

Introduction to python

Part 1

Administration

Fire alarm

- Fire exit to the rear of the Meeting Suite(Argyle House)/Outside the café area(JCMB)

Toilets

- Near the building entrance(Argyle House)/Near the café area(JCMB)

Refreshments

- Reception (Argyle House)/Café(JCMB)

Structure

- Divided into two 1.5h sessions.
- 20-minute coffee break in between.
- Each session has 1h of teaching and half an hour of exercises.
- 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](#)

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.



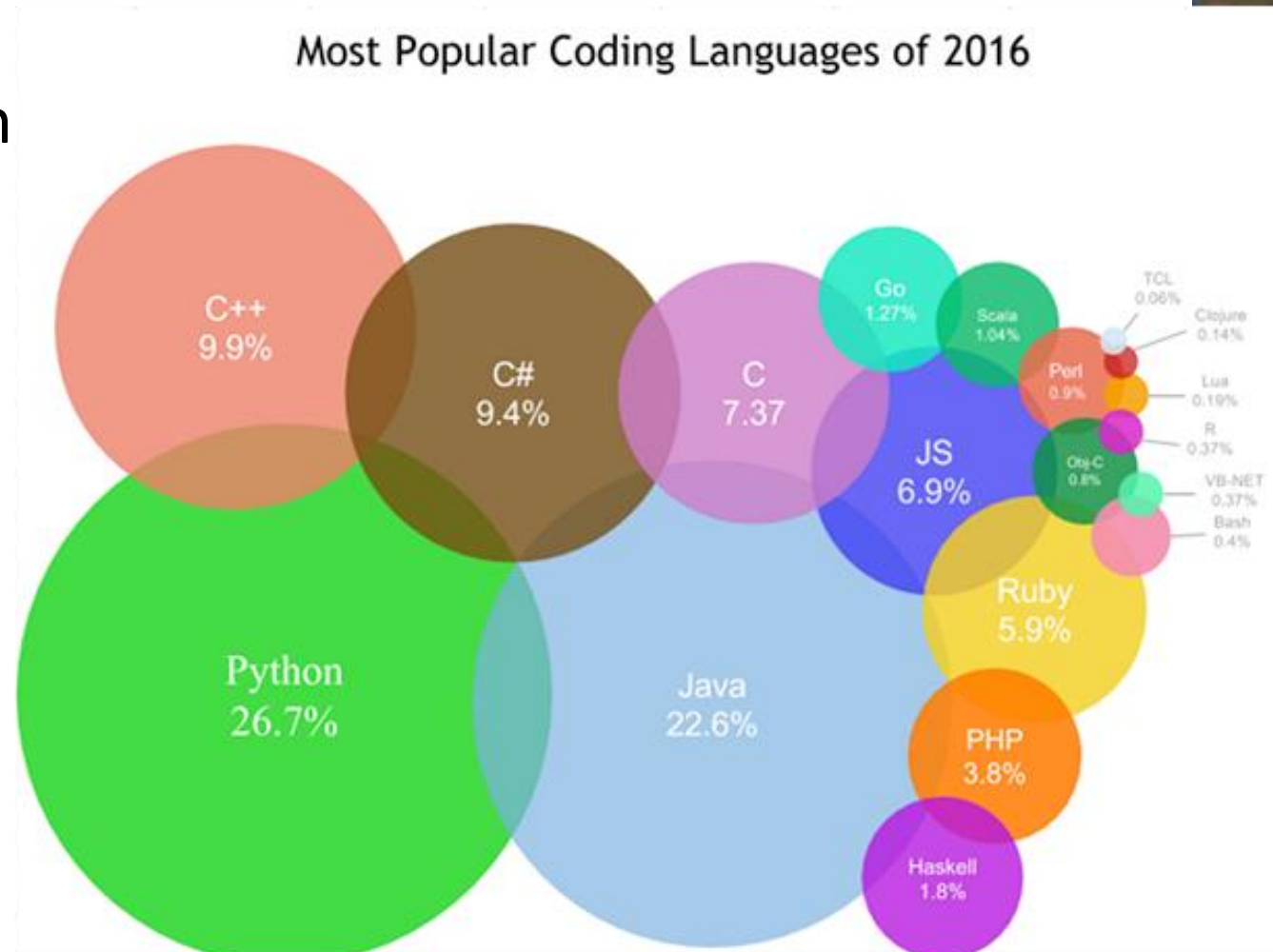
[Python natalensis by A. Smith](#) on Wikimedia Commons

