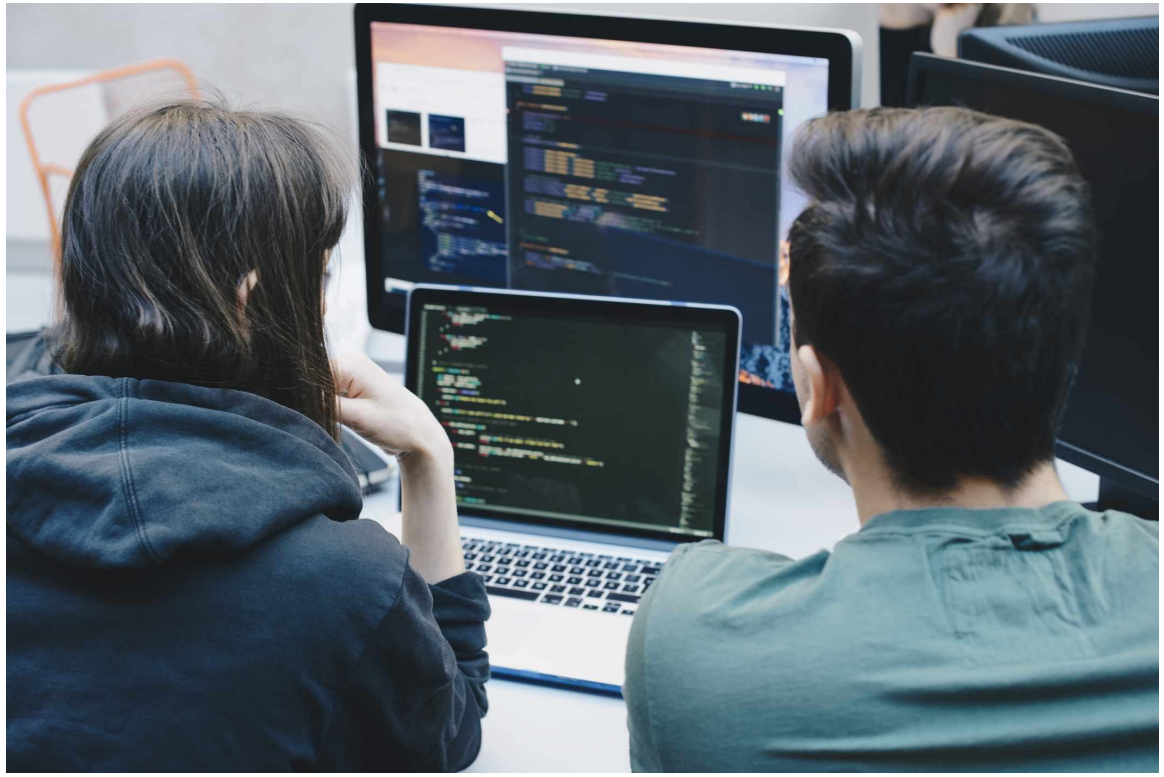
Intro to Programming I: Intro to Python

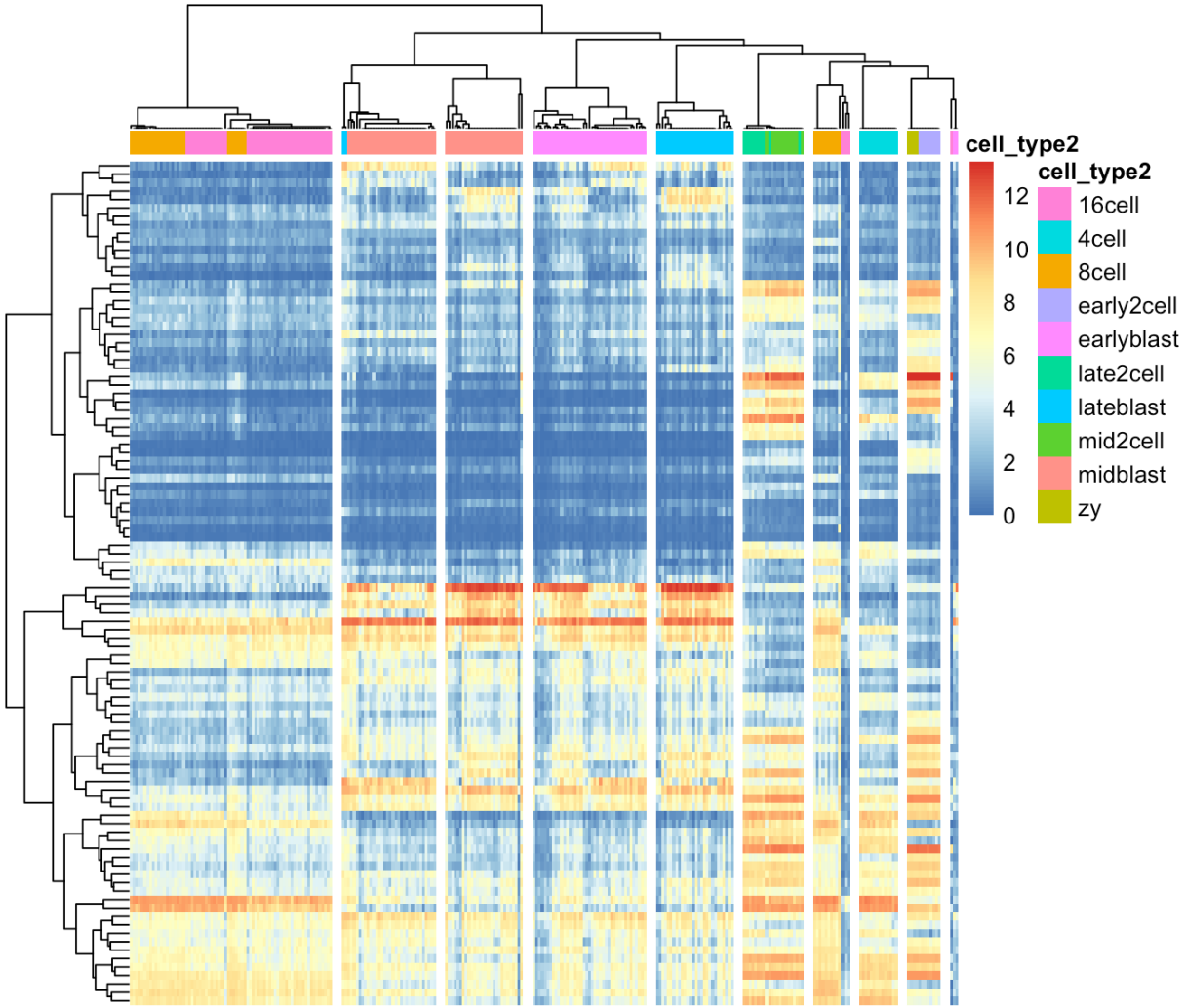
June 20th, 2024

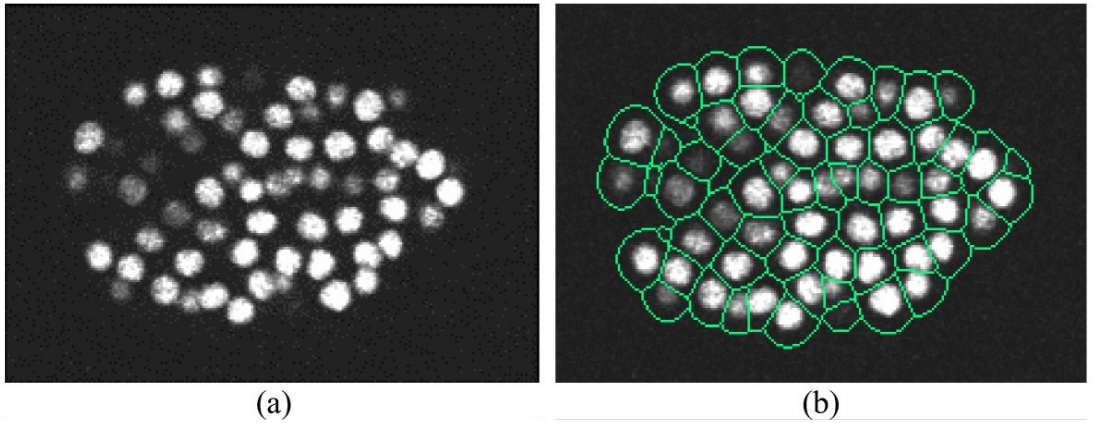
Motivation

