TMTplus Introduction to Scientific Programming

Mahdi Farnaghi, Mahdi Khodadadzadeh & Robert Ohuru

March 2021

Note from the teaching staff

The materials that we use for this course are somewhat new and have been heavily reworked. We will likely have made mistakes in this work or have glossed over issues that deserved better or different treatment. Where you find issues worth noting please do report these to us as it will allow to repair and improve.

By the way, welcome to the Exercise book for Introduction to Scientific Programming! We believe this book is self-explanatory, so go ahead and practice.

Chapter 2

Variables, Statements & Expressions

The purpose of this exercise is to kick you off on the programming environment, on some simple scripts, and on getting a bit of output.

2.1 Introduction

The main goal of each exercise is to use the Python programming language to review concepts you have learned during the lectures. You will also be given code or code snippets to study and understand.

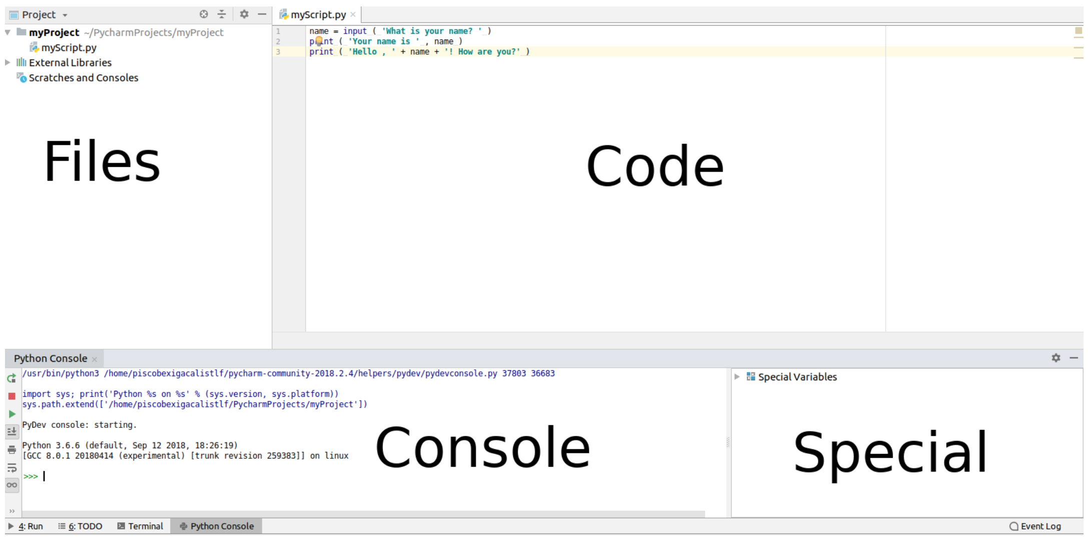
In this exercise in particular, you will work with:

- variables,
- statements,
- expressions.

Using PyCharm

Let us first discuss how the script development environment PyCharm can be used for program development. As discussed in class, you should already have installed it using the ITC Software Manager, package Python. If not, please consider going through the last chapter of this document which describes the installation steps for PyCharm.

The first time you run PyCharm, you will need to set a number of things just for once. Important is to set the default Python interpreter to Python 3.8. A looped video that demonstrates this is on canvas, demo video PyCharm. Open this with a viewer that can handle animated gif.



```
name = input ( 'What is your name? ' )
print ( 'Your name is ' , name )
print ( 'Hello , ' + name + '! How are you?' )
```

```
/usr/bin/python3 /home/piscobexigacalistlf/pycharm-community-2018.2.4/helpers/pydev/pydevconsole.py 37803 36683

import sys; print('Python %s on %s' % (sys.version, sys.platform))
sys.path.extend(['/home/piscobexigacalistlf/PycharmProjects/myProject'])

PyDev console: starting.

Python 3.6.6 (default, Sep 12 2018, 18:26:19)
[GCC 8.0.1 20180414 (experimental) [trunk revision 259383]] on linux

>>> 
```

Figure 2.1: The PyCharm user interface

Next, start up PyCharm and take a quick look around. We refer to Figure 2.1. You should initially see four panes for this application. We describe them here briefly:

files At top left, there is the *project file management pane*, through which you can navigate through the file system and find files relevant to your project.

console Bottom left is the *Python console* pane, but this same serves other functions as well: it can be used as Terminal window to the operating system, as manager of your TODO list, and as a pane that the interpreter uses to report.

The Python console function is used to directly execute usually simple Python commands. You should see the Python prompt »> waiting for your command. Try it out and type 2 + 2 to it, followed by a return. (Or some more advanced expression, if you are daring . . .)

The Terminal function is an operating system terminal that can be useful, as it allows to enter direct commands. At a later stage, we will run programs such as pip to install external libraries directly from here.