History

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

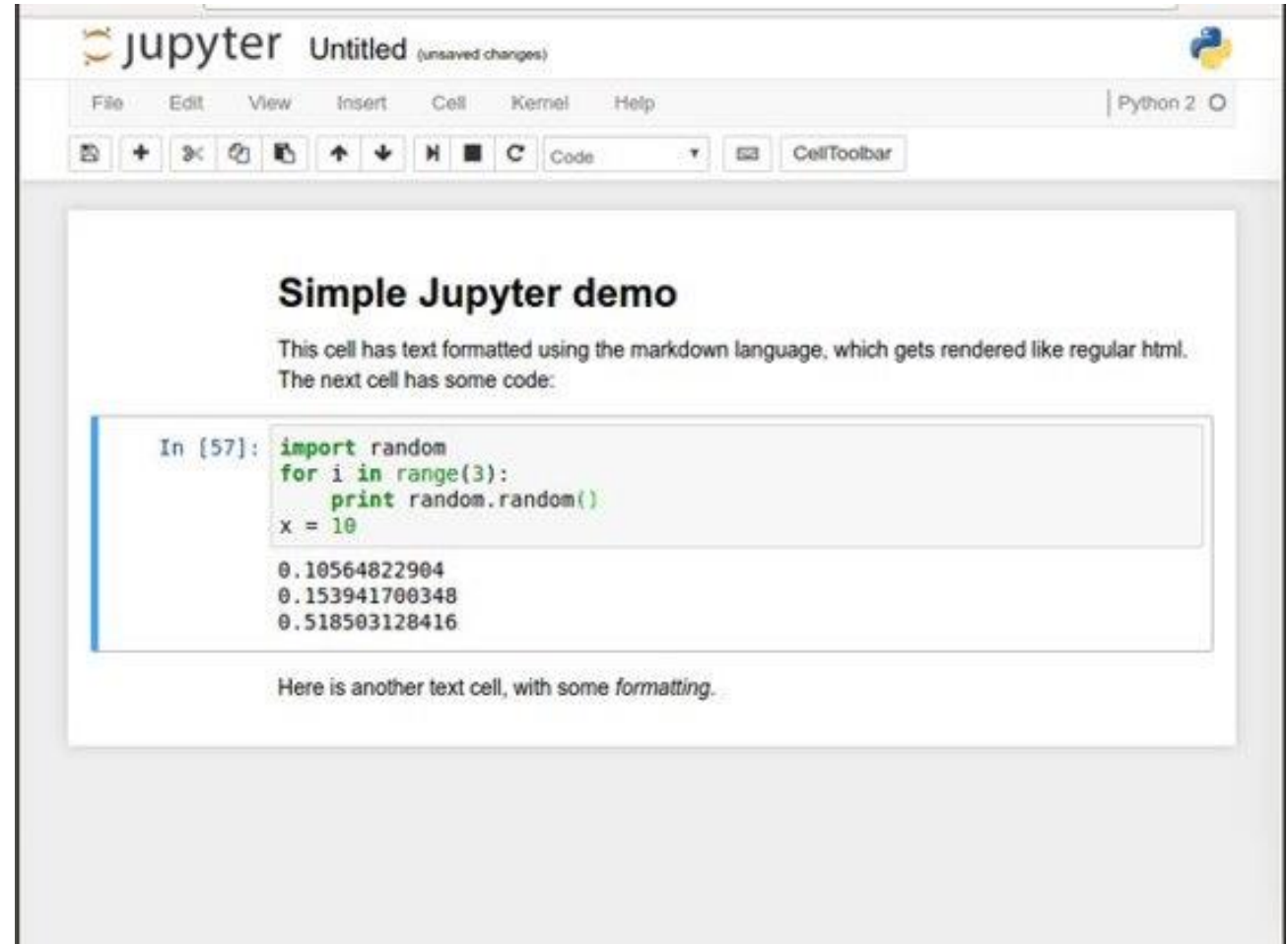
Then we must write text files in .py

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



1. Start Noteable

Introduction to Python

Course Content

Course Management

Control Panel

Content Collection

Course Tools

Evaluation

Grade Centre

Users and Groups

Customisation

Packages and Utilities

Help

Course Content

Welcome

Welcome to the Introduction to Python LEARN course, run by the Digital Skills and Training team in 1... build your confidence in self-study. For the duration of the course, we will be covering general progr...

Please note that this is an extra-curricular course, you won't receive any credits for taking it but you v...

For many people, the Python programming language has strong appeal. Since it first appeared in 19... reasons, Python has developed a large and active scientific computing and data analysis community... and general software development in academia and industry. Python offers a lot of versatile tools fo... statsmodels and others. For this course though, we will be covering the basic foundations of Python. This is a taught, training room course but the notebooks can also be studied independently. During t... yourself. The classroom course is divided into 1h 30m sessions with a coffee break in between. Each...

This course uses the Noteable service as a teaching environment. It provides a user-friendly and has... have to upload the course material to Noteable. Please follow the instructions below to set up your t...

Setup Instructions

To be ready for the course, first you will need to set up your Noteable environment.

1. **Open Noteable** by the link provided on this page. It is not necessary to share any of your details.
2. You will now be taken to your Jupyter Homepage.
3. Click the button **+GitRepo** near the top right.
4. Type in
`https://git.ecdf.ed.ac.uk/digital-skills/python-intro`
and click **OK**.

