

The dangers of cut & paste

Example: Let's look at a program which asks a user for two positive integers divisible by 5 and then computes the following formula

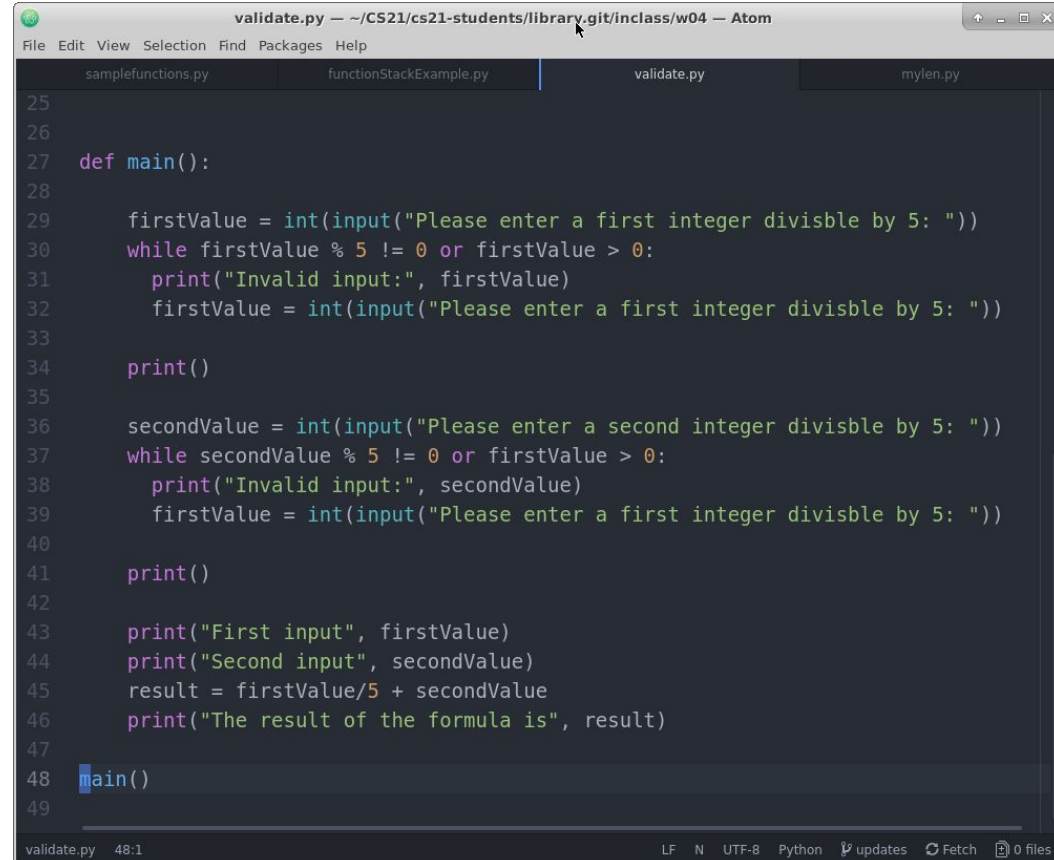
```
result = firstValue / 5 + secondValue
```

Approach 1: Using cut and paste

We cut and paste the code block that asks the user for input.

But it is full of bugs!!!!

Can you find them all?



