

Advanced
computer usage

Writing code

Running code on
supercomputers



Command
Line Linux



Python

High Performance
Computing



COMPASS



Overleaf



Git & GitHub

Developing software



Python
Packages

Data analysis
& visualization

Writing up
results

Today's Schedule

Monday

10am-10:30am: Introductions & Pre-workshop survey

10:30am-noon: Intro to Python & Hands-on Experience

noon-1pm: Lunch (provided)

1pm-2pm: Python Hands-on experience

Preworkshop Survey

<https://tinyurl.com/compasspreworkshopsurvey>

Our website: <https://www.vikrammanikantan.com/compass.html>

Python Fundamentals

COMPASS Workshop

```
Instructors = ["Vikram Manikantan",  
               "Erik Wessel",  
               "Vasilis Paschalidis"]
```

Goals

- Become familiar with the basics of python programming
- Begin to write code on your own

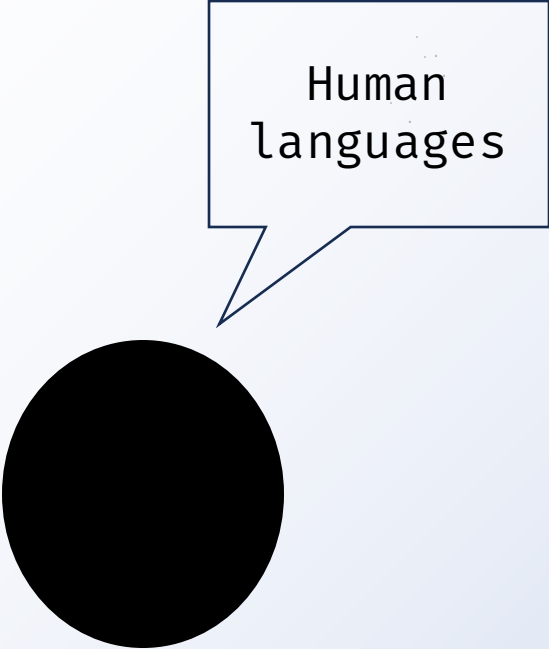
Goals

- Become familiar with the basics of python programming
- Begin to write code on your own

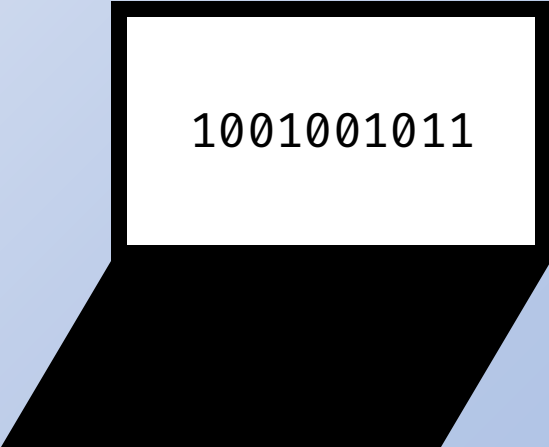
Contents

Print statements	.1
Variables & Data types	.2
Math with Python	.3
Conditional Programming	.4
Arrays	.5
Loops	.6
Functions	.7

What is a programming language?



Human
languages



1001001011

Communicating with computers

Human
languages



```
00000000 C2 30 REP #30
00000001 18 CLC
00000002 F8 SED
00000003 A9 34 12 LDA #1234
00000004 69 21 43 ADC #4321
00000005 8F 03 7F 01 STA $017F03
00000006 D8 CLD
00000007 E2 30 SEP #30
00000008 00 BRK
00000009 2012

r PB PC NUMxDI2C .A .X .Y SP DP DB
: 00 E012 00110000 0000 0000 0002 CFFF 0000 00
g 2000

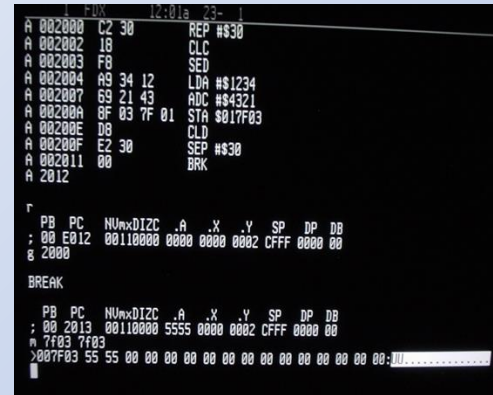
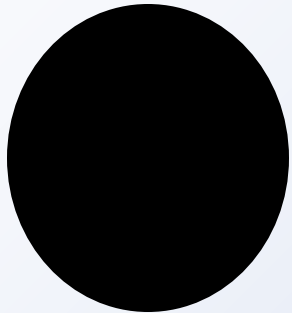
BREAK

PB PC NUMxDI2C .A .X .Y SP DP DB
: 00 2013 00110000 5555 0000 0002 CFFF 0000 00
m 7F03 7F03
>007F03 55 55 00 00 00 00 00 00 00 00 00 00 00 00 00 00:00
```

