

File I/O

Dr. Dongchul Kim

```
input ()
```

We have already seen the useful functions for grabbing input from a user:

```
input ()
```

Asks the user for a string of input, and returns the string

If you provide an argument, it will be used as a prompt.

CSCI3328 [~/PycharmProjects/CSCI3328] - .../example.py

File Edit View Navigate Code Refactor Run Tools VCS Window Help

CSCI3328 example.py example

example.py x

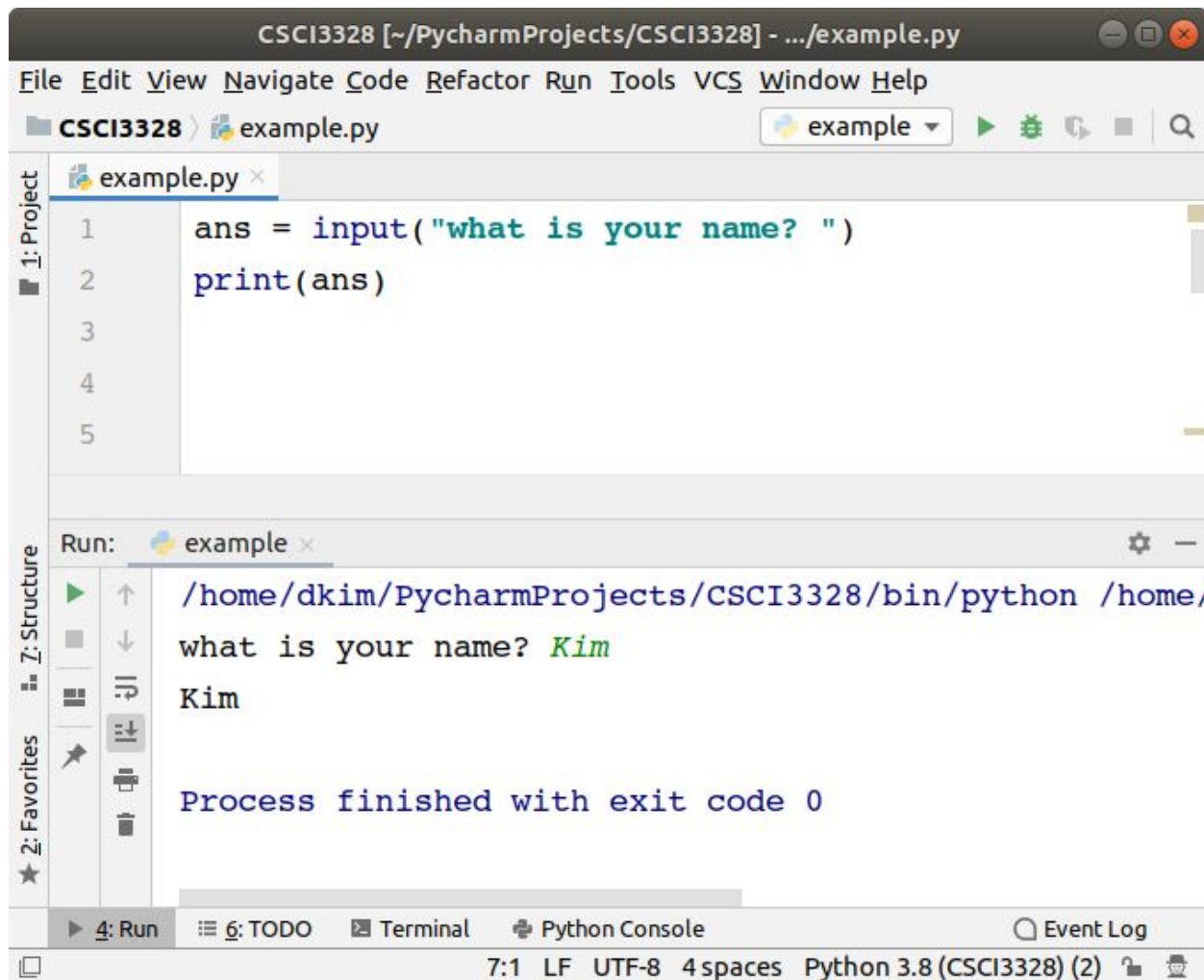
```
1 ans = input("what is your name? ")
2 print(ans)
3
4
5
```

Run: example x

```
/home/dkim/PycharmProjects/CSCI3328/bin/python /home/
what is your name? Kim
Kim
Process finished with exit code 0
```

4: Run | 6: TODO | Terminal | Python Console | Event Log

7:1 LF UTF-8 4 spaces Python 3.8 (CSCI3328) (2)