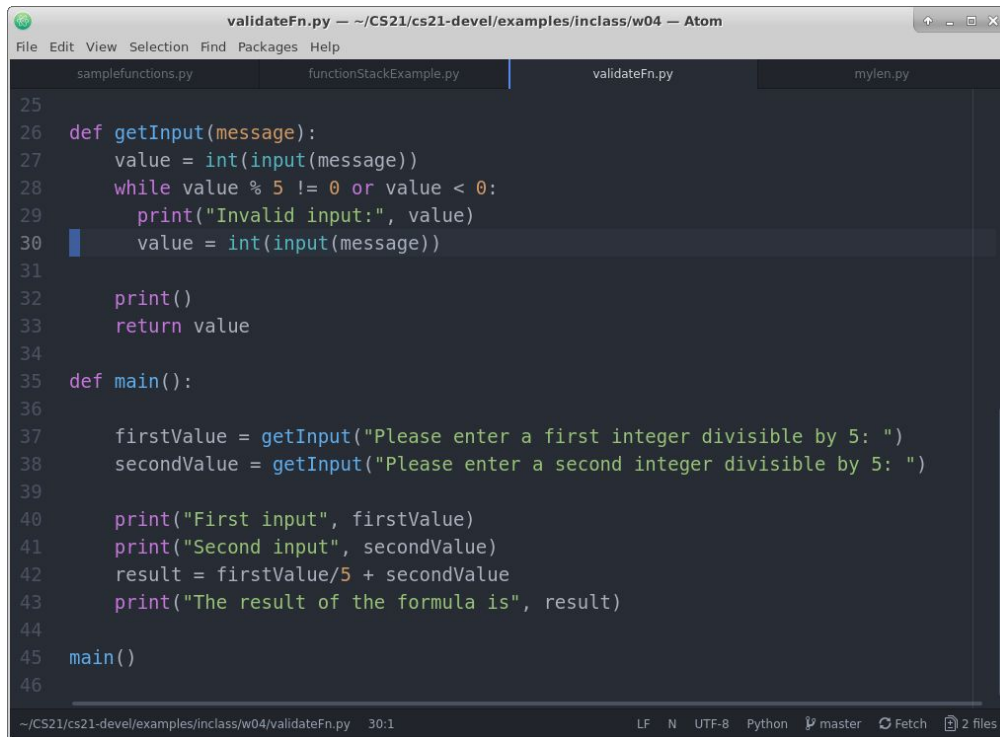
```
validate.py -- ~/CS21/cs21-students/library.git/inclass/w04 -- Atom
File Edit View Selection Find Packages Help
samplefunctions.py functionStackExample.py validate.py mylen.py
25
26
27 def main():
28
29     firstValue = int(input("Please enter a first integer divisble by 5: "))
30     while firstValue % 5 != 0 or firstValue > 0:
31         print("Invalid input:", firstValue)
32         firstValue = int(input("Please enter a first integer divisble by 5: "))
33
34     print()
35
36     secondValue = int(input("Please enter a second integer divisble by 5: "))
37     while secondValue % 5 != 0 or firstValue > 0:
38         print("Invalid input:", secondValue)
39         firstValue = int(input("Please enter a first integer divisble by 5: "))
40
41     print()
42
43     print("First input", firstValue)
44     print("Second input", secondValue)
45     result = firstValue/5 + secondValue
46     print("The result of the formula is", result)
47
48     main()
49
```

validate.py 48:1 LF N UTF-8 Python updates Fetch 0 files

Approach 2: Using a function

With a function:

- only need to fix bugs once
- abstracts the details so we don't need to worry about them in main()
- makes main() easier to read



```
validateFn.py — ~/CS21/cs21-devel/examples/inclass/w04 — Atom
File Edit View Selection Find Packages Help
samplefunctions.py  functionStackExample.py  validateFn.py  mylen.py
25
26 def getInput(message):
27     value = int(input(message))
28     while value % 5 != 0 or value < 0:
29         print("Invalid input:", value)
30         value = int(input(message))
31
32     print()
33     return value
34
35 def main():
36
37     firstValue = getInput("Please enter a first integer divisible by 5: ")
38     secondValue = getInput("Please enter a second integer divisible by 5: ")
39
40     print("First input", firstValue)
41     print("Second input", secondValue)
42     result = firstValue/5 + secondValue
43     print("The result of the formula is", result)
44
45     main()
46
~/CS21/cs21-devel/examples/inclass/w04/validateFn.py 30:1  LF N UTF-8 Python master Fetch 2 files
```

