UNC CHARLOTTE

The WILLIAM STATES LEE COLLEGE *of* ENGINEERING

Introduction to ML
Lecture 2: Intro to Python 1

Hamed Tabkhi

Department of Electrical and Computer Engineering,

University of North Carolina Charlotte (UNCC)

htabkhiv@uncc.edu

Ask!

The art and science of asking questions is the source of all knowledge.

- Thomas Berger

- Do not hesitate to ask!
- If something is not clear, stop me and as
- During exercises (you can also ask othe

Image by mohamed Hassan from Pixabay

Now let me ask something..

- Why do you want to learn Python/programming?
- What would you use Python for?

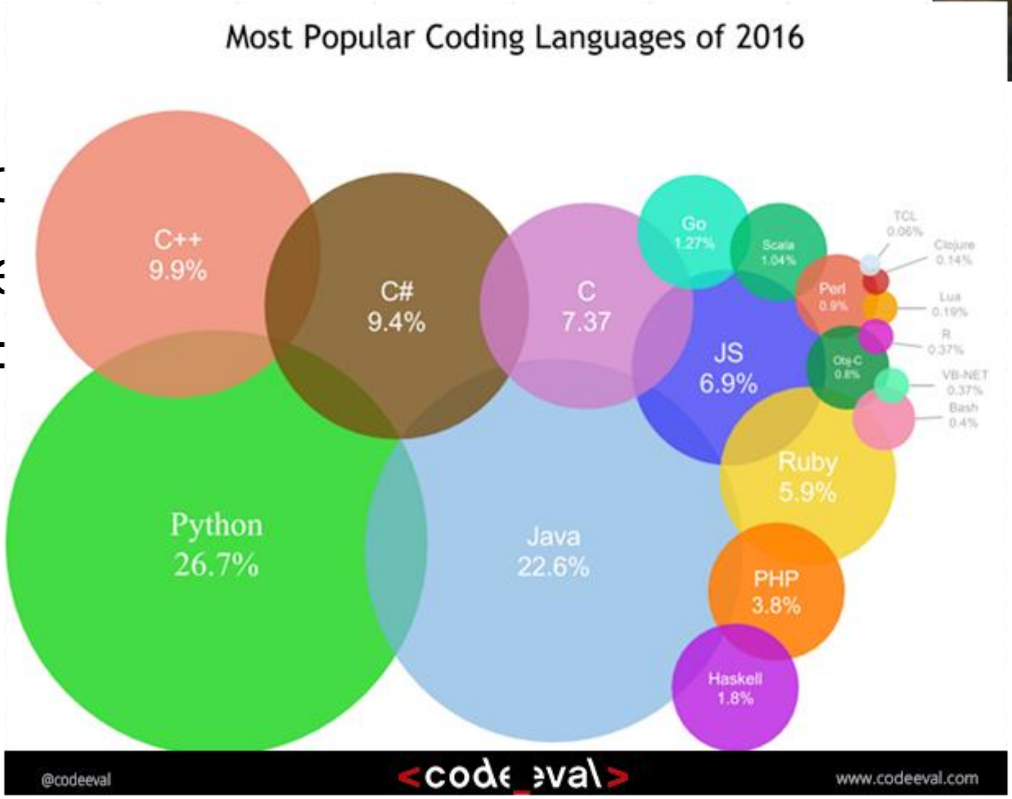
Failure

- Coding is all about trial and error.
- Don't be afraid of it.
- Error messages aren't scary, they are useful.

Python natalensis by A. Smith on Wikimedia Commons

History

- Started by Guid
  Rossum as a hc
- Now widely spre
- Open Source! F
- Versatile



Guido Van Rossum by [Doc Searls on Flickr]() CC-BY-SA

Python today

- Developed a large and active scientific computing and data analysis community
- Now one of the most important languages for
  - Data science
  - Machine learning
  - General software development
- Packages: NumPy, pandas, matplotlib, SciPy, scikit-learn, statsmodels

2 Modes

1. **IPython**

Python can be run interactively

Used extensively in research

2. **Python scripts**

What if we want to run more than a few lines of code?

Then we must write text files in .py

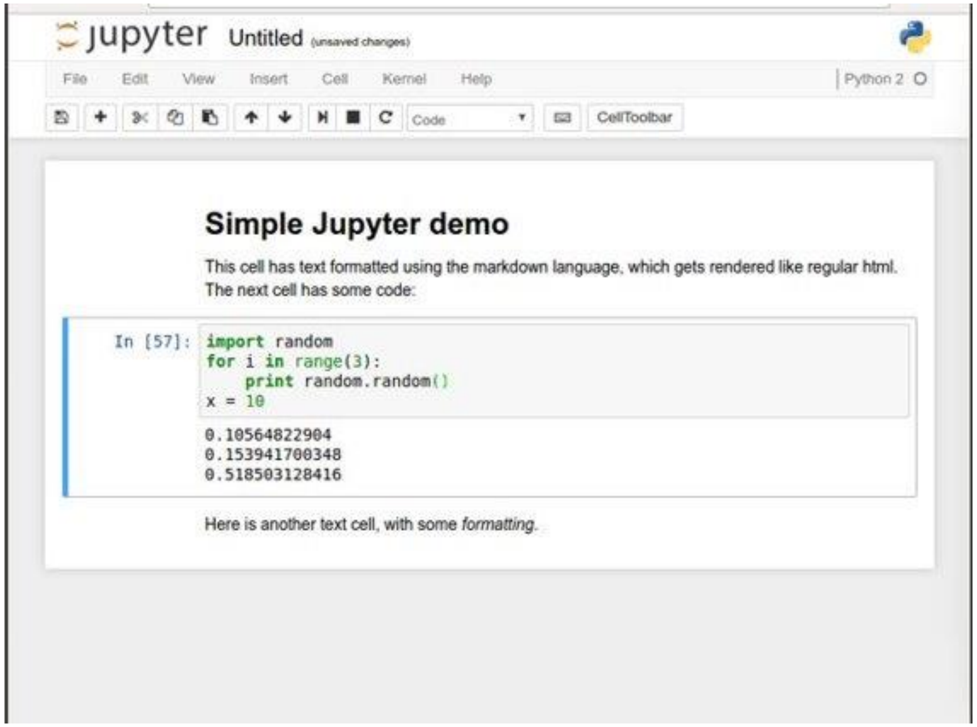
Time for a demo..

https://www.youtube.com/watch?v=BBwEF6WBUQs

Noteable (Jupyter notebooks)

- Easy to use environment
- Web-based
- Combines both text and code into one
- Come with a great number of useful packages

- https://jupyter.org/


jupyter
Untitled (unsaved changes)
File Edit View Insert Cell Kernel Help
Python 2
Code
CellToolbar

Simple Jupyter demo

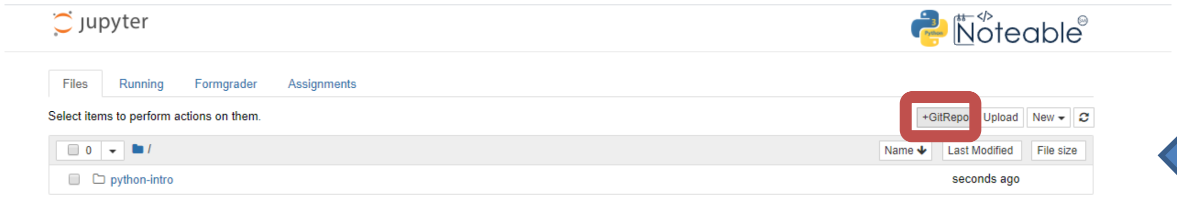
This cell has text formatted using the markdown language, which gets rendered like regular html. The next cell has some code:

```
In [57]: import random
         for i in range(3):
             print random.random()
         x = 10
```

```
0.10564822904
0.153941700348
0.518503128416
```

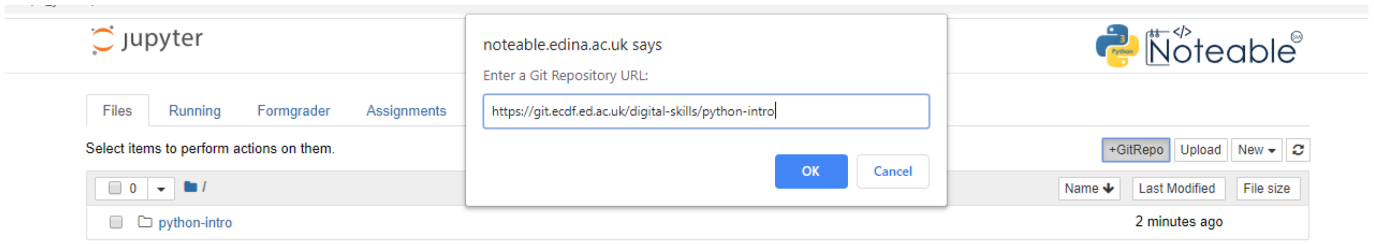
Here is another text cell, with some *formatting*.

