POWERING POTENTIAL

Python Programming Basics

WELCOME

Insert a RECENT and HIGH QUALITY photo of yourself here

Trainer Name

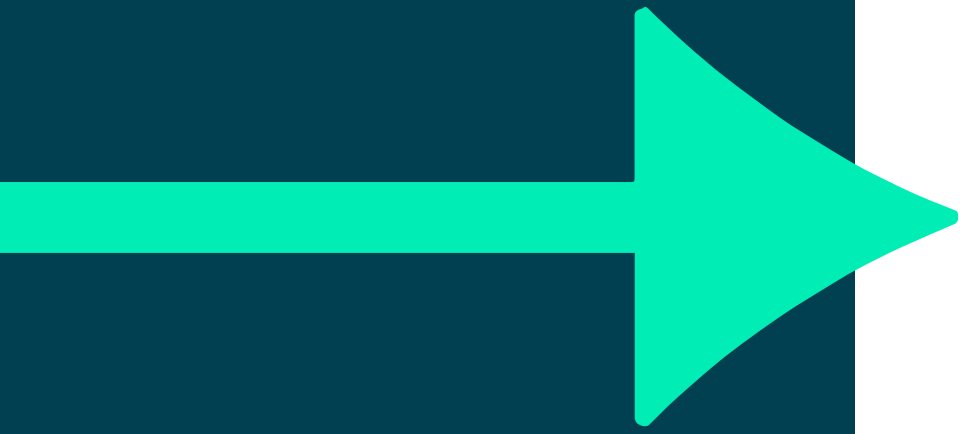
Trainer Role, QA

- A few key facts...
- Previous role
- Qualifications

SESSION OVERVIEW

- Engage in practical activities to support module evidence collection
- Introductions and ice-breaker activity
- Provide an overview of the 3-day class-based learning
- Provide support and guidance for the successful completion of Module 4B

LESSON OBJECTIVES

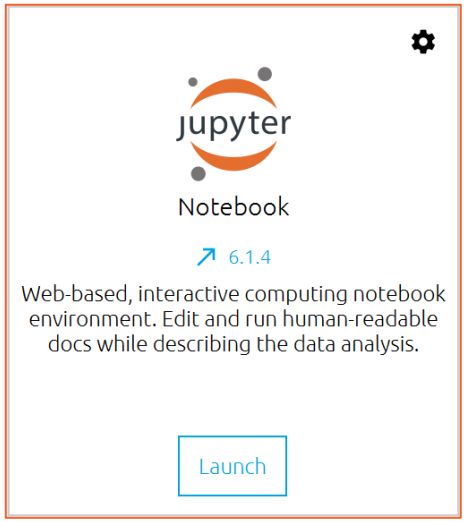


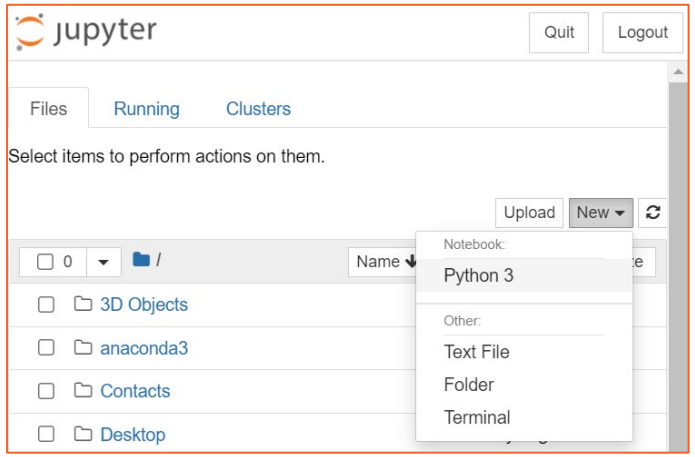
In this chapter, you'll learn about:

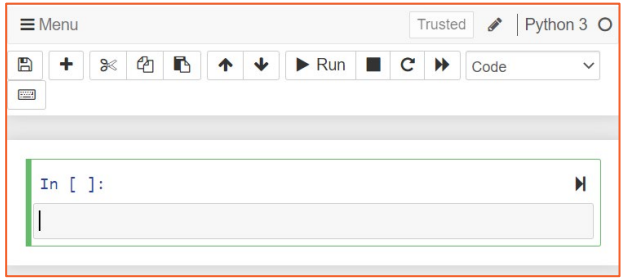
- Jupyter Notebooks
- Basic statements
- Numbers, strings, and Boolean variables
- Keyboard input
- Screen output

WHY JUPYTER NOTEBOOKS?

