SUMMER SCHOOL 2021

Introduction to Python Programming

Python Modules & Functions

Binjamin Barsch

Summary of commands used in this lecture

Command	Meaning
print()	**A Python function that prints text**

Up to now we have covered the basic principles of Python programming. The next few topics are supplementary topics that will help you program more efficiently and effectively.

We have been coding certain commands and functions in the Python interpreter. However, we haven't really been able to reuse the code. That is where this section comes in. We will focus on three ways that code is reused. The first is using functions, the second is using a Python file, and the third is a module. Method two and three are somewhat related.

Functions

A function is a block of code which only runs when it is called. You can pass it data, it performs certain tasks, and the returns a result. We have been using functions since we began this course. These are some of the functions you have already used:

- "`print()`"
- "`len()`"
- "`type()`"

Let us create our own function. If we were to add two numbers and print the result, we would do it in the following way:

```
>>>
>>> num1 = 40
>>> num2 = 2
>>> sum = num1 + num2
>>> print(sum)
42
>>>
```

But now if we want to do another sum with maybe different numbers we have to write out these 4 lines of code again. Whereas with a function we create a template of what these 4 lines do and makes it generic. Then to use it, we call the function with just need 1 line of code that contains the 2 numbers we want to add, and the function will return the result. How do we do this:

```
>>>
>>> def my_function_sum(number1, number2):
...  sum_var = number1 + number2
...  print(sum_var)
...
>>>
>>> my_function_sum(40,2)
42
>>>
```

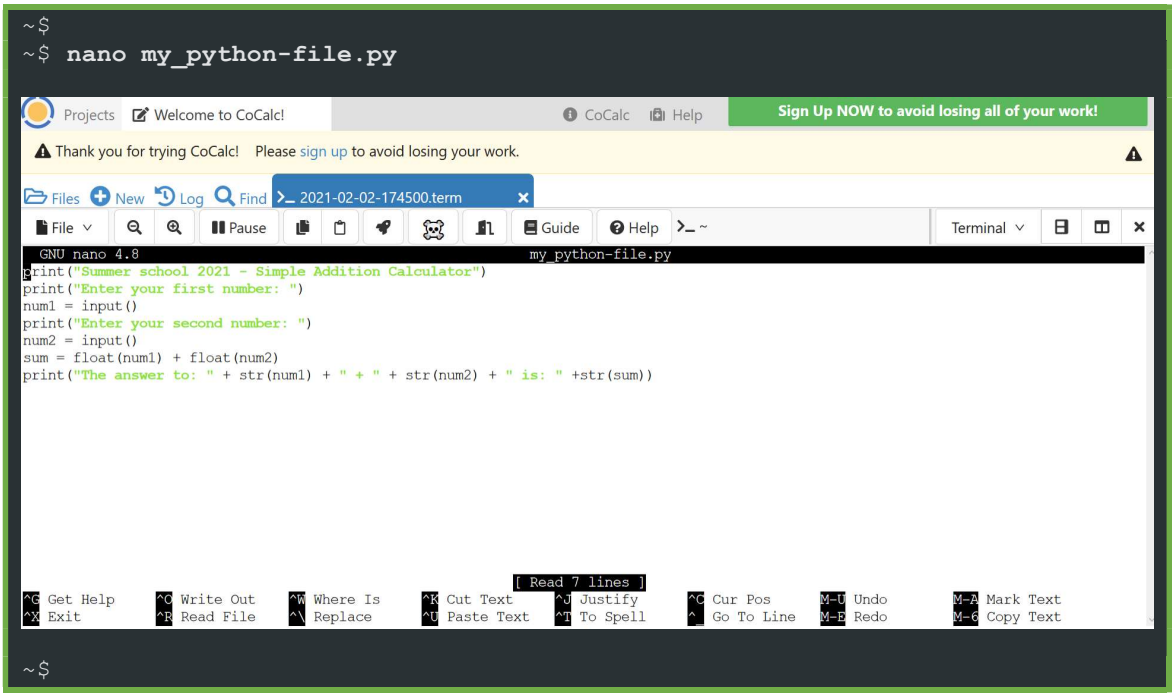
Clean, simple, and efficient!