2. Clone GitRepo(recommended)



Select items to perform actions on them.

python-intro seconds ago



noteable.edina.ac.uk says

Enter a Git Repository URL:

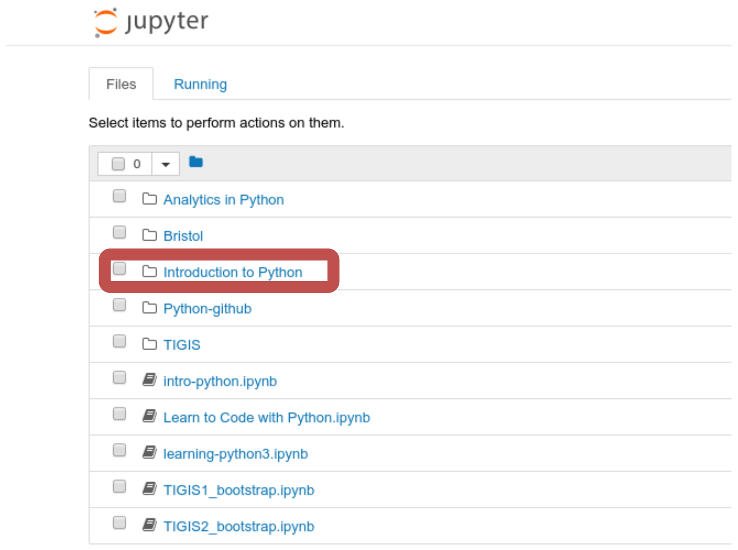
https://git.ecdf.ed.ac.uk/digital-skills/python-intro

Select items to perform actions on them.

python-intro 2 minutes ago

Pull down a Git repository

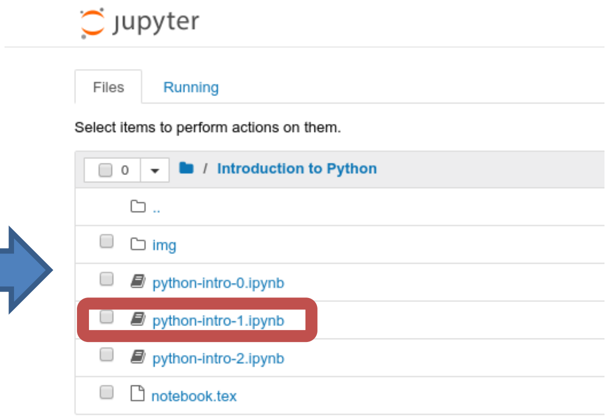
3. Starting a notebook


jupyter

Files Running

Select items to perform actions on them.

- Analytics in Python
- Bristol
- Introduction to Python
- Python-github
- TIGIS
- intro-python.ipynb
- Learn to Code with Python.ipynb
- learning-python3.ipynb
- TIGIS1_bootstrap.ipynb
- TIGIS2_bootstrap.ipynb


jupyter

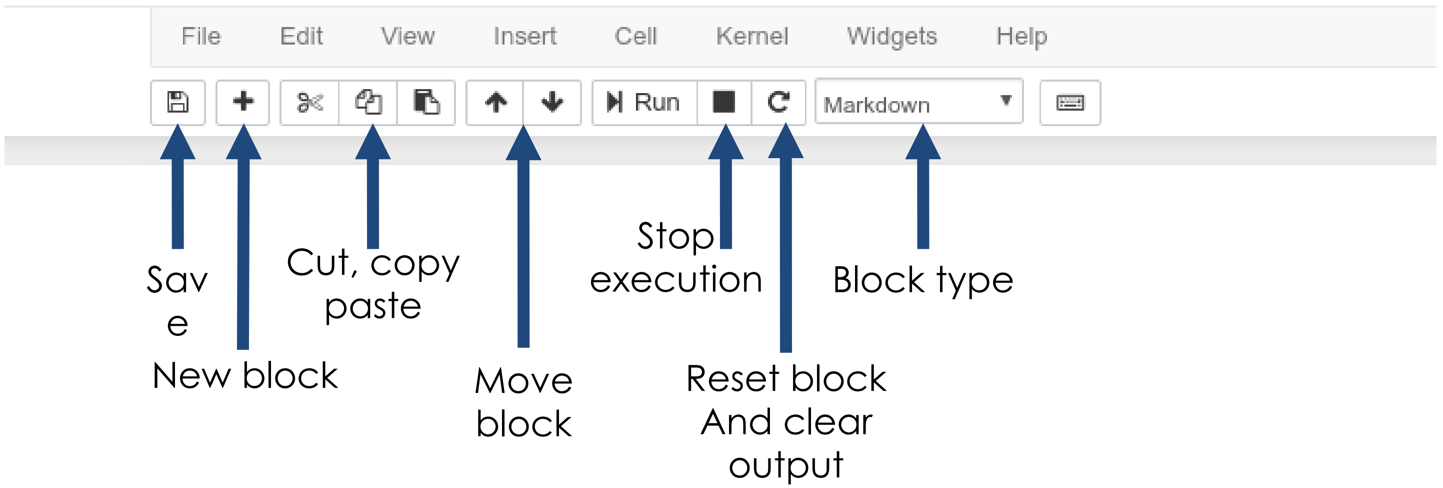
Files Running

Select items to perform actions on them.

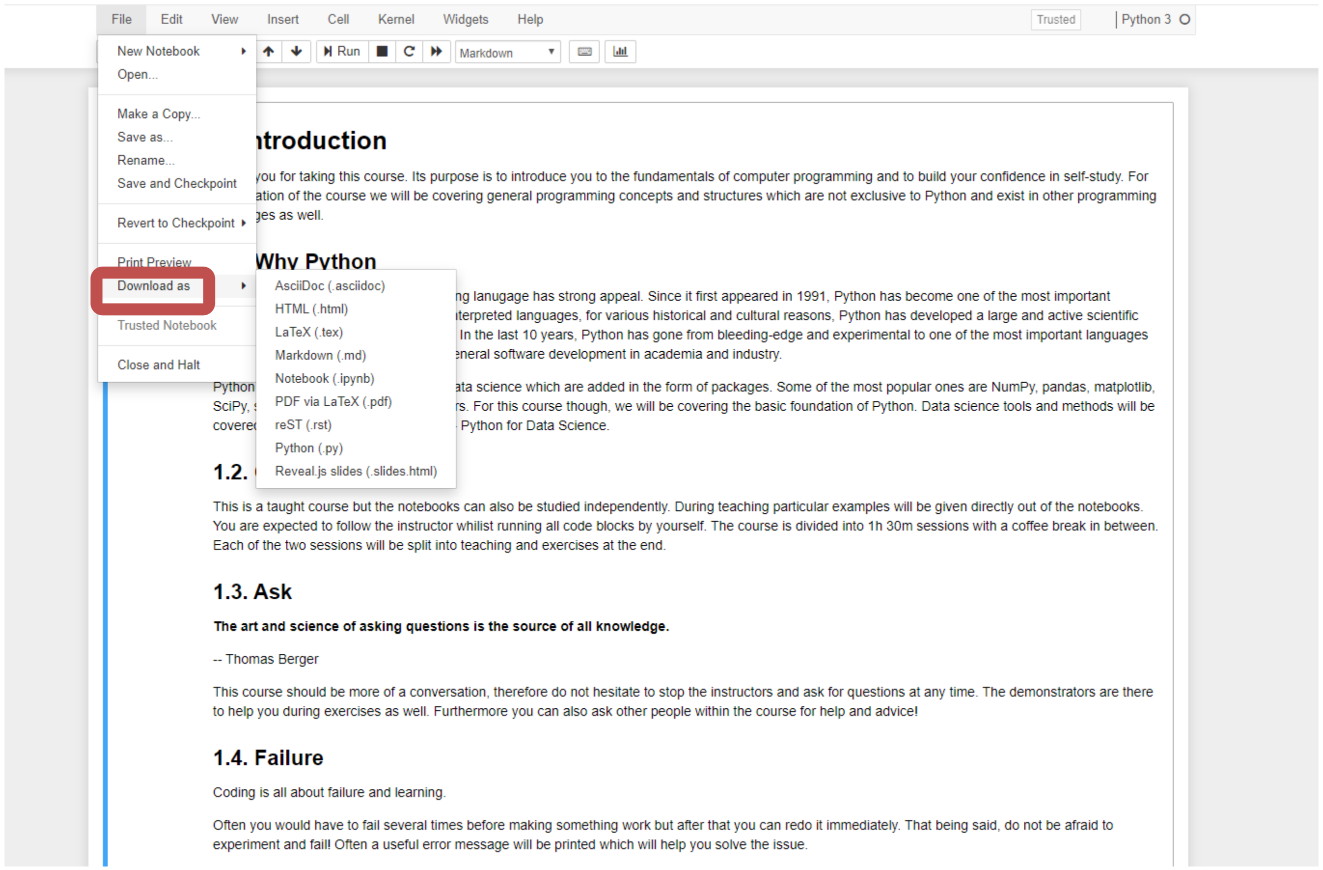
/ Introduction to Python

- ..
- img
- python-intro-0.ipynb
- python-intro-1.ipynb
- python-intro-2.ipynb
- notebook.tex

4. Toolbar


File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Run
Markdown
Sav
e
New block
Cut, copy
paste
Move
block
Stop
execution
Reset block
And clear
output
Block type

5. Download files



ntroduction

you for taking this course. Its purpose is to introduce you to the fundamentals of computer programming and to build your confidence in self-study. For ation of the course we will be covering general programming concepts and structures which are not exclusive to Python and exist in other programming ges as well.