- Jupyter Notebooks is a superb web-based interactive application
  - User Friendly
  - Easy to debug at each line of code

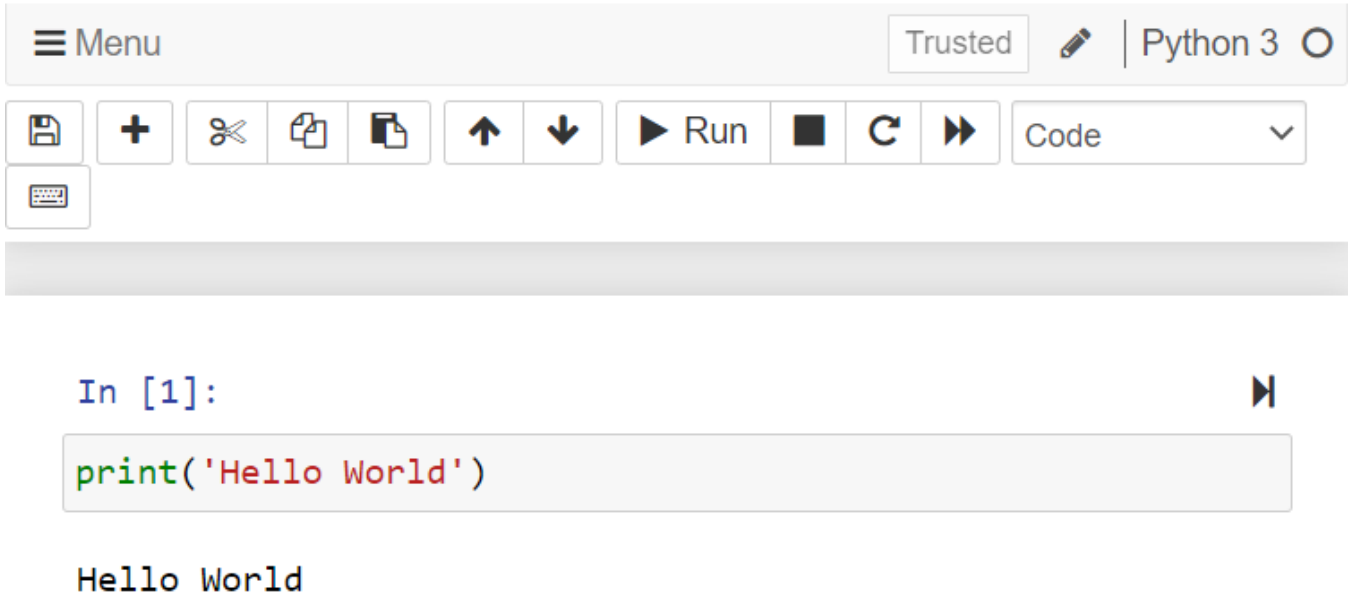






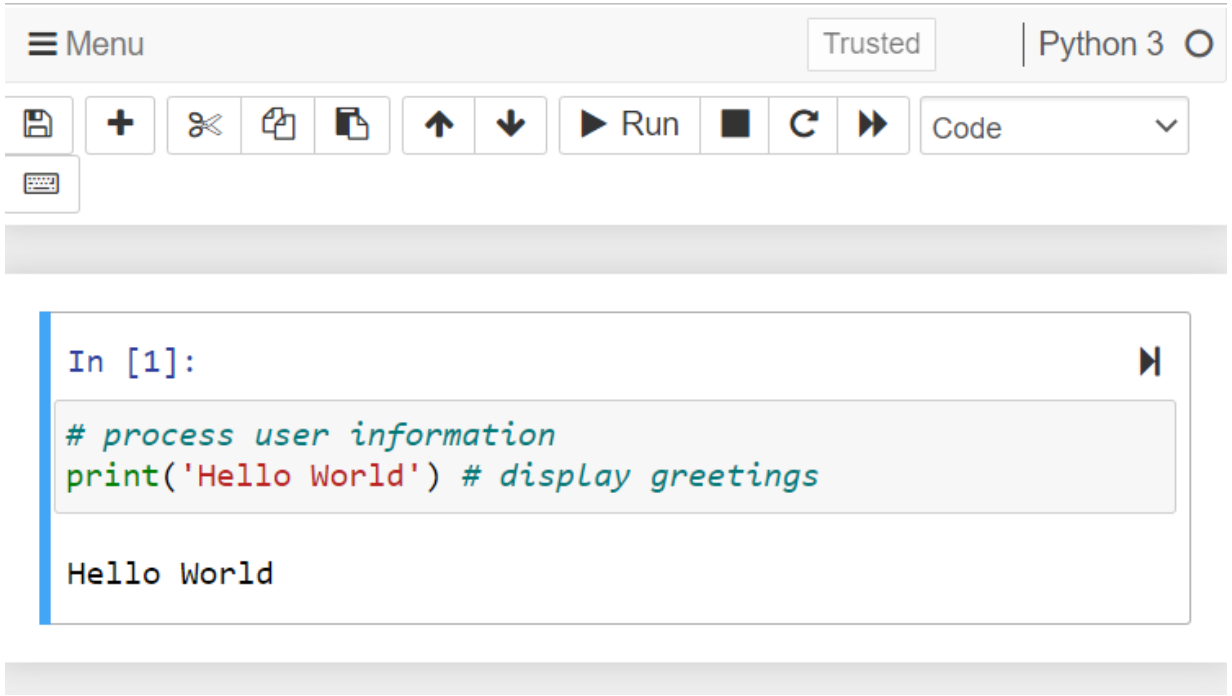
PRINTING TO CONSOLE WINDOW

- print ('Hello World')
- Click ▶ Run or SHIFT + Enter to run

