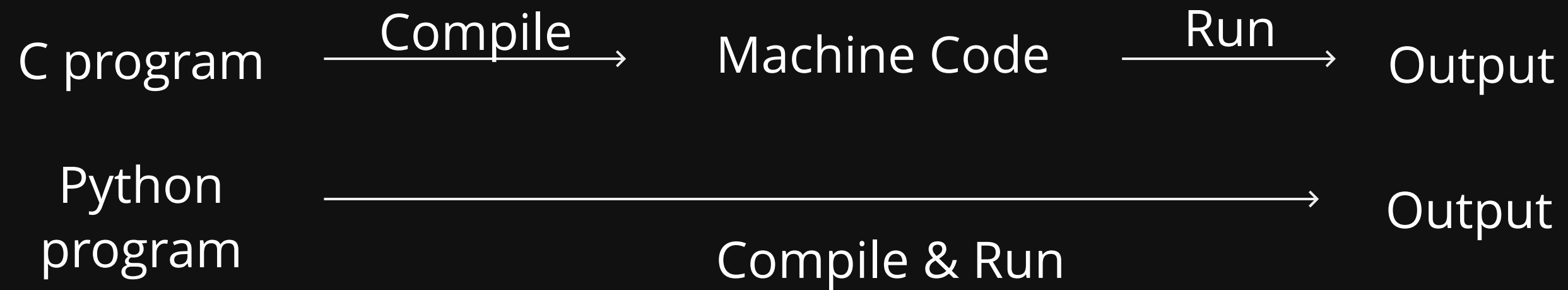


# COMP1531

## 1.4 - Python - Intro

# Python

**Interpreted V compiled**



# Python

Unlike C, Python is a very broad language with a rich range of libraries.

It's simply not possible for us to give you 100 lecture slides to teach you everything you need to know. For many of you, this will be the first course that you have to learn to take the basics from us, and then engage in self-learning to refine your knowledge of the language and its uses.

# Python

## CLI (Command line interface)

- Can be run inline
- Can be run as cli entry
- Can be run via a file

```
hsmith@vx1:~$ python3
Python 3.7.3 (default, Apr  3 2019, 05:39:12)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello")
Hello
>>>
hsmith@vx1:~$ python3 -c 'print("Hello")'
Hello
hsmith@vx1:~$ █
```

# Python - Basics

## Basic code (basics1.py)

```
1 name = "Giraffe"
2 age = 18
3 height = 2048.11 # mm
4
5 num1 = 3 ** 3
6 num2 = 27 // 3
7
8 print(name + ", " + str(age) + ', ' + str(height))
9 print(name, age, height, sep = ', ')
10 print(f"{name}, {age}, {height}")
11 print(type(name))
12 print(type(age))
13 print(type(height))
14 print(f"3 ** 3 == {num1}")
15 print(f"27 // 3 == {num2}")
```

- Garbage collection
- More info on data types

# Python - Basics

## Strings (basics2.py)

```
1 sentence = "My"  
2 sentence = sentence + " name is"  
3 sentence += " Pikachu"  
4  
5 print(sentence)  
6  
7 print("Hi!!" * 10)
```

Python strings are **immutable**

# Python - Basics

## Control structures, argc/argv (basics3.py)

```
1 import sys
2
3 argc = len(sys.argv)
4
5 empty = True
6 if argc > 0:
7     empty = False
8
9 if not empty:
10     if argc == 2:
11         print("Nearly there")
12     elif argc == 3:
13         if sys.argv[1] == "H" and sys.argv[2] == "I":
14             print("HI to you too")
15         else:
16             pass
17 else:
18     print("Please enter two letters as command line")
```

# Python - Basics

## Lists, loops (basics4.py)

```
1 names = [ "Hayden", "Rob", "Isaac" ]
2 names.append("Vivian")
3
4 for name in names:
5     print(name)
6
7 print("===")
8
9 names += [ "Eve", "Mia" ]
10 for i in range(0, len(names)):
11     print(names[i])
```

Python lists are very complicated arrays under the hood. You can read a lot [here](#), [here](#), and [here](#).