If we have a lot more functions and code that we want to reuse we can put that a Python file. To do that we first create a Python file, then write some code in it and then we can run it in the terminal without having to go to the Python interpreter. See below for the steps:

```
~$ touch my_python-file.py
~$ nano my_python-file.py
```

Now let us do a more involved example. Let us create a simple addition calculator.

We first create a file and add the following code:



We can display the code below:

```
~$ cat my_python-file.py
print("Summer school 2021 - Simple Addition Calculator")
print("Enter your first number: ")
num1 = input()
```

```
print("Enter your second number: ")
num2 = input()
sum = float(num1) + float(num2)
print("The answer to: " + str(num1) + " + " + str(num2) + " is: "
+str(sum))
~$
```

Now we can run it in the Linux terminal:

```
~$
~$ python my_python-file.py
Summer school 2021 - Simple Addition Calculator
Enter your first number:
40
Enter your second number:
2
The answer to: 40 + 2 is: 42.0
~$
```

We can take this even further. What happens if we want to use a Python file that performs certain generic functions that we can use. This type of file call a module and a collection of modules we call libraries.

Let us first create our own module. To do this, we first create two files, one called "**my_sum_module.py**" and the other called "**my_main_program.py**"

```
~$ touch my_sum_module.py
~$ touch my_main_file.py~$
```

In the nano editor, add the following code for each file:

```
~$ nano my_sum_module.py

# This function adds two numbers together

def sum_func():
    print("Summer school 2021 - Simple Addition Calculator")
    print("Enter your first number: ")
    num1 = input()
    print("Enter your second number: ")
    num2 = input()
    sum = float(num1) + float(num2)
    print("The answer to: " + str(num1) + " + " + str(num2) + " is: "
+str(sum))

~$ nano my_main_file.py

import my_sum_module

print("Start Program")

my_sum_module.sum_func()
```

```
~$
```

Now in the Linux terminal enter the following command:

```
~$ python my_main_file.py
Start Program
Summer school 2021 - Simple Addition Calculator
Enter your first number:
10
Enter your second number:
2
The answer to: 10 + 2 is: 12.0
~$
```

So what happened was that Python ran the file called "**my_main_file.py**" and it first imported the "**my_sum_module**", it then printed out "**Start Program**" and then it executed the function "**sum_func**" that is within "**my_sum_module**". If you have understood this, then you have understood the basic architecture of most Python programs.

One new feature that we have added at the top of "**my_sum_module.py:** " is:

```
# This function adds two numbers together
```

Any Python code that starts with the "**#**" symbol will result in the text after that being a comment. As it has no programming effect, it is just there for information.

Python has many built in modules that you can also import and install from the internet. For example if you wanted to install the module called "**numpy**" you would enter the following in the Linux terminal.

```
~$ pip install numpy
```

An example of a built-in module is called "**platform**" and you can use it like this:

```
>>> import platform
>>> x = platform.system()
>>> print(x)
Linux
>>>
```

You can also give the module an alias with a shorter name like:

```
>>> import platform as plt
>>> x = plt.system()
>>> print(x)
Linux
```

Another useful module is the calendar module which has features, for example:

```
>>> import calendar
>>> yy = 2017
>>> mm = 11
>>>
>>> # display the calendar
>>> print(calendar.month(yy, mm))
   November 2017
Mo Tu We Th Fr Sa Su
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30
```

Practice Exercise:

To help you understand some of these concepts here is a practice exercise:

