

Fundamentals of Computer Programming

Building a Programming Portfolio

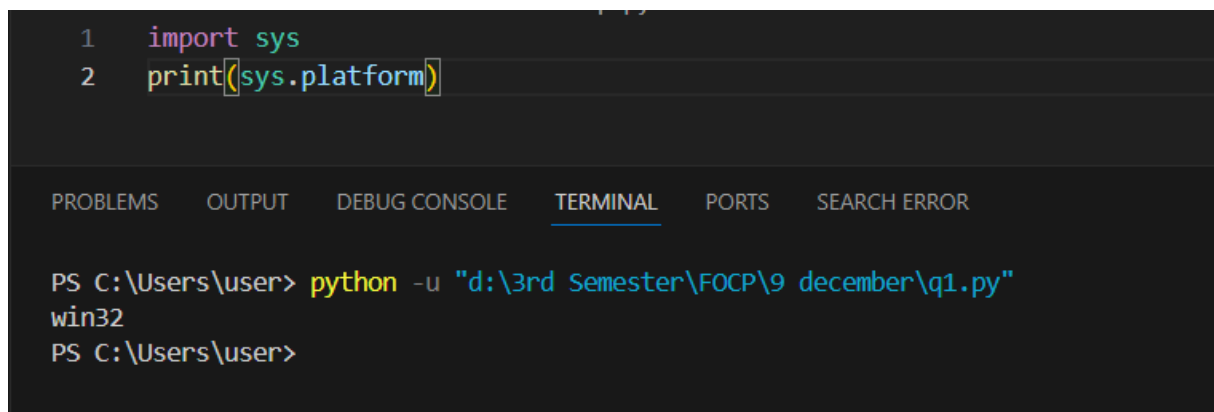
Week 5

You should be able to complete the following programs by the end of the week. By now you should understand why you should be saving your work to GitHub or similar. Possible solutions will be uploaded to the main module GitHub repository every week. If you follow that repo you should be able to receive notifications.

After this week you should be able to run your programs from the command-line, without having to use an IDE.

Note: Most of these programs process command-line arguments. In every case your program should not crash if no arguments are provided. In most cases it should just exit with some suitable error message.

1. Using command-line arguments involves the `sys` module. Review the docs for this module and using the information in there write a short program that when run from the command-line reports what operating system platform is being used.



```
1 import sys
2 print(sys.platform)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR

```
PS C:\Users\user> python -u "d:\3rd Semester\FOCP\9 december\q1.py"
win32
PS C:\Users\user>
```

2. Write a program that, when run from the command line, reports how many arguments were provided. (Remember that the program name itself is *not* an argument).

```
1 from sys import argv
2 print(len(argv)-1)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR

PS C:\Users\user> python -u "d:\3rd Semester\F0CP\9 december\q2.py"
0
PS C:\Users\user> python -u "d:\3rd Semester\F0CP\9 december\q2.py" ar ary arya ariannn
5
PS C:\Users\user> █
```

3. Write a program that takes a bunch of command-line arguments, and then prints out the shortest. If there is more than one of the shortest length, any will do.
Hint: Don't overthink this. A good way to find the shortest is just to sort them.

```
1 import sys
2 from sys import argv
3
4 list=argv[1:]
5 list.sort(key=len)
6 print(list)
7 try:
8     print(f"the shortest word is: {list[0]}")
9 except:
10    print("Word is not passed")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR

PS C:\Users\user> python -u "d:\3rd Semester\F0CP\9 december\q3.py"
[]
Word is not passed
PS C:\Users\user> python -u "d:\3rd Semester\F0CP\9 december\q3.py" apple a ae
['a', 'ae', 'apple']
the shortest word is: a
PS C:\Users\user> █
```

4. Write a program that takes a URL as a command-line argument and reports whether or not there is a working website at that address.
Hint: You need to get the HTTP response code.
Another Hint: StackOverflow is your friend.

```

1  import sys
2  from sys import argv
3  from urllib.request import urlopen
4  from urllib.error import *
5
6  try:
7      user_link=argv[1]
8      link=urlopen(user_link)
9
10 except IndexError as i:
11     print("No url provided and", i)
12
13 except HTTPError as h:
14     print("HTTP Error",h)
15
16 except URLError as h:
17     print("URL Error",h)
18
19 else:
20     print("Page found successfully!")

