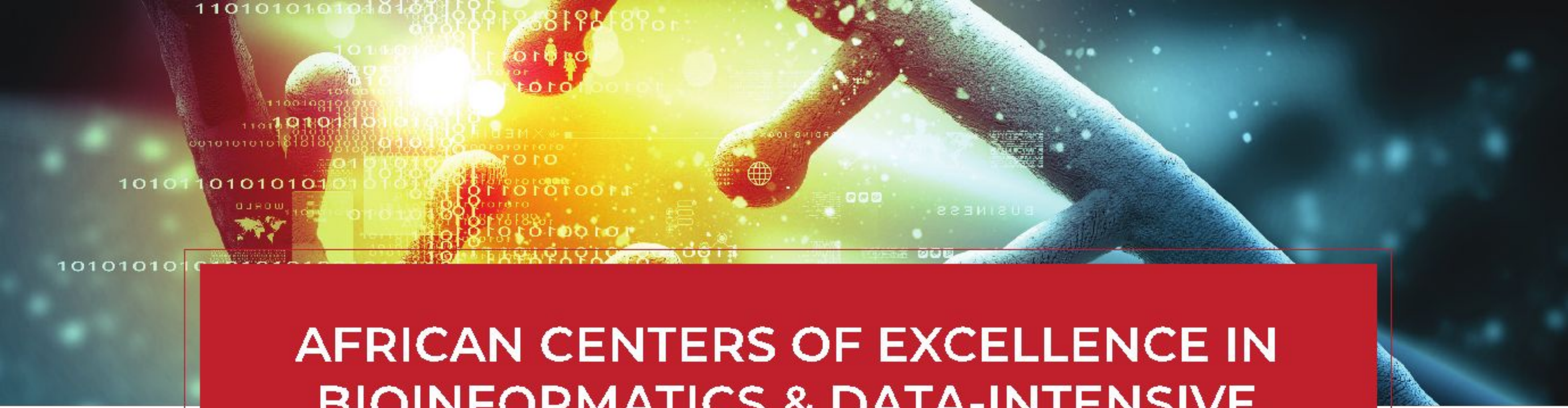
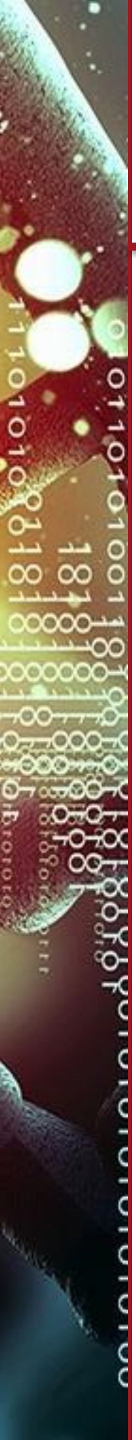


AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

ACE Workshop on HPC with Python
Day 1 HPC Python



Welcome to the Workshop!

- About me
- What I expect
  - follow along
  - lots of questions
- Schedule

AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

On Python

- First developed in late 1980’s and implemented in 1989 by Guido van Rossum

Why Python?

- easy-to-use
- expressive

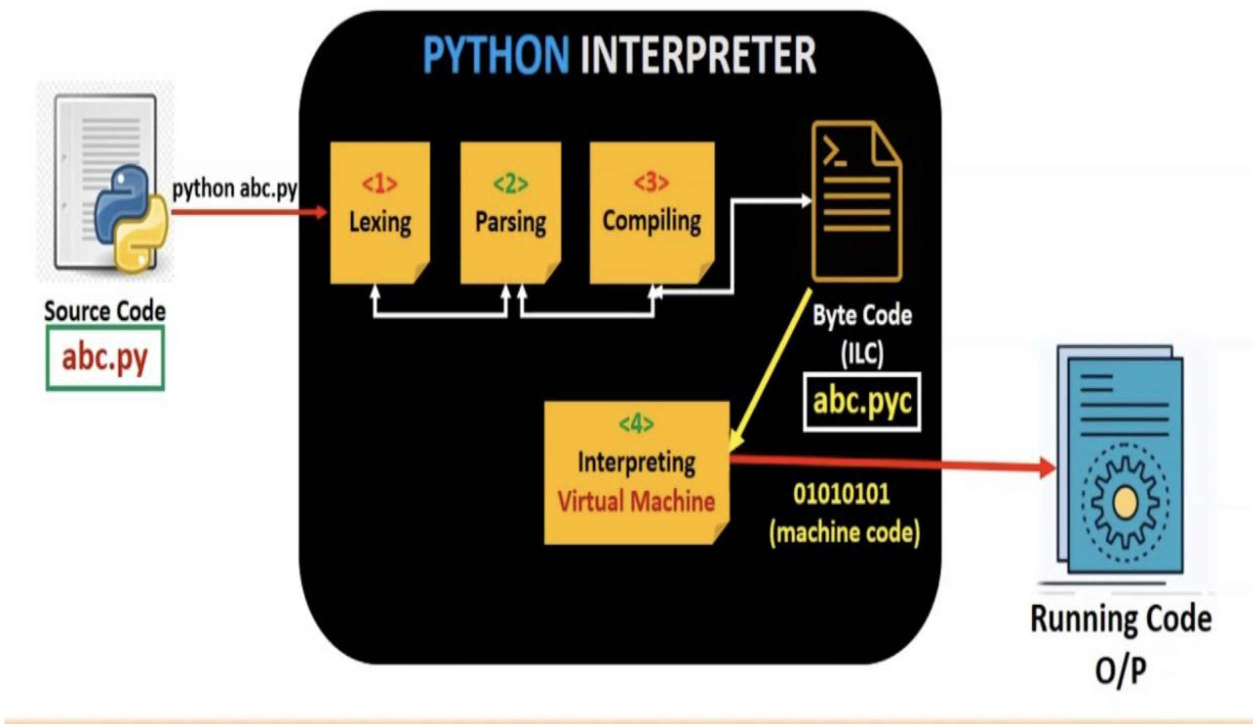
On Python: What Sort of Language

- Intepreted
- General Purpose
- Dynamically-Typed
- Fully Object-Oriented

AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

Interpreted Language

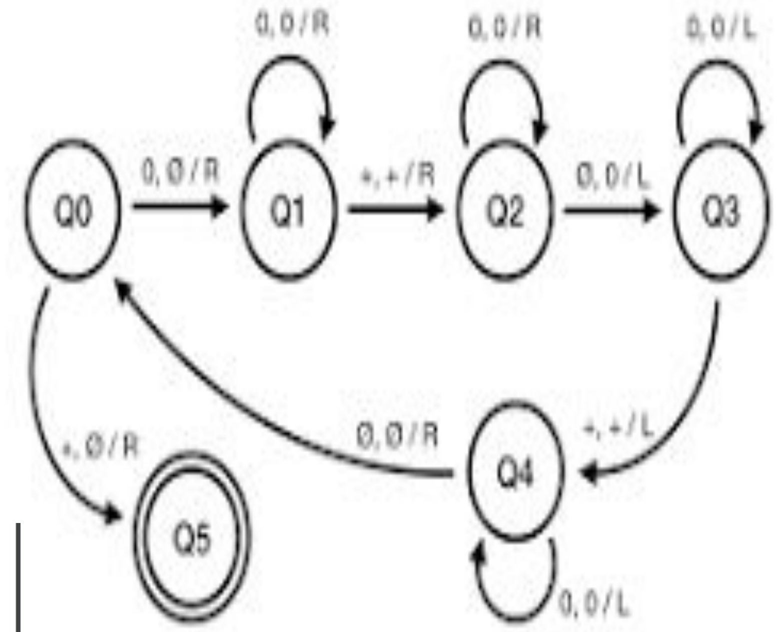
- Compiled versus Interpreted
  - Compiler
  - Machine-code
- Pipeline from User to Execution
  - User input
  - lexing
  - parsing
  - compiling
  - vm interpretation
  - machine-code/execution
- Cpython
  - Implemented in C

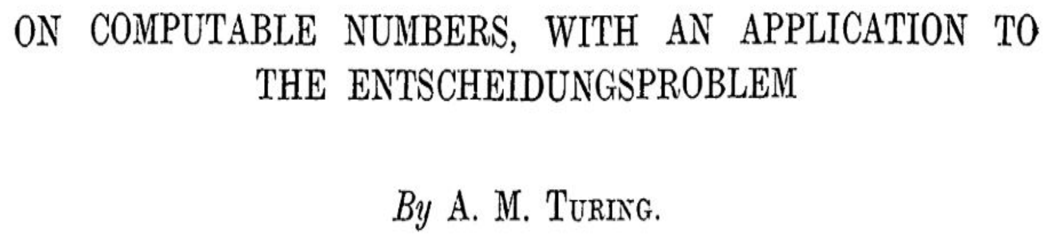


AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

General-Purpose

- Turing-computable
  - Turing-machine
  - Turing-machine as model
  - Human computer
  - Human computable




ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

Dynamically-Typed

- Static versus Dynamically Typed
- Implicit Versus Explicit
    - Initialization

Data Types
in Programming
Integer
Float/Double
Character
String
Boolean
Date/Time
Arrays
2d Arrays

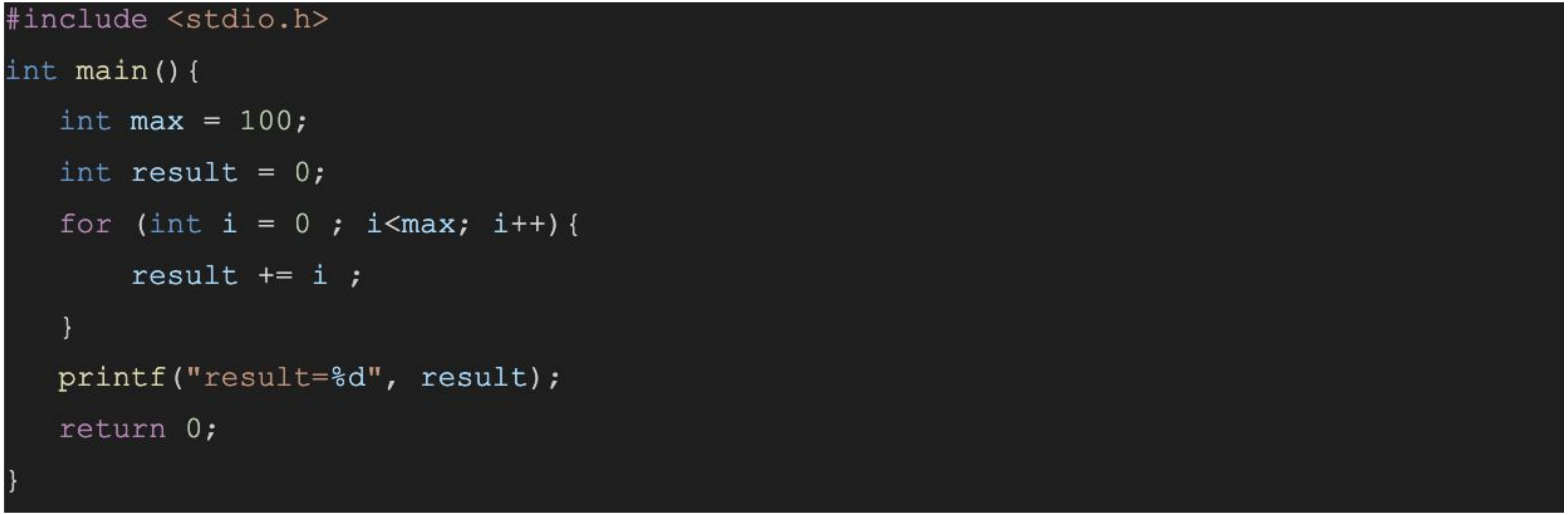
AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

Object-Oriented

- Object as Computer Science Data Structure
  - functions
  - data
  - example e.g. soccer ball
- Python and OOP
  - everything is an object, types, integers, functions etc.
  - dot-notation
  - function called **method** accessed via **object.method()**
  - data is called **attribute** accessed via **object.attribute**

AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

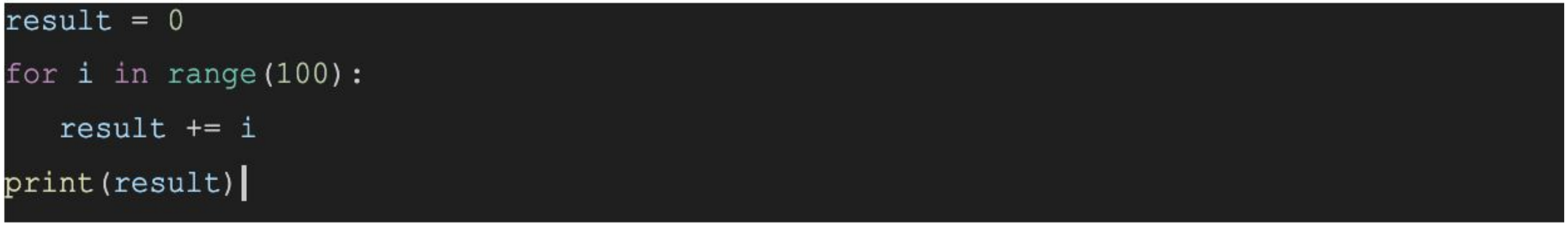
Comparison with C via example



```
#include <stdio.h>
int main(){
   int max = 100;
   int result = 0;
   for (int i = 0 ; i<max; i++){
       result += i ;
   }
   printf("result=%d", result);
   return 0;
}
```

AFRICAN
CENTERS
OF EXCELLENCE
IN BIOINFORMATICS &
DATA-INTENSIVE SCIENCE

Example of Dynamic



```
result = 0
for i in range(100):
   result += i
print(result)
```

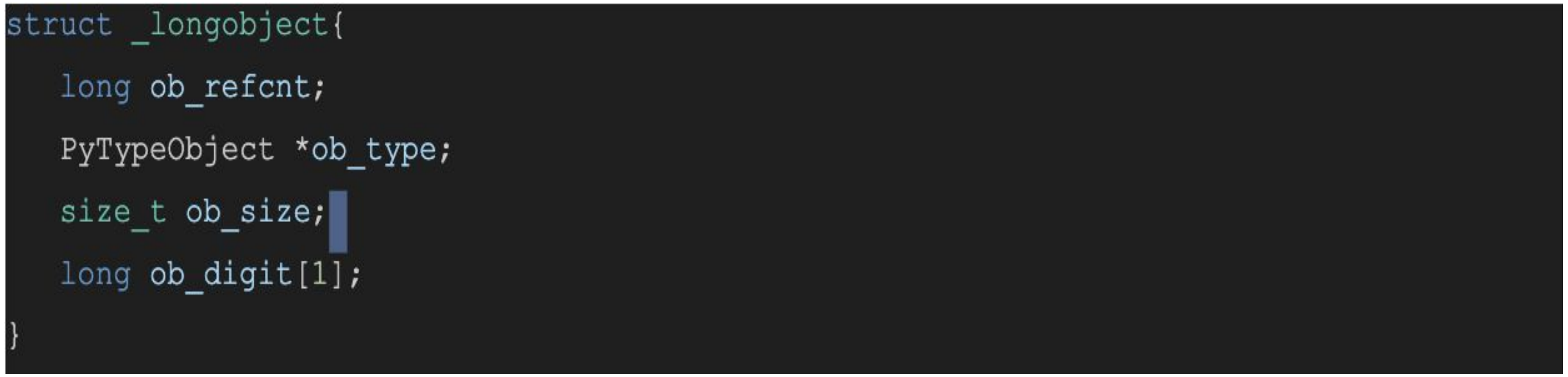
AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

Python Objects

As per documentation:

Every object has an identity, a type and a value. An object's identity never changes once it has been created; you may think of it as the object's address in memory. The 'is' operator compares the identity of two objects; the id() function returns an integer representing its identity.