The Run function is where the interpreter shows results from running your scripts.

code At top tight, mostly the screen centerpiece is your coding window. This is where you develop code to work. This pane normally shows the contents of your .py file.

special This pane is directly connected to the Python console, provides useful information that is under the hood of your current project. It includes variables and objects that you created in the Python console and gives you access to them if the need arises.

In this exercise, you will mostly work in **code** and **console** panes. We advise you to adopt a consistent work attitude of saving files, keeping versions, all in one place. We also want to suggest that you have a paper lab logbook in which you write up your findings and answers to exercises, where this is not code. Obviously, you may prefer using a digital logbook over a paper-based logbook.

1.2 User input and program output

Using PyCharm, create a new file and type into it the following text (do not copy and paste this text because pdf documents like these use different character encodings, which a.o. do not carry over quote characters):

Ex 2.1

```
name = input("What␣is␣your␣name?␣")
print ("Your␣name␣is" , name)
print ("Hello,␣" + name + "!␣How␣are␣you?")
```

By convention, Python program/script files are given names that end in the file extension `.py`. You should probably save the Python script as `greetings.py`

Once you are happy with entering the script above, find out how you can run a script in PyCharm, and run it. Ensure you know how to navigate the menus for a script run, but also take note of the keystroke to achieve this.

Ex 2.2

Explain, code line by code line, what the `greetings.py` program does.

Ex 2.3

Try to answer the following questions. You can use the online Python documentation to verify your answer:

- What kind of coding thing is `input()`?
- What data type does `input()` return?
- What kind of statement is the first statement of `greetings.py`?
- What kind of coding thing is `name`? Is `name` a valid name for a variable?

You can test the effect of input() by using it in interactive mode with the Python command line that PyCharm also offers. Try out the following statements, each time providing a different input.

Ex 2.4

```
>>> name = input ("What␣is␣your␣name?␣")
>>> name
>>> type(name)
```

- What happens if you enter nothing?
- What happens if you enter a number?
- What happens if you enter two words?

Which further observations can you make?

Ex 2.5

- What does a print statement without parameter do?
- What does a print statement with one object (a number, a string, or a numeric expression) do?

Ex 2.6

Furthermore, you may discover these things by making print statements in interactive mode. For instance, like these:

```
>>> x = 'hello'
>>> x
>>> print (x)
>>> print (x, x)
```

Use of parameter sep in print() allows for a separator between text parts. Try the following code and understand what it does.

```
>>> print (x, x, sep='+')

>>> print (x, x, sep=',')
```

As a third way to understand print(), try the following statement:

Ex 2.7

```
>>> help(print)
```

What does this tell you? Does this also work if you want to learn more about the print statement?

1.3 More about strings

Let us find out a bit more how strings operate. Execute the following statements and observe their effect.

Ex 2.8

Note: The character sequence \n is common in programming languages. It represents a newline character that can be inserted in text. Likewise, \t is a tab character.

```
>>> print ('"Hello␣World!"')

>>> print ('Hello\nWorld!')

>>> print ('Hello' + "World!")

>>> print ('Hello', 'World!')

>>> print ('Hello' + '␣␣' + 'World!')

>>> print ('Hello' ,'World!', sep='␣␣')

>>> print ('He' + 2 * 'l' + 'o␣Wo' + 'rld!')

>>> print (3 * 'Hello,␣' + 'World!')
```

What exactly does the * operator achieve?

Ex 2.9

Execute the following statements and observe their effect:

```
>>> print('Hello','world',end='.')

>>> print('Hello','world',end=',')

>>> print('Hello','world',end='Finish')

>>> print ('Hello','world',end='\n')

>>> print ('Hello','world',end='\n\n')
```

What is the use of end?

1.4 Numerical operators

Ex 2.10

Execute the following statements in interactive mode using the Python command line.

```
>>> a = 3 - 4 + 10
>>> b = 5 * 6

>>> c = 7.0 // 8.0

>>> a = a + 1

>>> d = a + b + c
```

Now, with that understanding, complete the following table:

variable	type	value
a	int	
b		
c		
d		

You may need the help of Python to find the answers! Use the `type()` built-in function to this end.

Arithmetic operators

Ex 2.11

Give a description for each of the following statements and list the data type of the result:

statement	description	type
`a = 1 + 2`	add 1 and 2 and assign to a	int
`b = a`		
`a = 7.0 // 8.0`		
`a = 7 // 8`		
`a = 9.0 ** 10`		
`a = 5 % 2`		
`a = int(2.6)`		
`a = round(2.6)`		

1.5 Expressions and their evaluation

Ex 2.12 Complete the table below by using Python and the following abbreviations: T=True, F=False.

statement	**result**
`1 > 2`	F
`1 == 1`	
`10 == 11`	
`10 != 10`	
`2 < 4`	
`1==1 and 2 > 4`	
`1==1 or 2 > 4`	