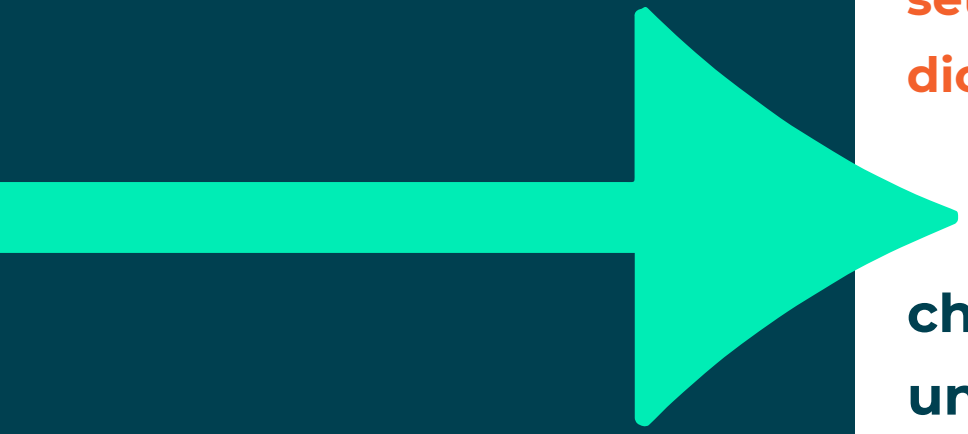


COMMENTS IN CODE

- Help readers understand your code
- Use a **#** to ignore the following chars
- Are ignored when code runs



DATA STRUCTURES IN PYTHON



Data Structures in Python:

- Primitive data types
- Complex data types (collections)

Data Structures in Python at a glance:

