try_except_sample.txt')
4
```
- Terminal Output:

```
PS C:\\Users\\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python
Traceback (most recent call last):
  File "c:\\Users\\rryan\\Downloads\\Untitled-1.py", line 3, in <module>
    os.remove('C:\\\\Users\\\\rryan\\\\try_except_sample.txt')
FileNotFoundError: [WinError 2] 지정된 파일을 찾을 수 없습니다: 'C:\\\\Users\\\\rryan\\\\try_except_sample.txt'
```

Right Editor:

- File: Untitled-1.py
- Code:

```
C: > Users > rryan > Downloads > Untitled-1.py
1 import os
2
3 try :
4     os.remove('C:\\\\Users\\\\rryan\\\\try_except_sample.txt')
5 except :
6     print('파일이 없습니다.')
7
```
- Terminal Output:

```
PS C:\\Users\\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python
파일이 없습니다.
PS C:\\Users\\rryan>
```

Selection 05. 파일 입출력의 심화 내용

2. 예외처리 (try, except 문)

- (예시) 문자열 중에서 '파이썬' 글자의 위치를 모두 찾아서 출력하는 코드
- '파이썬' 글자가 위치한 첨자 0과 13을 찾은 후 더 이상 없기 때문에 오류가 발생

```
C: > Users > rryan > Downloads > Untitled-1.py > ...
1 myStr = '파이썬은 재미 있어요. 파이썬을 공부하고 싶어요.'
2 strPosList = []
3 index = 0
4
5 while True :
6     index = myStr.index('파이썬', index)
7     strPosList.append(index)
8     index = index + 1
9
10 print('파이썬 글자 위치 : ', strPosList)
11
```

```
C: > Users > rryan > Downloads > Untitled-1.py > ...
1 myStr = '파이썬은 재미 있어요. 파이썬을 공부하고 싶어요.'
2 strPosList = []
3 index = 0
4
5 while True :
6     try :
7         index = myStr.index('파이썬', index)
8         strPosList.append(index)
9         index = index + 1
10    except :
11        break
12
13 print('파이썬 글자 위치 : ', strPosList)
14
```

Selection 05. 파일 입출력의 심화 내용

2. 예외처리 (try, except 문)

- except 문 뒤에 아무것도 표기하지 않으면 모든 종류의 오류를 처리하고,
필요하다면 오류의 종류에 따라서 서로 다른 오류를 처리 할 수 있음

The screenshot shows a Python code editor interface. On the left, there's a sidebar with various icons. The main area contains the following code:

```
try :  
    실행할 문장들  
except 예외_종류 1 :  
    오류일 때 실행할 문장들  
except 예외_종류 2 :  
    오류일 때 실행할 문장들  
  
9  p = int('파이썬 글자 위치 : ', strPosList)  
10  
11
```

Below the code, there are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is active, showing a command prompt and a traceback:

```
PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python312/python.exe c:/Users/rryan/Downloads/Untitled-1.py  
Traceback (most recent call last):  
  File "c:/Users/rryan/Downloads/Untitled-1.py", line 6, in <module>  
    index = myStr.index('파이썬', index)  
           ^^^^^^  
ValueError: substring not found  
PS C:\Users\rryan>
```

A red arrow points from the error message in the terminal to a tooltip window on the right. The tooltip lists various Python exceptions with their descriptions. The `ValueError` entry is highlighted with a red border.

종류	설명
ImportError	import 문에서 오류가 발생할 때
IndexError	리스트 등 첨자의 범위를 벗어날 때
KeyError	딕셔너리에서 키가 없을 때
KeyboardInterrupt	프로그램 실행 중 [Ctrl]+[C]를 누를 때
NameError	변수명이 없는 것에 접근할 때
RecursionError	재귀 호출의 횟수가 시스템에서 설정한 것보다 넘칠 때(1000번)
RuntimeError	실행 도중 오류가 발생할 때
SyntaxError	exec()나 eval()에서 문법상 오류가 발생할 때
TypeError	변수형의 오류가 발생할 때 예 '문자열-문자열' 연산
ValueError	함수의 매개변수에 잘못된 값을 넘길 때 예 int('파이썬')
ZeroDivisionError	0으로 나눌 때
IOError	파일 처리 등 오류일 때

〈 대표적인 예외 종류 〉

Selection 05. 파일 입출력의 심화 내용

2. 예외처리 (try, except 문)

- (예시) 오류의 종류에 따라서 다른 처리를 진행하는 코드

The image shows two side-by-side instances of Microsoft Visual Studio Code (VS Code) running on a Windows operating system. Both instances have dark themes applied.

Left Window (Code Editor):

- Title Bar:** File, Edit, Selection, View, Go, Run, ...
- File Explorer:** Shows a file named "Untitled-1.py".
- Code Editor:** Displays the following Python script:

