



**COMPASS**

Community Platform for Agricultural Sciences

# Introduction to python

Marco Lopez · Viviana Ortiz

19 Julio 2021

Slides modified from:  
Open Educational Resources. The University of Edinburgh  
Licensed under a CC BY 4.0 license  
<https://open.ed.ac.uk/5861-2/>

# Structure

- Divided into two 1h sessions.
- 10-minute break in between.
  
- Try to follow within your notebook and run all examples shown on the slides.

# 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 ask.
- During exercises (you can also ask others).



Image by [mohamed Hassan from Pixabay](#)



[Python natalensis](#) by [A. Smith](#) on Wikimedia Commons

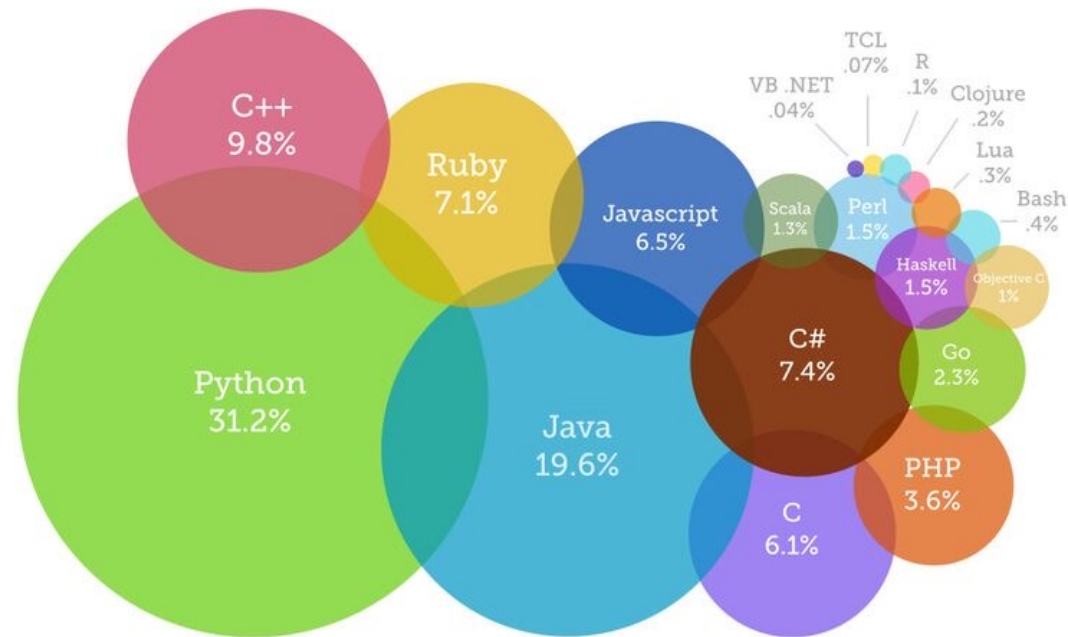
# History

- Started by Guido Van Rossum as a hobby
- Now widely spread
- Open Source! Free!
- Versatile



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

Most popular coding languages 2020



# 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

# The Anaconda Ecosystem



Sign in

Home

Environments

Learning

Community

ANACONDA NUCLEUS  
Join Now

Discover premium data science content

Documentation

Anaconda Blog

Twitter YouTube GitHub

Applications on  Channels Refresh

 <b>Datalore</b> Online Data Analysis Tool with smart coding assistance by JetBrains. Edit and run your Python notebooks in the cloud and share them with your team. <a href="#">Launch</a>	 <b>IBM Watson Studio Cloud</b> IBM Watson Studio Cloud provides you the tools to analyze and visualize data, to cleanse and shape data, to create and train machine learning models. Prepare data and build models, using open source data science tools or visual modeling. <a href="#">Launch</a>	 <b>JupyterLab</b> 3.0.14 An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture. <a href="#">Launch</a>	 <b>Jupyter Notebook</b> 6.3.0 Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis. <a href="#">Launch</a>
 <b>Qt Console</b> 5.0.3 PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more. <a href="#">Launch</a>	 <b>Spyder</b> 4.2.5 Scientific PYTHON Development Environment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features <a href="#">Launch</a>	 <b>Glueviz</b> 1.0.0 Multidimensional data visualization across files. Explore relationships within and among related datasets. <a href="#">Install</a>	 <b>Orange 3</b> 3.26.0 Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox. <a href="#">Install</a>