Lab 15-1

- Develop a program that displays a main menu and various sub-menus utilizing a while loop, if-else statements, and functions dedicated to menu display.
- The program should include a menu module comprising multiple functions. Each function displays a different type of menu, such as the main menu, utility menu, game menu, and multimedia menu. These functions will not accept any input arguments nor return values; their sole purpose is to present selectable options to the user.
- Upon starting, the program should import the menu module and display the main menu.
- Based on the user's selection from the main menu, the corresponding submenu should be displayed.
- The program should continue to display menus and sub-menus in a loop until the user chooses to exit or returns to the main menu from a submenu.
- Demonstrate the program's functionality with sample input and output scenarios (next slide)

```
Main Menu
1.Utility
2.Game
3.Multimedia
4.Exit
Please input: 1
```

```
Utility Menu
1.Calculator
2.Email
3.Note
4.Main menu
Please input: 1
```

I am sorry. It's not ready yet.

```
Utility Menu
1.Calculator
2.Email
3.Note
4.Main menu
Please input: 2
```

I am sorry. It's not ready yet.

```
Utility Menu
1.Calculator
2.Email
3.Note
4.Main menu
Please input: 4
```

```
Main Menu
1.Utility
2.Game
3.Multimedia
4.Exit
Please input: 2
```

```
Game Menu
1.Poker
2.Blackjack
3.Main menu
Please input: 1
```

I am sorry. It's not ready yet.

```
Game Menu
1.Poker
2.Blackjack
3.Main menu
Please input: 3
```

```
Main Menu
1.Utility
2.Game
3.Multimedia
4.Exit
Please input: 3
```

```
Multimedia Menu
1.Music player
2.Camera
3.Download Youtube
4.Main menu
Please input: 4
```

```
Main Menu
1.Utility
2.Game
3.Multimedia
4.Exit
Please input: 7
```

I am sorry. Please input correctly.

```
Main Menu
1.Utility
2.Game
3.Multimedia
4.Exit
Please input: 4
```

Thank you! Bye~

File Handling

The key function for working with files in Python is the `open()` function.

The `open()` function takes two parameters; filename, and mode.

There are four different methods (modes) for opening a file:

"r" - Read - Default value. Opens a file for reading, error if the file does not exist

"a" - Append - Opens a file for appending, creates the file if it does not exist

"w" - Write - Opens a file for writing, creates the file if it does not exist

"x" - Create - Creates the specified file, returns an error if the file exists

Open a file and Write on the file

The `open()` function returns a file object, which has a `write()` method for writing a content of the file.

For example,

```
f = open("test.txt", "a")  
  
f.write("Hello, Dr. Kim!")  
  
f.close()
```

```
1 f = open("test.txt", "a")
2 f.write("Hello, Dr. Kim!")
3 f.close()
4
5
6
7
8
9
```

```
▶ ↑ /home/dkim/CSCI3328/bin/python /home/dkim/PycharmProjects/CSCI3328/exam
```

```
◻ ↓ Process finished with exit code 0
```


Open a file and Reading

The `open()` function returns a file object, which has a `read()` method for reading the content of the file.

For example,

```
f = open("test.txt", "r")  
  
print(f.read())
```

Project	example.py	test.txt
1	<code>f = open("test.txt", "r")</code>	1 Hello, Dr. Kim! ✓
2	<code>print(f.read())</code>	2
3	<code>f.close()</code>	
4		
5		

Run: example ▾ ⚙️ -

7: Structure
2: Favorites

```
↑ /home/dkim/CSCI3328/bin/python /home/dkim/PycharmProjects/CSCI3328/example.py  
↓ Hello, Dr. Kim!  
⏪  
⏩  
🔄  
🗑️  
Process finished with exit code 0
```

Project	example.py ×	test.txt ×
1	<code>f = open("test.txt", "r")</code>	1 Hello, Dr. Kim!
2	<code>print(f.read())</code>	2 This is a test.
3	<code>f.close()</code>	3 Bye!
4		4
5		

Run: example ×

Structure
Favorites

```
▶ ↑ /home/dkim/CSCI3328/bin/python /home/dkim/PycharmProjects/CSCI3328/example.py  
■ ↓ Hello, Dr. Kim!  
☰ ↻ This is a test.  
✦ ↻ Bye!  
☰ ☑  
★ 4: Run  
☰ ☑ Process finished with exit code 0
```

Delete a file

To delete a file, you must import the `os` module, and run its `os.remove()` function.

For example,

```
import os

os.remove("test.txt") // error?
```

Check if File exist

To avoid getting an error, you might want to check if the file exists before you try to delete it.