Now if you close this notebook and return to your Jupyter Homepage you will see a new folder has a

Noteable

Web-based Python development environment

Lectures

my Learn

Self-Enrol

help

Introduction to Python

Course Content > Noteable

Edit Mode is: OFF

Connect to Noteable

* Indicates unsaved changes.

PERSONAL DATA

Send your user ID? ☒

Send your name? ☐
Check the box to send your name to the external tool.

Send your email address? ☐
Check the box to send your email address to the external tool.

Click **Submit** to proceed. Click **Cancel** to go back.

Cancel

Submit



Edina

Start Personal Container

Standard Notebook ▼

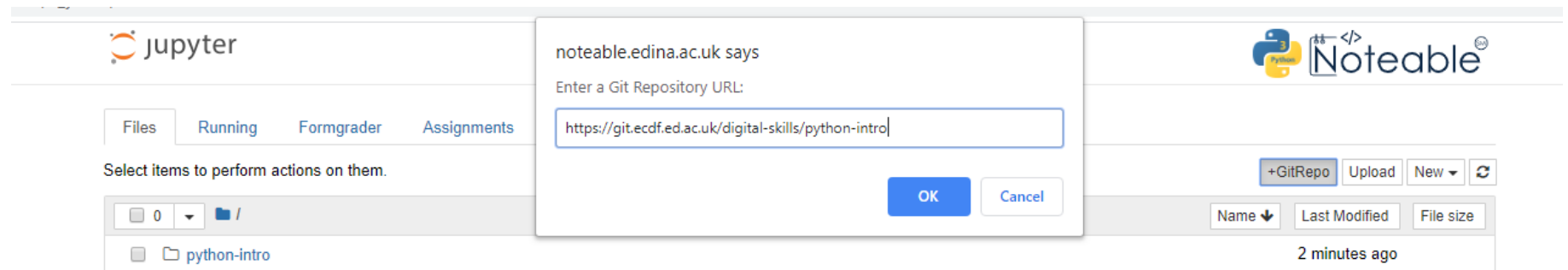
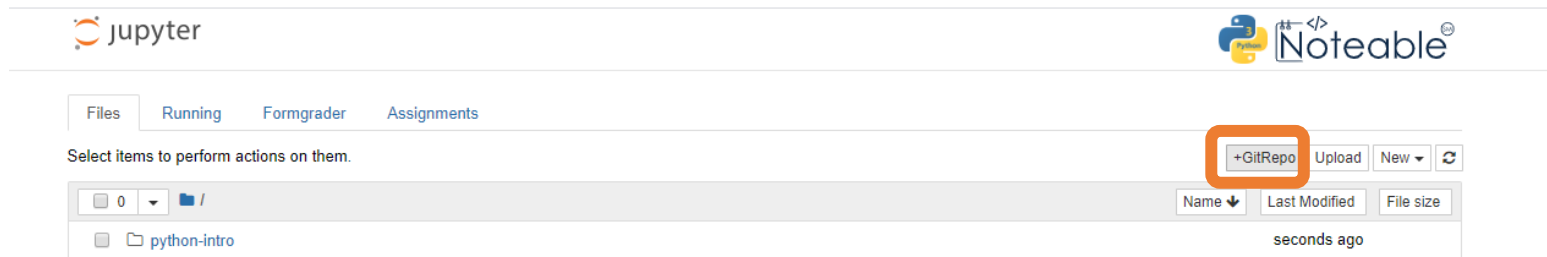
Start

You have a running container:

Reconnect

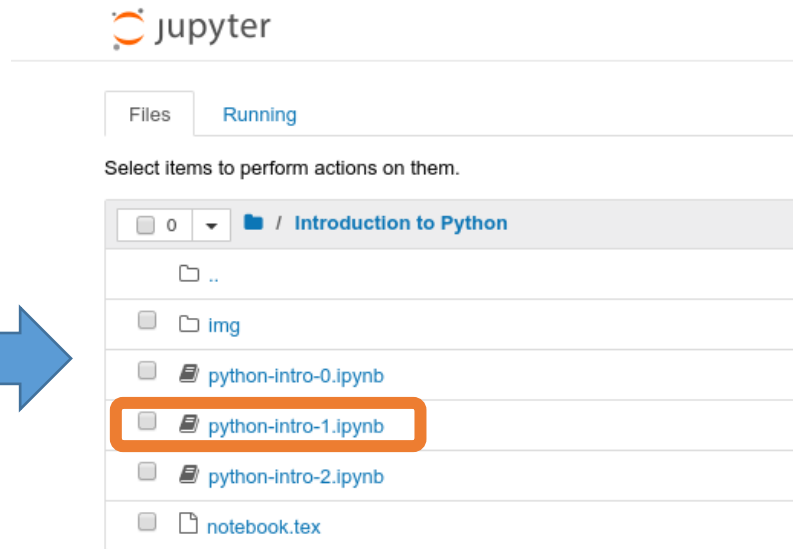
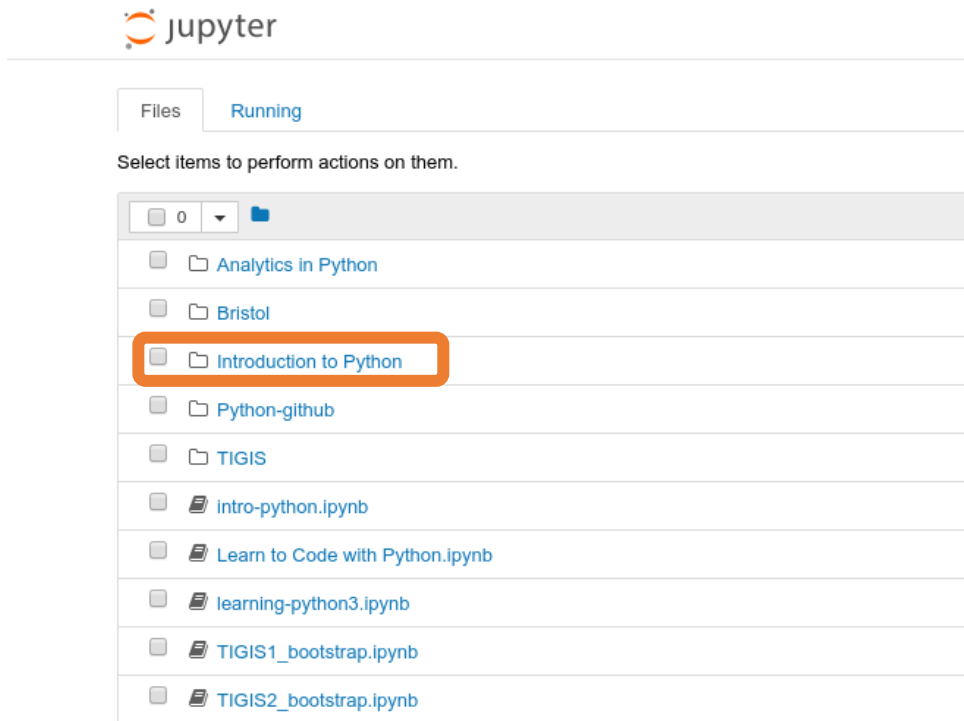
Shut Down