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR

```

PS C:\Users\user> python -u "d:\3rd Semester\FOCP\9 december\q4.py"
No url provided and list index out of range
PS C:\Users\user> python -u "d:\3rd Semester\FOCP\9 december\q4.py" https://youtube.com
Page found successfully!
PS C:\Users\user>

```

5. Last week you wrote a program that processed a collection of temperature readings entered by the user and displayed the maximum, minimum, and mean. Create a version of that program that takes the values from the command-line instead. Be sure to handle the case where no arguments are provided!

```

1  import sys
2
3  def process_temperatures(temperatures):
4      max_temp = max(temperatures)
5      min_temp = min(temperatures)
6      mean_temp = sum(temperatures) / len(temperatures)
7
8      print(f"Maximum Temperature: {max_temp}")
9      print(f"Minimum Temperature: {min_temp}")
10     print(f"Mean Temperature: {mean_temp:.2f}")
11
12  def main():
13     if len(sys.argv) < 2:
14         print("No temperature readings provided. Please provide values as command-line arguments.")
15         return
16
17     try:
18         temperatures = [float(arg) for arg in sys.argv[1:]]
19         process_temperatures(temperatures)
20     except ValueError:
21         print("Error: All temperature readings must be numeric values.")
22
23  if __name__ == "__main__":
24     main()

```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS SEARCH ERROR

```

PS C:\Users\user> python -u "d:\3rd Semester\FOCP\9 december\q5.py"
No temperature readings provided. Please provide values as command-line arguments.
PS C:\Users\user> python -u "d:\3rd Semester\FOCP\9 december\q5.py" 32 34 35 45 46 45
Maximum Temperature: 46.0
Minimum Temperature: 32.0
Mean Temperature: 39.50
PS C:\Users\user>

```

6. Write a program that takes the name of a file as a command-line argument, and creates a backup copy of that file. The backup should contain an exact copy of the contents of the original and should, obviously, have a different name. *Hint: By now, you should be getting the idea that there is a built-in way to do the heavy lifting here! Take a look at the "Brief Tour of the Standard Library" in the docs.*

```

1  from shutil import copyfile
2  from sys import argv
3  origin_filename=None
4  try:
5      origin_filename=argv[1]
6      filename,extension=origin_filename.split('.')
7      target_filename=(f"{filename}_backup.{extension}")
8
9      copyfile(origin_filename,target_filename)
10     print("file copied")
11
12 except IndexError as i:
13     print("file name not given", i)
14 except ValueError as v:
15     print("cannot find file extension", v)
16 except FileNotFoundError as f:
17     filename=origin_filename
18     print(f"Cannot open {filename}.",f)

```

The image shows a code editor interface with a file explorer on the left and a code editor on the right. The file explorer shows a directory named '9 DECEMBER' containing several files: doc_backup.txt, doc.txt, Exercises 03.pdf, Exercises 04.pdf, Program04.py, program05.py, ques6.py (selected), tempCodeRunnerFile.py, and week.py. The code editor shows the contents of 'ques6.py', which is a Python script that takes a filename as an argument and copies it to a backup file. The script includes error handling for IndexError, ValueError, and FileNotFoundError. Below the code editor, there is a terminal window showing the execution of the script. The terminal output shows that the script was run twice: first with no arguments, which resulted in an 'IndexError: list index out of range' error, and then with 'doc.txt' as an argument, which resulted in the message 'file copied'.

```
5 | origin_filename=argv[1]
6 | filename,extension=origin_filename.split('.')
7 | target_filename=(f"{filename}_backup.{extension}")
8 |
9 | copyfile(origin_filename,target_filename)
10 | print("file copied")
11 |
12 | except IndexError as i:
13 |     print("file name not given", i)
14 | except ValueError as v:
15 |     print("cannot find file extension", v)
16 | except FileNotFoundError as f:
17 |     filename=origin_filename
18 |     print(f"Cannot open {filename}.",f)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR

```
PS D:\3rd Semester\F0CP\9 december> python -u "d:\3rd Semester\F0CP\9 december\ques6.py"
file name not given list index out of range
PS D:\3rd Semester\F0CP\9 december> python -u "d:\3rd Semester\F0CP\9 december\ques6.py" doc.txt
file copied
PS D:\3rd Semester\F0CP\9 december>
```

Programming Portfolio 05 V1.0 2022-08-11 AMJ