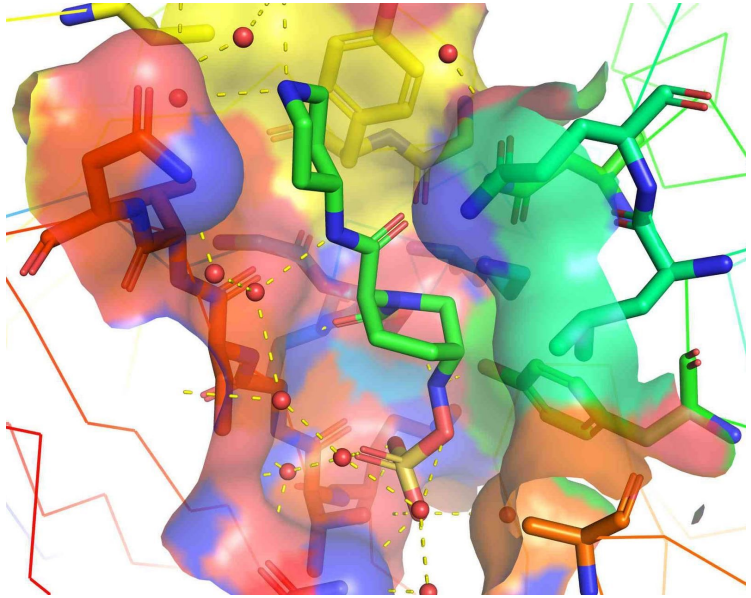
- **Computer Programming:** Creating a schedule of events for a computer to follow to complete a task.
- Why learn programming?