1001001011

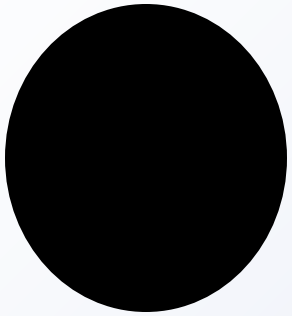
Languages range in difficulty

Human languages

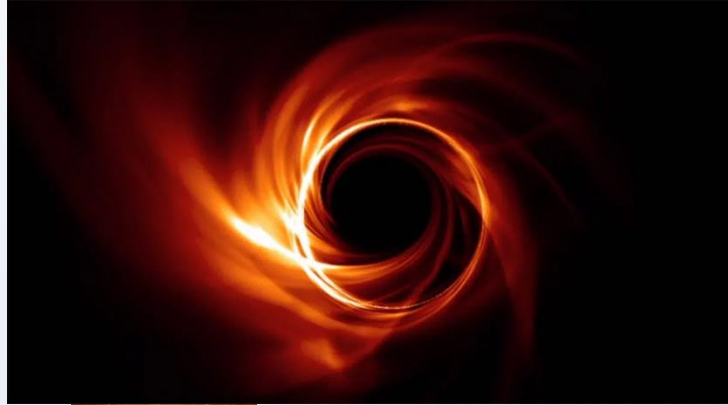


1001001011

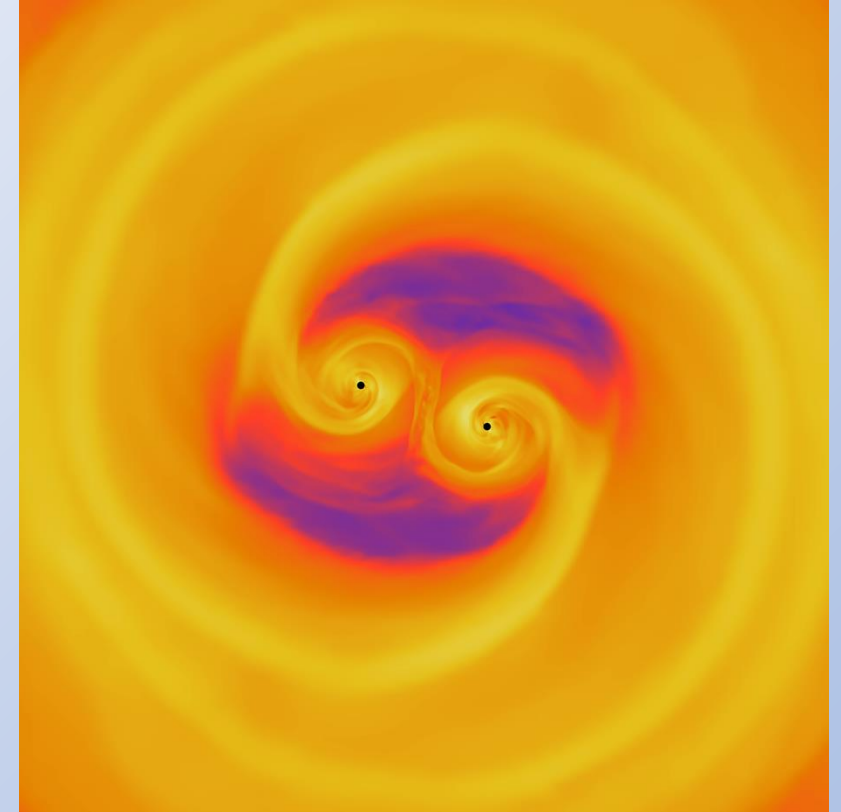
Humans + Computers = Infinite Possibilities



