COMP1531

🔨 Python

1.3 - Introduction

In this lecture

Why?

- Python is a valuable tool to learn and necessary for the project

What?

- Learning a second language
- Python vs C
- Core python language features
- Python versions

Python is a high level scripting language with rich libraries that has common applications in building simple services, utility tools, and all forms of data science.

Python is the universal go-to language if you had to pick up just one programming language.

```python
def times_tables(size):
    lst = []
    for i in range(size):
        for j in range(size):
            lst.append(f"{i} * {j}")
    return lst
```

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

Learning another language

Learning another programming language is a very comfortable exercise, particularly if the language is from the same programming paradigm.

Other major factors dictate differences between languages - i.e. does it deal with pointers? Is it a typed language?

	Procedural	**Object-oriented**	**Typed**	**Pointers**	**Compiled**
C	Yes	No	Yes	Yes	Yes
C++	Yes	Yes	Yes	No	Yes
Java	No	Yes	Yes	No	Yes
Python	Yes	Yes	Can be	No	No
Javascript	Yes	Yes	Can be	No	No

Of course, there are syntax differences! But syntax differences are easy to pick up.

Learning another language

In the case of learning another language like Python, the main hurdles we have to overcome are:

- Python does not have types, unlike C
- Python has object-oriented components (which we can somewhat ignore), unlike C
- Python does not deal with pointers, unlike C (yay)
- Python is often written at a "higher level" (more abstract)
- Python does not have an intermediate compilation step, like C

Python vs C

Write a function that takes two numbers, and returns a list of those two numbers along with its sum

Python

```python
def add_two_numbers(one, two):
    three = one + two
    return [ one, two, three ]
```

C

```c
int add_two_numbers(int one, int two) {
    int three = one + two
    int *arr = malloc(sizeof(int) * 3);
    arr[0] = one;
    arr[1] = two;
    arr[2] = three;
    return arr;
}
```

Python vs C

Interpreted V compiled

$$\begin{tikzcd}
\text{C program} \arrow[r, "\text{Compile}"] & \text{Machine Code} \arrow[r, "\text{Run}"] & \text{Output} \\
\text{Python program} \arrow[rr, "\text{Compile \& Run}"'] & & \text{Output}
\end{tikzcd}$$

Learning Python

Because you already know C, we will teach Python very quickly and mainly focus on the differences between Python and C.

Unlike C, Python has a sprawling set of capabilities - the language will *feel* much bigger, and therefore you might feel you have a poorer grasp on it.

Don't expect to know everything about Python this term. Just focus on only learning what you need to solve a problem at hand, and you will learn more super quick.

Read more about the python language here.

Python

Python can be run via the command line directly without compiling:

```
$ python3 myfile.py
```

```
print("Hello world!")
```

myfile.py

Python - Basics

```python
name = "Giraffe"
age = 18
height = 2048.11 # mm

num1 = 3 ** 3
num2 = 27 // 3

print("=== Printing Items ===")
print(name + ", " + str(age) + ', ' + str(height))
print(name, age, height, sep = ', ')
print(f"{name}, {age}, {height}")

print("=== Printing Types ===")
print(type(name))
print(type(age))
print(type(height))

print("=== Printing Mixed ===")
print(f"3 ** 3 == {num1}")
print(f"27 // 3 == {num2}")
```

types.py

- Garbage collection
- More info on data types

Python - Strings

```python
sentence = "My"
sentence = sentence + " name is"
sentence += " Pikachu"

print(sentence)

print("Hi!!" * 10)
```