# Jupyter notebooks

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



The screenshot shows the Jupyter web interface. At the top left is the Jupyter logo and name. On the top right are 'Quit' and 'Logout' buttons. Below the header are tabs for 'Files', 'Running', and 'Clusters'. A message says 'Select items to perform actions on them.' with 'Upload', 'New', and a refresh icon to the right. The main area shows a file browser for the path '/ Documents / Projects / UIS\_dataanalysis\_talleres / Python-basics-main'. It contains a table of files:

	Name	Last Modified	File size
<input type="checkbox"/>	..	seconds ago	
<input type="checkbox"/>	latex-example.ipynb	4 days ago	1.89 kB
<input type="checkbox"/>	lib-versions.ipynb	5 days ago	3.23 kB
<input type="checkbox"/>	minicurso-analisis_de_datos-01-intro.ipynb	4 days ago	13.5 kB
<input type="checkbox"/>	minicurso-analisis_de_datos-02-python_basico.ipynb	4 days ago	40.6 kB
<input type="checkbox"/>	minicurso-analisis_de_datos-03.1-bibliotecas-manipulacion_de_datos-numpy.ipynb	4 days ago	11.4 kB

# 1. Start Anaconda



Sign in

Applications on  Channels Refresh

Application	Version	Action
Datalore		Launch
IBM Watson Studio Cloud		Launch
JupyterLab	3.0.14	Launch
Jupyter Notebook	6.3.0	Launch
Qt Console	5.0.3	Launch
Spyder	4.2.5	Launch
Glueviz	1.0.0	Install
Orange 3	3.26.0	Install

# 2. Download GitRepo

MarcoLopez / Python-basics

Unwatch 2 Star 0 Fork 0

Code Issues Pull requests Actions Projects Wiki Security Insights

main Python-basics / minicurso-analisis\_de\_datos-01-intro.ipynb Go to file

vivianaortizl Update reviewed files Latest commit 11db9db 4 days ago History

1 contributor

465 lines (465 sloc) 13.1 KB

Raw Blame

### Minicurso Analisis de Datos usando Python

#### Parte 1 - Introduccion

main 1 branch 0 tags

Go to file Add file

Code

MarcoLopez Update machine\_learning.md 323c41e 4 hours ago 41 commits

Categories-of-Machine-Learning.jpg Add files via upload 6 hours ago



### 3. Starting a notebook



Files Running Clusters

Select items to perform actions on them.

0 / Documents / Projects / UIS

- ..
- Intro\_python\_ppt
- Python-basics-main**
- 2021\_jul\_dic\_UIS\_talleres.xlsx