1001001011



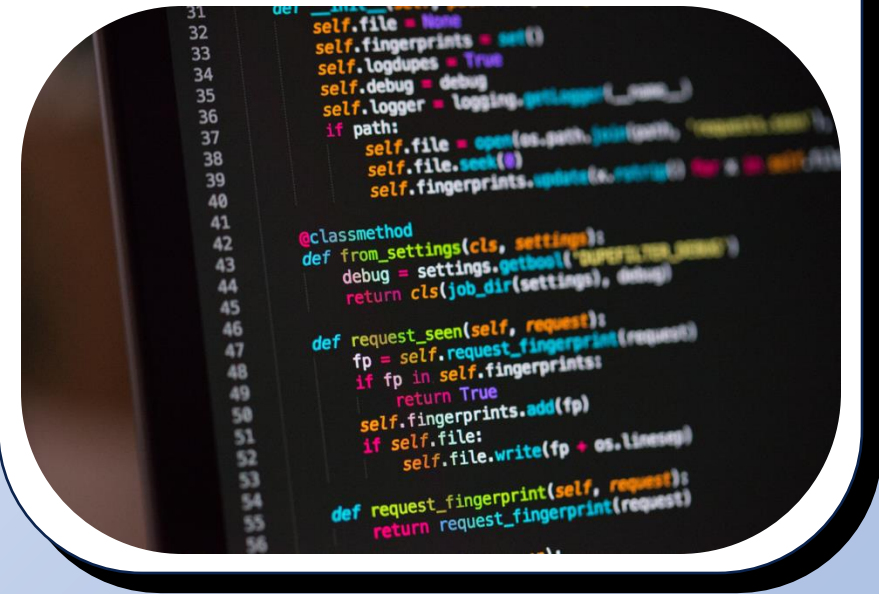
Credits to CK Chan, who will be talking about his work on Thursday!



One of my simulations

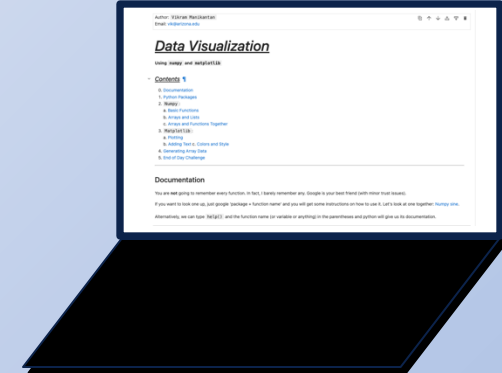
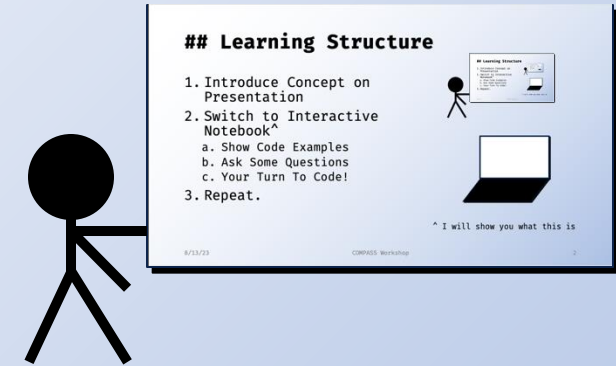
What is Python?