```
1 num1 = input('숫자1 : ')
2 num2 = input('숫자2 : ')
3
4 try :
5     num1 = int(num1)
6     num2 = int(num2)
7     while True :
8         res = num1 / num2
9         print(res)
10
11 except ValueError :
12     print('문자열은 숫자로 변환할 수 없습니다.')
13
14 except ZeroDivisionError :
15     print('0으로 나눌 수 없습니다.')
16
17 except KeyboardInterrupt :
18     print('Ctrl + C를 눌렀습니다.')
19
```

Right Window (Terminal):

- Title Bar:** File, Edit, Selection, View, Go, Run, ...
- Sidebar:** PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS.
- Terminal Output:** Shows the execution of the script in a PowerShell-like environment (PS). The user inputs "10" for num1 and "10" for num2. The script runs successfully, printing the result 10.0 multiple times. Two specific lines of output are highlighted with red boxes:
 - "문자열은 숫자로 변환할 수 없습니다." (Line 12)
 - "0으로 나눌 수 없습니다." (Line 15)

Selection 05. 파일 입출력의 심화 내용

2. 예외처리 (try, except, else, finally 문)

- try 문에서 예외가 발생할 수 있는 코드를 포함하고 예외가 발생하면 except 문이 실행
- else 문은 try에서 예외가 발생하지 않았을 때 실행되는 코드로 항상 사용할 필요는 없음
- finally 문은 try와 except 후에 항상 실행되어야 하는 코드를 포함하는 데 사용

The screenshot shows a code editor and a terminal window. The code in the editor is:

```
C: > Users > rryan > Downloads > Untitled-1.py > ...
1 num1 = input('숫자1 : ')
2 num2 = input(['숫자2 : '])
3
4 try :
5     num1 = int(num1)
6     num2 = int(num2)
7
8 except :
9     print('오류가 발생했습니다.')
10
11 else :
12     print(num1, '/', num2, '=', num1/num2)
13
14 finally :
15     print('이 부분은 무조건 나옵니다.')

PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python Untitled-1.py
숫자1 : 100
숫자2 : 50
100 / 50 = 2.0
이 부분은 무조건 나옵니다.