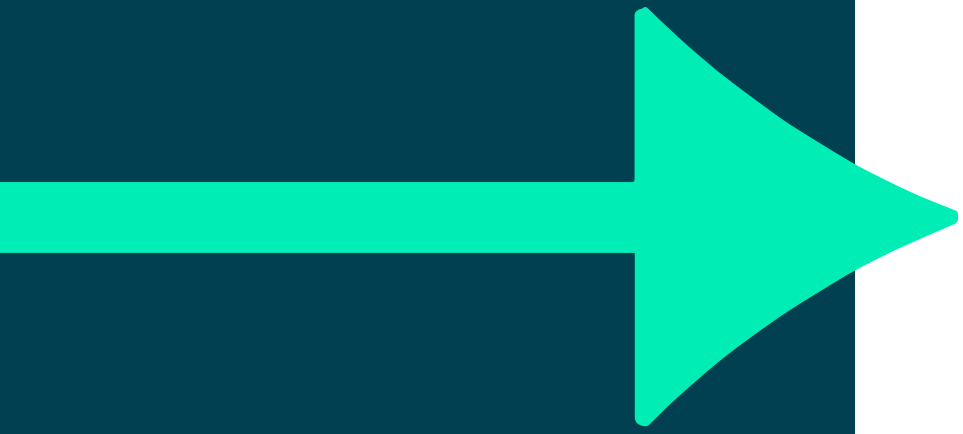
- **primitive data types: integer, string, Boolean, float**
- **complex data types (collections)**

list	ordered	changeable	allow duplicates
tuple	ordered	unchangeable	allow duplicates
set	unordered	changeable	no duplicates
dictionary	unordered/ ordered (Python 3.7)	changeable	no duplicates

changeable = mutable

unchangeable = immutable

LISTS VERSUS ARRAYS



Python does not have arrays as a built-in data type, however arrays are introduced in a Python library (NumPy).

Lists can be used as arrays.

Arrays vs Lists - what is the difference?

- Arrays are fixed in size - lists are dynamic
- Arrays can only store data of the same type - lists can store different data types
  - Lists can be used as arrays - but not the other way round
- Array are slightly more memory-efficient

DATA TYPES IN PYTHON

There Are Three Basic Variable Types:

- Numbers: Integer and Float
  - 1,2,3, 1.23, 0.0005
- Character or String
  - 'Hello world' or "Hello world"
- Boolean **True** or **False**
  (case-sensitive)

```
age=21
salary = 2000.78
companyName='QA Ltd'
isRegistered = True
hasLicence = False
```

type is determined automatically; value can change

VARIABLE NAMING STANDARDS

- Use letters, not punctuation

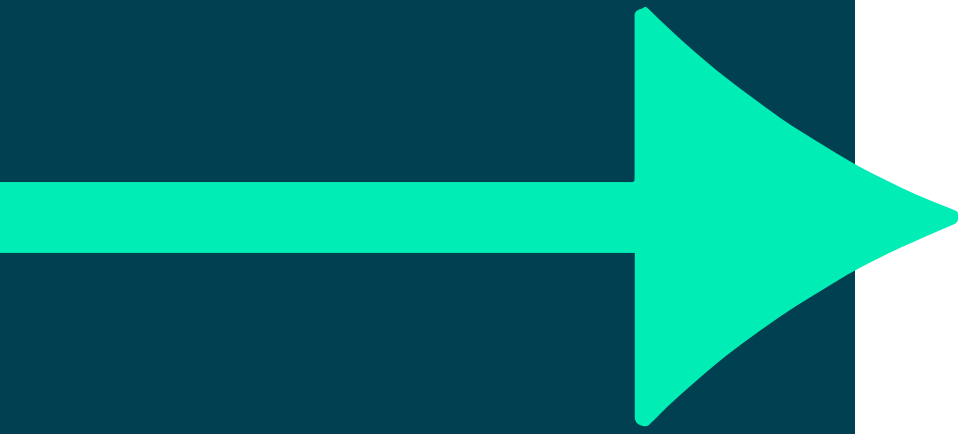
```
salay$ = 1500          ✗
my-city='London'       ✗
my city='London        ✗
```

```
my_city='London'       ✓
myCity='London'        ✓
```