2. Clone GitRepo(recommended)

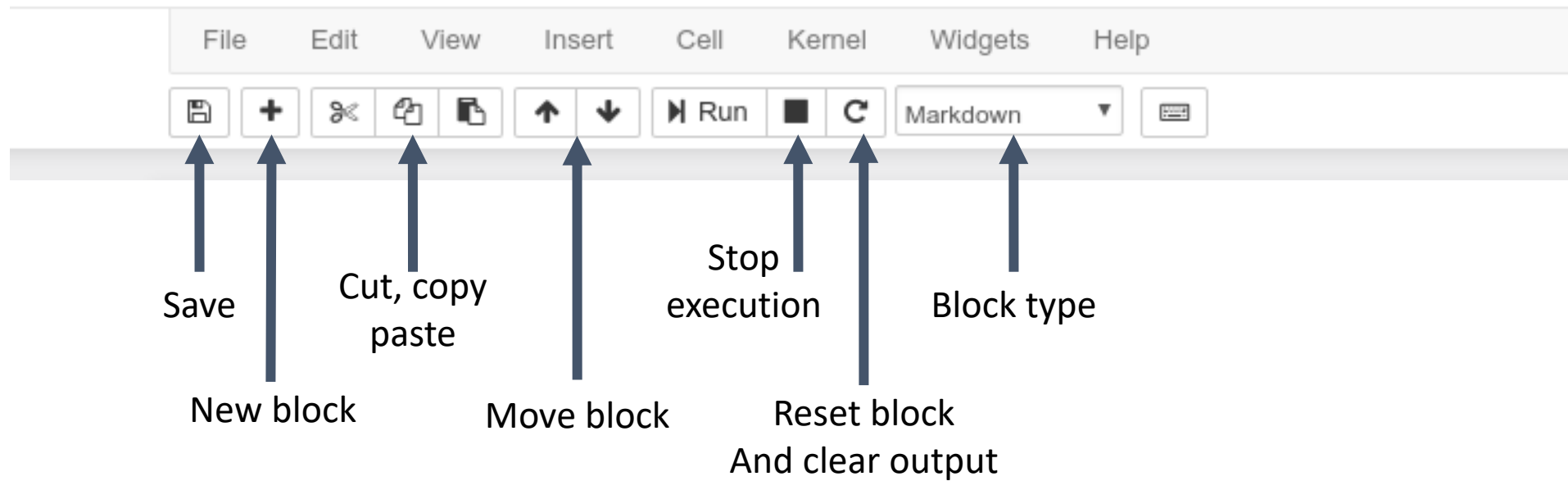


Pull down a Git repository

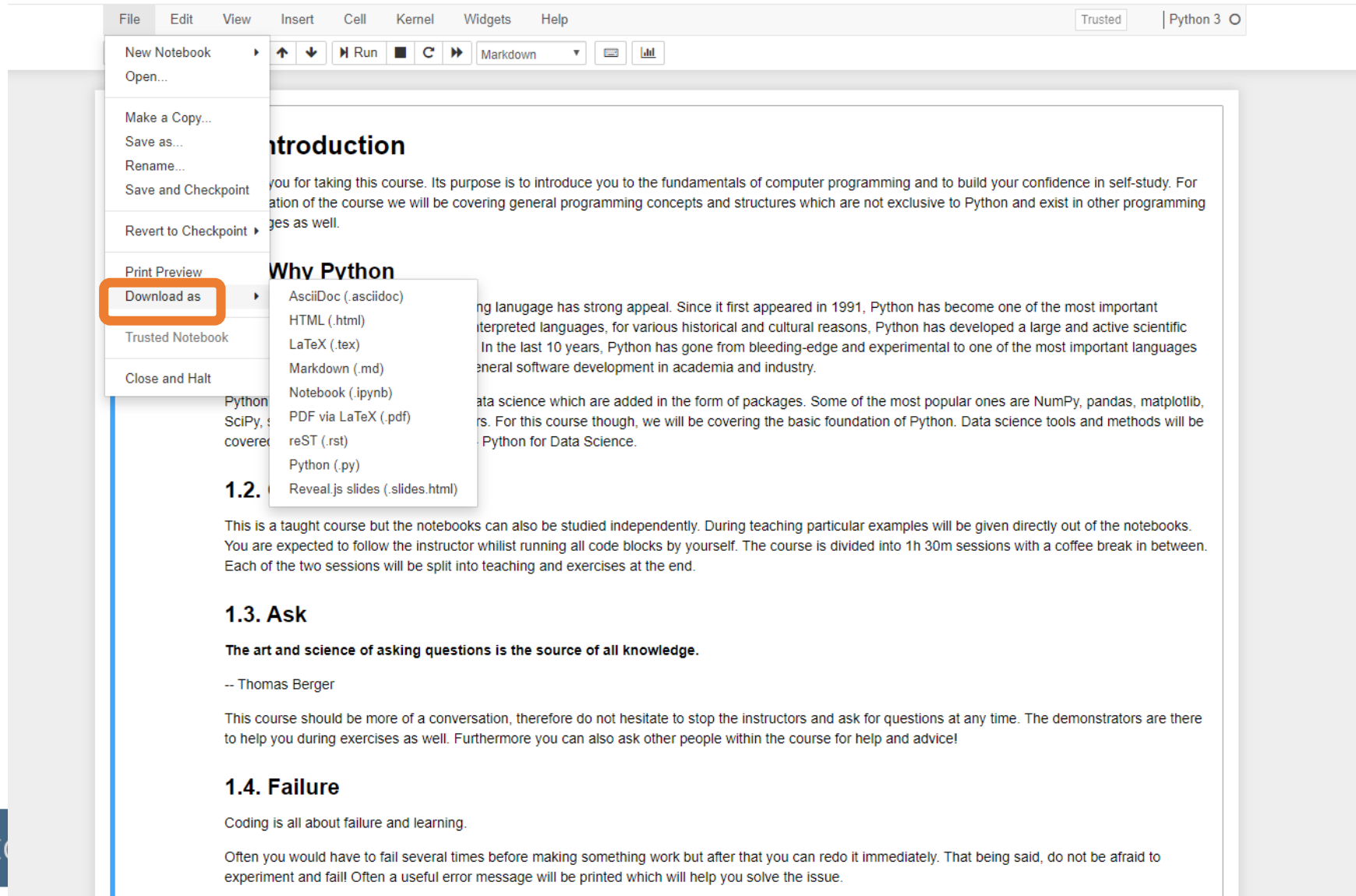
3. Starting a notebook



4. Toolbar



5. Download files



The screenshot shows a Jupyter Notebook interface. The 'File' menu is open, and the 'Download as' option is highlighted with an orange rectangle. 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 background content of the notebook includes a title 'Introduction', a paragraph about the course purpose, a section 'Why Python', and several numbered sections (1.2, 1.3, 1.4) discussing the course structure and the importance of asking questions and learning from failure.

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

New Notebook
Open...

Make a Copy...
Save as...
Rename...
Save and Checkpoint
Revert to Checkpoint

Print Preview
Download as
Trusted Notebook
Close and Halt

AsciiDoc (.asciidoc)
HTML (.html)
LaTeX (.tex)
Markdown (.md)
Notebook (.ipynb)
PDF via LaTeX (.pdf)
reST (.rst)
Python (.py)
Reveal.js slides (.slides.html)

Introduction

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 the duration of the course we will be covering general programming concepts and structures which are not exclusive to Python and exist in other programming languages as well.