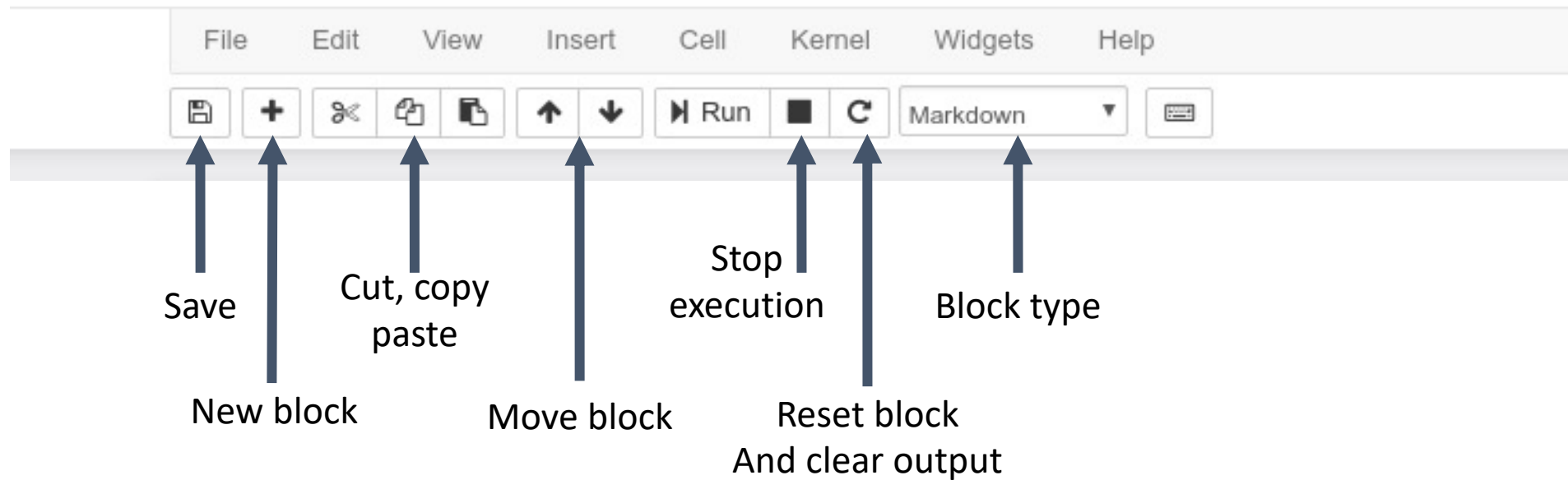
Files Running Clusters

Select items to perform actions on them.

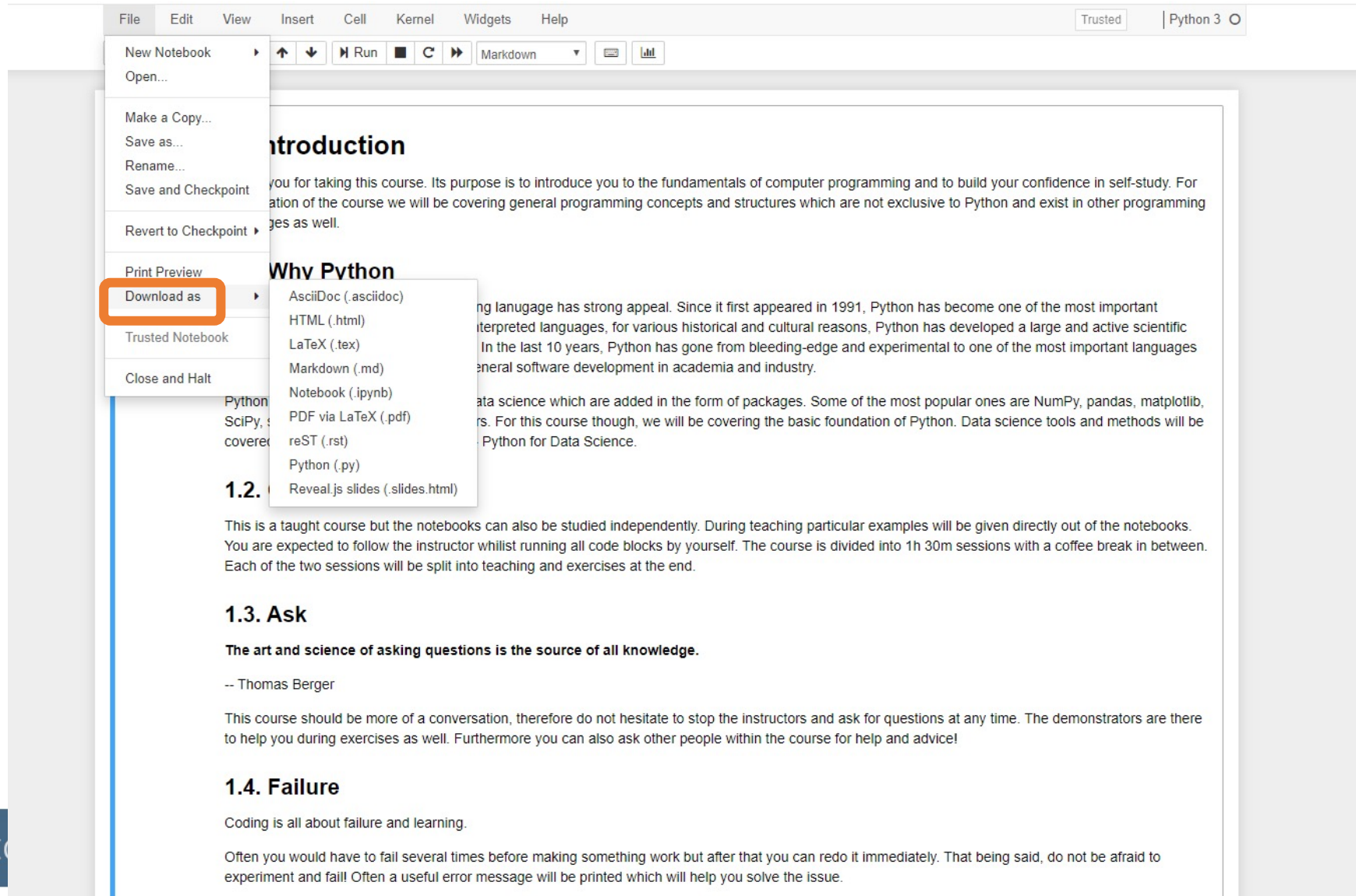
0 / Documents / Projects / UIS\_dataanalysis\_talleres / Python-basics-main