Why Python

ng lanugage has strong appeal. Since it first appeared in 1991, Python has become one of the most important terpreted languages, for various historical and cultural reasons, Python has developed a large and active scientific In the last 10 years, Python has gone from bleeding-edge and experimental to one of the most important languages eneral software development in academia and industry.

Python ata science which are added in the form of packages. Some of the most popular ones are NumPy, pandas, matplotlib, SciPy, rs. For this course though, we will be covering the basic foundation of Python. Data science tools and methods will be covered Python for Data Science.

1.2.

This is a taught course but the notebooks can also be studied independently. During teaching particular examples will be given directly out of the notebooks. You are expected to follow the instructor whilist running all code blocks by yourself. The course is divided into 1h 30m sessions with a coffee break in between. Each of the two sessions will be split into teaching and exercises at the end.

1.3. Ask

The art and science of asking questions is the source of all knowledge.

-- Thomas Berger

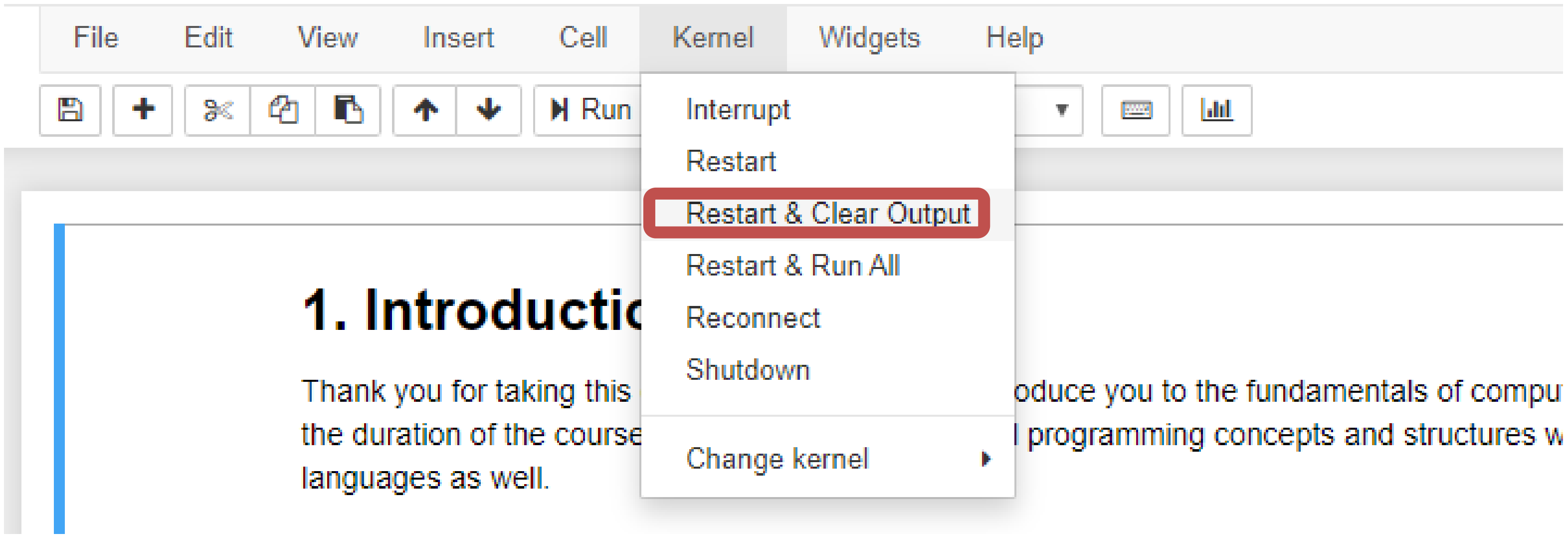
This course should be more of a conversation, therefore do not hesitate to stop the instructors and ask for questions at any time. The demonstrators are there to help you during exercises as well. Furthermore you can also ask other people within the course for help and advice!

1.4. Failure

Coding is all about failure and learning.

Often you would have to fail several times before making something work but after that you can redo it immediately. That being said, do not be afraid to experiment and fail! Often a useful error message will be printed which will help you solve the issue.

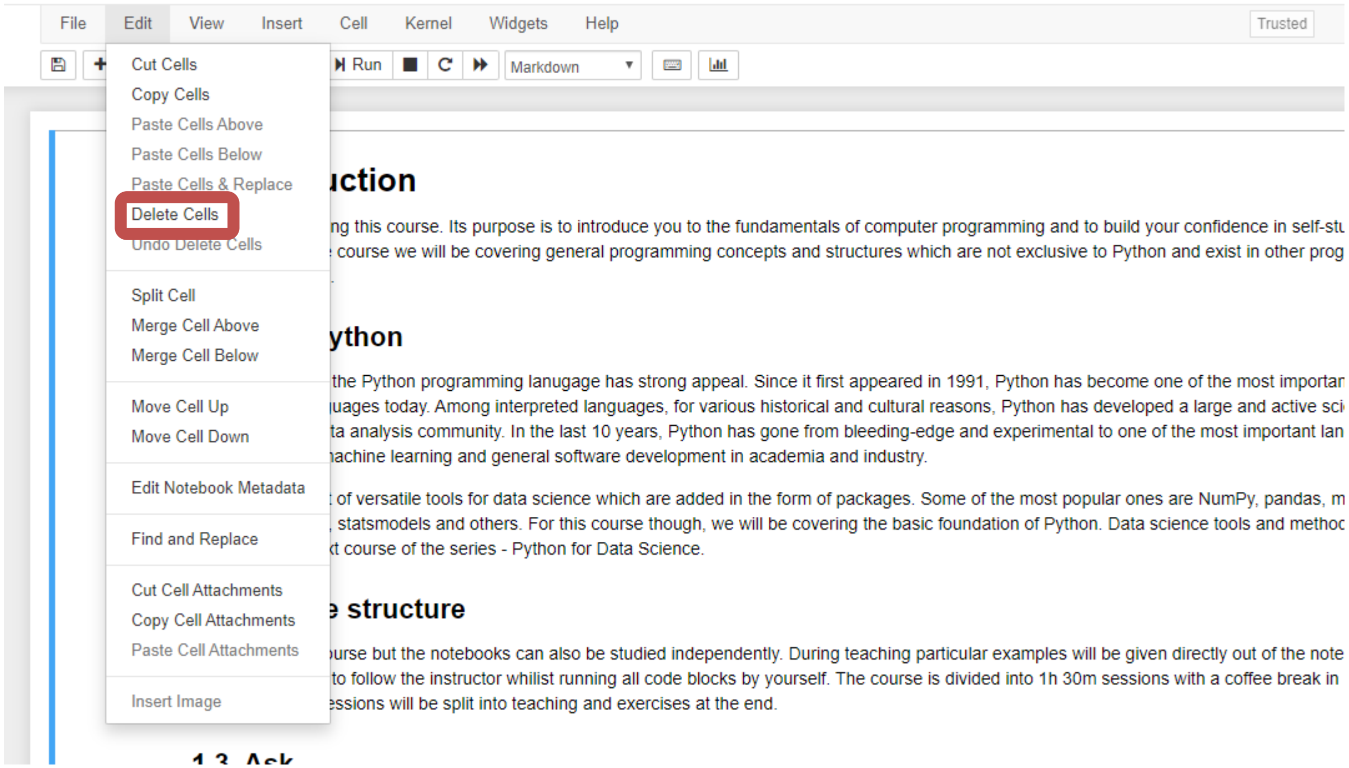
6. Kernel/Restart & Clear output



1. Introductio

Thank you for taking this oduce you to the fundamentals of compu
the duration of the course l programming concepts and structures w
languages as well.

7. Edit/Delete Cell



uction

ng this course. Its purpose is to introduce you to the fundamentals of computer programming and to build your confidence in self-stu
course we will be covering general programming concepts and structures which are not exclusive to Python and exist in other prog

ython

the Python programming lanugage has strong appeal. Since it first appeared in 1991, Python has become one of the most importar
uages today. Among interpreted languages, for various historical and cultural reasons, Python has developed a large and active sci
ta analysis community. In the last 10 years, Python has gone from bleeding-edge and experimental to one of the most important lan
achine learning and general software development in academia and industry.