Why Python

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

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

1.2. Course Structure

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 whilst 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

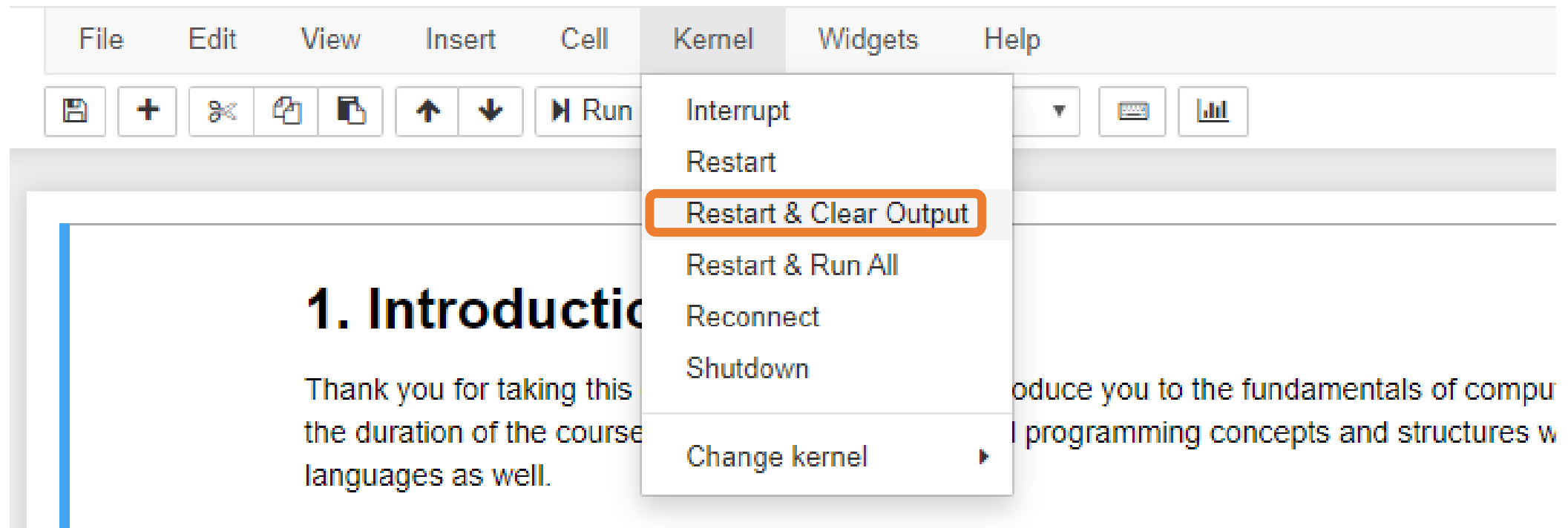
This course should be more of a conversation, therefore do not hesitate to stop the instructors and ask for questions at any time. The demonstrators are there to help you during exercises as well. Furthermore you can also ask other people within the course for help and advice!

1.4. Failure

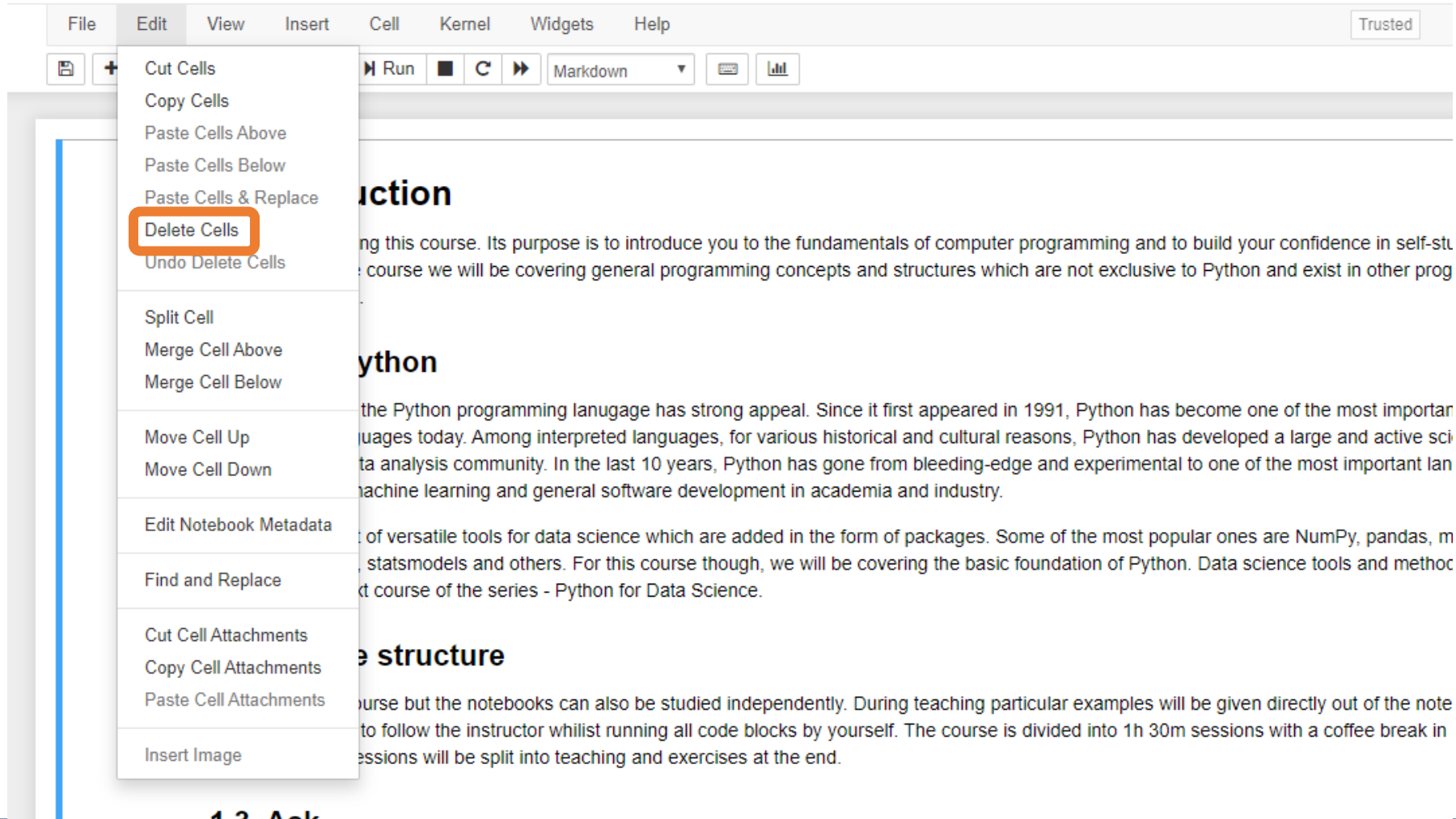
Coding is all about failure and learning.