Actually a pointer to a C struct:

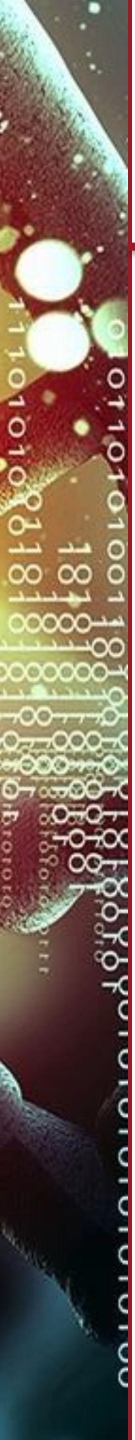


```
struct _longobject{
    long ob_refcnt;
    PyTypeObject *ob_type;
    size_t ob_size;
    long ob_digit[1];
}
```

AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

Why not Python

- Interpreted, so it is slower
- Incentivize sloppy coding-practice
- Indentation can be annoying (my opinion)
- Because Python is slow….



SciPy

- SciPy (pronounced "Sigh Pie") is a Python-based ecosystem of open-source software for mathematics, science, and engineering
  - NumPy
  - Pandas
  - Matplotlib
  - Juptyer

SciPy NumPy

AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

NumPy

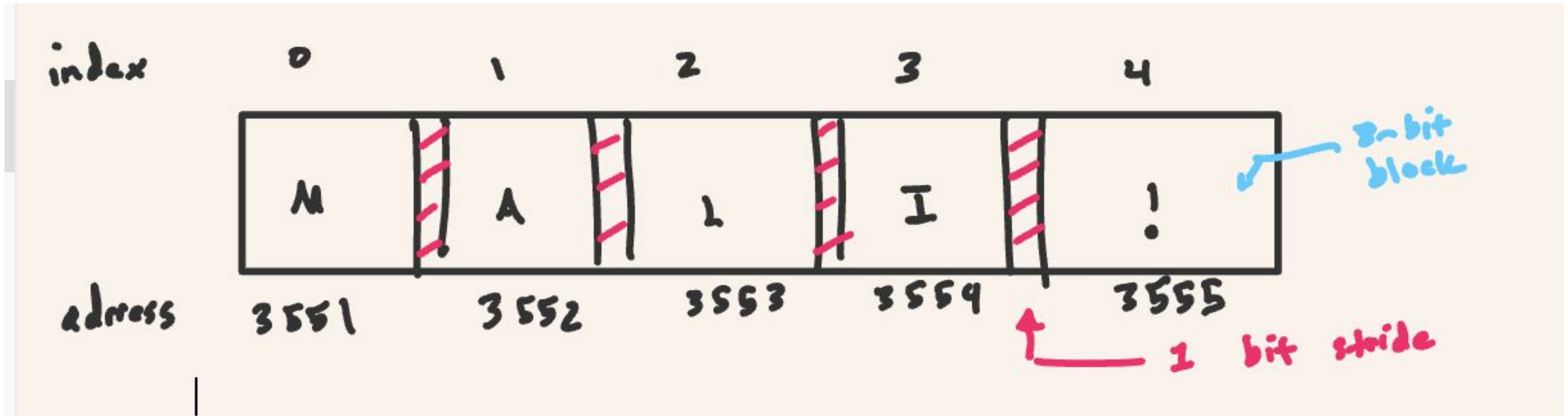
- Numerical Python i.e. NumPy
  - module
    - packaging up of code
    - makes it easy to install and use in code
- Fast because it is a wrapper on C code
- Built on top of the ndarray (“n-dimensional” array)
- Ndarray consists of two ingredients :
  - data buffer
  - meta-data about buffer

Array in General and Ndarray

- What is an array?
  - Array is a contiguous block of memory made up of homogeneous elements of fixed size of the same-type
- Anatomy of an array:
  - indices
  - stride
  - blocklength

AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

Structure of an Array



AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

NumPy

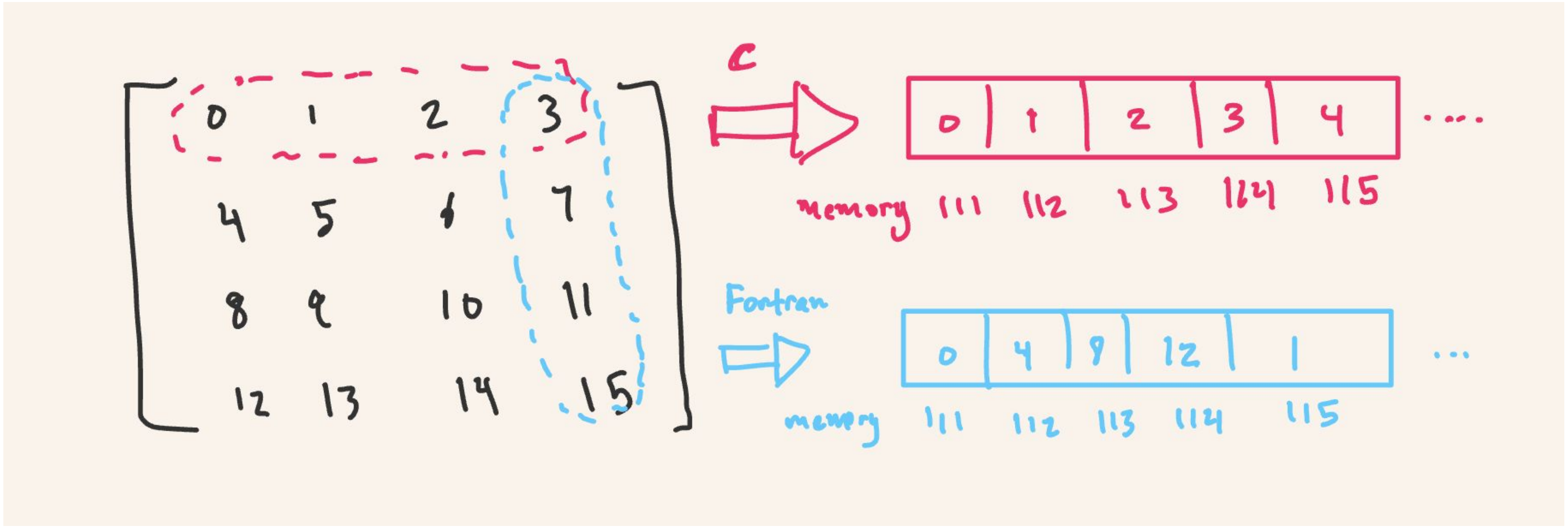
- Data buffer
  - size efficient
  - permits vectorized operations
- What about meta-data on buffer?

AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

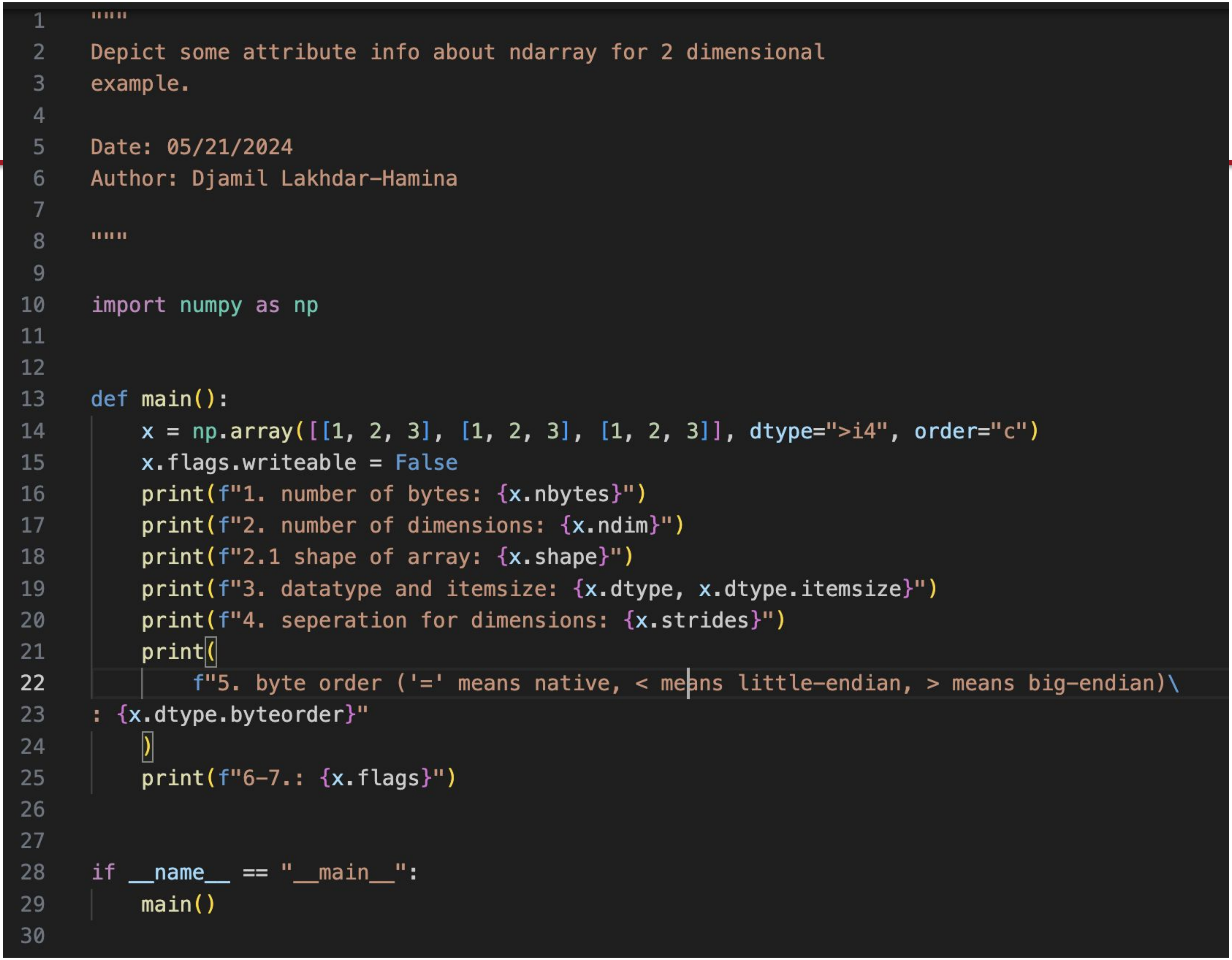
Meta-data on Buffer

1. The size of the whole array in bytes
2. The dimensions and shape of the array
3. Information (via the **dtype** object) about the interpretation of the basic data element.
4. The separation between elements for each dimension (the stride). This does not have to be a multiple of the element size.
5. The byte order of the data (which may not be the native byte order).
    a. big versus little endian
6. Whether the buffer is read-only.
7. Whether the array is to be interpreted as C-order or Fortran-order.

C versus Fortran Ordering



AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE



```python
"""
Depict some attribute info about ndarray for 2 dimensional
example.

Date: 05/21/2024
Author: Djamil Lakhdar-Hamina

"""


import numpy as np


def main():
    x = np.array([[1, 2, 3], [1, 2, 3], [1, 2, 3]], dtype=">i4", order="c")
    x.flags.writeable = False
    print(f"1. number of bytes: {x.nbytes}")
    print(f"2. number of dimensions: {x.ndim}")
    print(f"2.1 shape of array: {x.shape}")
    print(f"3. datatype and itemsize: {x.dtype, x.dtype.itemsize}")
    print(f"4. seperation for dimensions: {x.strides}")
    print(
        f"5. byte order ('=' means native, < means little-endian, > means big-endian)\
: {x.dtype.byteorder}"
    )
    print(f"6-7.: {x.flags}")


if __name__ == "__main__":
    main()
```

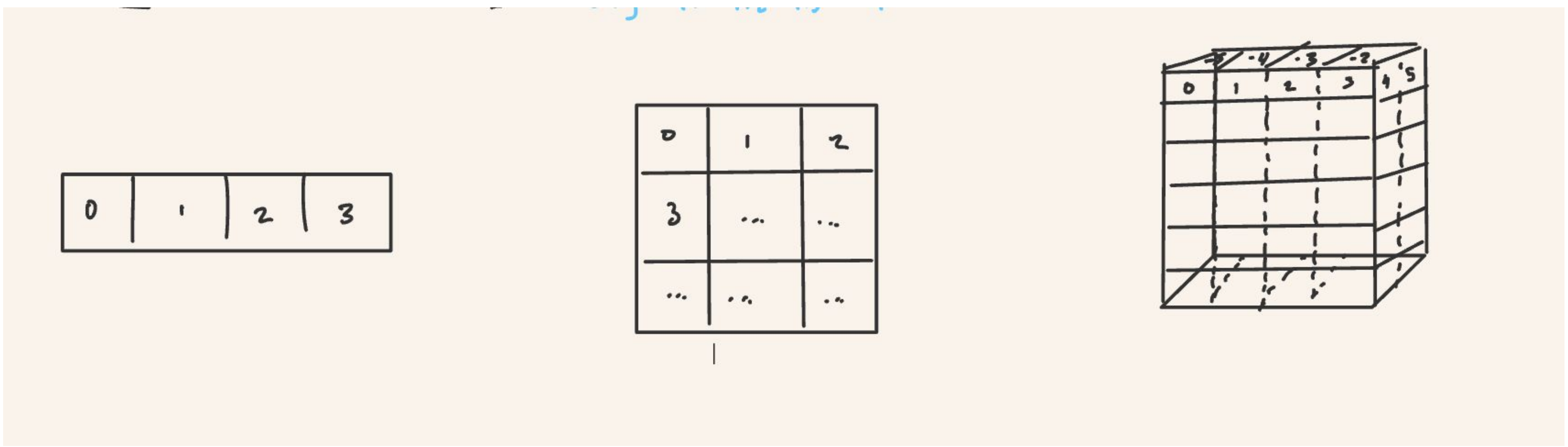
AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