- No 'reserved word'

```
print = 10             ✗
```

VARIABLE NAMING STANDARDS



Case Sensitive

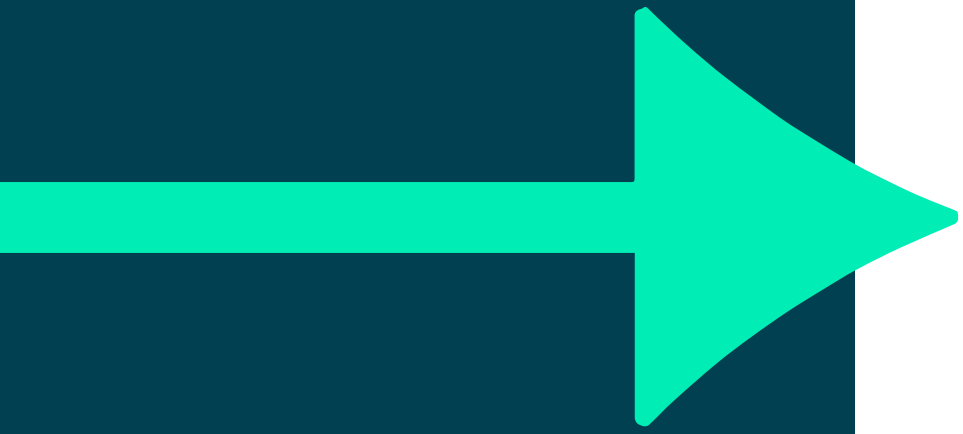
- Use lowercase letters for consistency

What will be displayed?

```
age=32
Age=21
print(age)
```

32

ARITHMETIC OPERATIONS



The arithmetic operations in Python are:

- Addition **+**
- Subtraction **–**
- Multiplication *****
- Division **/**
- Floor (integer) division **//**
- Modulus (modular division) **%**
- Exponentiation (raising to a power) ******

ARITHMETIC OPERATIONS: THE DIFFERENT TYPES OF DIVISION

```
x1 = 5
x2 = 3

# division
d1 = x1 / x2
# floor (integer) division
d2 = x1 // x2
# modular division (remainder of integer division)
d3 = x1 % x2
```

The results of the three types of divisions are:

d1 = 1.6666666667

d2 = 1

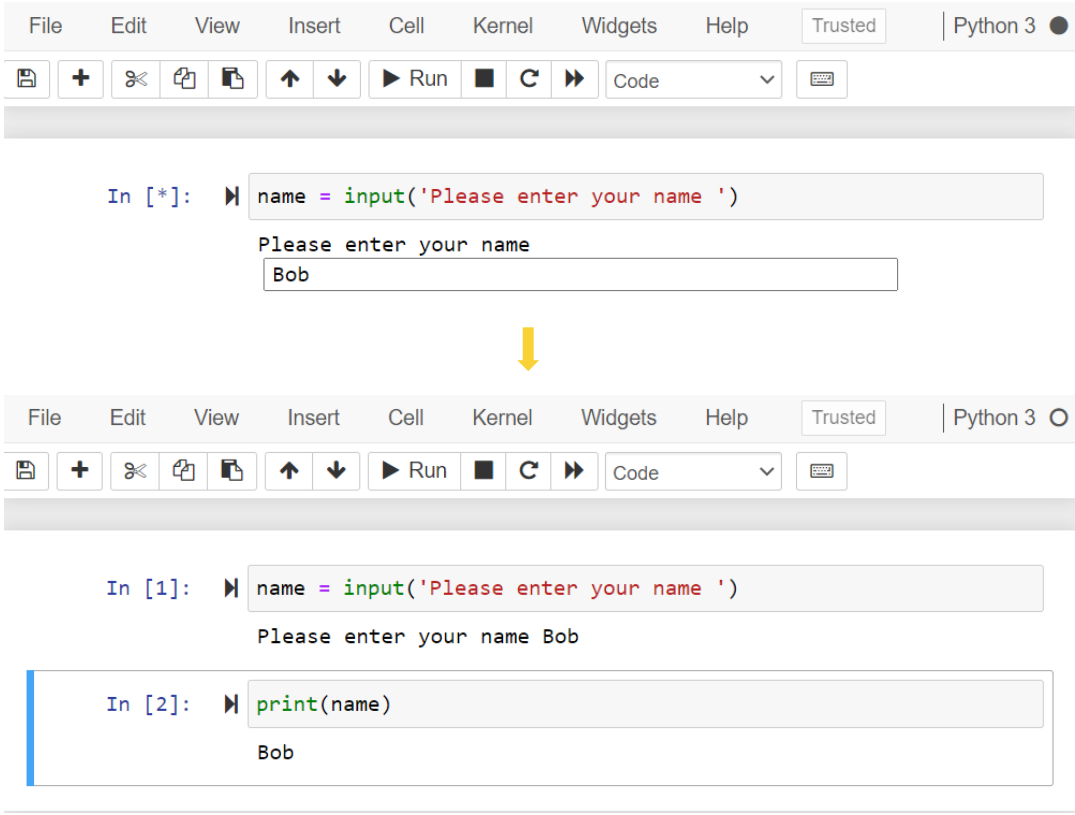
d3 = 2

The floor (integer) division doesn't round, it **truncates** to obtain the integer result.