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.

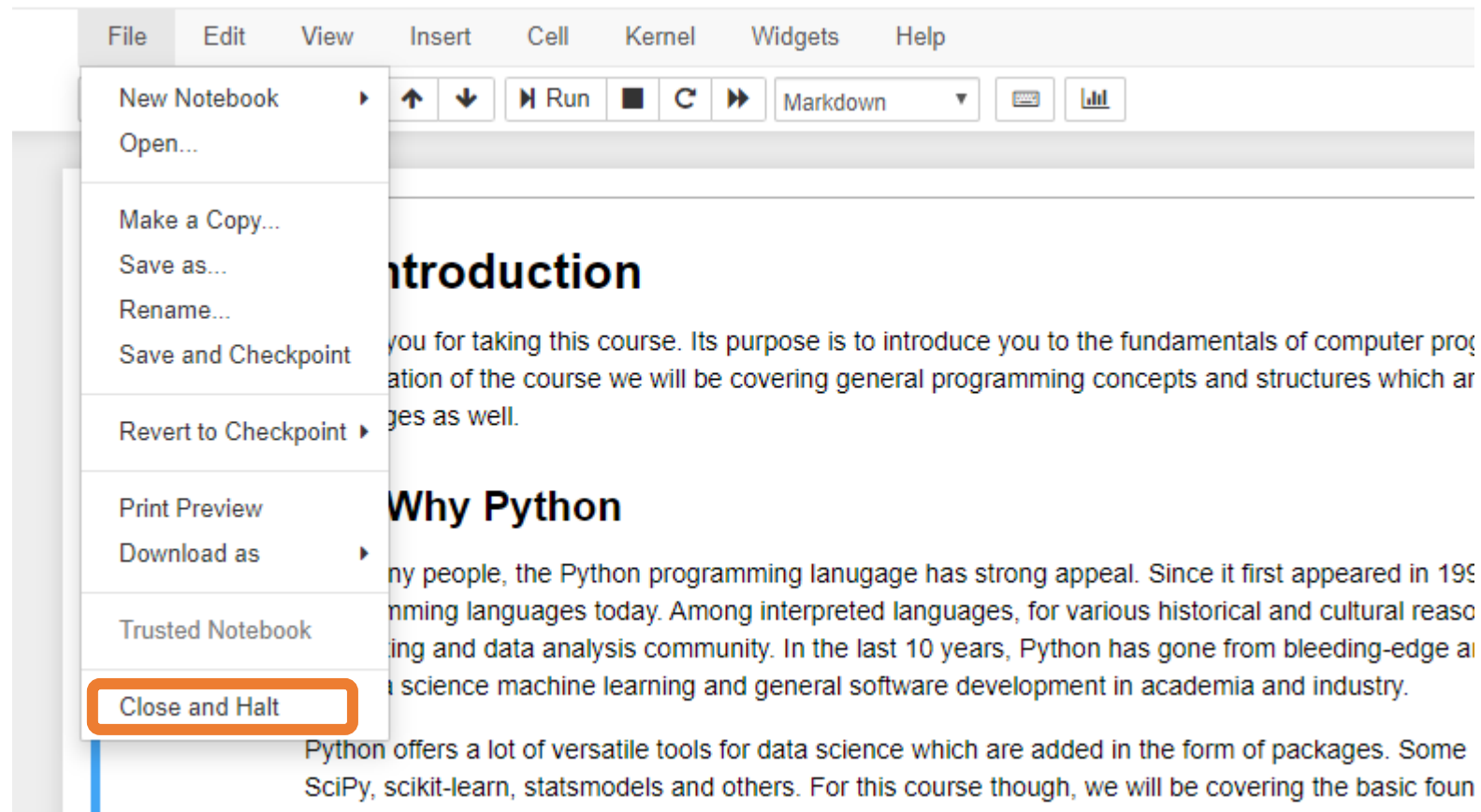
6. Kernel/Restart & Clear output



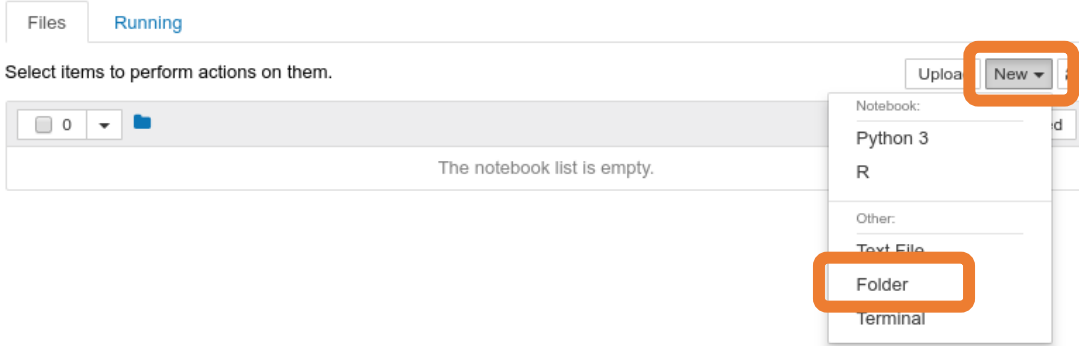
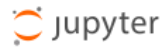
7. Edit/Delete Cell



8. File/ Close & Halt



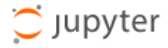
9. Create a folder



10. Rename



11. Upload files



Files [Running](#)

Select items to perform actions on them.

0 ▾

/ Introduction to Python

Name ▾

Last Modified

..

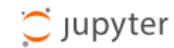
seconds ago

The notebook list is empty.

Upload

New ▾

↺



Files [Running](#)

Select items to perform actions on them.

0 ▾

/ Introduction to Python

Name ▾

Last Modified

The notebook list is empty.

python-intro-0.ipynb	<div><div>Upload</div><div>Cancel</div></div>
python-intro-1.ipynb	<div><div>Upload</div><div>Cancel</div></div>