Dimensionality and Shape

- dimension as numbers needed to pick out one element
- geometrically as different lattices like coordinate-systems
- shape is each how many spaces in each dimension all taken together in tuple form
- 1-d = vector , 2-d = matrix

AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

Dimension and Shape

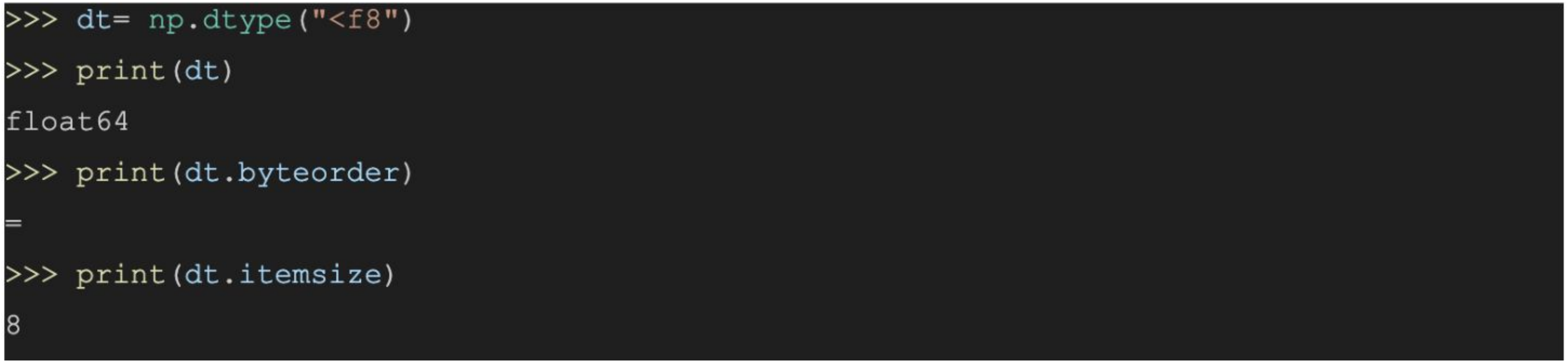


dtype class

- Type of data (integer, float, Python object, etc.)
- Size of data (how many bytes is in *e.g.* the integer)
- Byte order of the data (little-endian or big-endian)
- If the data type is structured data type, an aggregate of other data types
    - what are the names of the “fields” of the structure, by which they can be accessed,
    - what is the data-type of each field, and
    - which part of the memory block each field takes.
- If the data type is a sub-array, what is its shape and data type.

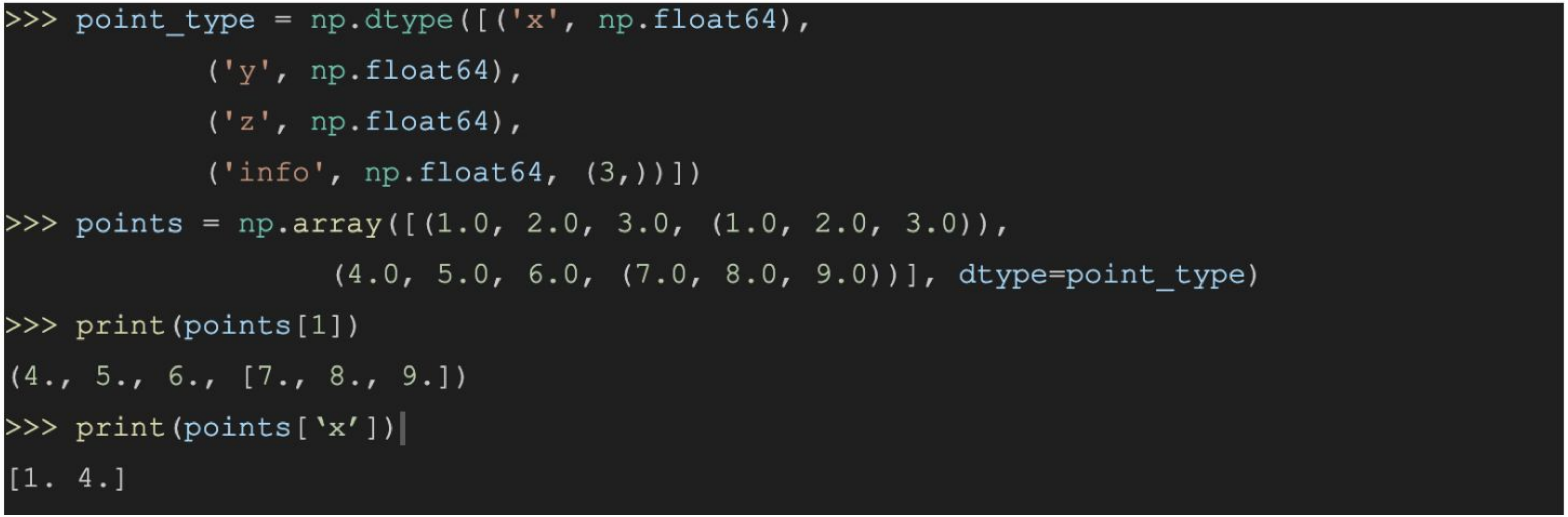
AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

Creating our own Datatype



```
>>> dt= np.dtype("<f8")
>>> print(dt)
float64
>>> print(dt.byteorder)
=
>>> print(dt.itemsize)
8
```

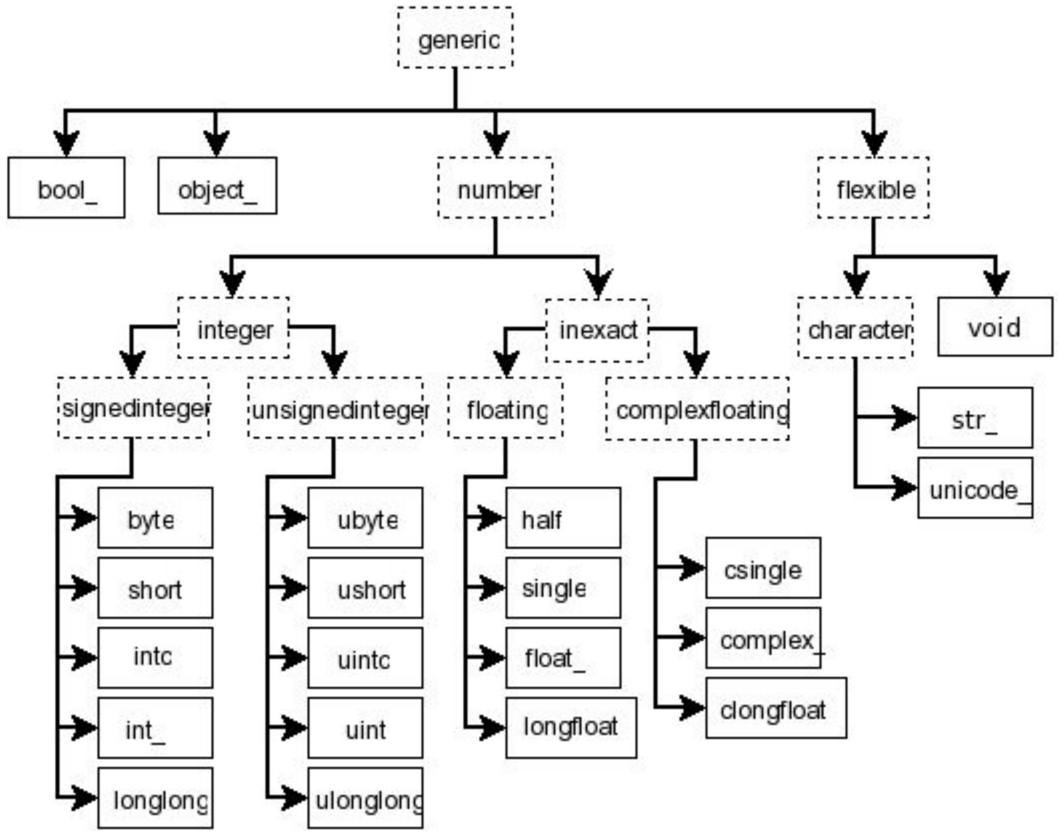
Structured Array



```
>>> point_type = np.dtype([('x', np.float64),
             ('y', np.float64),
             ('z', np.float64),
             ('info', np.float64, (3,))])
>>> points = np.array([(1.0, 2.0, 3.0, (1.0, 2.0, 3.0)),
                   (4.0, 5.0, 6.0, (7.0, 8.0, 9.0))], dtype=point_type)
>>> print(points[1])
(4., 5., 6., [7., 8., 9.])
>>> print(points['x'])
[1. 4.]
```

AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

Types and their Hierarchy



AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

Exercise 0: Setting up Environment

- need to make sure everyone has:
  - python 3.10-3.12
  - github account
  - git command-line tool
  - virtual studio code
    - if you have some place to code don’t worry about this

AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

Git repository for workshop

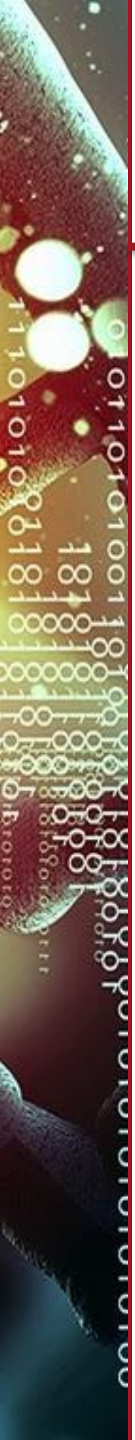
- Set up our git repository

AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

Initializing Arrays

- From Lists
- From Tuples
- From a Combo
- Nested Lists

AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE



Create Arrays from Functions

- array of zeros
- array of ones
- array of any number
- **np.arange**
- **np.linspace**
- **np.random** way

AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

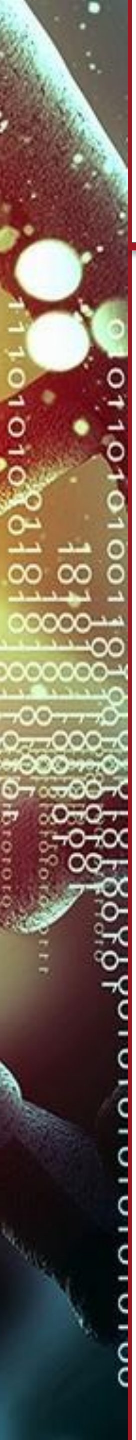
Exercise # 1 :

Create an array from nested tuples and make each number a 32-bit integer.

Hint #1 : use the dtype keyword argument
Hint #2 : a tuple is for instance (1,2,3)

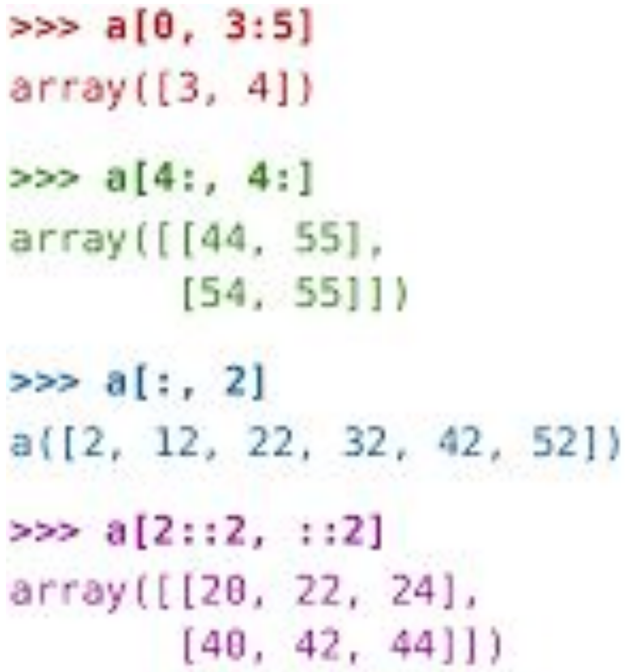
Array of Characters

- representation of strings as bytes
- byte-strings

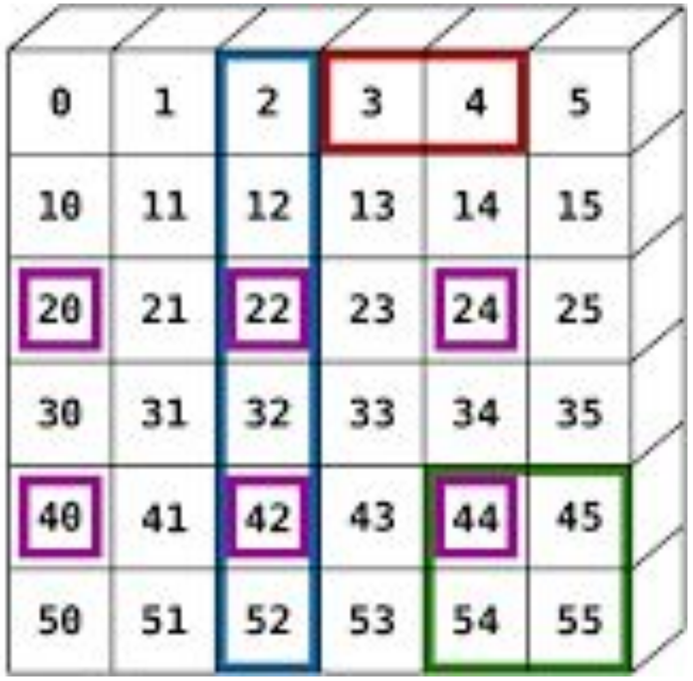


Indexing and Slicing

- Indexing is picking out elements: x[0]
- Slicing picking out a group: x[start:stop:step]



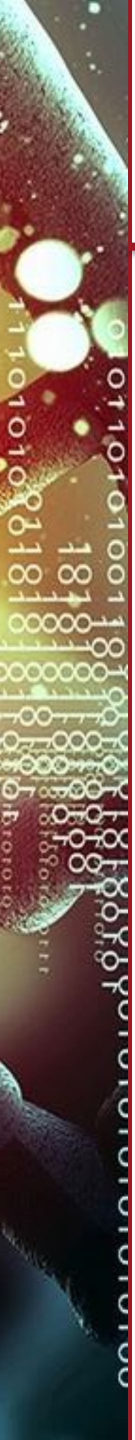
```
>>> a[0, 3:5]
array([3, 4])

>>> a[4:, 4:]
array([[44, 55],
       [54, 55]])

>>> a[:, 2]
a([2, 12, 22, 32, 42, 52])

>>> a[2::2, ::2]
array([[20, 22, 24],
       [40, 42, 44]])
```



AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

Indexing and Slicing

- indexing 1-d (vector)
- indexing 2-d (matrix)
- indexing 3-d (dim0, dim1, matrix)



AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

View and Copying

- When we slice we create views
    - arrays as mutable objects
    - view modification = array modification
- If you want new array use **.copy()**
    - creates a brand new array , own unique id
    - modify new array , no modification of old

AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

Exercise # 2: 5 minutes

Create an array of string-bytes, include all the characters. Then slice that string byte array. Extra credit: reverse the string.