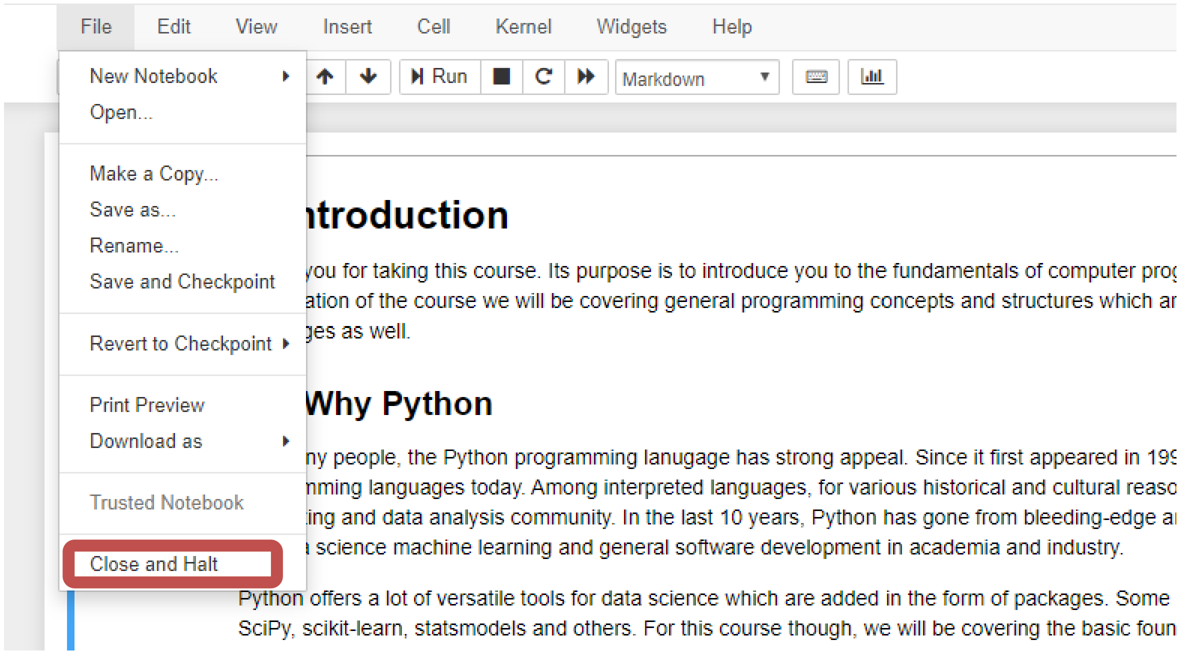
of versatile tools for data science which are added in the form of packages. Some of the most popular ones are NumPy, pandas, m
, statsmodels and others. For this course though, we will be covering the basic foundation of Python. Data science tools and methoc
t course of the series - Python for Data Science.

e structure

urse but the notebooks can also be studied independently. During teaching particular examples will be given directly out of the note
to follow the instructor whilst running all code blocks by yourself. The course is divided into 1h 30m sessions with a coffee break in
ssions will be split into teaching and exercises at the end.

1.3 Ask

8. File/ Close & Halt



ntroduction

you for taking this course. Its purpose is to introduce you to the fundamentals of computer prog
ation of the course we will be covering general programming concepts and structures which ar
ges as well.

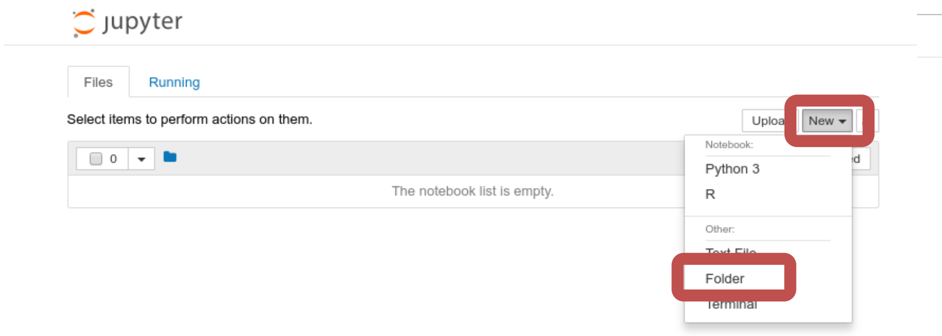
Why Python

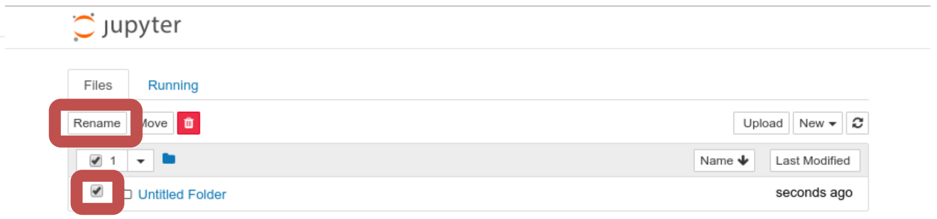
ny people, the Python programming lanugage has strong appeal. Since it first appeared in 19
nming languages today. Among interpreted languages, for various historical and cultural reaso
ing and data analysis community. In the last 10 years, Python has gone from bleeding-edge a
science machine learning and general software development in academia and industry.

Python offers a lot of versatile tools for data science which are added in the form of packages. Some
SciPy, scikit-learn, statsmodels and others. For this course though, we will be covering the basic foun

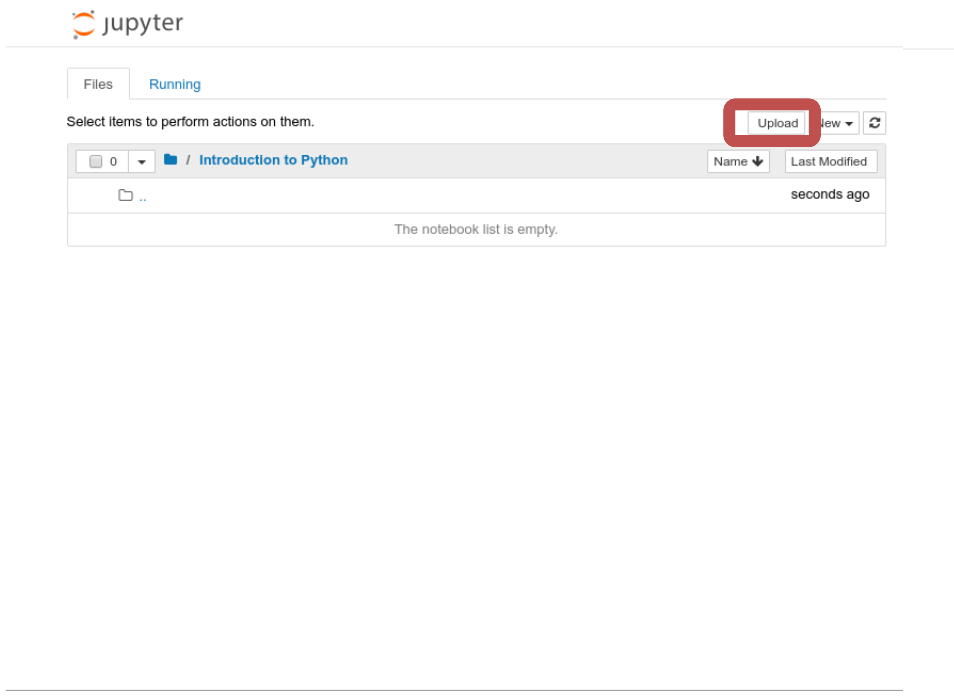
9. Create a folder

10.Rename





11. Upload files

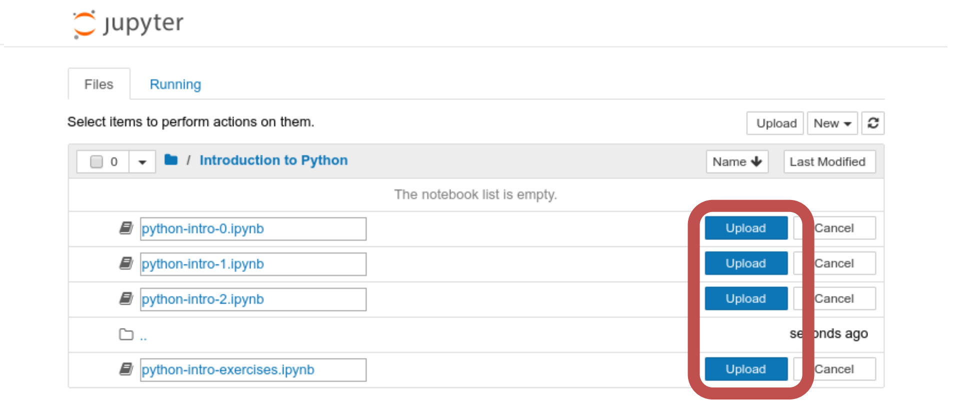


Select items to perform actions on them.

/ Introduction to Python

.. seconds ago

The notebook list is empty.



Select items to perform actions on them.

/ Introduction to Python

The notebook list is empty.

python-intro-0.ipynb Upload Cancel

python-intro-1.ipynb Upload Cancel

python-intro-2.ipynb Upload Cancel

.. seconds ago

python-intro-exercises.ipynb Upload Cancel

Running blocks

- By pressing the Run button
- Shift + Enter – runs block
- Alt + Enter – creates a new block

Let us start

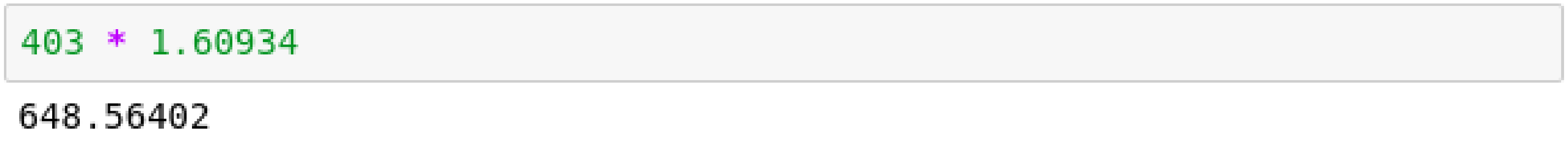
If you like to follow along, you can open your own notebook. But please try to keep up with my presentation, as you still have time for exercises after the teaching.