python-intro-2.ipynb	<div><div>Upload</div><div>Cancel</div></div>
<div>..</div> <div>seconds ago</div>	
python-intro-exercises.ipynb	<div><div>Upload</div><div>Cancel</div></div>

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

Why should we care?

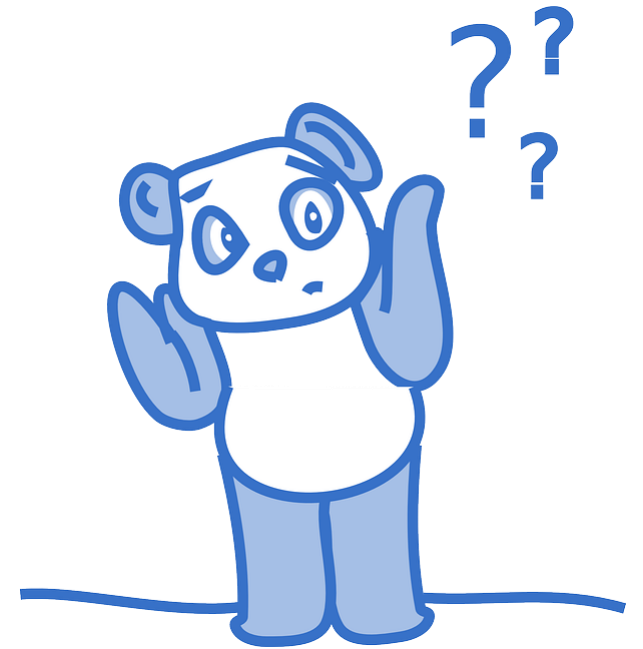


Image by [Cler-Free-Vector-Images on Pixabay](#)


```
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

Another example

```
x = 14  
y = 42  
not (  )  
False
```

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

```
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

```
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 earth 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 and fun exercises.(notebooks 0 and 1)
20-minute break afterwards.

Failure is progress!

Ask us anything. Ask among yourselves as well.