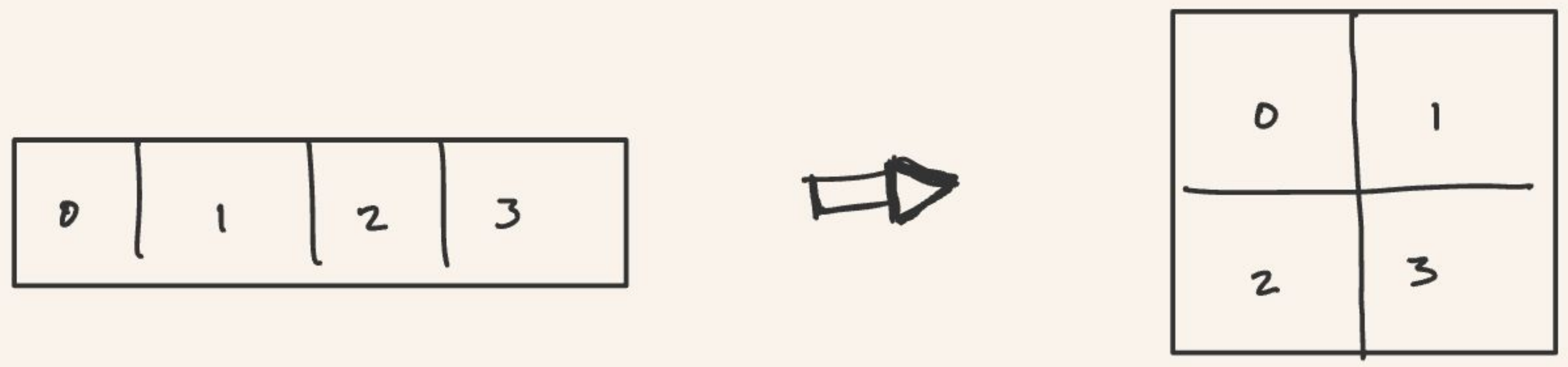
hint: the b in front of ’’ i.e. b’string’ matters

hint: use **np.frombuffer**

AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

Reshaping

- Take array of one dimension and shape and change it to another dimension and shape

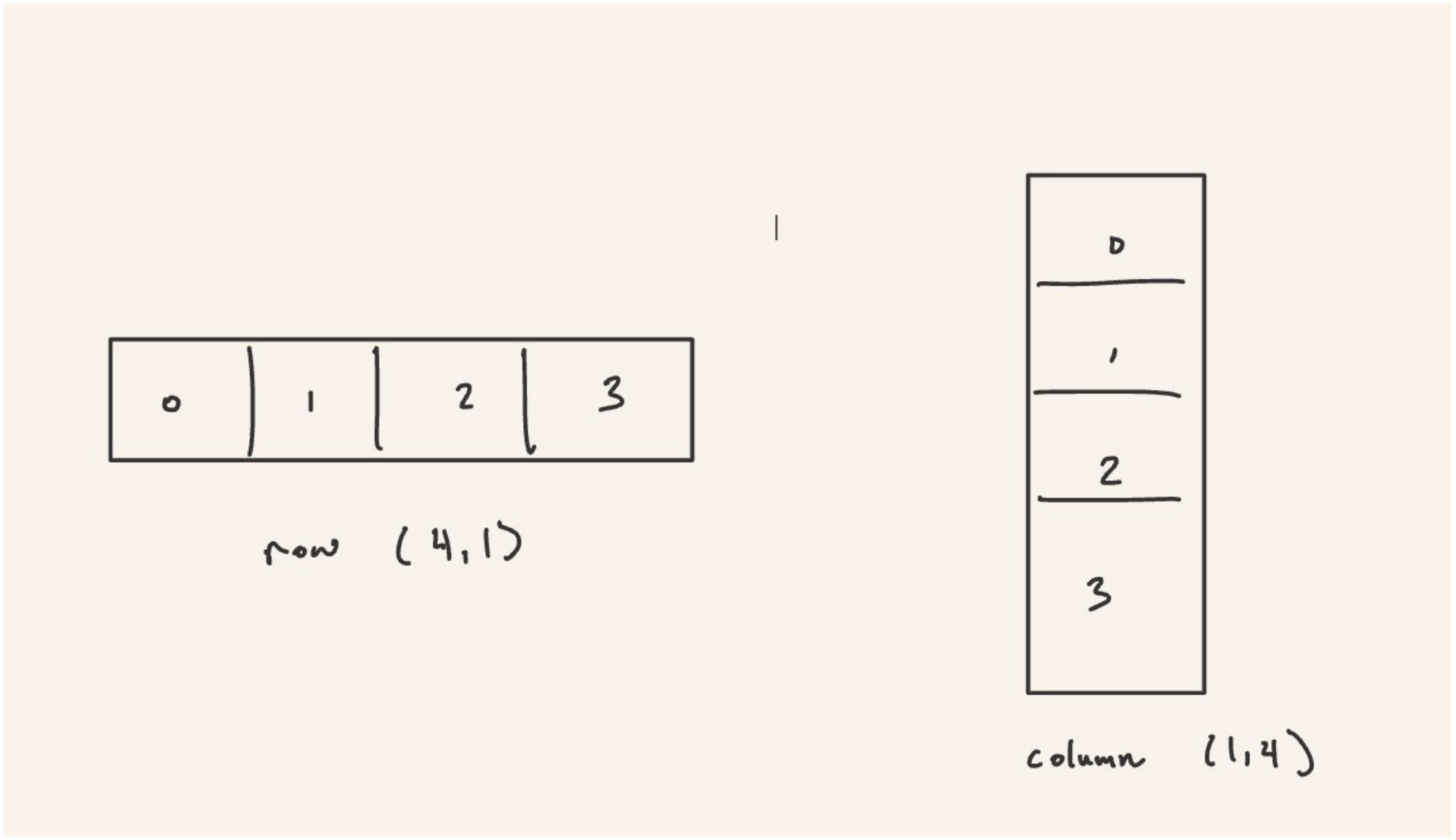


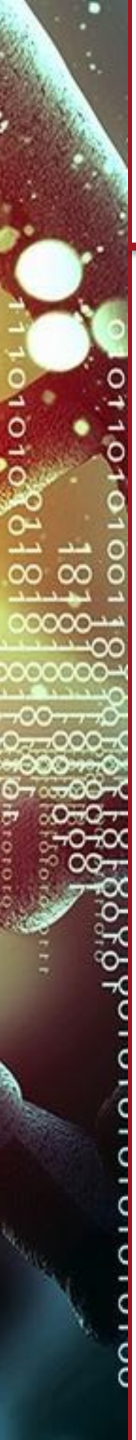
Fundamental Rule of Reshaping

Rule: If you have an array with n elements and wish to reshape it into an array with dimensions (n0,n1,n2….n) then n%(n0*n1*n2…*n)= 0 where % is modulus operator.

AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

Row and Column Vector





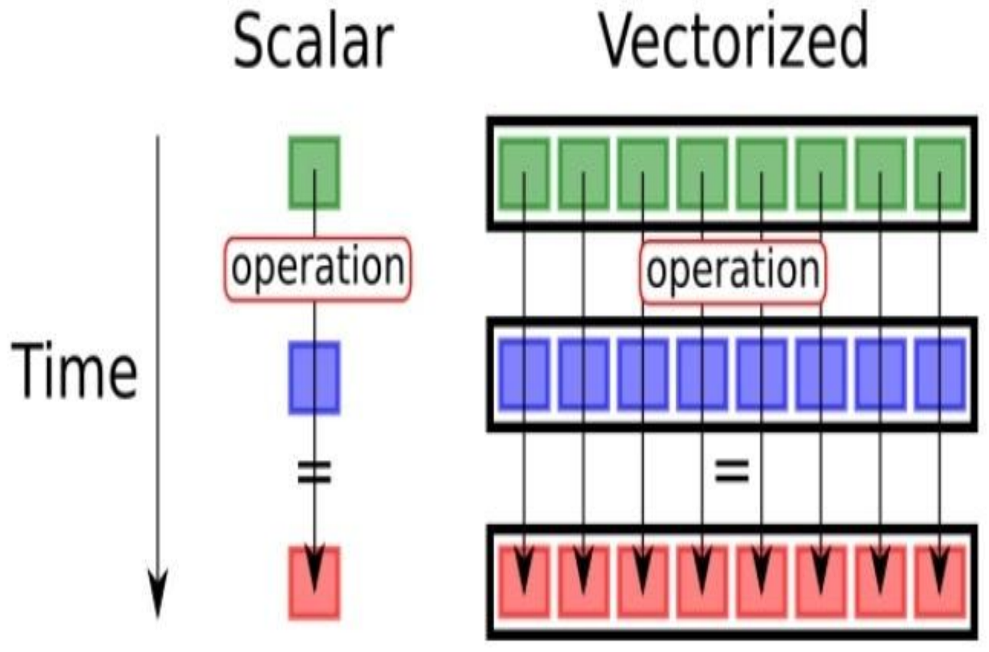
Joining, Splitting, Aggregating

- Joining
  - bringing together arrays to form new arrays
    - np.concatenate
    - np.hstack
    - np.vstack
- Splitting
  - the opposite
    - np.split
    - np.hsplit
    - np.vsplit

AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

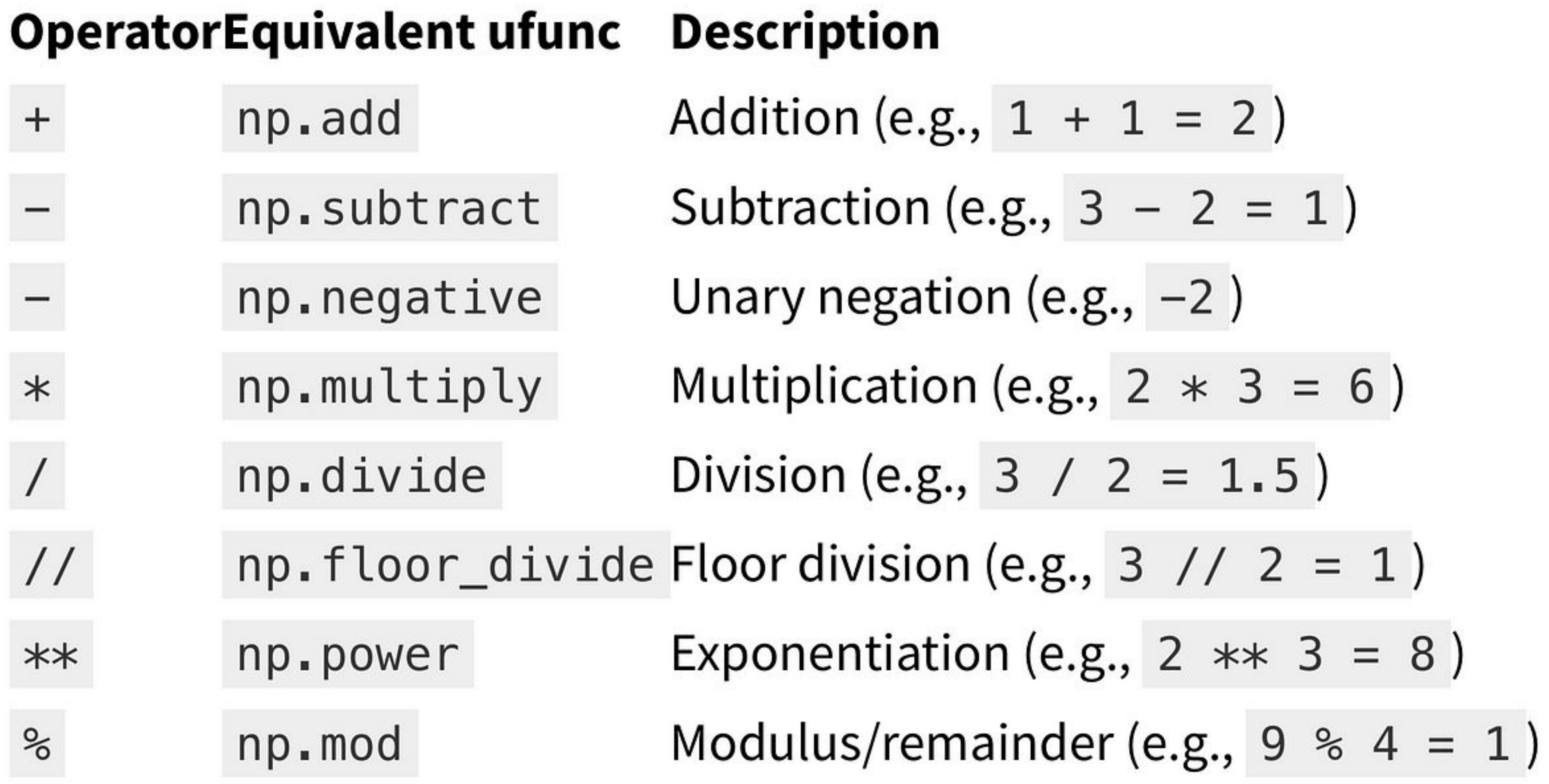
Universal Functions i.e. ufuncs

- NumPy and the Ufunc
  - vectorized operation
    - concurrent execution
    - operand versus operator
    - unary vrs. binary
- Overloaded functions
  - “overwriting” behavior
    - based on type



Comparison between scalar (classical) computation and vectorization.