strings.py

Python strings are **immutable**

Python - Lists & Tuples

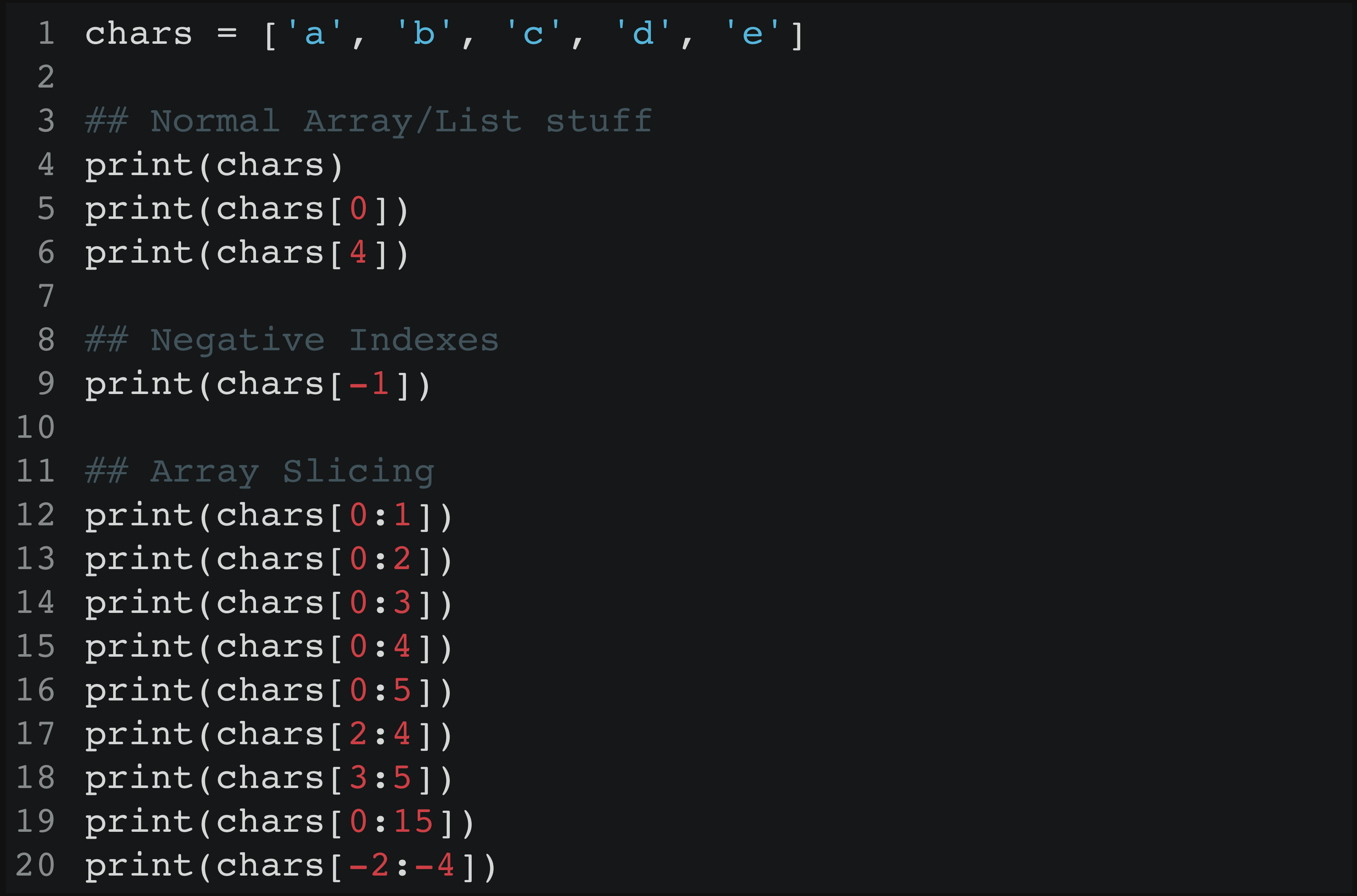
```python
# This is a list
names = ['Hayden', 'Jake', 'Nick', 'Emily']
print(f"1 {names}")
print(f"2 {names[0]}")
names[1] = 'Jakeo'
names.append('Rani')
print(f"3 {names}")

print('=====================================')

# This is a tuple
animals = ('Giraffe', 'Turtle', 'Elephant')
print(f"4 {animals}"
print(f"4 {animals[0]}"
# animals[1] = 'Dog' PROHIBITED
animals.append('Koala')
```

lists-tuples.py

[Lists] are for mutable ordered structures of the same type

[Tuples] are for immutable ordered structures of any mix of types

Python - Slicing



```python
chars = ['a', 'b', 'c', 'd', 'e']

## Normal Array/List stuff
print(chars)
print(chars[0])
print(chars[4])

## Negative Indexes
print(chars[-1])

## Array Slicing
print(chars[0:1])
print(chars[0:2])
print(chars[0:3])
print(chars[0:4])
print(chars[0:5])
print(chars[2:4])
print(chars[3:5])
print(chars[0:15])
print(chars[-2:-4])
```

slicing.py

Lists/Tuples can be "sliced" to extract a subset of information about them.
This is a real standout feature of python.

Python - Control Structures

```python
# Note the following:
#  - Indentation and colon denotes nesting, not braces
#  - Conditions generally lack paranthesis
#  - pass used to say "do nothing"
#  - i++ is not a language feature

number = 5
if number > 10:
        print("Bigger than 10")
elif number < 2:
        pass
else:
        print("Number between 2 and 9")

print("--------------------------")

i = 0
while i < 5:
        print("Hello there")
        i += 1

print("--------------------------")
for i in range(5):
        print("Hello there")
```

control-structures.py

Python - Functions

```python
def get_even(nums):
        evens = []
        for i in range(len(nums)):
                if number % 2 == 0:
                        evens.append(number)
        return evens

all_numbers = [1,2,3,4,5,6,7,8,9,10]
print(get_even(all_numbers))
```

functions.py

Python - Dictionaries

```python
student = {
     'name': 'Emily',
     'score': 99,
     'rank': 1,
}

print(student)
print(student['name'])
print(student['score'])
print(student['rank'])

student['height'] = 159
print(student)
```

dictionaries.py

Think of dictionaries like structs. You use them when you need a "collection" of items that are identified by a string description, rather than a numerical index (lists)