# Python - Basics

## Tuples (basics5.py)

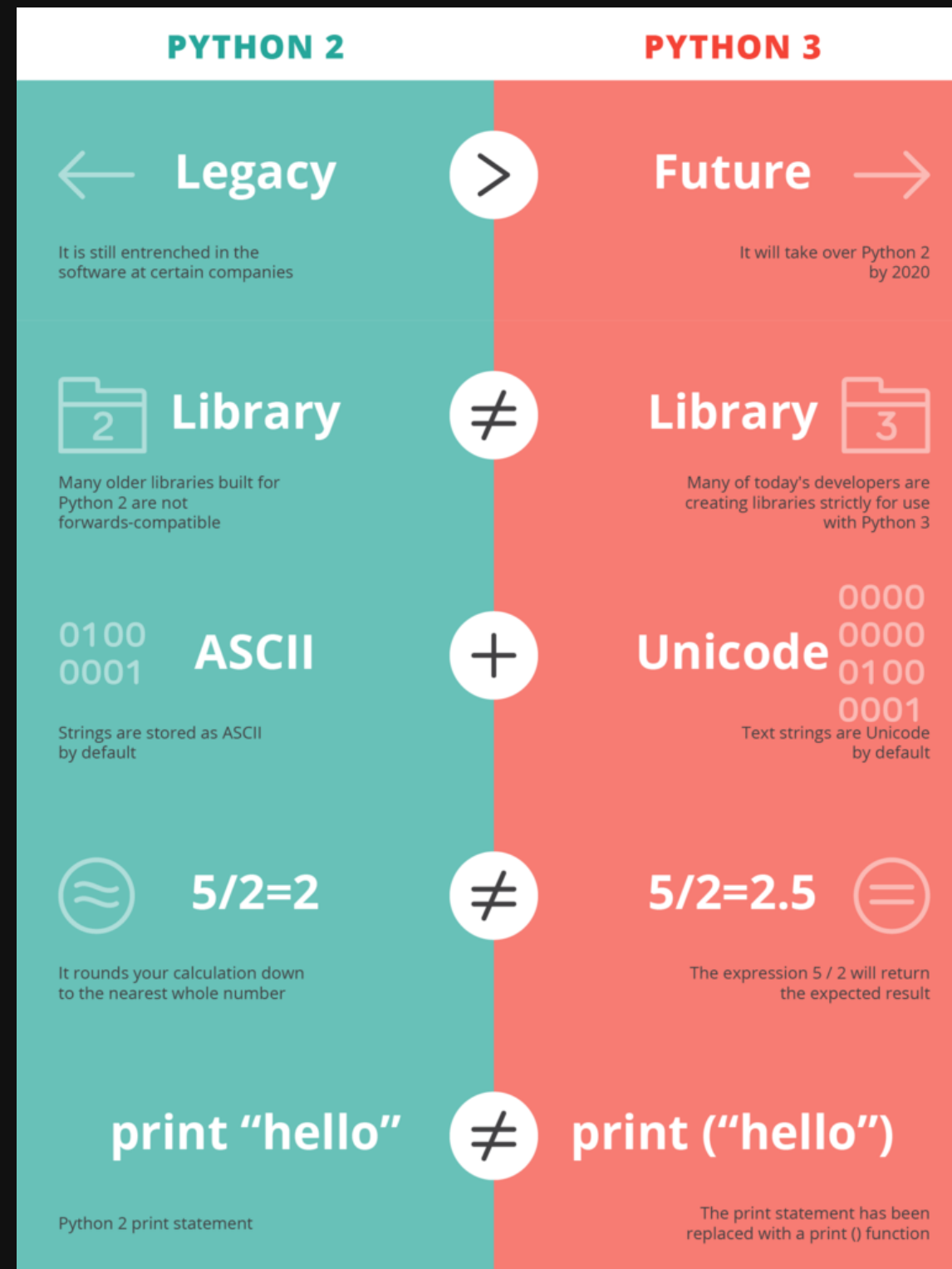
```
1 x = 5
2 y = 6
3 point = (x, y)
4 print(point)
5
6 a, b = point # destructuring
7 print(f"{a}, {b}")
8
9 names = [ "Giraffe", "Llama", "Penguin" ]
10 for id, name in enumerate(names):
11     print(f"{id} {name}")
```

- lists are mutable, tuples are immutable

# Python

python2

python3



# Why Python?

- Rapidly build applications due to high level nature
- Very straightforward toolchain to setup and use
- It's very structured compared to other scripting languages
- Useful in data science and analytics applications