Common Overloaded **ufuncs**

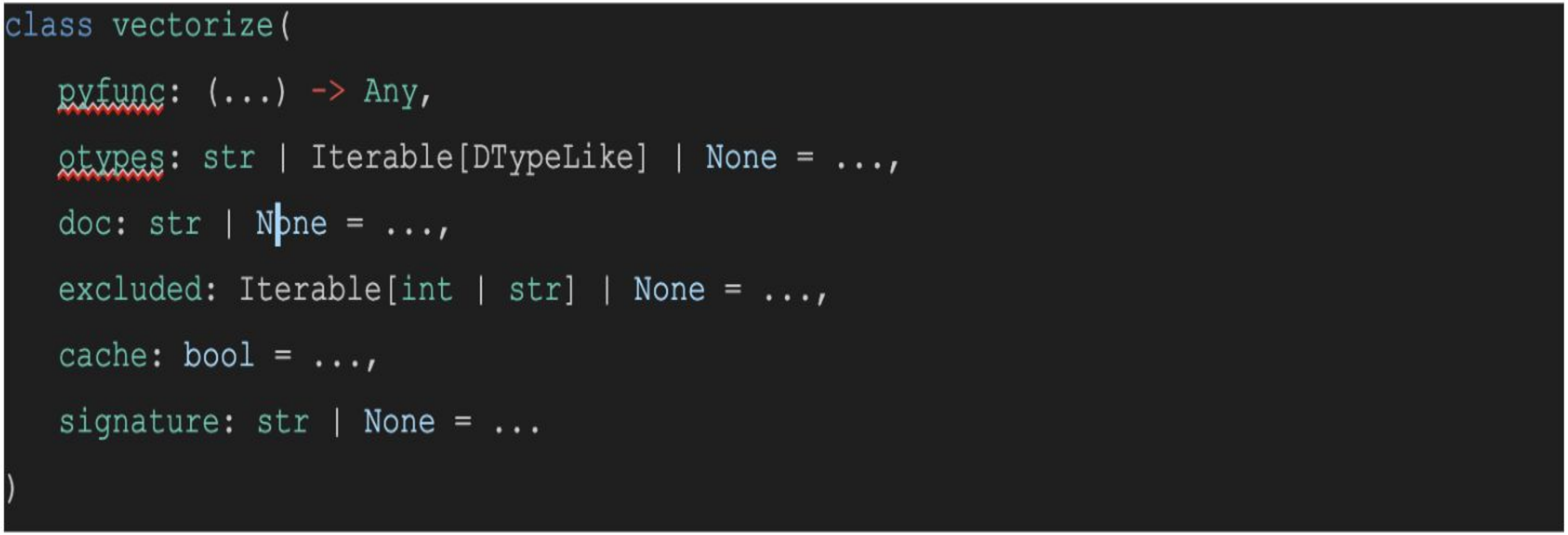


Operator	Equivalent ufunc	Description
`+`	`np.add`	Addition (e.g., `1 + 1 = 2`)
`−`	`np.subtract`	Subtraction (e.g., `3 − 2 = 1`)
`−`	`np.negative`	Unary negation (e.g., `−2`)
`∗`	`np.multiply`	Multiplication (e.g., `2 ∗ 3 = 6`)
`/`	`np.divide`	Division (e.g., `3 / 2 = 1.5`)
`//`	`np.floor_divide`	Floor division (e.g., `3 // 2 = 1`)
`∗∗`	`np.power`	Exponentiation (e.g., `2 ∗∗ 3 = 8`)
`%`	`np.mod`	Modulus/remainder (e.g., `9 % 4 = 1`)

AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

Custom Ufunc

- We can build our own ufuncs via **vectorize** or **@vectorize**



```
class vectorize(
    pyfunc: (...) -> Any,
    otypes: str | Iterable[DTypeLike] | None = ...,
    doc: str | None = ...,
    excluded: Iterable[int | str] | None = ...,
    cache: bool = ...,
    signature: str | None = ...
)
```

Reductions

In computer science there are operations called reductions. A reduction takes some array [a,b,c,...n] and chains together all elements with an operator let us call it * to produce a single value x i.e. x= a*b*c…*n. We can call **reduce** on a ufunc to produce that single value.

- attributes of **ufuncs**
- **reduce**
- **accumulate**

AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

Statistical Aggregations

Function Name	NaN-safe Version	Description
`np.sum`	`np.nansum`	Compute sum of elements
`np.prod`	`np.nanprod`	Compute product of elements
`np.mean`	`np.nanmean`	Compute mean of elements
`np.std`	`np.nanstd`	Compute standard deviation
`np.var`	`np.nanvar`	Compute variance
`np.min`	`np.nanmin`	Find minimum value
`np.max`	`np.nanmax`	Find maximum value
`np.argmin`	`np.nanargmin`	Find index of minimum value
`np.argmax`	`np.nanargmax`	Find index of maximum value
`np.median`	`np.nanmedian`	Compute median of elements
`np.percentile`	`np.nanpercentile`	Compute rank-based statistics of elements
`np.any`	N/A	Evaluate whether any elements are true
`np.all`	N/A	Evaluate whether all elements are true

AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

Exercise # 3

Create a random array with 1000 elements, index 100 of those elements in steps 2,and then 1) add-reduce them 2) find the median of them

hint #1 : remember that slicing syntax is array[start:stop:step]

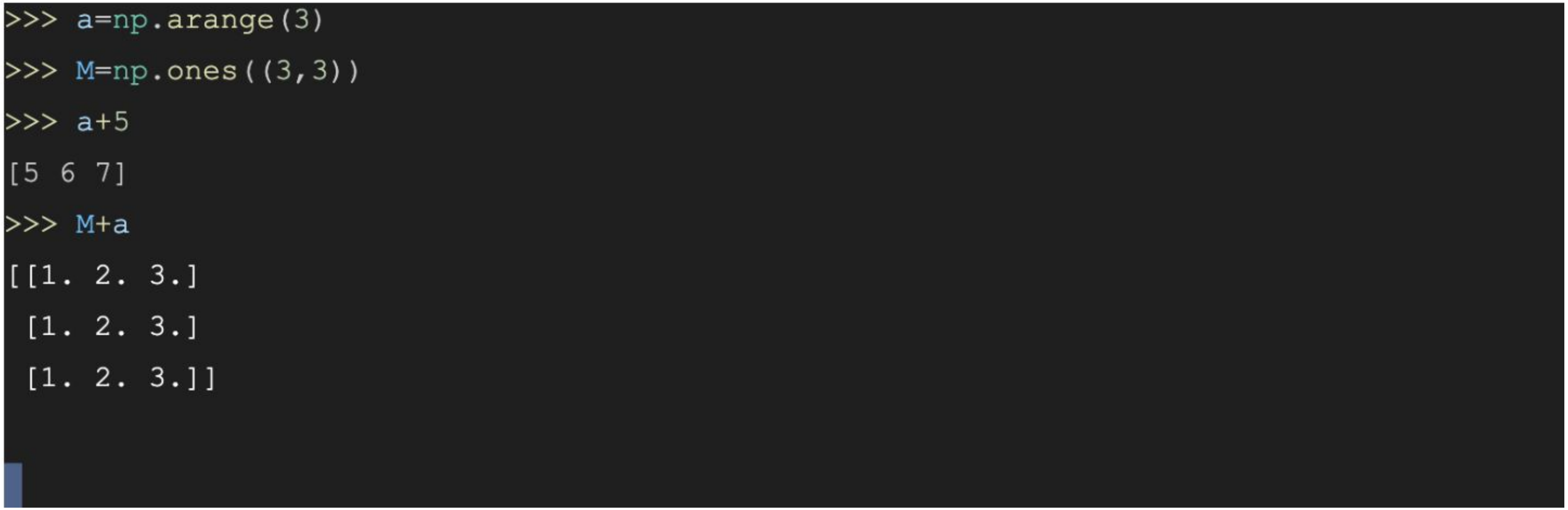
AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

Broadcasting

- Broadcasting refers to a technique of operating conjointly on arrays of different dimensions
- set of rules for applying binary ufuncs on arrays of differing dimensions.
- It allows us to apply vectorized operations to vectors of different sizes, retaining the efficiency of the numpy array and its ufuncs.
- For arrays of the same size binary operations just operate on an element-by-element basis. This is what we came to expect.

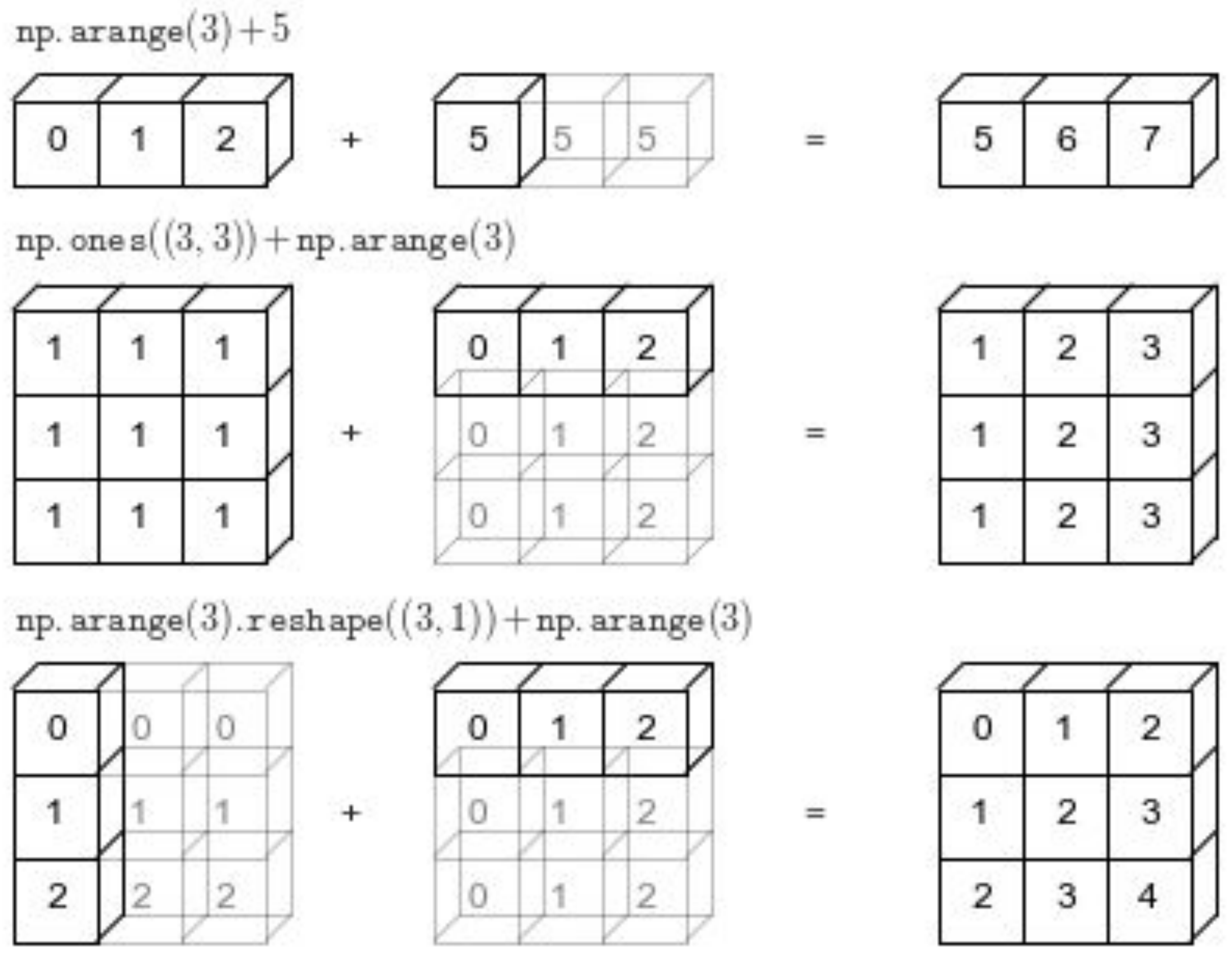
AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

Puzzling Example

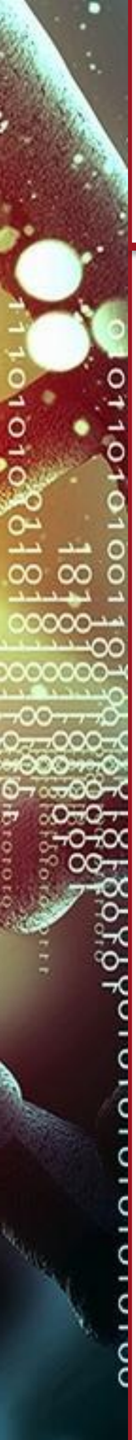


```
>>> a=np.arange(3)
>>> M=np.ones((3,3))
>>> a+5
[5 6 7]
>>> M+a
[[1. 2. 3.]
 [1. 2. 3.]
 [1. 2. 3.]]
```

What happened?



AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE



Rules of Broadcasting

1. If the two arrays differ in their dimensions, the shape of the one with the fewer is padded with ones on its leading left side.
2. If the shape of the two arrays does not agree in any dimension, the array with shape equal to one in that dimension is stretched to match that other shape.
3. If in any dimension the sizes disagree and neither is equal to 1, an error is raised.

AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

Example # 1 : One-Operand Broadcast

Let us add 1-d of shape (3,) to 2-d array

Example #2 : Two-Operand Broadcast

AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

Exercise #4 : Broadcast Calculation

You have a column vector with shape (1,4) and a row vector with shape (4,1) what array can we expect from their addition?

List out the steps with the rules, then create the vectors and execute it in python.