USER INPUT USING KEYBOARD

Input with a Prompt
`input(<prompt>)`

```
name = input('Please enter your name ')
print(name)
```



PUTTING STRINGS TOGETHER

```
username = 'Bob'
print('Hello' , username)
print('Hello' + username)
print('Hello ' + username)
```

```
Hello Bob
HelloBob
Hello Bob
```

Cannot add numbers and strings

```
age = 21
print('Your age is ' + age)
Message = 'Your age is ' + age
```

keyboard input is always text... even if it 'looks' like a number

```
age = input('Please enter your age ')
age = age + 1
```

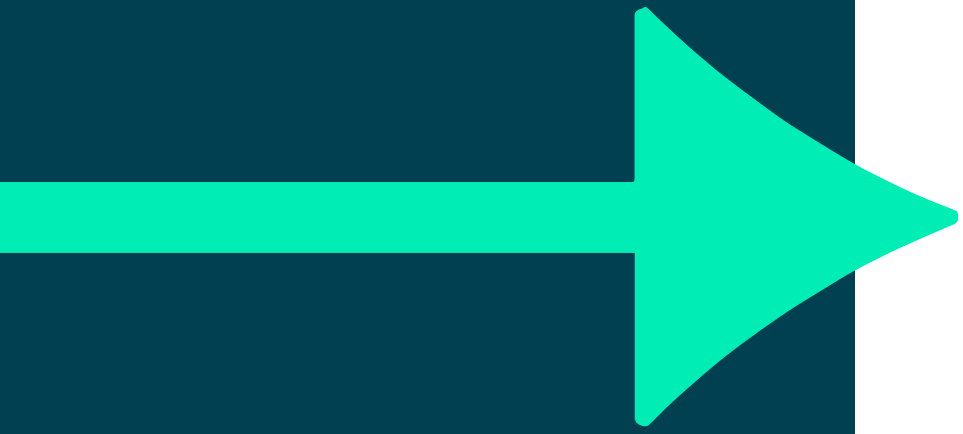
```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-2-972fe510ffb0> in <module>
----> 1 age = age + 1

TypeError: can only concatenate str (not "int") to str
```

