# 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 - If Statements

**if...**

```
>>> if(condition):
...     statement_1
...     statement_2
...
```

**if...else**

```
>>> if(condition):
...     statement_1
...     statement_2
... else:
...     statement_3
...     statement_4
```

# Interpreter - If Statements

**if...else if**

```
>>> if(condition_1):
...     statement_1a
...     statement_2a
... else if(condition_2):
...     statement_1b
...     statement_2b
```

**if...else if...else**

```
>>> if(condition_1):
...     statement_1a
...     statement_2a
... else if(condition_2):
...     statement_1b
...     statement_2b
... else if(condition_3):
...     statement_1c
...     statement_2c
... else:
...     statement_1d
```

# Interpreter - If Statements

**Nested if**

```
>>> if(condition_1):
...   if(condition_1A):
...     statement_1A_a
...     statement_2A_a
...   else:
...     statement_1A_b
...     statement_2A_b
... else:
...   if(condition_2A):
...     statement_2A_a
...     statement_2A_b

```

# Interpreter - List, Tuples and Loops

**for loop 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 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

# Exercises 2 - 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.