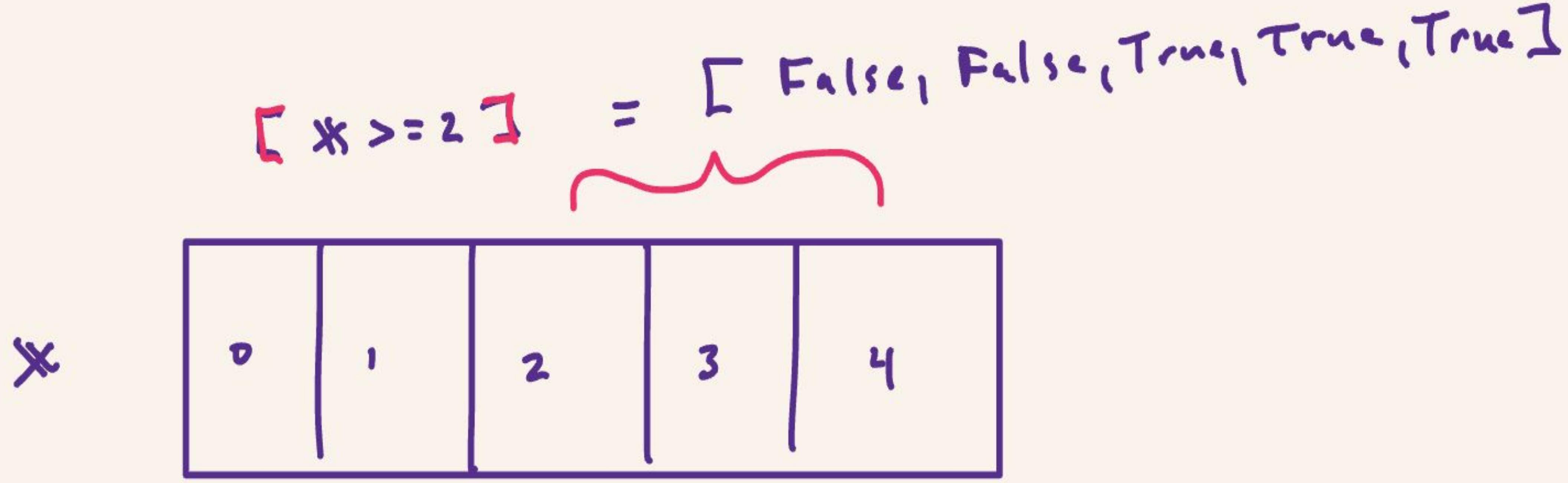
AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

Comparisons, Booleans, Masks

- Booleans
  - datatype of true or false, 0 and 1’s
  - OG datatype in computer science
  - written True or False in python
- Comparisons
  - functions returning booleans
  - in NumPy
- Masks
  - using booleans and ufuncs to builds indexes

AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

Comparison Statement and Mask



np.any, np.any, and masks

- element-by-element
- any of them
- all of them
- subset arrays with boolean expressions

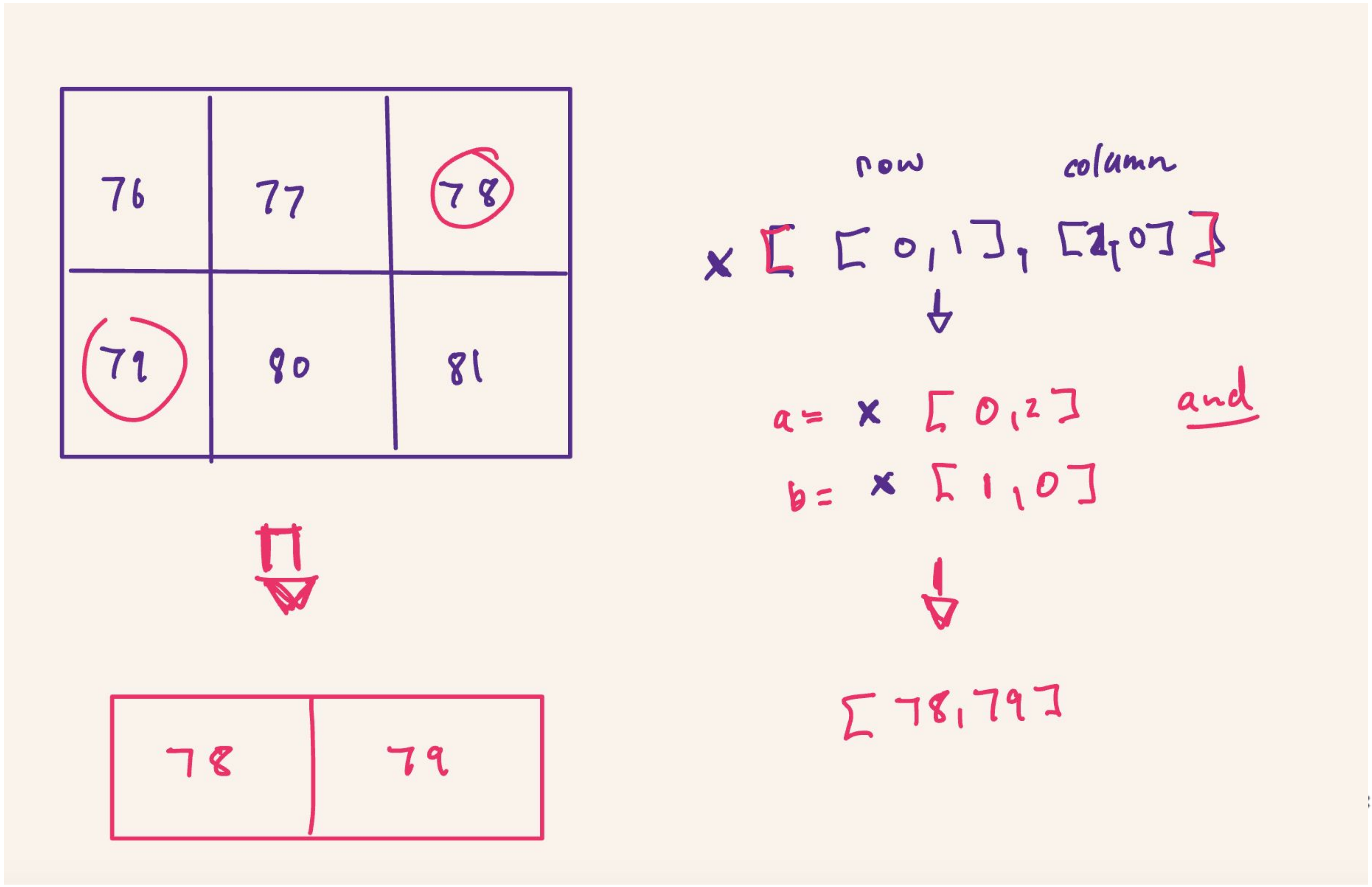
AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

Fancy Indexing

- Pass an array of indices to select multiple elements
- With fancy indexing, the resultant array's shape reflects the index array as opposed to the array being indexed

AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

One and Multi-Dimensional Fancy



Exercise #5: Comparison and Fancy

Create a random array x of length 150 with integers between 0 and 100

Produce a boolean array by checking which elements are less than 54

Use that array to index the random array x

reshape that array into (3,50) matrix

Use fancy indexing to pick out some elements from it

Numba

According to Numba’s documentation:

“ Numba is a compiler for Python array and numerical functions that gives you the power to speed up your applications with high performance functions written directly in Python.

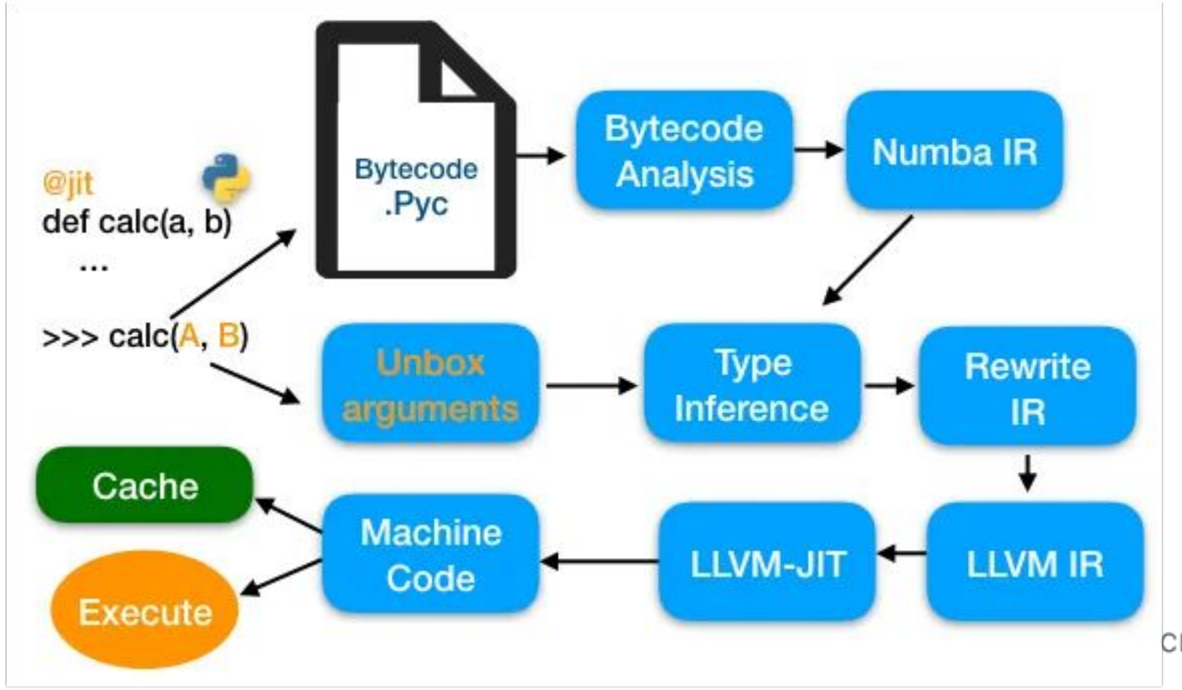
Numba generates optimized machine code from pure Python code using the LLVM compiler infrastructure. With a few simple annotations, array-oriented and math-heavy Python code can be just-in-time optimized to performance similar as C, C++ and Fortran, without having to switch languages or Python interpreters.

Numba’s main features are:

- on-the-fly code generation (at import time or runtime, at the user’s preference)
- native code generation for the CPU (default) and GPU hardware
- integration with the Python scientific software stack (thanks to Numpy)

LLVM and Optimized Code

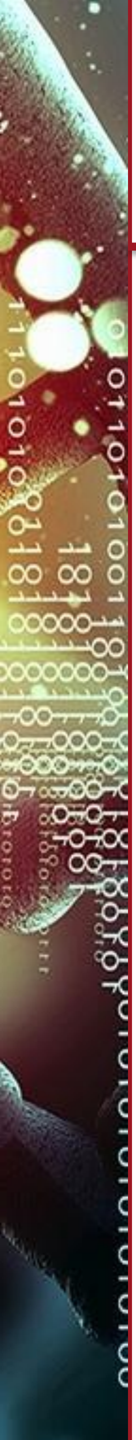
- “ **LLVM** is a set of compiler and toolchain technologies[4] that can be used to develop a frontend for any programming language and a backend for any instruction set architecture.”



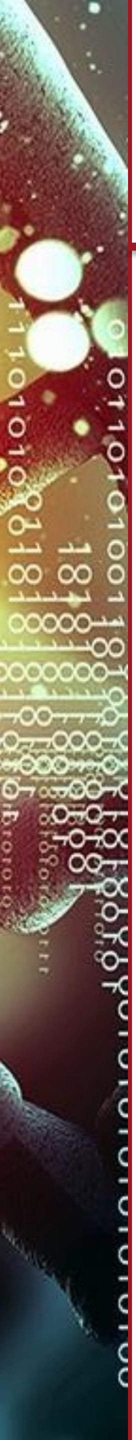
CE

Just-in-Time Compilation (JIT)

- In a compiled language, a program is compiled into machine-code before it is run
  - distinction in process between compilation versus run-time
- JIT “combines the two processes”, it compiles code during run-time
  - human-interactivity, flexibility, and ease of interpreter
  - efficiency and performance of compiled language



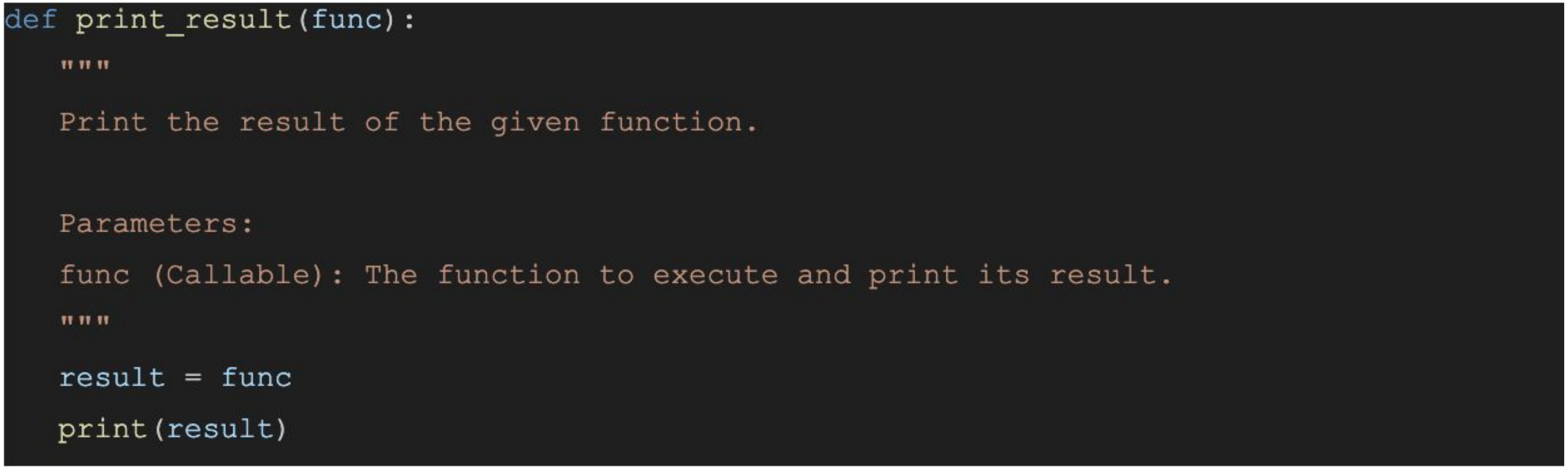
AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE



Python as Functional Language

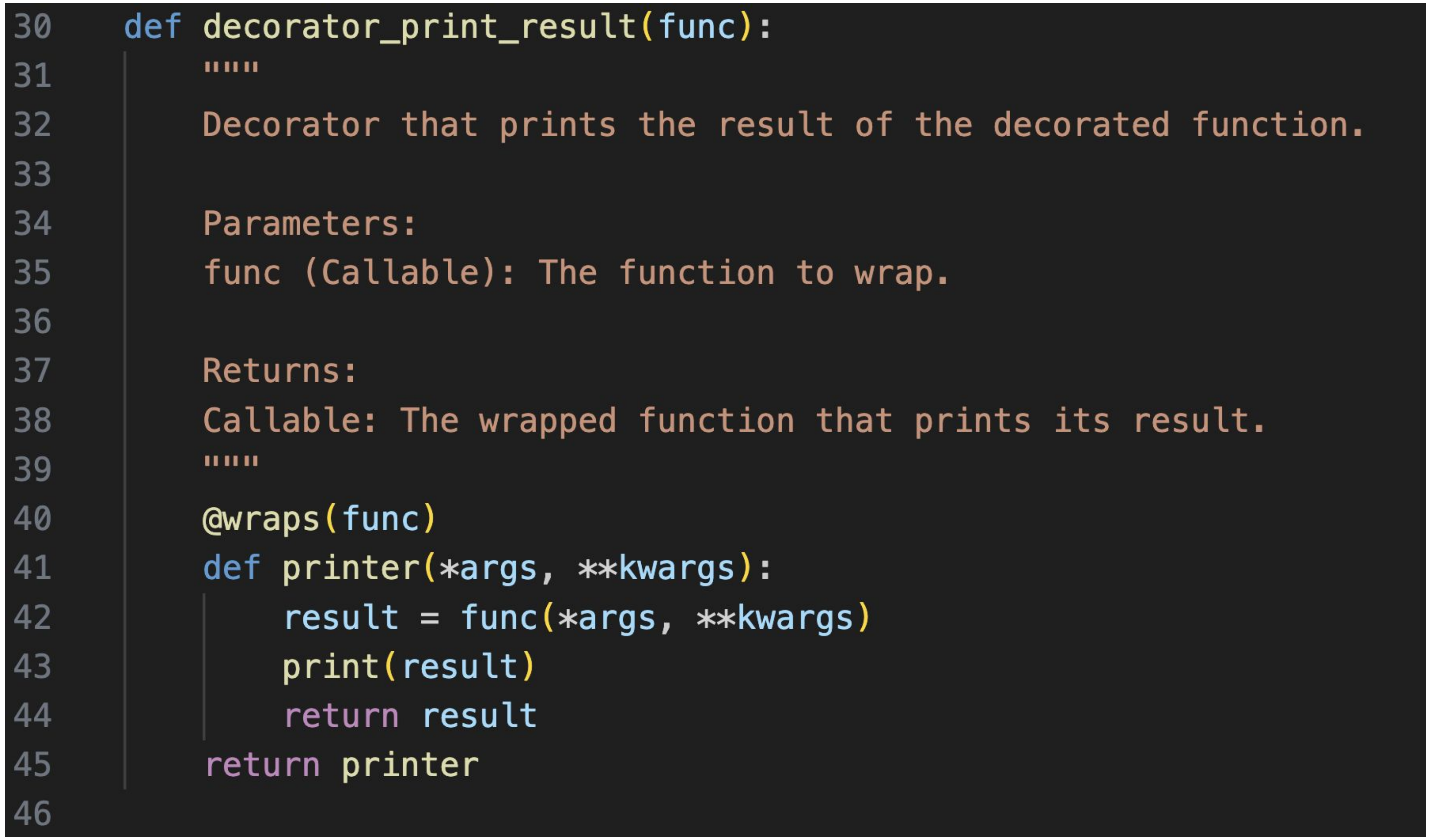
- Functional Languages
    - functions as first class citizens
- Decorator as second-order functions
    - modify behavior of function
- Decorator as “syntactic-sugar”
    - “syntactic-sugar” means an expression that simplifies a more complex expression