CASTING

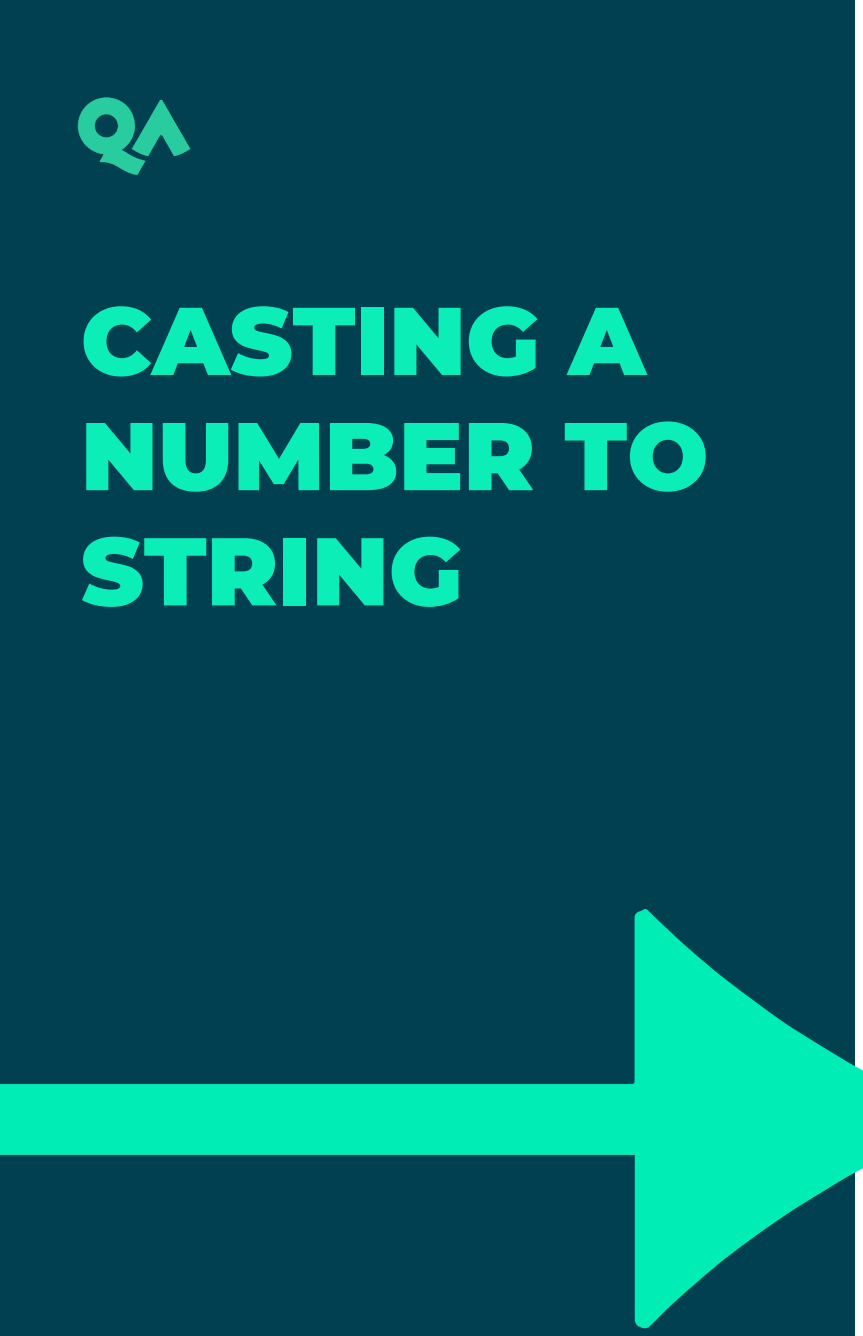
Keyboard Is Text, So We Use Casting to Convert It to Other Types

```
age = int(input('What is your age? '))
age = age + 1
print('Next year you will be', age ,'years old')
```

```
What is your age?  21
Next year you will be 22 years old
```



```
age = input('What is your age? ')
age = int(age)
age = age + 1
print('Next year you will be', age ,'years old')
```



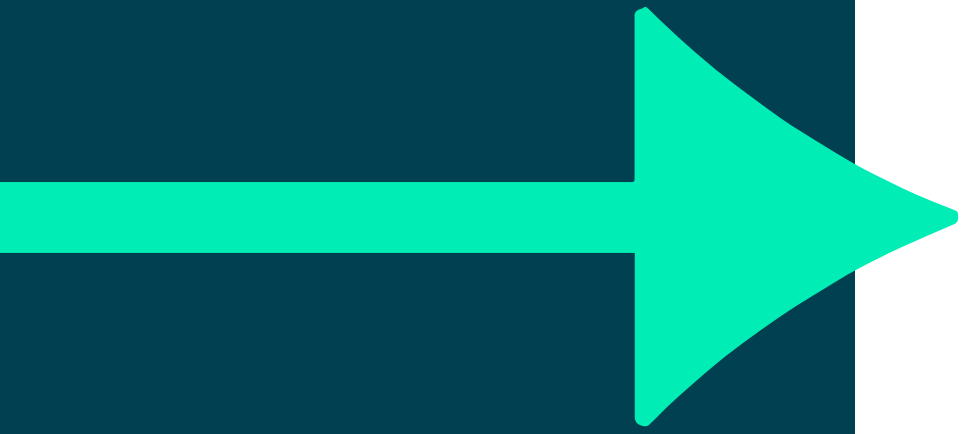
CASTING A NUMBER TO STRING

Use the str() function

```
## Find the average of a few numbers

total = 1 + 3 + 5 + 7 + 9 + 11
average = total / 6

print("Total is = " + str(total))
print("Average is = " +
str(average))
```

CASTING FLOATS

```
price = int(input('What is the price? '))

totalPrice = price * 1.2
```