  - To become a programmer
  - To think computationally
  - To apply computational tools to problems



Motivation



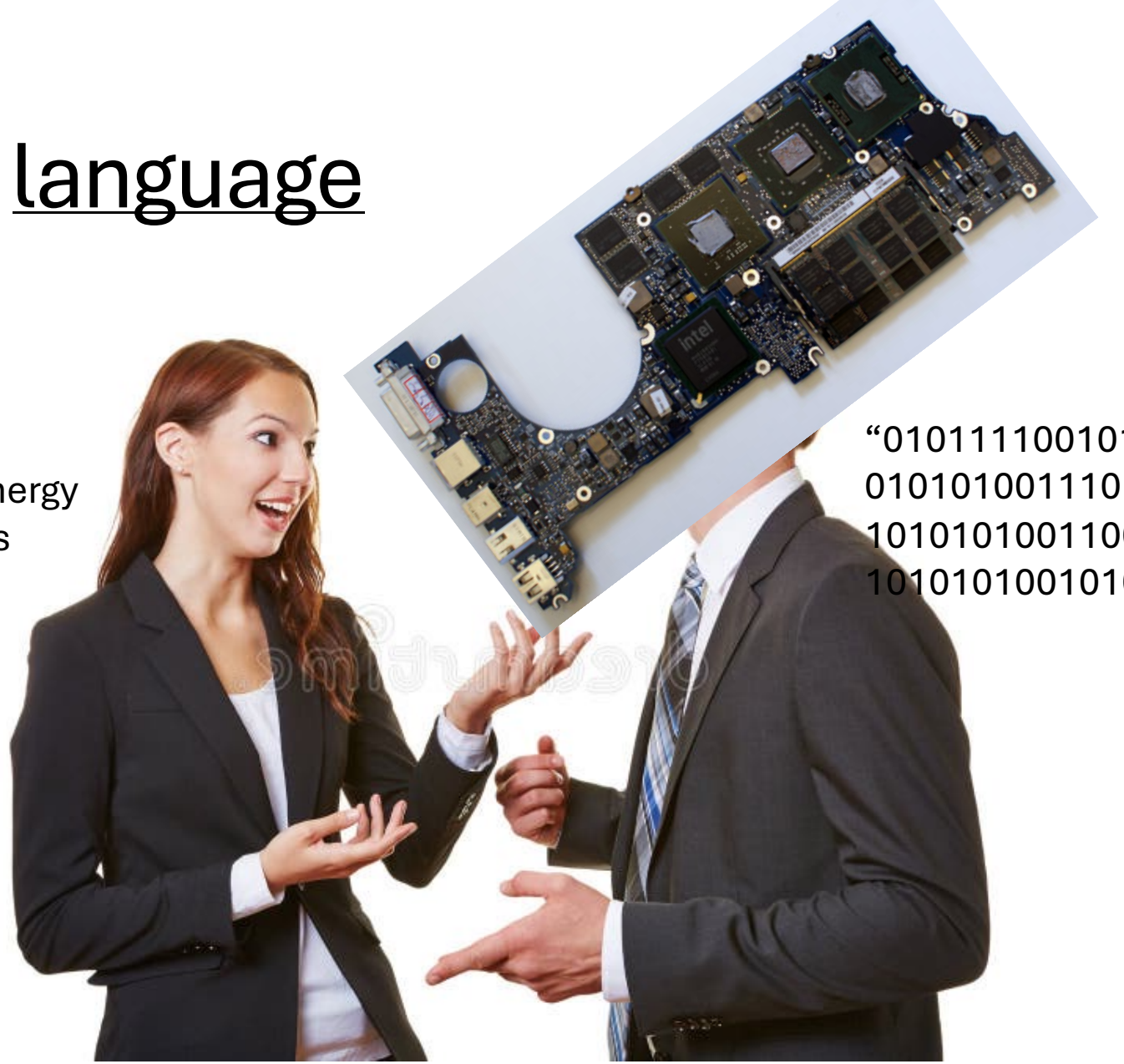




What do **you** use programming for?