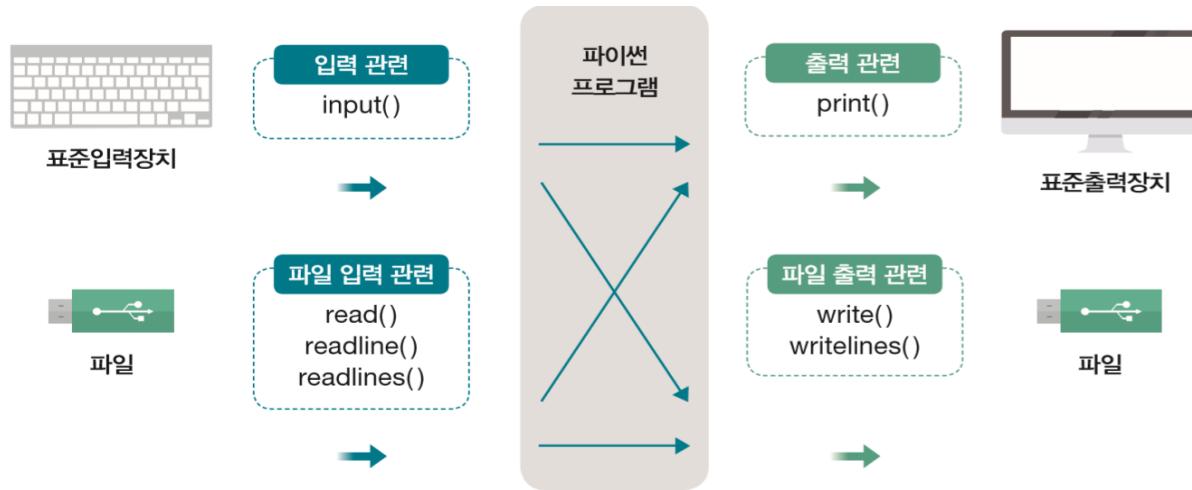
PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python Untitled-1.py
숫자1 : 100
숫자2 :
오류가 발생했습니다.
이 부분은 무조건 나옵니다.