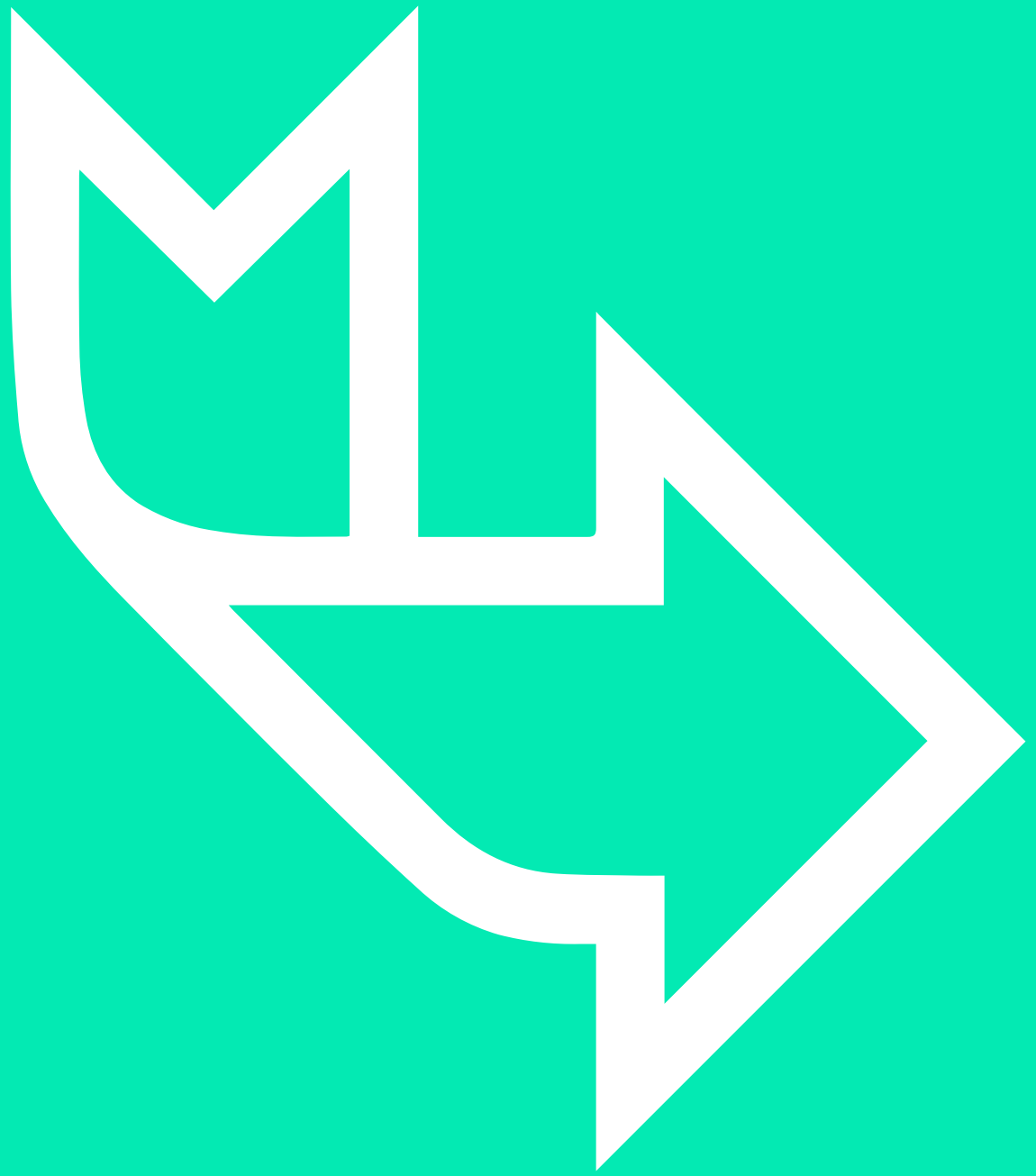


```
price = float(input('What is the price? '))

totalPrice = price * 1.2
```

SUMMARY

In this chapter, you learned about:

- Python in Jupyter Notebook
- Basic statements
- Numbers, strings, and Boolean variables
- Keyboard input
- Screen output
- Casting



Further Reading

https://www.python.org/