- ..
- latex-example.ipynb
- lib-versions.ipynb
- minicurso-analisis\_de\_datos-01-intro.ipynb
- minicurso-analisis\_de\_datos-02-python\_basico.ipynb**
- minicurso-analisis\_de\_datos-03.1-bibliotecas-manipulacion\_de\_datos-numpy.ipynb
- minicurso-analisis\_de\_datos-03.2-bibliotecas-manipulacion\_de\_datos-matplotlib.ipynb
- minicurso-analisis\_de\_datos-03.3-bibliotecas-manipulacion\_de\_datos-pandas.ipynb

# 4. Toolbar



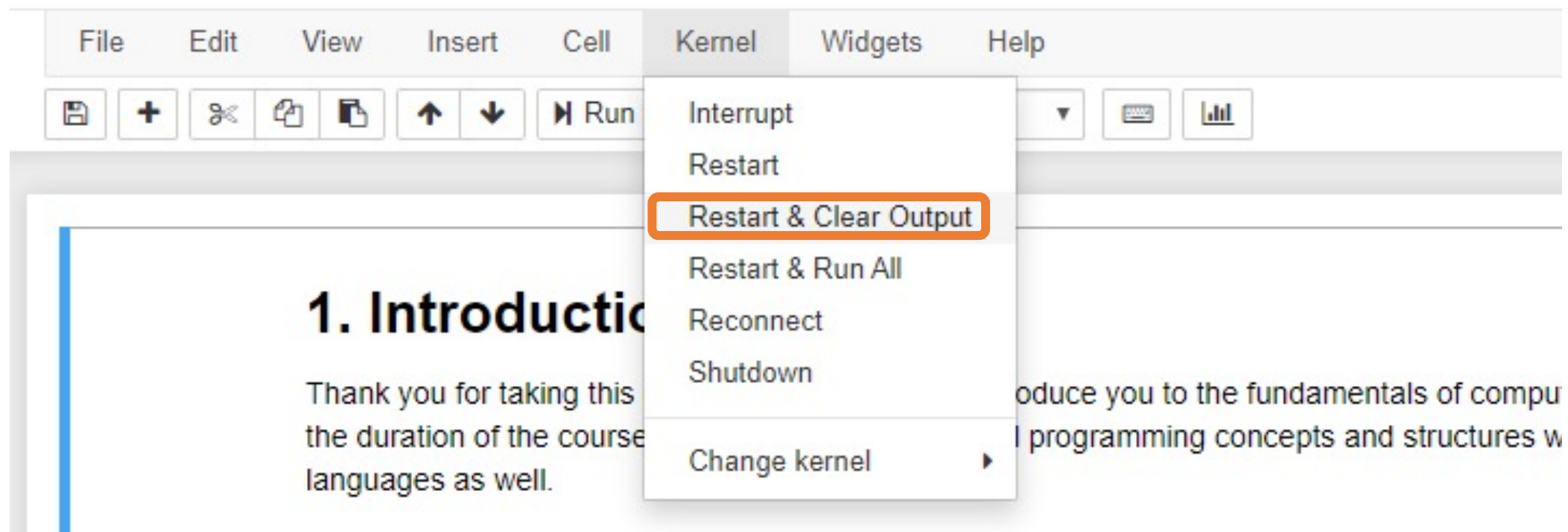
# 5. Download files



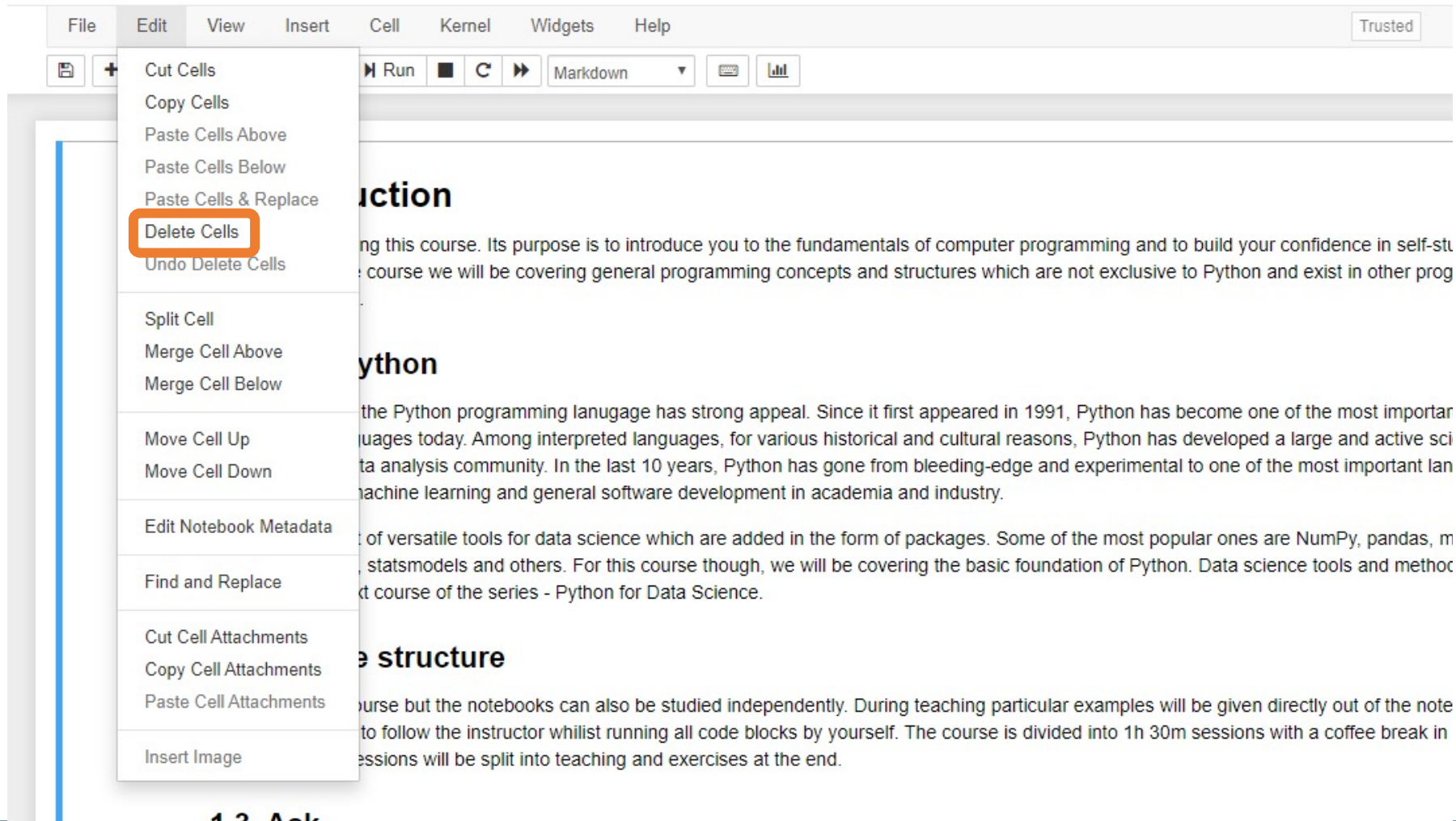
The screenshot shows a Jupyter Notebook interface with a 'File' menu open. The 'Download as' option is highlighted with an orange box. A sub-menu is visible, listing various file formats for download: AsciiDoc (.asciidoc), HTML (.html), LaTeX (.tex), Markdown (.md), Notebook (.ipynb), PDF via LaTeX (.pdf), reST (.rst), Python (.py), and Reveal.js slides (.slides.html). The notebook content in the background includes sections on 'Introduction', 'Why Python', and '1.2. This is a taught course but the notebooks can also be studied independently...'. The interface also shows a 'Trusted' status and 'Python 3' kernel.