PS C:\Users\rryan>
```

The code demonstrates exception handling. It first attempts to convert user input into integers. If an error occurs (e.g., non-integer input), it prints an error message. If no error occurs, it prints the division result. Finally, it prints a message that executes regardless of whether an error occurred or not.

Chapter 11. 요약

- ✓ 파일 입출력은 키보드를 사용하는 대신 파일을 읽어서 내용을 입력하거나 화면 대신 파일에 내용을 출력
- ✓ 표준 입출력과 파일 입출력 함수는 아래와 같음



- ✓ 파일 입출력은 ① 파일 열기 → ② 파일 처리 → ③ 파일 닫기 단계를 거침
- ✓ 파일의 내용을 한 행씩 읽어 오려면 `readline()` 함수를 사용하고, `readlines()` 함수를 사용하면 파일의 내용을 통째로 읽어서 리스트에 저장

Chapter 11. 요약

- ✓ 파일이 없을 때 오류를 방지하려면 `os.path.exists()` 함수를 사용하여 해당 경로에 파일이 없다는 메시지를 출력
- ✓ 텍스트 파일은 우리가 읽을 수 있는 글자로 구성된 파일이고 이를 제외한 그림 파일, 음악 파일, 동영상 파일 등은 이진(Binary) 파일로 글자가 아닌 비트(bit) 단위로 의미가 있음
- ✓ 이진 파일을 처리하려고 할 때는 `read()` 함수를 이용해서 한 바이트(byte)씩 읽고, `write()` 함수를 사용해서 한 바이트씩 쓰는 작업을 통해 처리
- ✓ 파이썬의 `shutil` 모듈과 `os` 모듈 등은 파일이나 디렉터리를 다룰 수 있는 다양한 함수를 제공
- ✓ 예외처리는 오류가 발생할 때 파이썬이 처리하지 않고 사용자가 작성한 코드를 실행하는 방식

감사합니다.

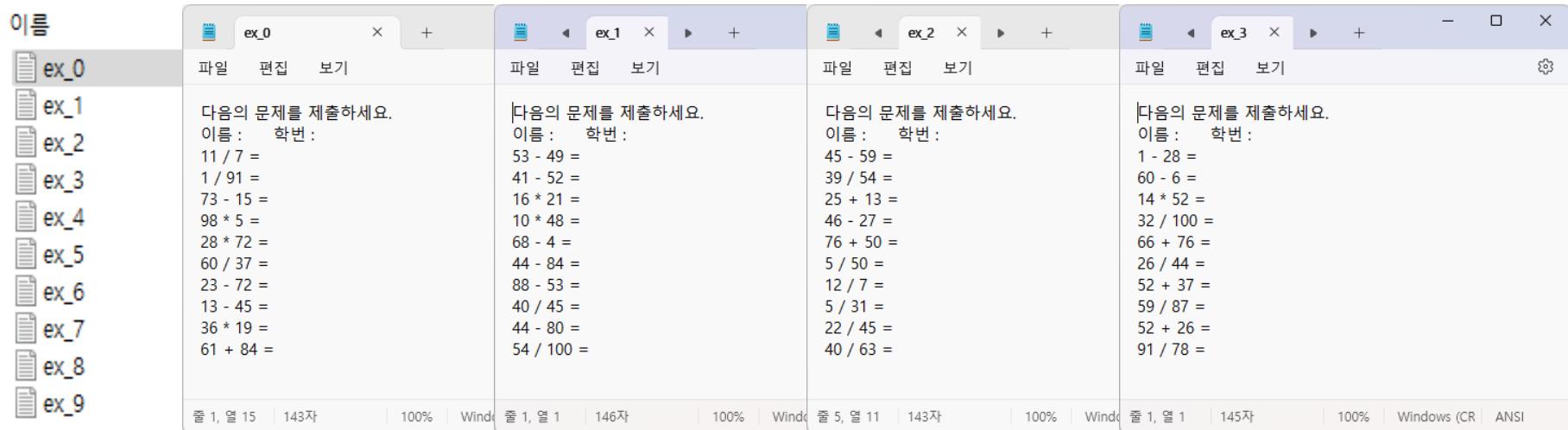
우창우

Dr.woo@chungbuk.ac.kr

실습&과제 : (제출처) Dr.woo@chungbuk.ac.kr, (기한) 5.11(토) 까지

1. 사칙 연산 문제지 10개 만들기

- 쓰기용 파일을 10개 만들고 각각 파일에 사칙 연산 문제를 랜덤하게 출제하는 프로그램



팀프로젝트 현황

조 번호	프로젝트 주제	조장	조원
1조	물리 엔진과 충돌 처리 기능, 그리고 캐릭터 디자인을 반영한 슈팅 게임 (슈퍼 마리오)	조형준	고태경, 김다민
2조	다중 미니게임 클라이언트를 위한 통합 아키텍처 개발	김민혁	전영우, 김정민
3조	팀프로젝트 수행능력 및 파이썬 프로그래밍 역량 강화를 위한 방탈출 게임 개발	홍성진	김태영, 정새연
4조	재학생의 완벽한 한 끼 보장을 위한 맛집 서비스 플랫폼 개발	이규민	우태현, 전수혁
5조	날씨에 따른 일정 관리를 위한 캘린더 앱	배정민	박상인, 서범교, 송설희
6조	교내 학생 간의 물품 중고거래를 위한 시스템 SW를 통한 웹사이트 제작	박조현	김건우, 오다영
7조	최근 인기 있는 음악의 키워드를 활용한 시기별 음악을 찾아주는 프로그램 개발	박주현	권정욱, 정현준
8조	보드게임 (like 클루)	김규현	김준후, 조윤정
9조	학생들을 위한 학습도우미 앱	신종환	신승우, 한강민
11조	학교 졸업사정 Q&A 챗봇	한준영	고태영, 이관학, 육광민
12조	학생의 수업의 참여율 감소 문제를 해결하기 위한 수업 참여도 인공지능 기술개발	윤시훈	전준석, 김민경
13조	AI와 데이터베이스를 활용해 복잡성을 최소화한 화장품 추천 웹사이트	김준호	황지연, 이용희
14조	게임개발 이해도 향상을 위한 고도엔진을 이용한 퍼즐게임 개발	배수환	이한결, 신혜원
15조	학생들의 경제적 부담 완화를 위한 택시 동승 서비스 개발	박성범	이태정, 김민석