Agenda

- Variables
- Types
- Arithmetic operators
- Boolean logic
- Strings
- Printing
- Exercises

Python as a calculator

- Let us calculate the distance between Edinburgh and London in km

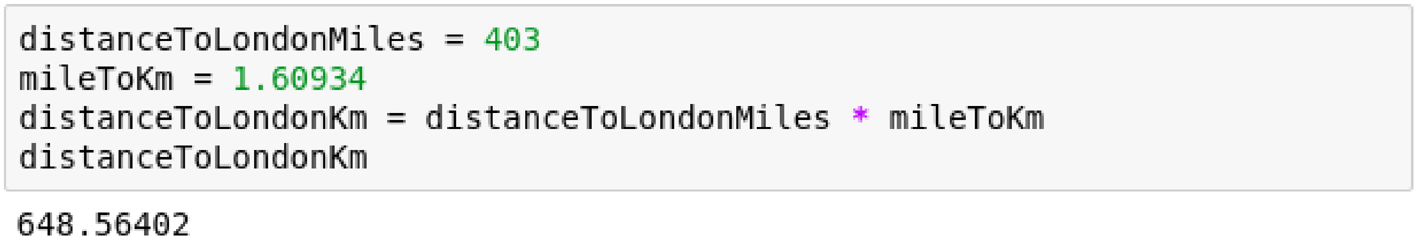


```
403 * 1.60934
```

```
648.56402
```

Variables

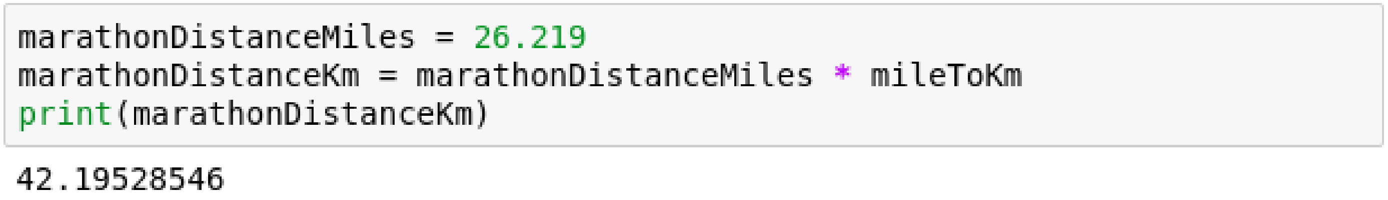
- Great calculator but how can we make it store values?
- Do this by defining variables
- Can later be called by the variable name
- Variable names are case sensitive and unique



```
distanceToLondonMiles = 403
mileToKm = 1.60934
distanceToLondonKm = distanceToLondonMiles * mileToKm
distanceToLondonKm
```

```
648.56402
```

We can now reuse the variable mileToKm in the next block without having to define it again!



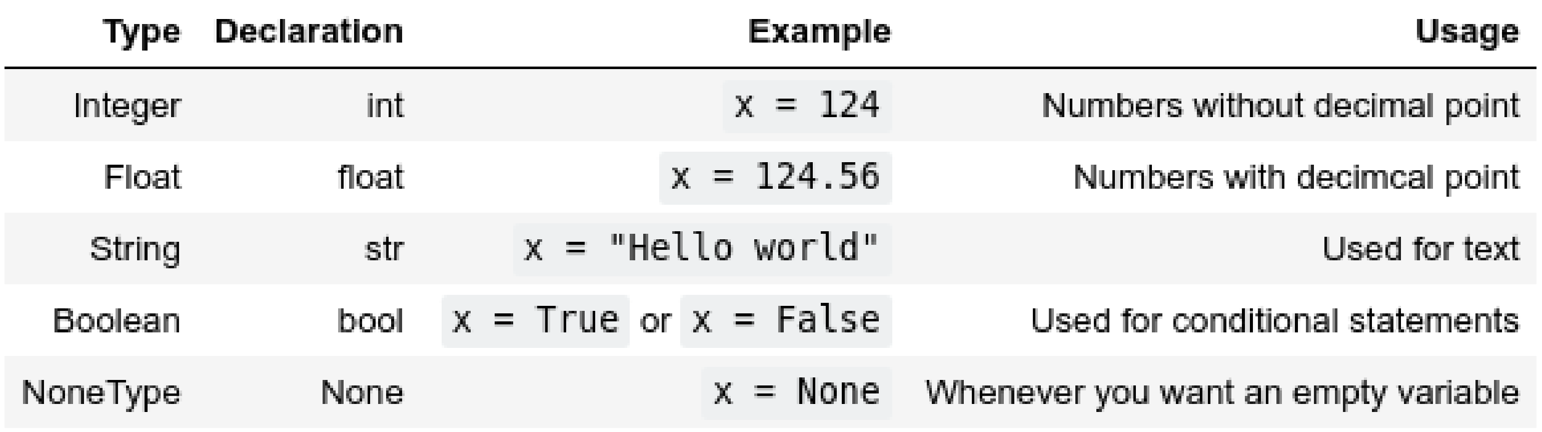
```
marathonDistanceMiles = 26.219
marathonDistanceKm = marathonDistanceMiles * mileToKm
print(marathonDistanceKm)
```

```
42.19528546
```

Types

Variables actually have a type, which defines the way it is stored.

The basic types are:



Type	Declaration	Example	Usage
Integer	int	`x = 124`	Numbers without decimal point
Float	float	`x = 124.56`	Numbers with decimcal point
String	str	`x = "Hello world"`	Used for text
Boolean	bool	`x = True` or `x = False`	Used for conditional statements
NoneType	None	`x = None`	Whenever you want an empty variable

Why should we care?

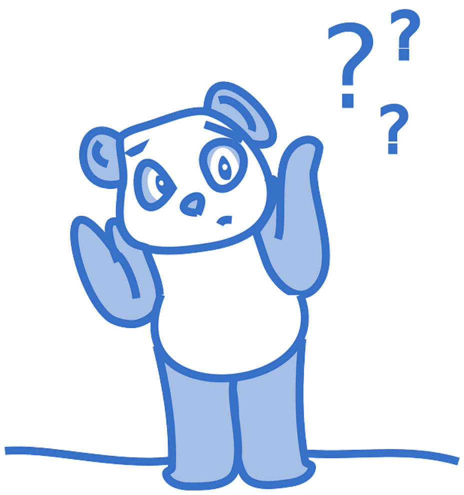
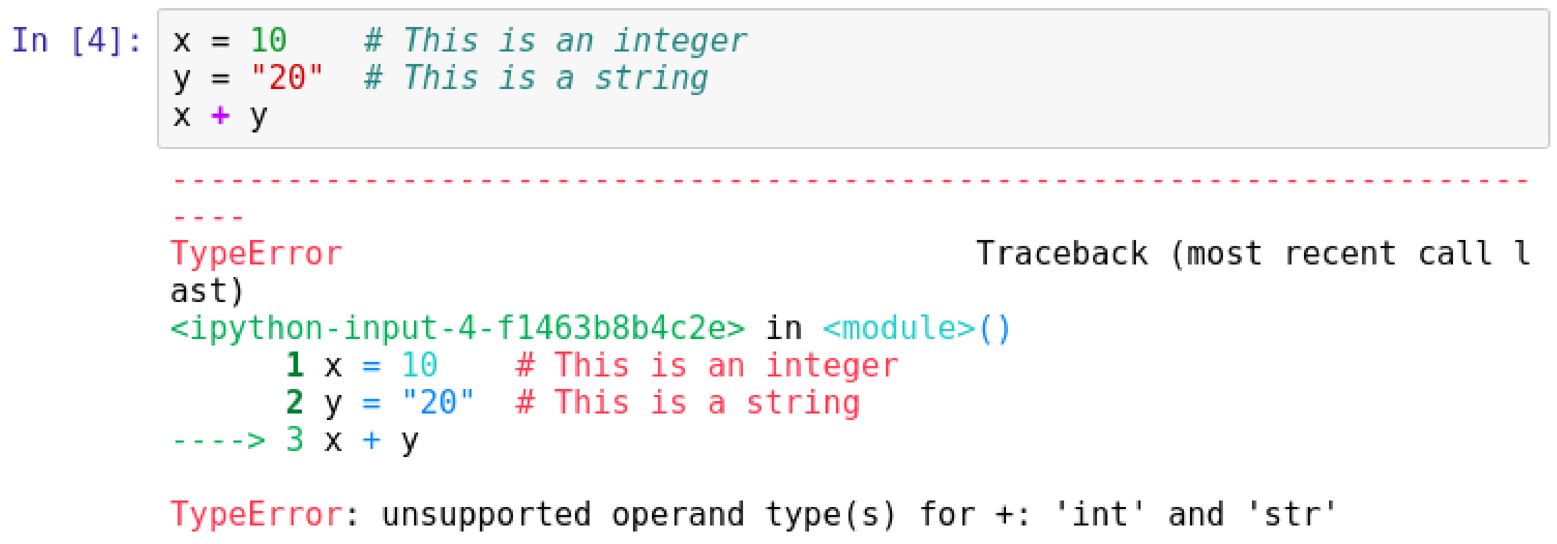