Python - Combining

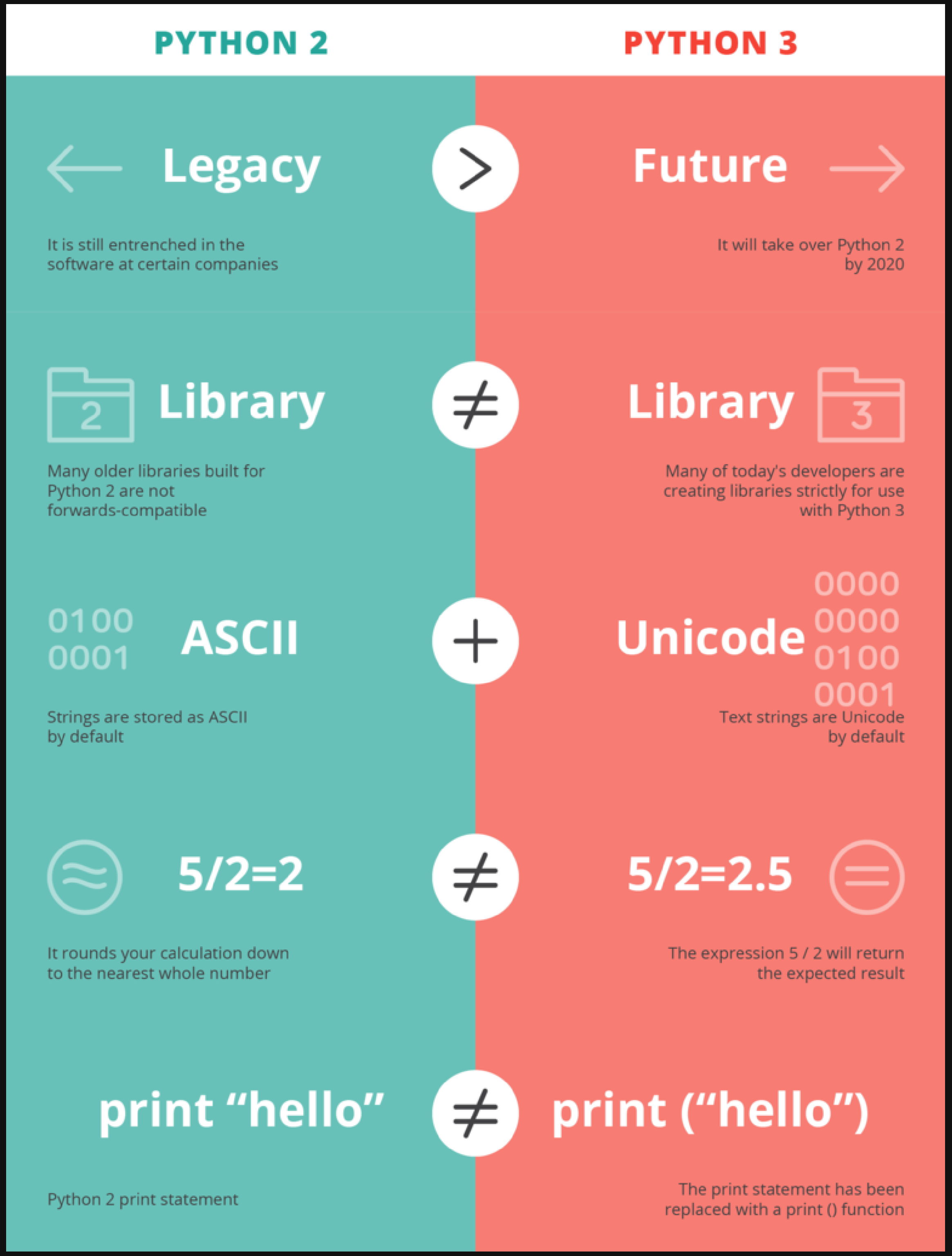
```python
student1 = { 'name' : 'Hayden', 'score': 50 }
student2 = { 'name' : 'Nick', 'score': 91 }
student3 = { 'name' : 'Emily', 'score': 99 }
students = [student1, student2, student3]

print(students)

# Approach 1
num_students = len(students)
for i in range(num_students):
  student = students[i]
  if student['score'] >= 85:
    print(f"{student['name']} got an HD")

# Approach 2
for student in students:
  if student['score'] >= 85:
    print(f"{student['name']} got an HD")
```

combining1.py

It's possible to create data structures of other data structures

Python

python2



PYTHON 2		PYTHON 3
Legacy — It is still entrenched in the software at certain companies	>	**Future** — It will take over Python 2 by 2020
Library — Many older libraries built for Python 2 are not forwards-compatible	≠	**Library** — Many of today's developers are creating libraries strictly for use with Python 3
ASCII (0100 0001) — Strings are stored as ASCII by default	+	**Unicode** (0000 0000 0100 0001) — Text strings are Unicode by default
5/2=2 — It rounds your calculation down to the nearest whole number	≠	**5/2=2.5** — The expression 5 / 2 will return the expected result
print "hello" — Python 2 print statement	≠	**print ("hello")** — The print statement has been replaced with a print () function

python3

Python

In COMP1531, we will be using python 3.7 for everything.

If you're on the CSE machines, you can run the python interpreter with "python3" - this will automatically use python3.7

If you're on a local machine, we recommend you run python with "python3.7" - this will ensure you use the right version

Feedback