# 6. Kernel/Restart & Clear output

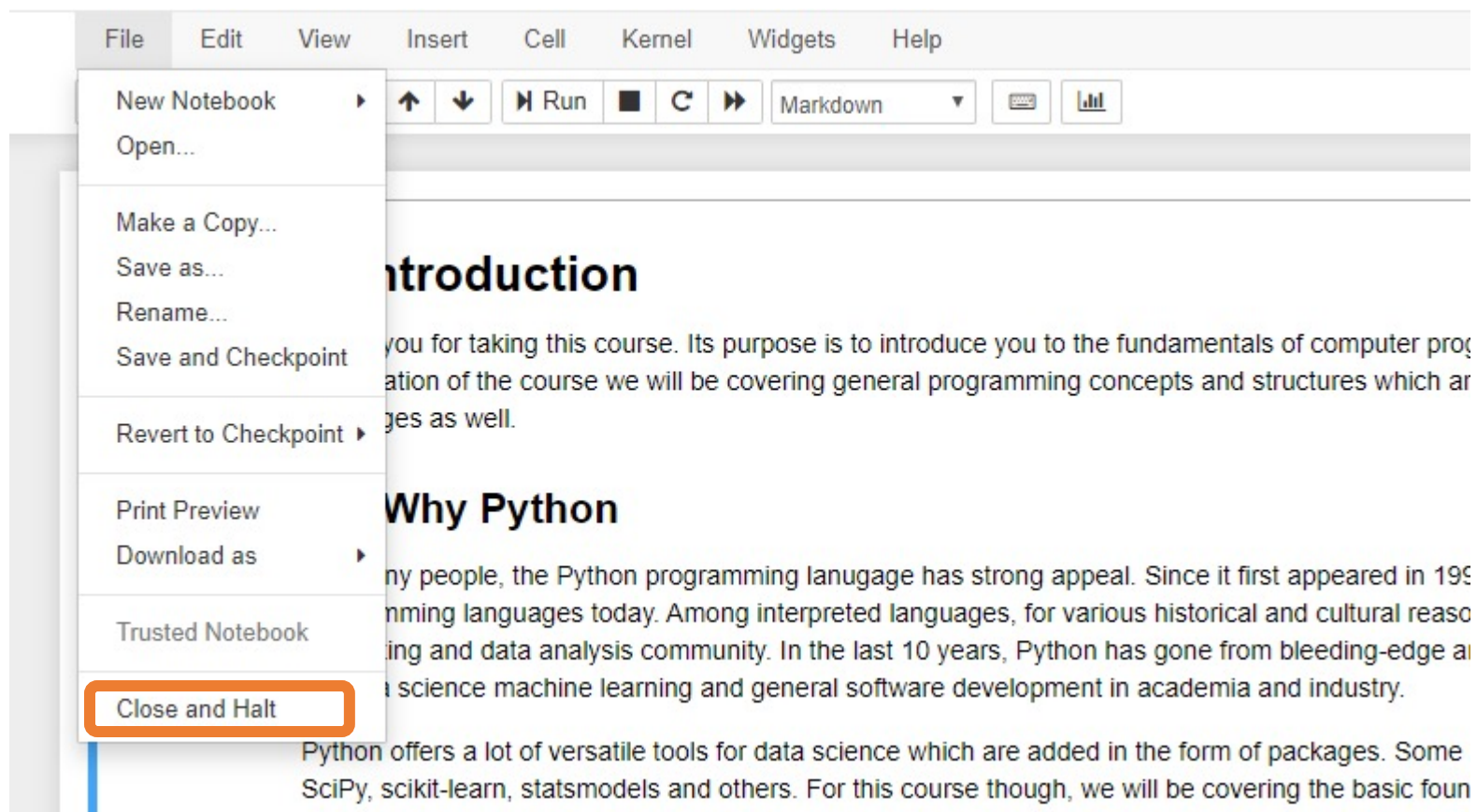


# 7. Edit/Delete Cell



The image shows a Jupyter Notebook interface. The top menu bar includes 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. A 'Trusted' badge is visible in the top right corner. The 'Edit' menu is open, displaying various options. The 'Delete Cells' option is highlighted with an orange border. Other options in the menu include 'Cut Cells', 'Copy Cells', 'Paste Cells Above', 'Paste Cells Below', 'Paste Cells & Replace', 'Undo Delete Cells', 'Split Cell', 'Merge Cell Above', 'Merge Cell Below', 'Move Cell Up', 'Move Cell Down', 'Edit Notebook Metadata', 'Find and Replace', 'Cut Cell Attachments', 'Copy Cell Attachments', 'Paste Cell Attachments', and 'Insert Image'. The background shows a notebook cell with text, including the word 'Introduction' and 'Python'.

# 8. File/ Close & Halt



## 9. Create a folder

jupyter

Files Running

Select items to perform actions on them.

0

The notebook list is empty.

Upload New

Notebook:

- Python 3
- R

Other:

- Text File
- Folder
- Terminal

## 10. Rename

jupyter

Files Running

Rename Move

1

Untitled Folder

seconds ago

Upload New

Name Last Modified

# 11. Upload files



Files **Running**

Select items to perform actions on them.

**Upload** New ↕ ↻

0 / Introduction to Python

Name	Last Modified
..	seconds ago
The notebook list is empty.	



Files **Running**

Select items to perform actions on them.

Upload New ↕ ↻

0 / Introduction to Python

Name	Last Modified