Image by Clker-Free-Vector-Images on Pixabay



```
In [4]: x = 10    # This is an integer
        y = "20"  # This is a string
        x + y
```

```
---------------------------------------------------------------------------
----
TypeError                                 Traceback (most recent call l
ast)
<ipython-input-4-f1463b8b4c2e> in <module>()
      1 x = 10    # This is an integer
      2 y = "20"  # This is a string
----> 3 x + y

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

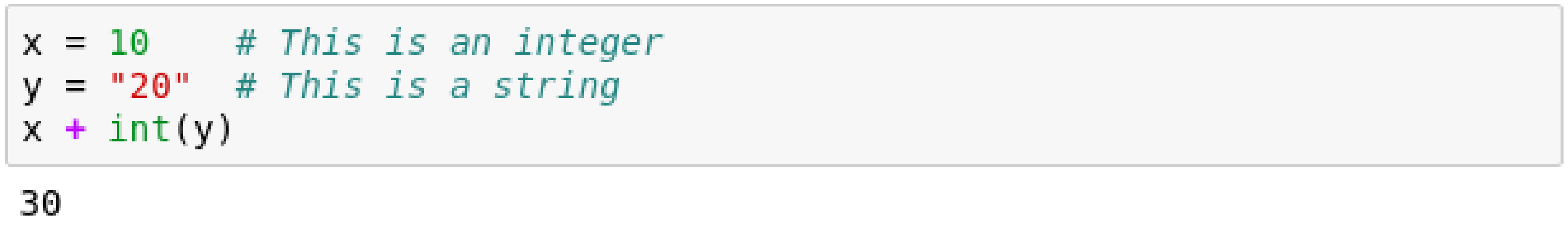
Important lesson to remember!
We can't do arithmetic operations on variables of different types. Therefore make sure that you are always aware of your variables types!

You can find the type of a variable using **type()**. For example type **type(x)**.

Casting types

Luckily Python offers us a way of converting variables to different types!

Casting – the operation of converting a variable to a different type



```
x = 10     # This is an integer
y = "20"   # This is a string
x + int(y)
```

```
30
```

Similar methods exist for other data types: **int()**, **float()**, **str()**

Quick quiz

```
x = "10"
y = "20"
x + y
```

What will be the result?

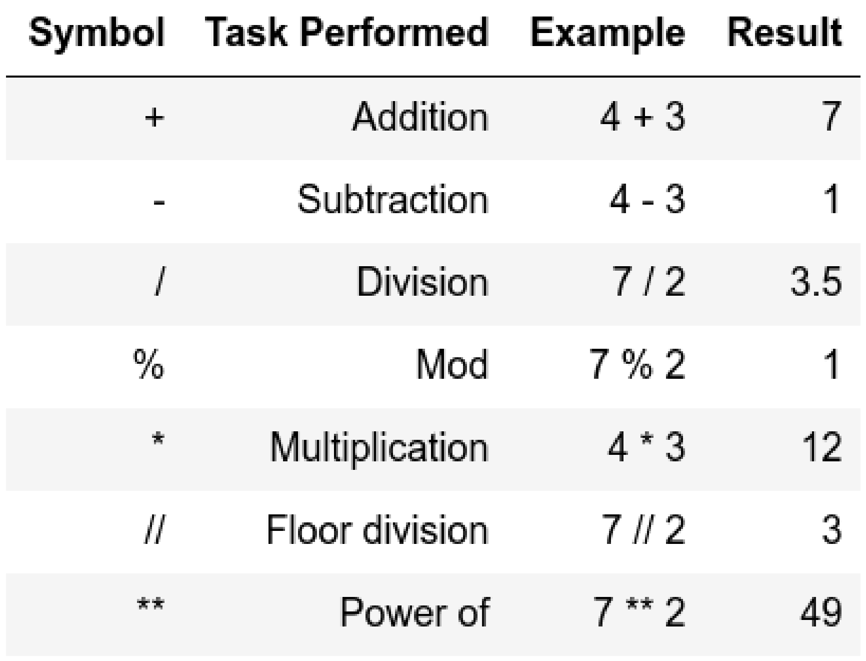
```
'1020'
```

Arithmetic operations

Similar to actual Mathematics.

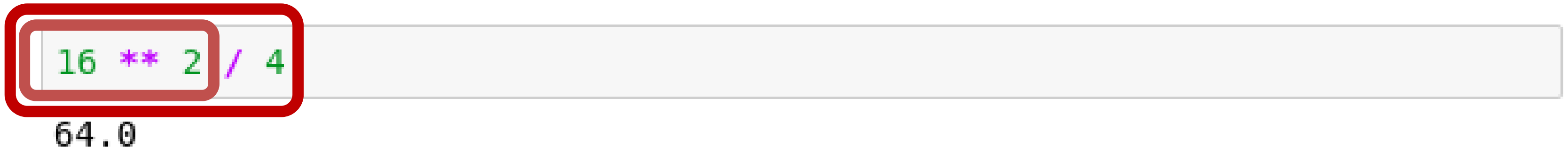
Order of precedence is the same as in Mathematics.

We can also use parenthesis ()



Symbol	Task Performed	Example	Result
+	Addition	4 + 3	7
-	Subtraction	4 - 3	1
/	Division	7 / 2	3.5
%	Mod	7 % 2	1
*	Multiplication	4 * 3	12
//	Floor division	7 // 2	3
**	Power of	7 ** 2	49

Order precedence example



```
16 ** 2 / 4
```

```
64.0
```

Quick quiz

```python
4 + 3 ** 2
```

13

vs

```python
(4 + 3) ** 2
```

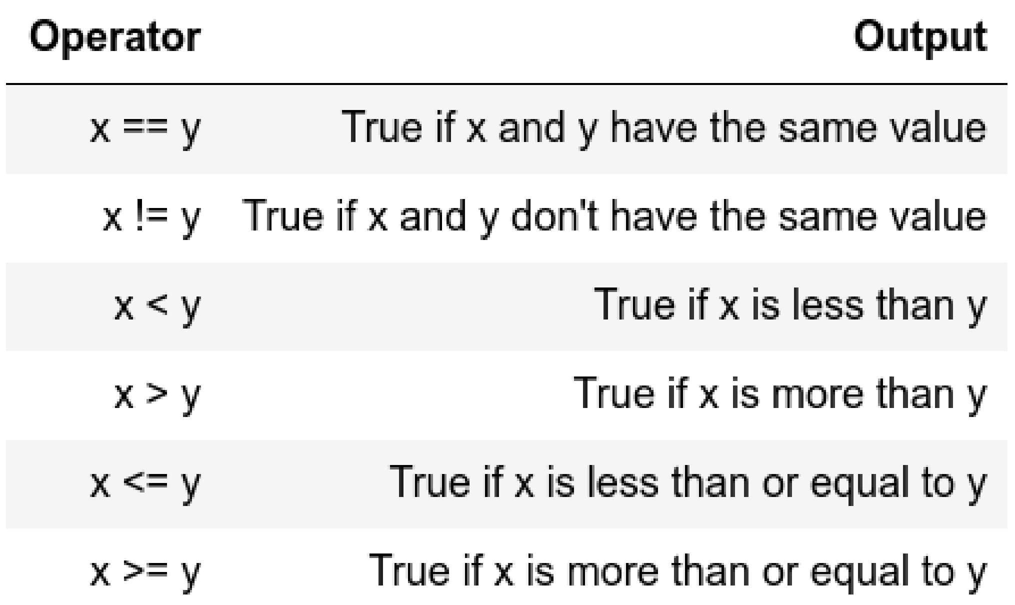
49

Comparison operators

- I.e. comparison operators
- Return Boolean values

(i.e. True or False)

- Used extensively for conditional statements



Operator	Output
x == y	True if x and y have the same value
x != y	True if x and y don't have the same value
x < y	True if x is less than y
x > y	True if x is more than y
x <= y	True if x is less than or equal to y
x >= y	True if x is more than or equal to y

Comparison examples

```
x = 5         # assign 5 to the variable x
x == 5        # check if value of x is 5
```