1. Programming Language
2. High-Level (human readable)
3. General Purpose (multi-functional)



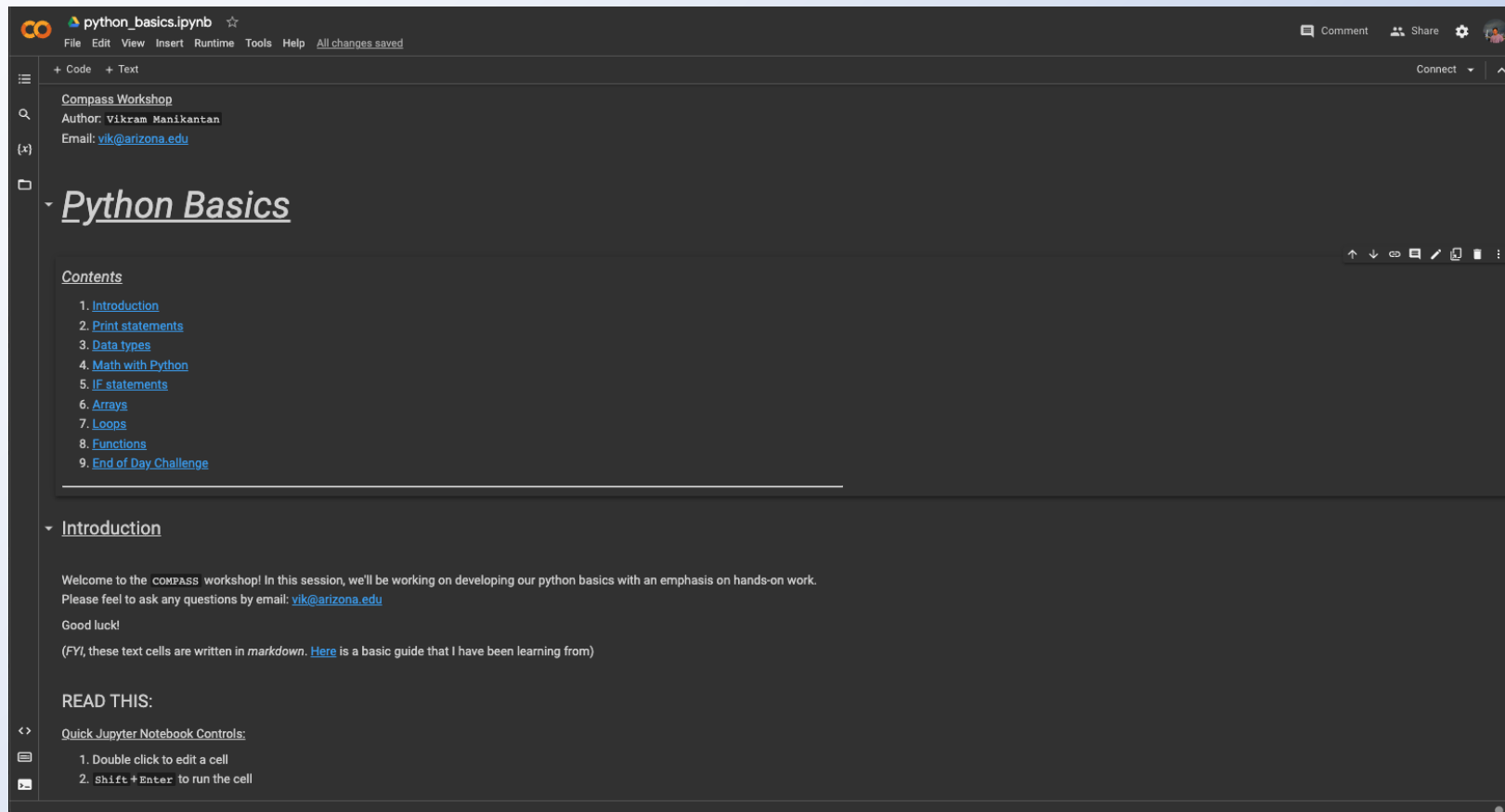
Session Structure

1. Introduce Concept on Presentation
2. Switch to Code
 - a. Show Code Examples
 - b. Ask Some Questions
 - c. Your Turn To Code!
3. Repeat.



Collaboratory Notebook

tinyurl.com/compasspython



Print Statements: in Python

- Why are print statements important:
 - Understanding Errors
 - Communicating to User
 - Understanding Your Code
- Python syntax:
`print("output here")`

```
>> python3
Python 3.10 on Darwin...
>>>
>>> print("Hello, World!")
Hello World!
>>> print("Bye, World.")
Bye, World.
>>> exit()
```

Print Statements: in Python

Your turn:

Print Statements

Print statements are essential when coding. But figuring out what you can and cannot print is not always straightforward.

Example

```
[ ]: ###  
# print the phrase: 'hello world'  
###  
  
print('hello world')
```

Questions

```
[ ]: ###  
# Q. would you expect a difference between these two statements?  
###  
  
print('2')  
print(2)
```

```
[ ]: ###  
# Q. what do you think is going to happen here?  
###  
  
print(2 '2')
```

```
[ ]: ###  
# fix the above statement  
###  
  
""" YOUR CODE HERE """
```

```
[ ]: ###  
# Q. What goes wrong here?
```

Variables & Data Types

- Variable is a storage location with a name and contents
- Every variable has a data type that classifies what it's storing

```
>>> x = 5.0
>>> y = 10
>>> print("x = ", x)
x = 5.0
>>> print("y = ", y)
y = 10
```

Variables & Data Types

- Interactive notebook explanations and examples

What is a Data Type?

A *data type* classifies the data stored within *variables*. Our data can take on many types, here are some examples:

```
[1]: # integer, just like in math
print(type(1))

<class 'int'>

[2]: # float or double, what you might call a real number in math
print(type(1.1))

<class 'float'>

[3]: # string, any sequence of ASCII characters (including numbers)
print(type('abc 123'))

<class 'str'>

[4]: # boolean, True or False
print(type(True))

# there are many more you will get to know

<class 'bool'>
```

▼ Example ¶

Let's calculate the force on an object according to Newton's second law: $\vec{F} = m\vec{a}$

```
[5]: # the mass
m = 10 # kg

# acceleration
a = 9.81 # m/s^2
```