Python is a language

“Calculate the energy of a protein in this conformation.”



“0101111001010100101001001010101010100111011001010111100101010101001100100110110100101011010100101001010100101001001”

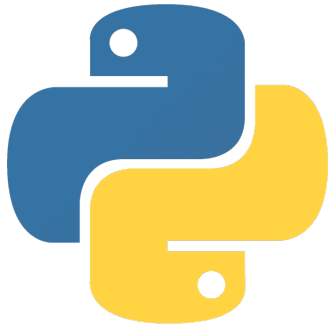
Python is a language

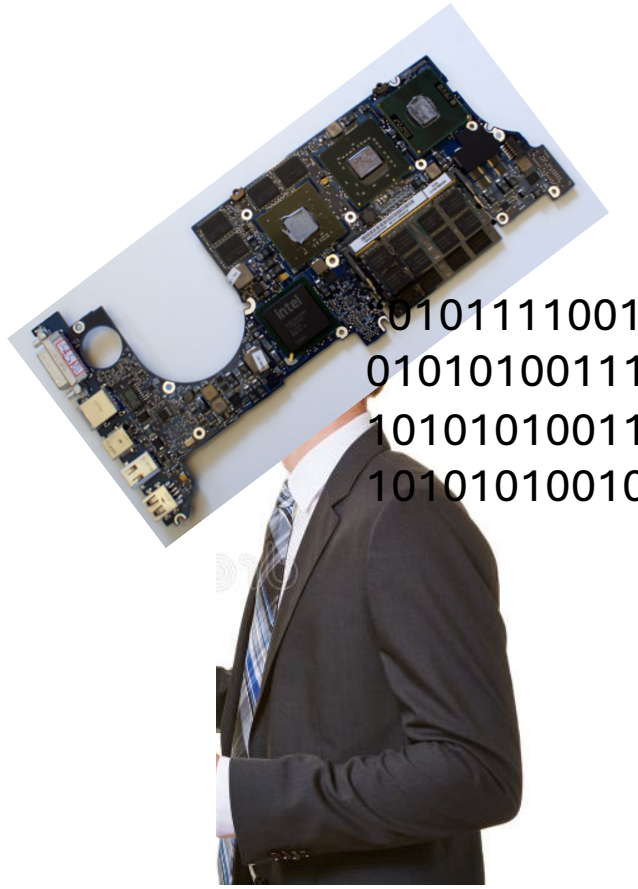
“Calculate the energy of a protein in this conformation.”

(You are here)

```
num1 = int(num1str)
num2 = int(num2str)

valid_operator = True
if operator == "+":
  result = num1 + num2
elif operator == "*":
  result = num1 * num2
elif operator == "-":
  result = num1 - num2
elif operator == "/":
  result = num1 / num2
else:
  valid_operator = False
```