```
True
```

Note that `==` *is not the same as* `=`

```
x > 7
```

```
False
```

Logical operators

- Allows us to extend the conditional logic
- Will become essential later on

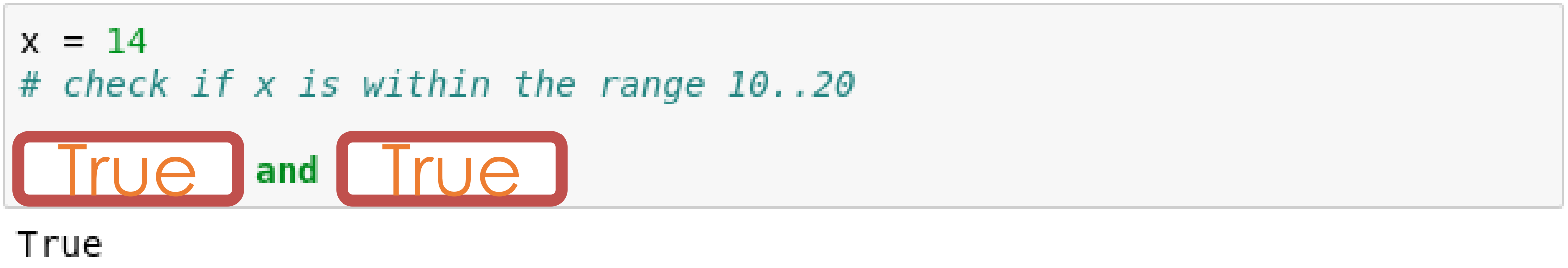
Operation	Result
x or y	True if at least on is True
x and y	True only if both are True
not x	True only if x is False

a	not a
False	True
True	False

a	b	a and b	a or b
False	False	False	False
False	True	False	True
True	False	False	True
True	True	True	True

Truth-table definitions of `bool` *operations*

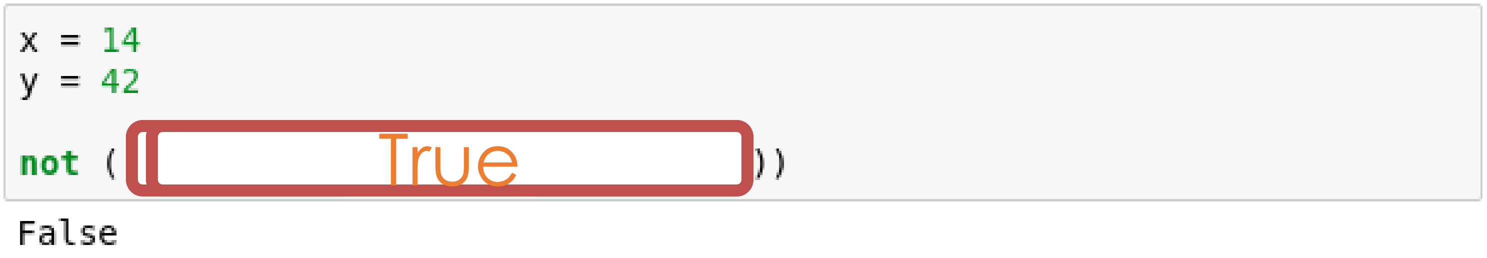
Combining both



```
x = 14
# check if x is within the range 10..20
True and True
```

```
True
```

Another example



```
x = 14
y = 42

not ( True ))
```

```
False
```

That wasn't very easy to read was it?
Is there a way we can make it more readable?

```python
x = 14
y = 42

xDivisible = ( x % 2 ) == 0 # check if x is a multiple of 2
yDivisible = ( y % 3 ) == 0 # check if y is a multiple of 3

not (xDivisible and yDivisible)
```

```
False
```

Strings

- Powerful and flexible in Python
- Can be added
- Can be multiplied
- Can be multiple lines

Strings

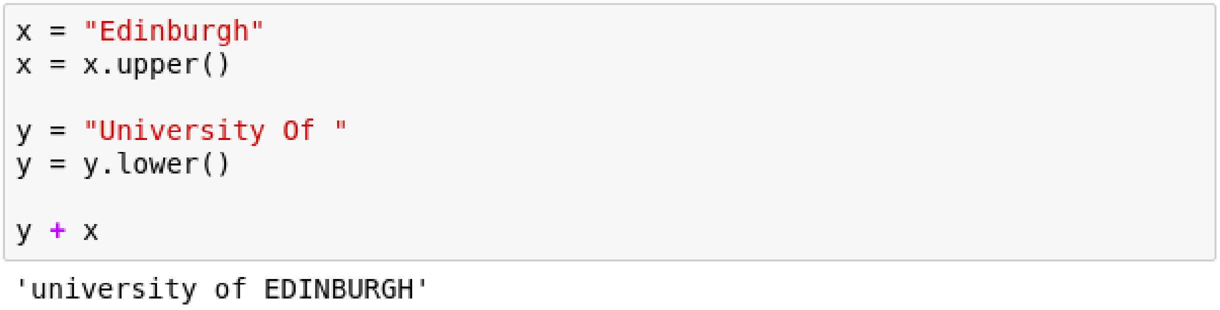
```
x = "Python"
y = "rocks"
x + " " + y
```

```
'Python rocks'
```

```
x = "This can be"
y = "repeated "
x + " " + y * 3
```

```
'This can be repeated repeated repeated '
```

Strings



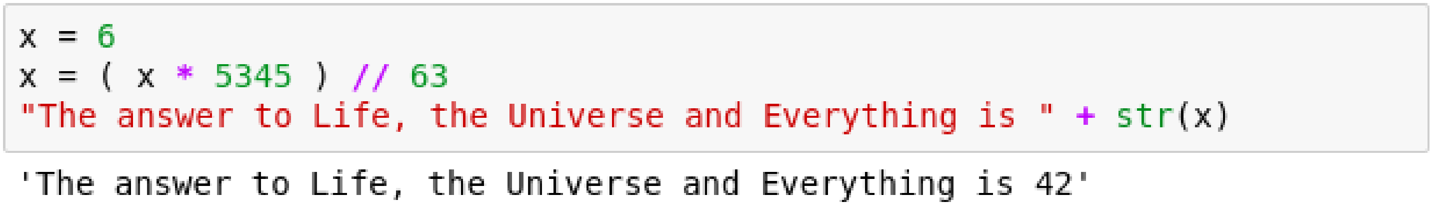
```
x = "Edinburgh"
x = x.upper()

y = "University Of "
y = y.lower()

y + x
```

```
'university of EDINBURGH'
```

These are called methods and add extra functionality to the String.
If you want to see more methods that can be applied to a string simply type in **dir('str')**

Mixing up strings and numbers

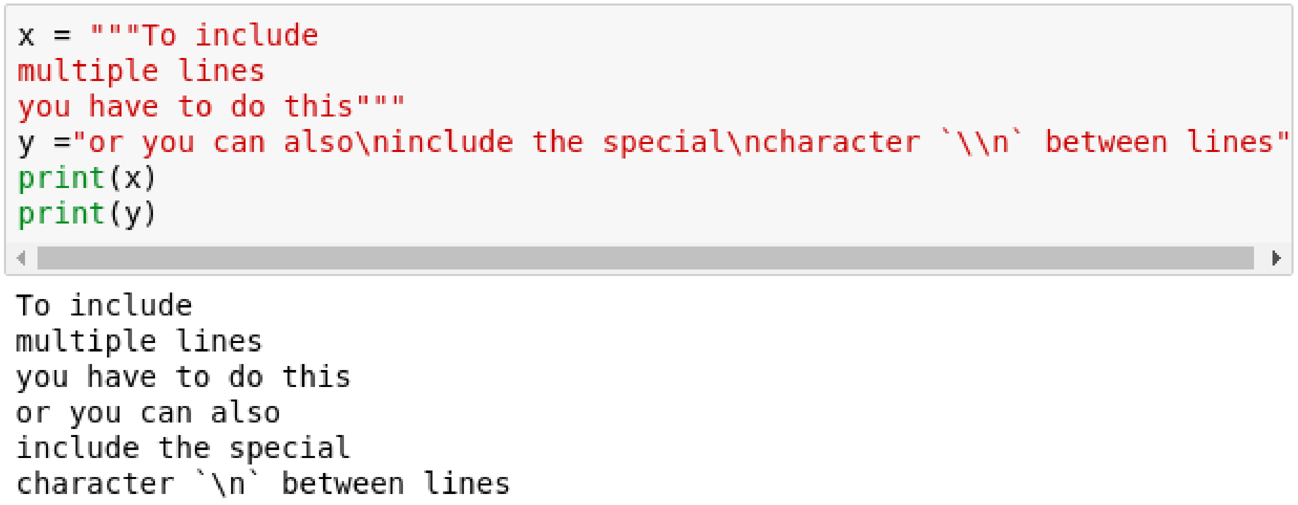
Often we would need to mix up numbers and strings. It is best to keep numbers as numbers (i.e. int or float) and cast them to strings whenever we need them as a string.