The notebook list is empty.	
python-intro-0.ipynb	
python-intro-1.ipynb	
python-intro-2.ipynb	
..	seconds ago
python-intro-exercises.ipynb	

# Running blocks

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



# Other operations

- File/Save and Checkpoint
- File/Revert to Checkpoint
- Tab completion
- Introspection

# 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	<code>int</code>	<code>x = 124</code>	Numbers without decimal point
Float	<code>float</code>	<code>x = 124.56</code>	Numbers with decimal point
String	<code>str</code>	<code>x = "Hello world"</code>	Used for text
Boolean	<code>bool</code>	<code>x = True</code> or <code>x = False</code>	Used for conditional statements
NoneType	<code>None</code>	<code>x = None</code>	Whenever you want an empty variable

Why should we care?

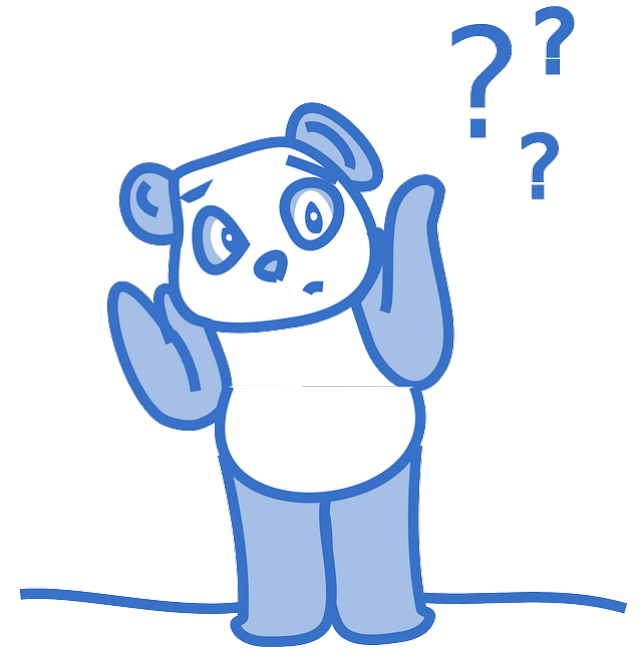


Image by [Clker-Free-Vector-Images on Pixabay](#)

```
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

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

13

vs

```
(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 one is True
x and y	True only if both are True
not x	True only if x is False

a	not a	a	b	a and b	a or b
False	True	False	False	False	False
True	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

# 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

```
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

# Exercise time

Simple exercises (notebooks minicurso 02 and 03)  
10-minute break afterwards.

Failure is progress!

Ask us anything. Ask among yourselves as well.