Basic Math

- Basic mathematic operations carry over
 - $+$, $-$, $/$, $*$, etc.
- See examples on the interactive notebook

```
>>> a = 5
>>> b = 10.0
>>> print("a x b = ", a*b)
a x b = 50.0
```


Basic Math

- Interactive Notebook:
 - Order of Operations
 - Solving the Quadratic Equation
- (I am going to stop reminding us to go to the notebook)

Question

Order of Operations

How do order of operations work in python? I am going to let you figure this one out by yourself!

```
[ ]: # some variables to get you started
a = 2
b = 3
c = 7

[ ]: # have a play around with these operations: +, -, *, /, **
# using each variable once, what's the craziest number you can get?

""" YOUR CODE HERE """
```

Challenge

Solve this quadratic equation:

$$3x^2 + 5x + 10 = 0$$

If you've forgotten (I don't blame you):

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Using what you have learnt about variables, types, operations and order of operations, you should be able to do this!

```
[ ]: """ YOUR CODE HERE """
```

Conditional Programming

- This is what makes programming powerful
- In words:
 - if statement is true, do this code,
 - if not, do this other code

```
>>> if x > 3:  
>>>     print("x is big!")  
>>> else:  
>>>     print("x is small")
```

Lunch Break

Arrays and Lists

- Storing collections of data under one name
- Variables are a single storage location.
- Arrays are many storage locations (near each other)
- ***Important:*** the count starts at 0

```
>>> fav_bhs = ["M87",  
               "Sagittarius A*", "OJ 287"]  
>>> print(fav_bhs[0])  
M87
```

Loops

- For repeating code!
- Syntax:
for some condition, do
this code repetitively
- In the simplest case,
counting from 0 to 9...
- ...but why just to 9?

```
>>>for counter in range(10):  
>>>     print(counter)  
0  
1  
2  
3  
...  
9
```


Functions

- Compartmentalization of your code
- Reusability, without copying the same code again, and again...
- ***Very important***

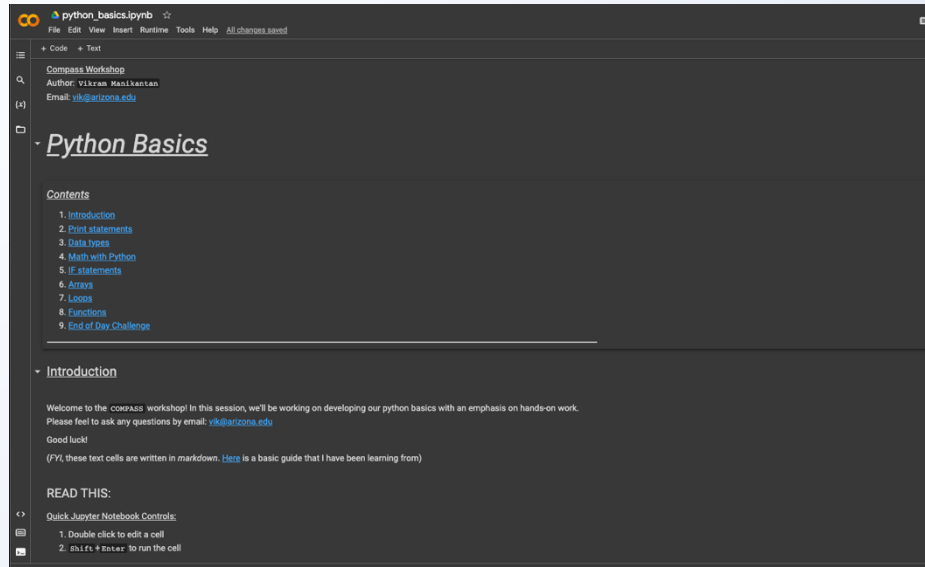
```
>>> def calc_force(m, a):  
>>>     f = m*a  
>>>     return f  
>>> ff=calc_force(10, 9.81)  
>>> print(ff)  
98.1000000000001
```


Variable Names

- ❑ Lowercase, uppercase, and underscores
- ❑ NO spaces, dashes, quotation marks, backslashes, etc.

```
>>> variableName  
>>> not a variable name  
>>> bad-variable-name  
>>> good_variable_name
```

and Open a Terminal



```
Last login: Mon Aug 14 21:52:02 on ttys001

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
clear
vik@mac compass (main) >> clear
```

Print Statements: in Shell

- Syntax:
Echo + “output”

```
>> echo 'Hello, World!'
Hello, World!
>>
```


Accessing Python from Terminal

- In terminal, type:
python (or python3)

```
>> python3  
Python 3.10 on Darwin...  
>>>
```