```
x = 6
x = ( x * 5345 ) // 63
"The answer to Life, the Universe and Everything is " + str(x)
```

```
'The answer to Life, the Universe and Everything is 42'
```

Multiline strings

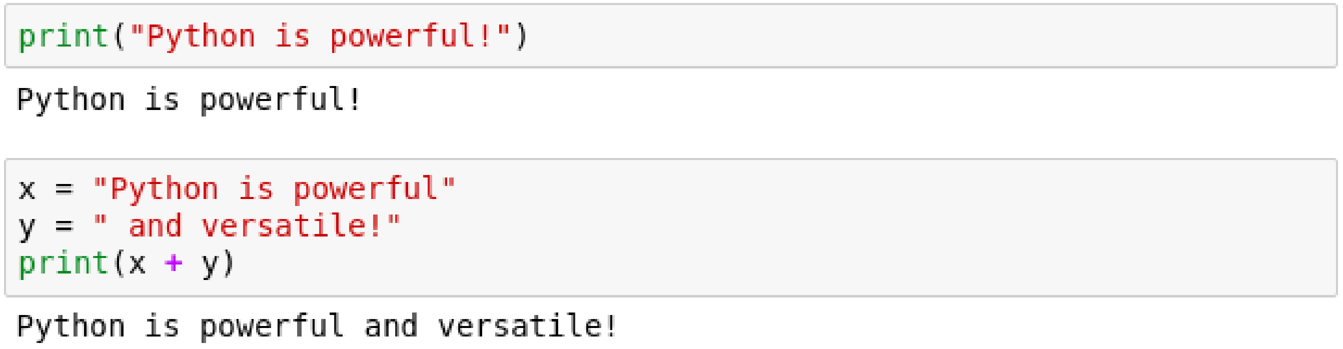


```python
x = """To include
multiple lines
you have to do this"""
y ="or you can also\ninclude the special\ncharacter `\\n` between lines"
print(x)
print(y)
```

```
To include
multiple lines
you have to do this
or you can also
include the special
character `\n` between lines
```

Printing

- When writing scripts, your outcomes aren't printed on the terminal.
- Thus, you must print them yourself with the print() function.
- Beware to not mix up the different type of variables!



```
print("Python is powerful!")
```

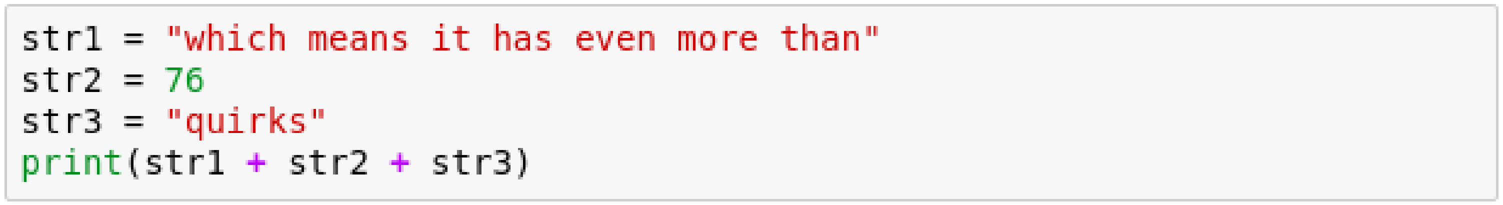
```
Python is powerful!
```

```
x = "Python is powerful"
y = " and versatile!"
print(x + y)
```

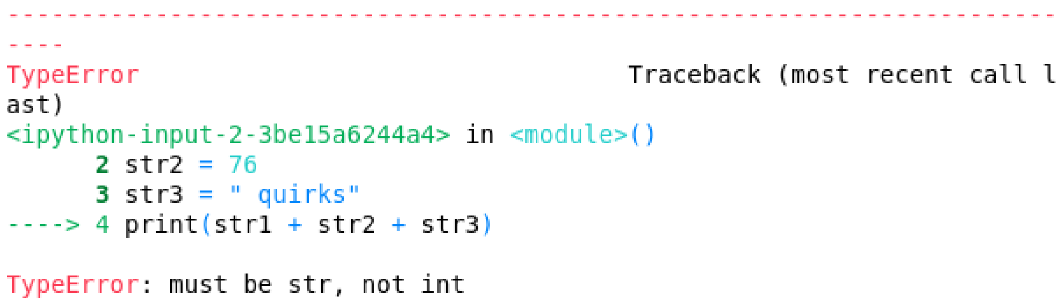
```
Python is powerful and versatile!
```

Quick quiz

Do you see anything wrong with this block?

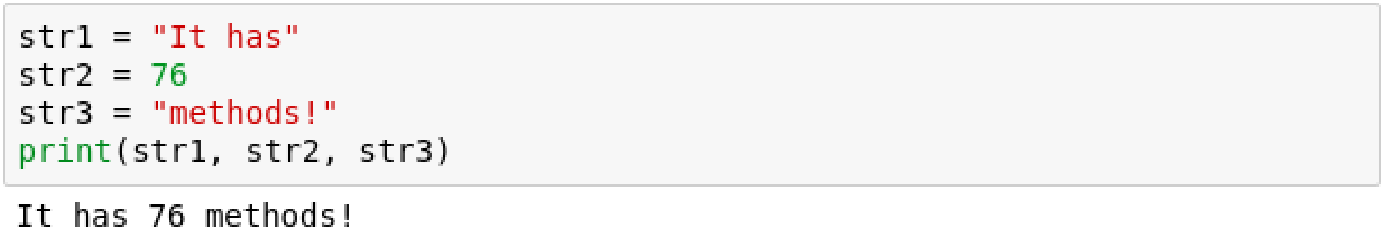


```
str1 = "which means it has even more than"
str2 = 76
str3 = "quirks"
print(str1 + str2 + str3)
```



```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call l
ast)
<ipython-input-2-3be15a6244a4> in <module>()
      2 str2 = 76
      3 str3 = " quirks"
----> 4 print(str1 + str2 + str3)

TypeError: must be str, not int
```

Another more generic way to fix it



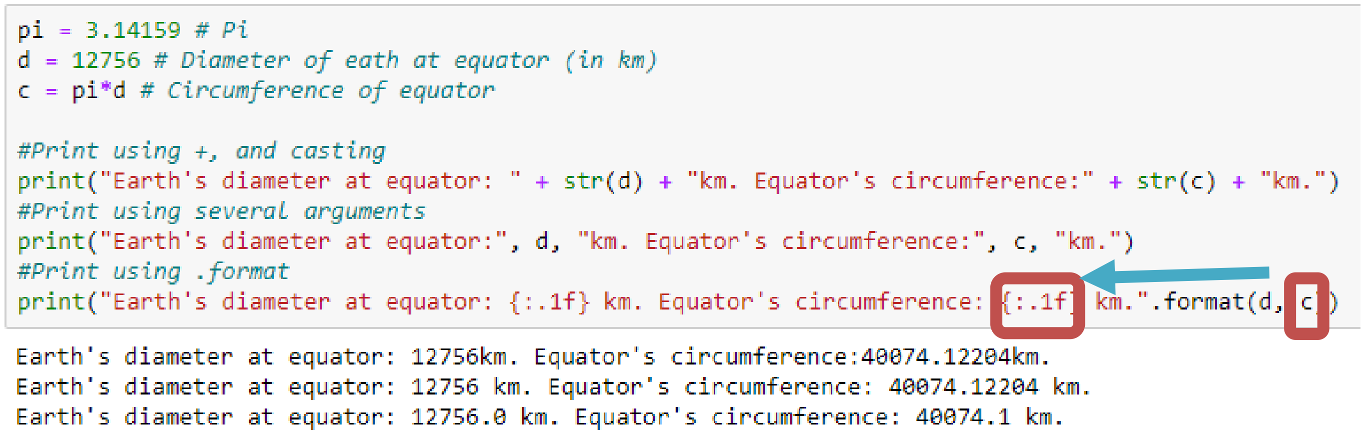
```
str1 = "It has"
str2 = 76
str3 = "methods!"
print(str1, str2, str3)
```

```
It has 76 methods!
```

If we comma separate statements in a print function we can have different variables printing!

Placeholders

- A way to interleave numbers is



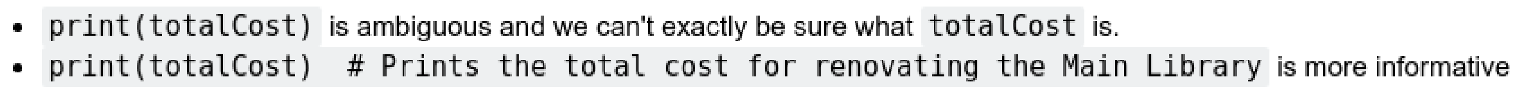
```
pi = 3.14159 # Pi
d = 12756 # Diameter of eath at equator (in km)
c = pi*d # Circumference of equator

#Print using +, and casting
print("Earth's diameter at equator: " + str(d) + "km. Equator's circumference:" + str(c) + "km.")
#Print using several arguments
print("Earth's diameter at equator:", d, "km. Equator's circumference:", c, "km.")
#Print using .format
print("Earth's diameter at equator: {:.1f} km. Equator's circumference: {:.1f} km.".format(d, c))
```

```
Earth's diameter at equator: 12756km. Equator's circumference:40074.12204km.
Earth's diameter at equator: 12756 km. Equator's circumference: 40074.12204 km.
Earth's diameter at equator: 12756.0 km. Equator's circumference: 40074.1 km.
```

- Elegant and easy
- more in your notes

Commenting

- Useful when your code needs further explanation. Either for your future self and anybody else.
- Useful when you want to remove the code from execution but not permanently
- Comments in Python are done with #



- `print(totalCost)` is ambiguous and we can't exactly be sure what `totalCost` is.
- `print(totalCost)  # Prints the total cost for renovating the Main Library` is more informative