# 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 15, 2025

# 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)
-0.4480736161291701

```

Multiple methods can be imported

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

```

# Plotting - Matplotlib

## Matplotlib

```
1 import matplotlib.pyplot as plt
2
3 plt.plot(X_data, Y_data)
4 plt.title("Title of plot")
5 plt.xlabel("X Axis Lable")
6 plt.ylabel("Y axis Label")
7
8 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

```
1 #! /usr/bin/python
2
3 print("Hello World")
4
5
```

Terminal

```
1 python3 hello.py
2
```

# Exercises 3

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

# arrays and multidimensional vectors

# Handling Arrays & Multidimensional Vectors

**Vector Operation**

$$
\begin{aligned}
\vec{a} \cdot \vec{b} &= \sum_{i=0}^{N} a_i b_i \\
&= \begin{pmatrix} 20 & -3 & 5 \end{pmatrix} \begin{pmatrix} 15 \\ -2.249 \\ 1 \end{pmatrix} \\
&= 20(15) - 3(-2.249) + 5(1) \\
&= 300 + 6.747 + 5 \\
&= 311.747
\end{aligned}
$$

# Handling Arrays & Multidimensional Vectors

**Multidimensional Arrays**

$$\begin{bmatrix} 20 & 15 & 10 & 45 \\ -3 & -2.249 & 7 & 1.751 \\ 5 & 1 & 3 & 9 \end{bmatrix} = \begin{pmatrix} 20 \\ -3 \\ 5 \end{pmatrix} \begin{pmatrix} 15 \\ -2.249 \\ 1 \end{pmatrix} \begin{pmatrix} 10 \\ 7 \\ 3 \end{pmatrix} \begin{pmatrix} 45 \\ 1.751 \\ 9 \end{pmatrix}$$

# file I/O, exceptions and assertions

# File I/O

keyword: **open**

```
>>> fh = open("demofile.txt", "a")
>>> fh.write("My data file \n")
>>> fh.write("Results: %d", res)
>>> fh.close
```

# Exceptions and Assertions

This is a way to handle expected and unexpected errors.

1. Exceptions Handling
2. Assertion

```
try:
# Runs First
< code >
except:
# Runs if exceptions occurs in try block
< code >
else:
# Executes if try block succeeds.
< code >
finally:
# This code always runs executes.
< code >

```

# Exceptions and Assertions

## Exception Example

```
1  def read_file(path):
2  """ Return the content of a file at path"""
3  try:
4  fh = open(path, mode="rb")
5  data = f.read()
6  return data
7  except FileNotFoundError as err:
8  raise
9  else:
10 fh.close
11 finally:
12 print("Leaving file read routine")
13
```

# Python Basics - End

End of Basics.
Questions ?
Review

# Numerical and Scientific Python
Numpy and Scipy libraries

# Numerical Python - NumPy

Arrays could be made from:

1. Python list or tuples
2. Using functions that are dedicated to generating numpy arrays, such as *arange*, *linspace*, etc.
3. Reading data from files

```
from numpy import as np
v = array([1,2,3,4])
-----
[1,2,3,4]

```

```
M = np.array([[1, 2], [3, 4]])
-----
array([[1, 2],
[3, 4]])

```

# Exercises 4

Using the python programming language, write a code that implements the solution or finds the roots of the non-linear equation:
$3x^2 + 2x - 1 = 0$
using the

1. Bisection Method
2. Newton-Raphson's Method
3. Secant Method

as separate functions.

# Classes

Classes in Python are defined by the keyword **class**

```
>>> class myfunctions:
...
...     def add(x):
...         res = x + 2
...         return res
>>>
>>> yy = myfunctions.add(7)
>>> yy
9
```

# End of talk

Thank you