Computational Science and Scientific Computing Workshop

Data Programming - Python as a scientific computing tool

Elliot S. MENKAH, Ph.D
Daniella N. APEADU

National Institute for Mathematical Sciences
Kwame Nkrumah University of Science and Technology

October 14, 2025

data storage, loops, len and range, if statements

Interpreter - List, Tuples and Dictionaries

```
>>> x = ["Hey", "you", 5, 8.7]
>>> y = ("hello", "hi", "you")
>>> w =  {"foo": 1.0, "bar": 2.0 }
>>> print(type(x))
>>> <class 'list'>
>>> print(type(y))
>>> <class 'tuple'>
>>> print(type(w))
>>> <class 'dict'>

```

Empty list:

```
>>> x = []
>>> x
[ ]

```

Interpreter - List, Tuples and Dictionaries

Indexing and memory location:
Memory locations for storing data in list and tuples are indexed so that one could access data stored in a specific memory locations.

NB: By default, index locations begin from zero (0).

```
>>> z = [2, 3, 4, 5]
>>> num0 = z[0]
>>> print(num0)
2

```

Interpreter - List, Tuples and Loops

Loops, List and **range**:

```
>>> for i in z:
...     print(i)
...

```

```
2
3
4
5

```

range & **len** intrinsic functions

```
>>> range(4)
range(0,4)
>>> len(z)
4

```

0 = Starting index
4 = Total no. of numbers
4 = Number of elements in list **z**.

Interpreter - List, Tuples and Loops

range and **len** can be combined and used in loops:

```
>>> for i in range(len(z)):
...     print(i)
...
```

```
0
1
2
3
```

len gives length of list z, that is, 4.
range gives 4 integers used as indexes starting from index 0.

Interpreter - While loops and Boolean

while loops, **if** statements and **boolean**

```
>>> a = True:
>>> print(a)
True
>>> res = 0
>>> while  (a):
...     res += 1
...     print(res)
...     if (res >= 10):
...         a = False


```

```
1
2
3
4
...

```

boolean **a** changes to False and it is used to terminate loop in the condition test section

modules

Modules - import, help, dir

There are lots of libraries in Python that can be imported to use rather than having to build your own. This makes life much easier.
E.g. **math**

```
>>> import math

```

Docs of modules can be viewed with the **help** and **dir** methods.

```
>>> help(< module >)
>>> help(math)
...
...
...
>>> dir(math) or >>> print(dir(math))

```

help gives a comprehensive documentation of the module.
dir gives you the symbols contained in the method concerned.

Modules - import, help, dir

```
>>> help(math.log)
...
...
...
```

import math place the math class in current environment.

```
>>> math.log(10)
2.3025
>>> math.cos(2 * math.pi)
1.0

```

Modules - More on **import**

Partial or selective importation of modules.

In the event of wanting to import only a few symbols into your namespace, the **from** statement is made use of.

```
>>> from math import < symbol or method >
>>> from math import cos
>>> cos(90)
-0.448073616129170l
```

Multiple methods can be imported

```
>>> from math import cos, pi
>>> cos(2 * pi)
1.0
```

Plotting - Matplotlib

Matplotlib

```
import matplotlib.pyplot as plt

plt.plot(X_data, Y_data)
plt.title("Title of plot")
plt.xlabel("X Axis Lable")
plt.ylabel("Y axis Label")

plt.savefig("NameOfFile.png")
```

Listing: Plottin with Matplotlib

functions

Functions

Functions in Python are defined by the keyword **def**

```
>>> def func(x):
...     res = x + 1
...     return res
...
>>> d = func(4)
>>> d
5
```

Python scripts

Script

```
#! /usr/bin/python

print("Hello World")

```

Terminal

```
python3 hello.py
```

Exercises 2

Convert your code from Exercise 1 into a function (that returns a value), where the initial condition, **u**, and the time, **t**, are arguments.

Exercises 3 - Algorithm Development

Exercise A: Multiples of 3 & 5

If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The sum of these multiples is 23. Implement an algorithm, with Python, to find the sum of all the multiples of 3 or 5 below 1000.

Exercise B: Fibonacci sequence

Each new term of the Fibonacci sequence is generated by adding the previous two terms. By starting with 1 and 2, the first 10 terms will be:

1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...

By considering the terms of the Fibonacci sequence whose values do not exceed four million, find the sum of the even-valued terms.