For example,

```
import os

if os.path.exists("demofile.txt"):

    os.remove("demofile.txt")

else:

    print("The file does not exist")
```

Delete Folder

To delete an entire folder, use the `os.rmdir()` method.

For example,

```
import os

os.rmdir("myfolder")
```

```
readlines ()
```

`readlines()` reads until EOF using `readline()` and returns a list containing the lines.

If the optional `sizehint` argument is present, instead of reading up to EOF, whole lines totalling approximately `sizehint` bytes (possibly after rounding up to an internal buffer size) are read.

An empty string is returned only when EOF is encountered immediately.

readlines()

The screenshot shows the PyCharm IDE interface. The main editor window displays two files: `example.py` and `test.txt`. The `example.py` file contains the following Python code:

```
1 f = open("test.txt", "r")
2 lines = f.readlines()
3 print(lines)
4 print(lines[1])
5 f.close()
```

The `test.txt` file contains the following text:

```
1 Hello, Dr. Kim!
2 This is a test.
3 Bye!
```

The output of the script is shown in the Run console:

```
Run: example x
/home/dkim/CSCI3328/bin/python /home/dkim/PycharmProjects/CSCI3328/example.py
['Hello, Dr. Kim!\n', 'This is a test.\n', 'Bye!\n']
This is a test.

Process finished with exit code 0
```

The status bar at the bottom indicates that packages were installed successfully, including 'numpy' (today 5:30 PM). The system tray shows the time as 17:1, the file encoding as LF UTF-8, 4 spaces, and the Python version as Python 3.8 (CSCI3328).

readlines()

The screenshot shows the PyCharm IDE interface. The main editor window displays two files: `example.py` and `test.txt`. The `example.py` file contains the following Python code:

```
1 f = open("test.txt", "r")
2 lines = f.readlines(15)
3 print(lines)
4 f.close()
5
```

The `test.txt` file contains the following text:

```
1 Hello, Dr. Kim!
2 This is a test.
3 Bye!
4
```

The `Run` console at the bottom shows the execution of the script:

```
Run: example x
/home/dkim/CSCI3328/bin/python /home/dkim/PycharmProjects/CSCI3328/example.py
['Hello, Dr. Kim!\n']

Process finished with exit code 0
```

The status bar at the bottom indicates that packages were installed successfully, including 'numpy' (today 5:30 PM). The current settings are Python 3.8 (CSCI3328) with 7:1 LF UTF-8 4 spaces.

Lab 15-2: File-Based User Authentication System with Integrated Menu

Objective: Create a Python program that incorporates user authentication through file handling and integrates with the interactive menu system developed in Lab 15-1.

Task Description:

Initial Setup:

The program should feature a login/signup menu with two options: 'Login' and 'Signup'.

On selecting 'Signup', users should be prompted to create an ID and Password.

The program should save each pair of ID and Password in two separate files: id.txt and pass.txt, respectively.

User Authentication:

After signing up, the program redirects users back to the login/signup menu.

Users can then log in using their previously created ID and Password.

Implement a validation mechanism to check if the entered credentials match those stored in id.txt and pass.txt.

Menu Integration and Logout Feature:

Upon successful login, the program should display the main menu from Lab 15-1.

Modify the Lab 14-1 main menu to include a new option: "Option 4 - Log out".

Selecting "Log out" should bring the user back to the initial login/signup menu.

Program Termination:

Provide an option in the login/signup menu to exit the program.

|

Login/Signup Menu

- 1.Log in
- 2.Sign up
- 3.Exit

Please input: 2

Sign up

Please input an ID: dkim

Please input a password: 1234qwer

Login/Signup Menu

- 1.Log in
- 2.Sign up
- 3.Exit

Please input: 1

Log in

Please input your ID: dkim

Please input your password: abcd9876

ID or Password is incorrect!

Login/Signup Menu

- 1.Log in
- 2.Sign up
- 3.Exit

Please input: 1

Log in

Please input your ID: dkim

Please input your password: 1234qwer

Welcome back, dkim!

Main Menu

- 1.Utility
- 2.Game
- 3.Multimedia
- 4.Log out

Please input: 2

Game Menu

- 1.Poker
 - 2.Blackjack
 - 3.Main menu
- Please input: 3

Main Menu

- 1.Utility
- 2.Game
- 3.Multimedia
- 4.Log out

Please input: 4

You are logged out!

Login/Signup Menu

- 1.Log in
 - 2.Sign up
 - 3.Exit
- Please input: 3

Thank you! Bye~