A Simple Calculator

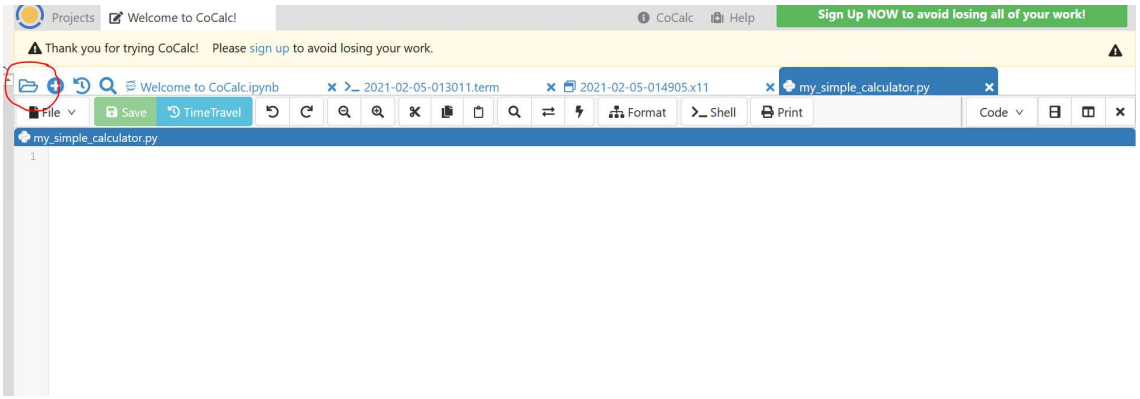
Let us create a calculator that asks the user to enter any two integer numbers and if they want to add, subtract, multiply or divide those two numbers. The calculator must then output the correct answer.

Now that you know about Python files, modules and functions you should always use them for any piece of code you want to do. Avoid using the Python interpreter for code that is going to be more than roughly 3 lines.

So, let us create a Python file called my_simple_calculator.py in the CoCalc Linux terminal:

```
~$
~$ touch my_simple_calculator.py
~$
```

And then open this file in the CoCalc text editor so we can code with proper features:



Copy the following code in the file that asks the user for their input:

```
# This file will ask the user for two integer numbers to either add,
subract, multiply or divide

print("Hello User!")

print("Please enter your first number: ")
num1 = input()
num1_int = int(num1)
# Use print for debugging
print(str(num1))


print("Please enter your second number: ")
num2 = input()
num2_int = int(num2)
print(str(num2))

print("Press 1. Add, 2. Subtract, 3. Multiply, 4. Divide: ")
option = input()
option_int = int(option)
print(str(option))
```

Let us see what happens if we run this in the Linux terminal now:

```
~$ python my_simple_calculator.py
Hello User!
Please enter your first number:
40
40
Please enter your second number:
2
2
Press 1. Add, 2. Subtract, 3. Multiply, 4. Divide:
```

```
3
3
~$
```

Now, we have to use a conditional statement to choose the appropriate option and print the answer. Copy the following code to you file directly under your existing code:

```
if option_int == 1:
    sum = num1_int + num2_int
elif option_int == 2:
    sum = num1_int - num2_int
elif option_int == 3:
    sum = num1_int * num2_int
elif option_int == 4:
    sum = num1_int / num2_int
else:
    print("invalid choice")

print("The answer is: " + str(sum))
```

Let us see what happens if we run this in the Linux terminal now:

```
~$ python my_simple_calculator.py
Hello User!
Please enter your first number:
40
40
Please enter your second number:
2
2
Press 1. Add, 2. Subtract, 3. Multiply, 4. Divide:
1
1
The answer is: 42
```

It works! Let us add one more feature. Let us put this code inside a function. All we do is that we call the function at the END of the file.

```
# This file will ask the user for two integer numbers to either add,
subract, multiply or divide

def simple_calc():
    print("Simple Calc Function")
    print("Hello User!")

    print("Please enter your first number: ")
    num1 = input()
    num1_int = int(num1)
    # Use print for debugging
    print(str(num1))

    print("Please enter your second number: ")
    num2 = input()
    num2_int = int(num2)
```

```
    print(str(num2))

    print("Press 1. Add, 2. Subtract, 3. Multiply, 4. Divide: ")
    option = input()
    option_int = int(option)
    print(str(option))

    if option_int == 1:
        sum = num1_int + num2_int
    elif option_int == 2:
        sum = num1_int - num2_int
    elif option_int == 3:
        sum = num1_int * num2_int
    elif option_int == 4:
        sum = num1_int / num2_int
    else:
        print("invalid choice")

    print("The answer is: " + str(sum))

simple_calc()
```

Challenge: What happens if the user chooses 0 for number 2 and the divide option ☹ - can you find a way to resolve that?

Summary

This was a really important lesson! We learnt about functions and modules. Mostly importantly the practice exercises showed you how you should structure your code. This is the end of lesson 6. On to the next lesson!