Example : **print_result**



```python
def print_result(func):
    """
    Print the result of the given function.

    Parameters:
    func (Callable): The function to execute and print its result.
    """
    result = func
    print(result)
```

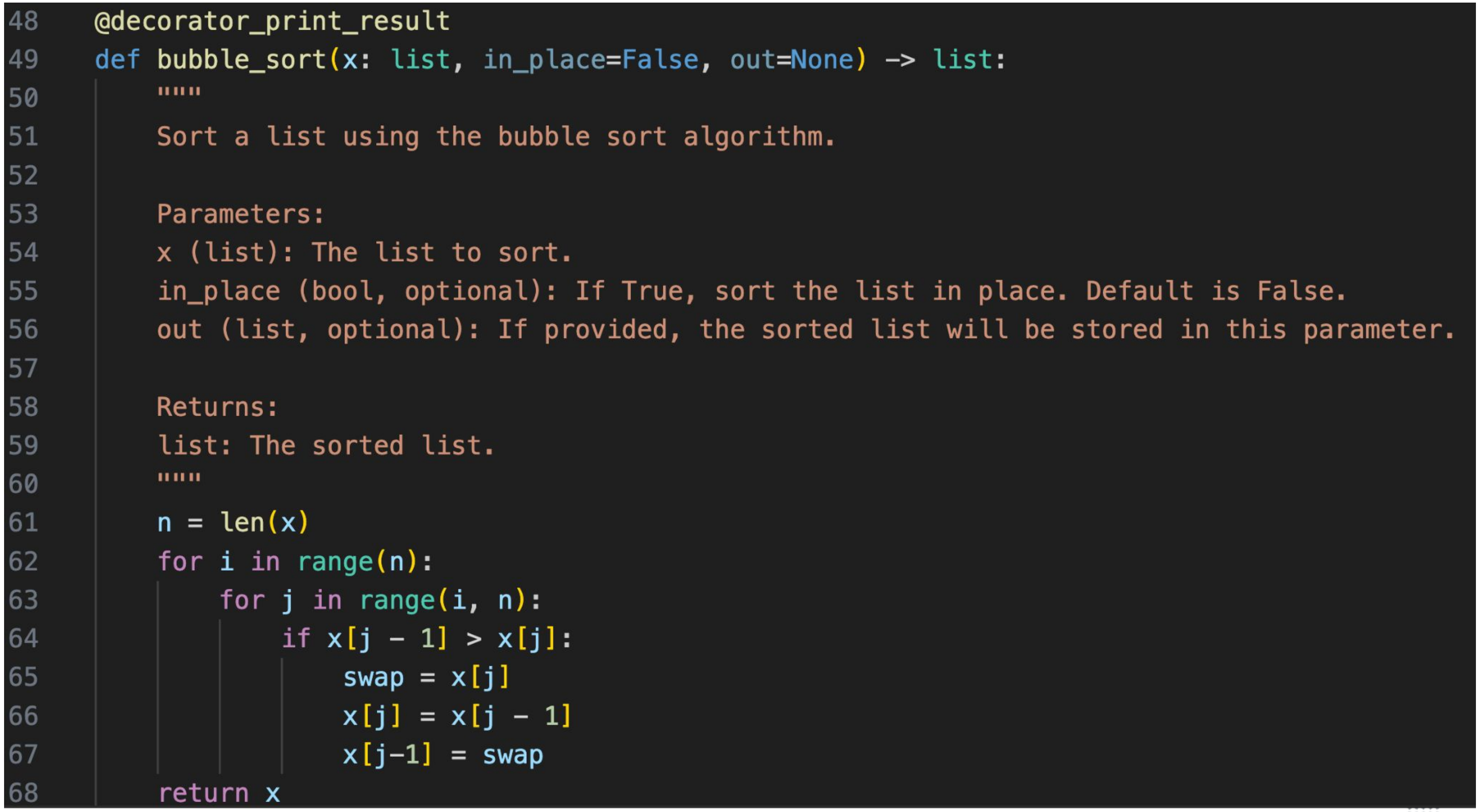
@print_result



```python
def decorator_print_result(func):
    """
    Decorator that prints the result of the decorated function.

    Parameters:
    func (Callable): The function to wrap.

    Returns:
    Callable: The wrapped function that prints its result.
    """
    @wraps(func)
    def printer(*args, **kwargs):
        result = func(*args, **kwargs)
        print(result)
        return result
    return printer

```

AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

Modifying function



```python
@decorator_print_result
def bubble_sort(x: list, in_place=False, out=None) -> list:
    """
    Sort a list using the bubble sort algorithm.

    Parameters:
    x (list): The list to sort.
    in_place (bool, optional): If True, sort the list in place. Default is False.
    out (list, optional): If provided, the sorted list will be stored in this parameter.

    Returns:
    list: The sorted list.
    """
    n = len(x)
    for i in range(n):
        for j in range(i, n):
            if x[j - 1] > x[j]:
                swap = x[j]
                x[j] = x[j - 1]
                x[j-1] = swap
    return x
```

AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

Basics of **@jit**

- Lazy vrs. Eager Evaluation
  - run-time inference of types
  - explicit type signatures
    - therefore before run-time

AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

Signatures

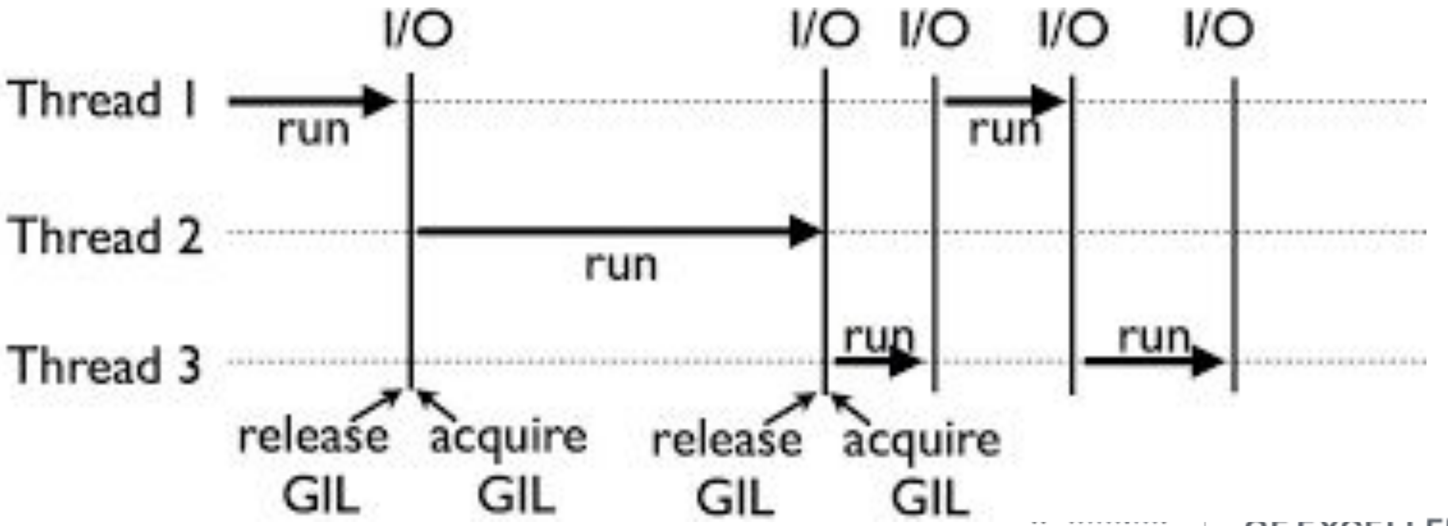
- Function signature is an expression, written in some kind of language, that tells us
  - what inputs go into a function
  - outputs come out of the function
- In Numba has form
  - **`output_type(input_type1, input_type2, …input_type_n)`**
  - `An array is simply some type followed by [:]`
- `Some examples :`

```
(float32[:](float32[:],float32[:],float32[:]))

(int32[:](float32,float32[:],float32[:]))
```

AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

Some Options for **@jit**

- nopython
  - nopython versus object mode
- nogil
  - locking mechanism (mutex)
- nocache
- parallel



Conditions for **parallel=True**

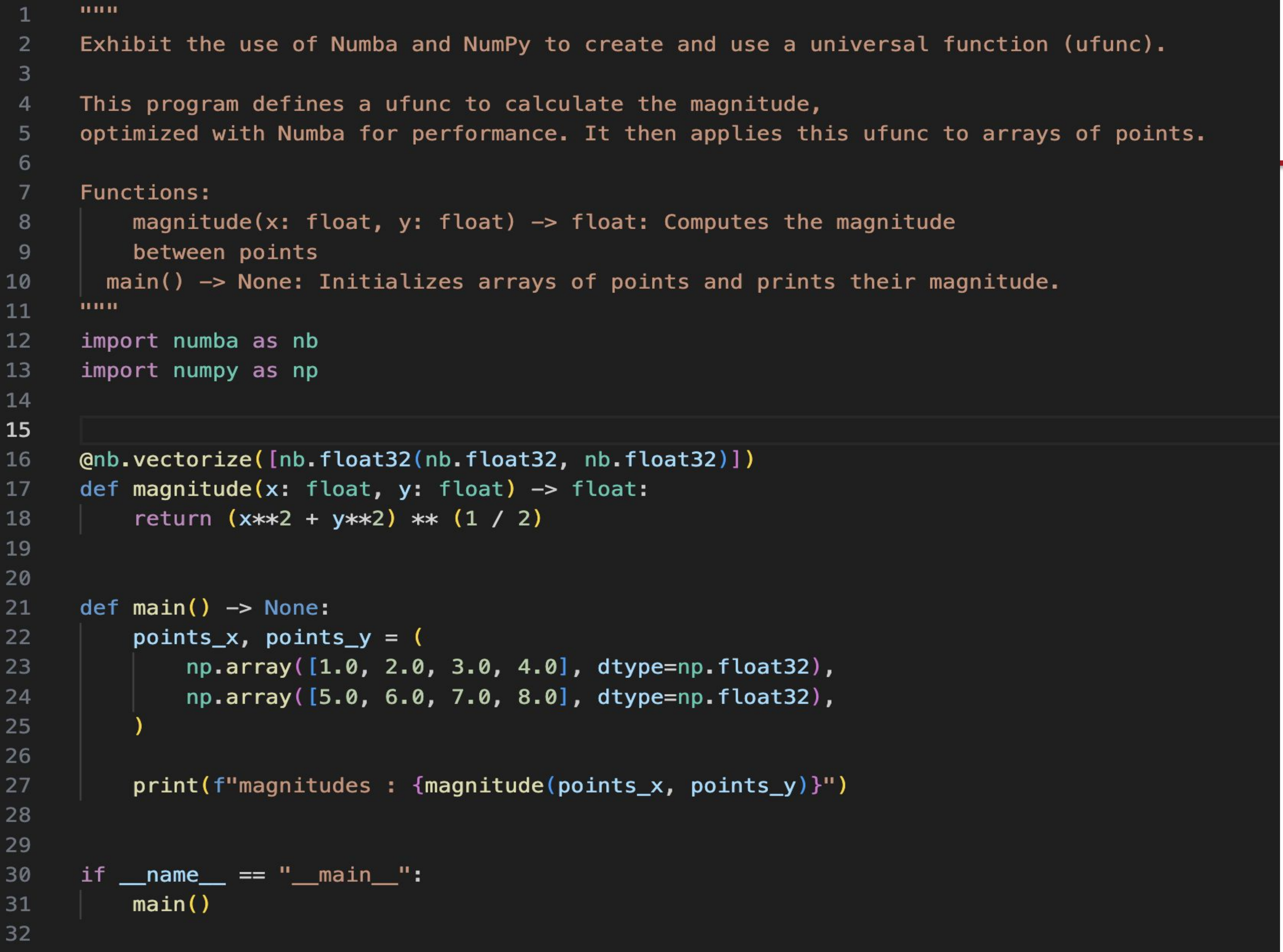
1. Common arithmetic functions between Numpy arrays, and between arrays and scalars, as well as Numpy ufuncs. They are often called element-wise or point-wise array operations e.g.:
   - unary operators: + - ~
   - binary operators: + - * / /? % | >> ^ << & ** //
   - comparison operators: == != < <= > >=
2. Numpy reduction functions sum, prod, min, max, argmin, and argmax.
3. array math functions mean, var, and std.
4. Numpy array creation functions zeros, ones, arange, linspace, and several random functions
5. Numpy dot function between a matrix and a vector, or two vectors
6. Array assignment in which the target is an array selection using a slice or a boolean array, and the value being assigned is either a scalar or another selection where the slice range or bitarray are inferred to be compatible.

AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

Numba and Ufuncs

- **vectorize** and **guvectorize**
  - **vectorize** takes functions that take vectors of same shape
  - **guvectorize** takes functions of different shape, takes functions of vectors of vectors
- Lazy and Eager mode
  - eager -> **ufunc**
  - lazy -> **DUfunc**
- Signature passed as list
- Less precise to more precise signature

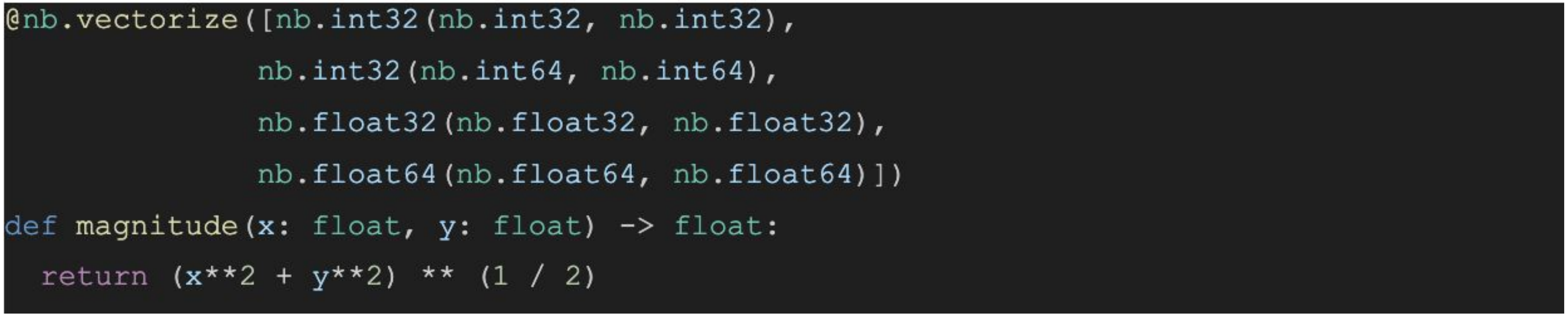
AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE



```python
"""
Exhibit the use of Numba and NumPy to create and use a universal function (ufunc).

This program defines a ufunc to calculate the magnitude,
optimized with Numba for performance. It then applies this ufunc to arrays of points.

Functions:
    magnitude(x: float, y: float) -> float: Computes the magnitude
    between points
  main() -> None: Initializes arrays of points and prints their magnitude.
"""
import numba as nb
import numpy as np



@nb.vectorize([nb.float32(nb.float32, nb.float32)])
def magnitude(x: float, y: float) -> float:
    return (x**2 + y**2) ** (1 / 2)


def main() -> None:
    points_x, points_y = (
        np.array([1.0, 2.0, 3.0, 4.0], dtype=np.float32),
        np.array([5.0, 6.0, 7.0, 8.0], dtype=np.float32),
    )

    print(f"magnitudes : {magnitude(points_x, points_y)}")


if __name__ == "__main__":
    main()
```

AFRICAN
CENTERS
OF EXCELLENCE
IN BIOINFORMATICS &
DATA-INTENSIVE SCIENCE

Less precise to more precise



```python
@nb.vectorize([nb.int32(nb.int32, nb.int32),
               nb.int32(nb.int64, nb.int64),
               nb.float32(nb.float32, nb.float32),
               nb.float64(nb.float64, nb.float64)])
def magnitude(x: float, y: float) -> float:
  return (x**2 + y**2) ** (1 / 2)
```

AFRICAN
CENTERS
OF EXCELLENCE
IN BIOINFORMATICS &
DATA-INTENSIVE SCIENCE

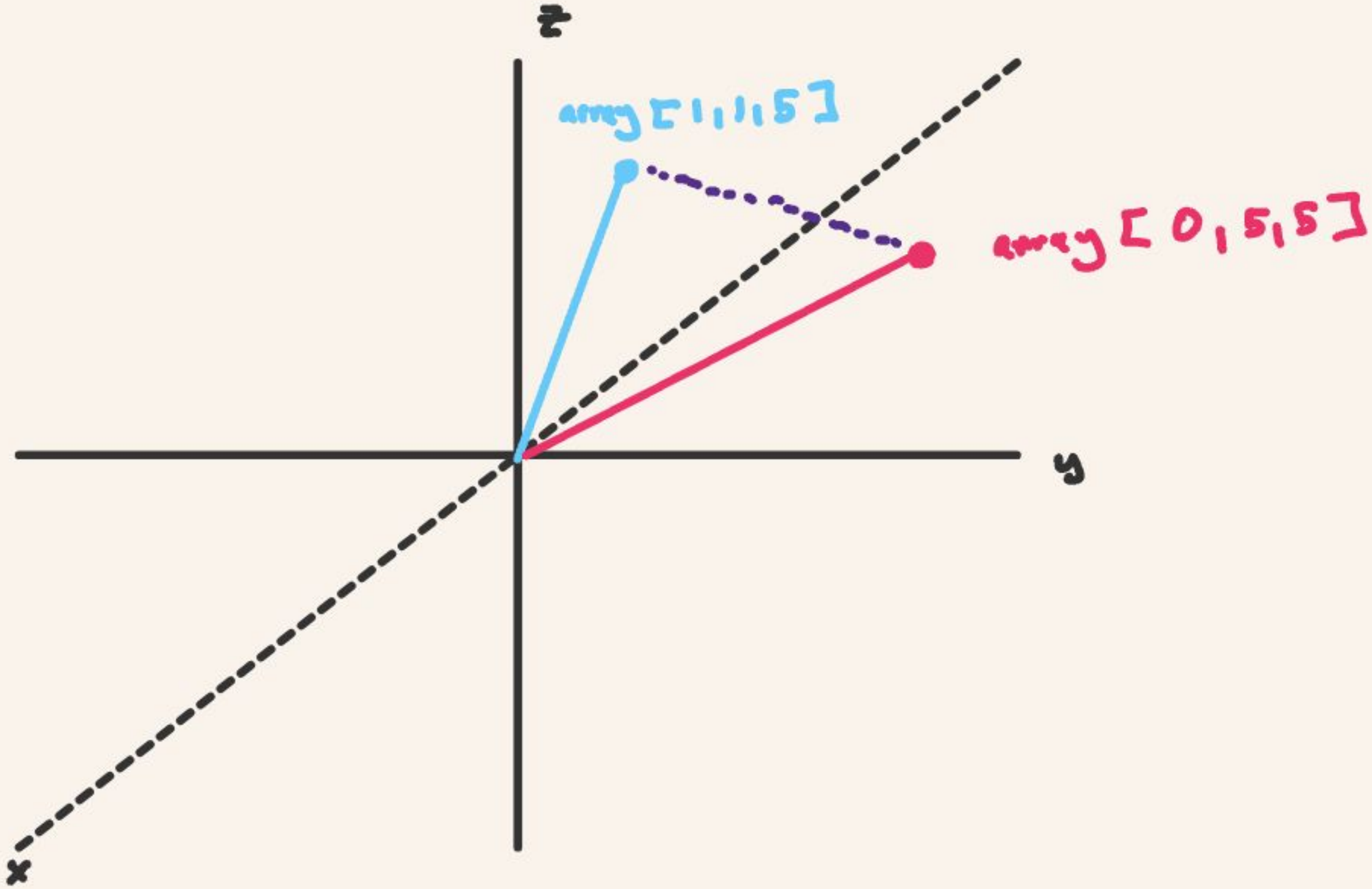
Why **@vectorize** or **@guvectorize**

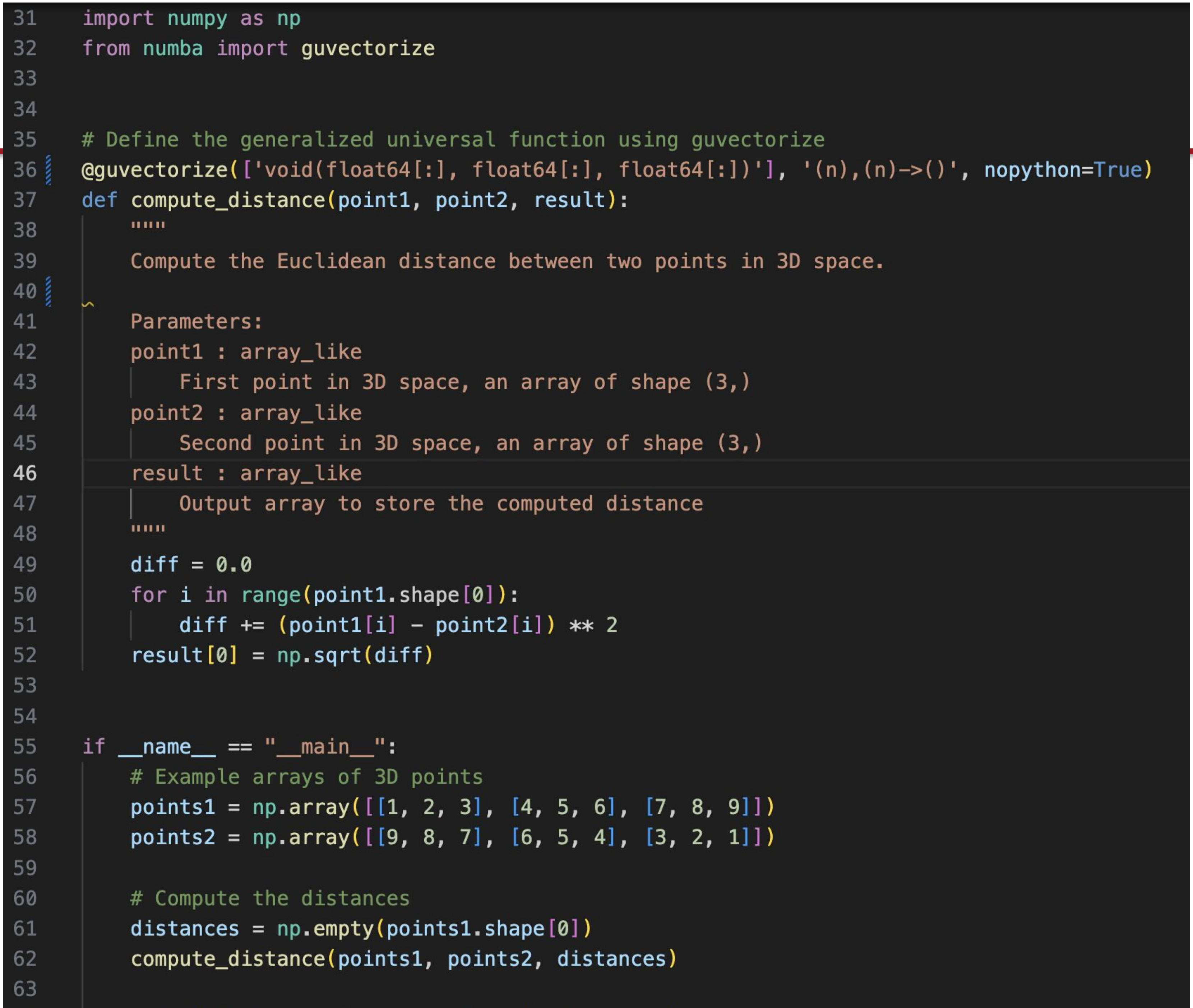
- Well because we get to leverage the features , attributes, and methods of a **ufunc**
- methods: reductions and aggregations e.g. , **reduce** and **mean**.
- attributes: dimension number, or the order (c or fortran ordered)

@guvectorize

- Takes n-dimensional and m-dimensional array -> k-dimensional array
- example : find distance between two points in 3-d space
    - function takes two array of arrays x and y, each array within the array x or y represents a point in 3-d space, find distance between x[i] and y[i] for all i

AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE


z
array [1,1,5]
array [0,5,5]
y
x



```python
import numpy as np
from numba import guvectorize


# Define the generalized universal function using guvectorize
@guvectorize(['void(float64[:], float64[:], float64[:])'], '(n),(n)->()', nopython=True)
def compute_distance(point1, point2, result):
    """
    Compute the Euclidean distance between two points in 3D space.

    Parameters:
    point1 : array_like
        First point in 3D space, an array of shape (3,)
    point2 : array_like
        Second point in 3D space, an array of shape (3,)
    result : array_like
        Output array to store the computed distance
    """
    diff = 0.0
    for i in range(point1.shape[0]):
        diff += (point1[i] - point2[i]) ** 2
    result[0] = np.sqrt(diff)


if __name__ == "__main__":
    # Example arrays of 3D points
    points1 = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
    points2 = np.array([[9, 8, 7], [6, 5, 4], [3, 2, 1]])

    # Compute the distances
    distances = np.empty(points1.shape[0])
    compute_distance(points1, points2, distances)

```

AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

Extra-Credit Exercise

produce a compute_difference_vector

AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

Performance through Monte Carlo

- calculate ***π*** with dart-board
- monte-carlo algorithm

<u>pseudo-code : calculate pi</u>

```
area = 4.0
for i in  number_samples:
    x = random()
    y = random()
    if $\sqrt{(x^2+y^2)} < 1$ :
        hit = hit+1
return area * hit/number_samples
```

AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

Speedup-resume

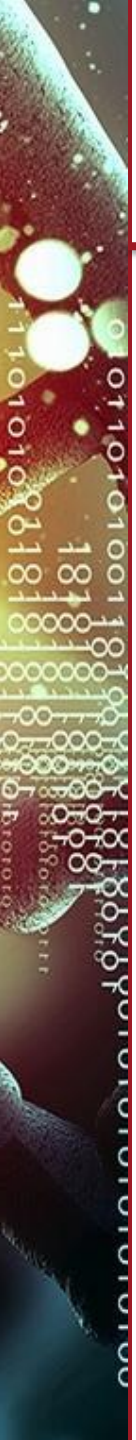
- **monte_carlo_pi :** baseline
- **jitted_monte_carlo** makes use of numba in eager-mode: 100x speedup
- **less_precision_jitted_monte_carlo** : 100x speedup
- **fast_math_jitted_monte_carlo** which is an option at the compiler level that may make our calculations less precise, degrades precision
- **fast_math_flags_jitted_monte_carlo** next makes use of LLVM-flags for fast math operations, we can restrict which mathematical operations actually get “fastmathed”.
- **parallel_jitted_monte_carlo**, which makes use of **parallel** or automatic parallelization.
- We have sped up our program by almost 500x!

AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

Some lessons from today

1. Use NumPy arrays
2. Make use of numpy ufuncs
3. Use Numba
4. Use **nopython** mode
5. Use *only* amount of precision needed
6. Use **fastmath** compiler optimization
7. Parallelize when possible

AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE



Conclusion:

- We did it!
- That was a lot good job.
- Tomorrow we parallelize!!!!

AFRICAN CENTERS OF EXCELLENCE IN BIOINFORMATICS & DATA-INTENSIVE SCIENCE

Exercise #4:

Tell me what this signature means:

```
(int64(int64,int64))
```

Code *any* **@jit** function that follows this signature.

Create a **@vectorize** version of such a function , use the .reduce() function on it.