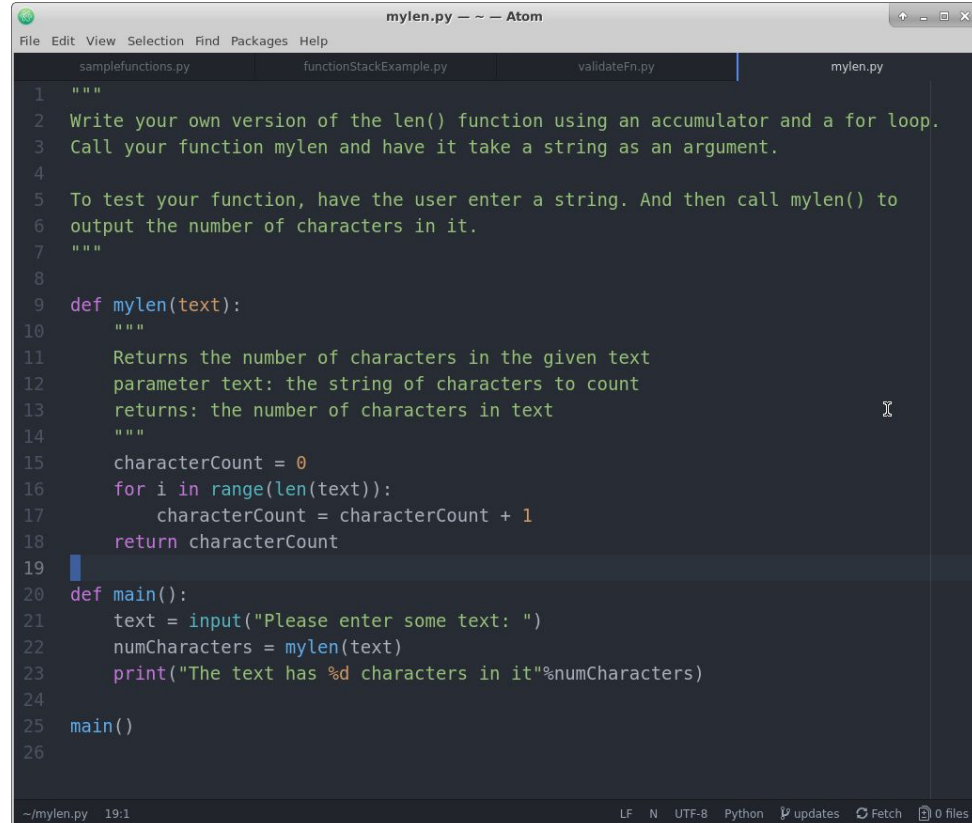
Exercise: Write your own len() function for strings

Analysis:

text = "test" # Input from user

characterCount = 0 # Initial Value

Iteration	i	characterCount = characterCount + 1
1	0	0 + 1 = 1
2	1	1 + 1 = 2
3	2	2 + 1 = 3
4	3	3 + 1 = 4



```
mylen.py --- Atom
File Edit View Selection Find Packages Help
samplefunctions.py functionStackExample.py validateFn.py mylen.py
1 """
2 Write your own version of the len() function using an accumulator and a for loop.
3 Call your function mylen and have it take a string as an argument.
4
5 To test your function, have the user enter a string. And then call mylen() to
6 output the number of characters in it.
7 """
8
9 def mylen(text):
10     """
11     Returns the number of characters in the given text
12     parameter text: the string of characters to count
13     returns: the number of characters in text
14     """
15     characterCount = 0
16     for i in range(len(text)):
17         characterCount = characterCount + 1
18     return characterCount
19
20 def main():
21     text = input("Please enter some text: ")
22     numCharacters = mylen(text)
23     print("The text has %d characters in it"%numCharacters)
24
25 main()
26
~/mylen.py 19:1 LF N UTF-8 Python updates Fetch 0 files
```

Stack

Terminology: A **stack** is a list where the last item added is the first to be removed

Analogies:

Pancake stack where we eat the top-most pancake first

Paper stack where we read the top-most paper first

Function stack

When we call a function, we create a new context. When the function completes, we return to the previous context.

Analogies:

Russian Matryoshka dolls

A dream within a dream (like Inception)



This process is called the **function stack**, or **call stack**, or **runtime stack**

Function scope

Local variables are variables that only exist within a function. When the function completes, these variables are destroyed (although their values may stick around).

Global variables are variables that can be seen by the entire program. Because global variables can cause subtle bugs, it is best practice to avoid global variables.

Scope refers to the lines of code where a variable exists. For example, the **scope** of a local variable in Python3 is the function it belongs to.

Heap

The **heap** contains all the values in our program

A variable lets us refer to raw data that is on the heap

When you say `value = 6`, the value 6 is created on the heap

The heap is like a common workspace for your program

Your program “checks out” values from the heap and then returns values when they aren’t needed anymore (like a library book)