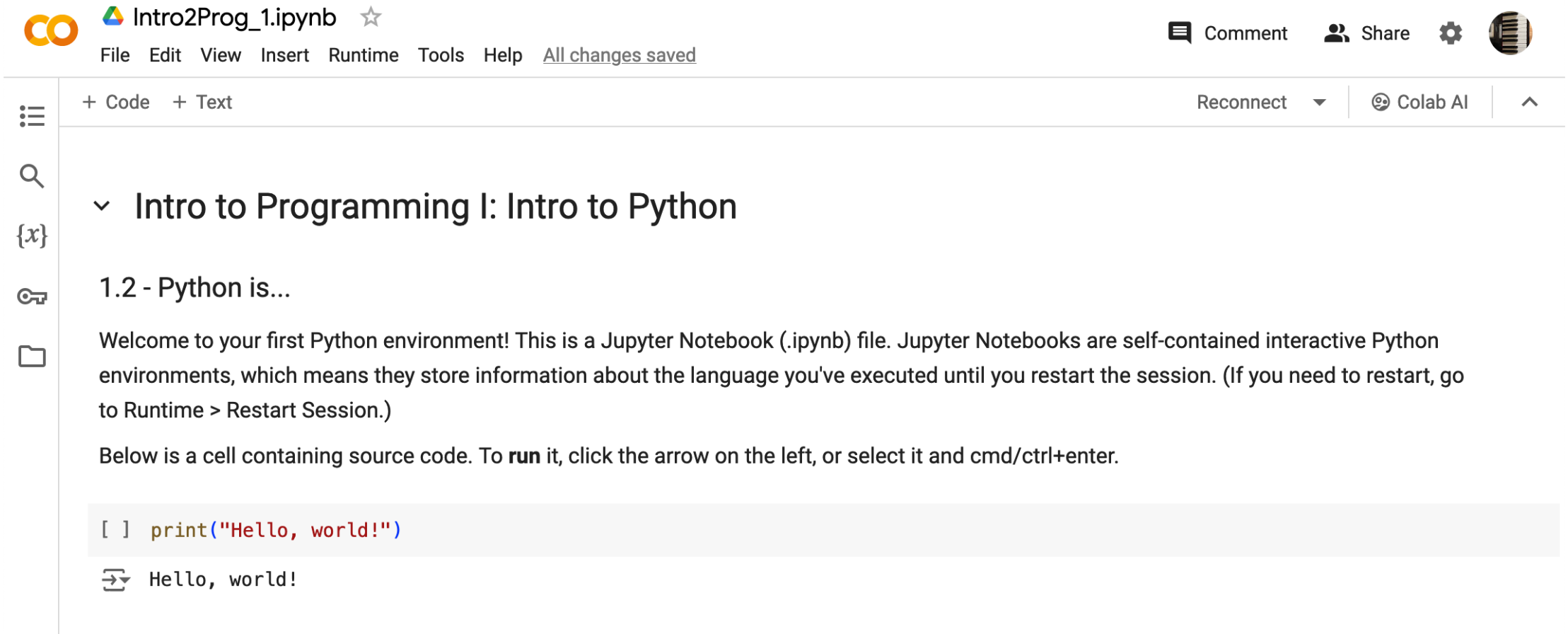


Human Language > **Programmer** > **Python** > **Python Interpreter** > **Machine Language**

+ easy to pick up
- slower than other languages

Let’s try Python!

- Colab link



Variables

```
>>> message = "What's up, Doc?"
>>> reply = ‘Not much’
>>> n = 17
>>> pi = 3.14159
>>> decide = True
>>> empty = None
```

Strings

Integers

Floating point

Boolean

NoneType

- Now we can fix our bug!

Operators

```
>>> print((2*6+3)/5-2)
>>> print(2**2,2**3,2**4)
>>> print(5 % 2)
>>> print("hello" + "sir")
>>> print(not (3 >= 0))
>>> print((3 > 0) and (3 < 0))
>>> print((3 == 0) or (3 < 0))
```

Arithmetic

Exponentiation

Modulus

String concatenation

Comparison and Logical

Functions

Function Definition

```
>>> def info(name, color="blue", age=34):
...     print(name, color, age)

>>> def distance(x1, y1, x2, y2):
...     dx = x2 - x1
...     dy = y2 - y1
...     return (dx**2 + dy**2)**0.5
```

Whitespace is significant!

Function Invocation

```
>>> info("Spunky")
```

Can you predict what this line of code will do?

In-Built Functions

Built-in Functions

A
abs()
aiter()
all()
any()
anext()
ascii()

B
bin()
bool()
breakpoint()
bytearray()
bytes()

C
callable()
chr()
classmethod()
compile()
complex()

D
delattr()
dict()
dir()
divmod()

E
enumerate()
eval()
exec()

F
filter()
float()
format()
frozenset()

G
getattr()
globals()

H
hasattr()
hash()
help()
hex()

I
id()
input()
int()
isinstance()
issubclass()
iter()

L
len()
list()
locals()

M
map()
max()
memoryview()
min()

N
next()

O
object()
oct()
open()
ord()

P
pow()
print()
property()

R
range()
repr()
reversed()
round()

S
set()
setattr()
slice()
sorted()
staticmethod()
str()
sum()
super()

T
tuple()
type()

V
vars()

Z
zip()

_
__import__()

Modules

```
import math

print (math.pi)
print (math.factorial(5))
print (math.cos(0))
print (math.log10(100))
```

Python has a way to put definitions in a file and use them in a script or in an interactive instance of the interpreter. Such a file is called a ***module****; definitions from a module can be imported into other modules or into the main module (the collection of variables that you have access to in a script executed at the top level and in calculator mode).*

Turtles!!!

- Back to colab!

```
import turtle
turtle.forward(100)
turtle.right(90)
turtle.backward(100)
```

```
from turtle import *
forward(100)
right(90)
backward(100)
```

```
help(turtle.forward)
```