Introduction to
python

Sichen.Liu@xjtlu.edu.cn

XJTLU | SCHOOL OF FILM AND TV ARTS

Xi'an Jiaotong-Liverpool University
西交利物浦大学

Self-introduction

- 2016.9-2021.6 Institute of Acoustics, Chinese Academy of Sciences

- 2021.7-2022.10 Tencent (Beijing) Worked as an audio algorithm researcher at Tencent Video

Structure

- 3 tutorials in total.
- Each tutorials will last about 100 minutes.
- 60% of teaching, 40% of exercise.
- Try to follow within your notebook and run all the 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!
- Thursday, 10 - 12 am @ SD 557 room
- E-mail me before coming
- Sichen.Liu@xjtlu.edu.cn

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?

History

- Now widely spread
- Open Source! Free!
- Versatile

Top 10 Programming Languages
1 python™
2 JavaScript
3 HTML
4 CSS
5 SQL
6
7
8 Objective-C
9 Rust
10 Scala
YoungWonks

Python today

- Python has 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

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?

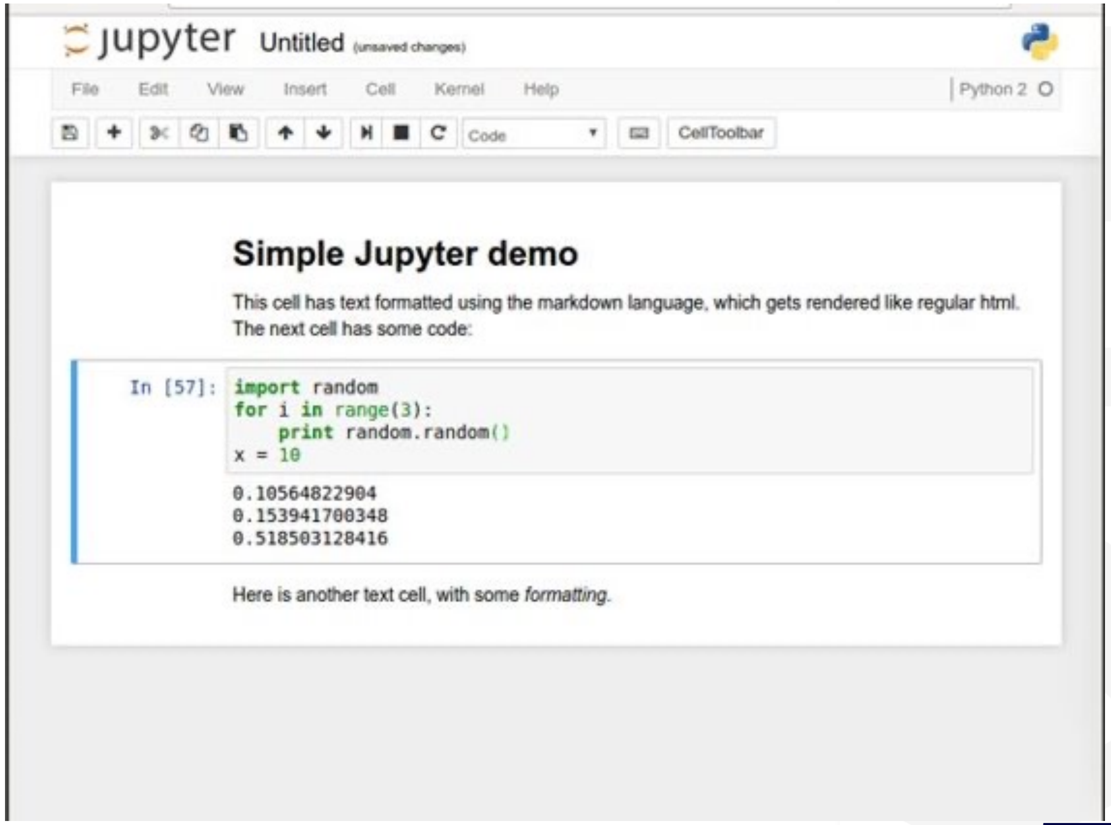
Then we must write text files in .py

Failure

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

Jupyter notebooks

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


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

Simple Jupyter demo

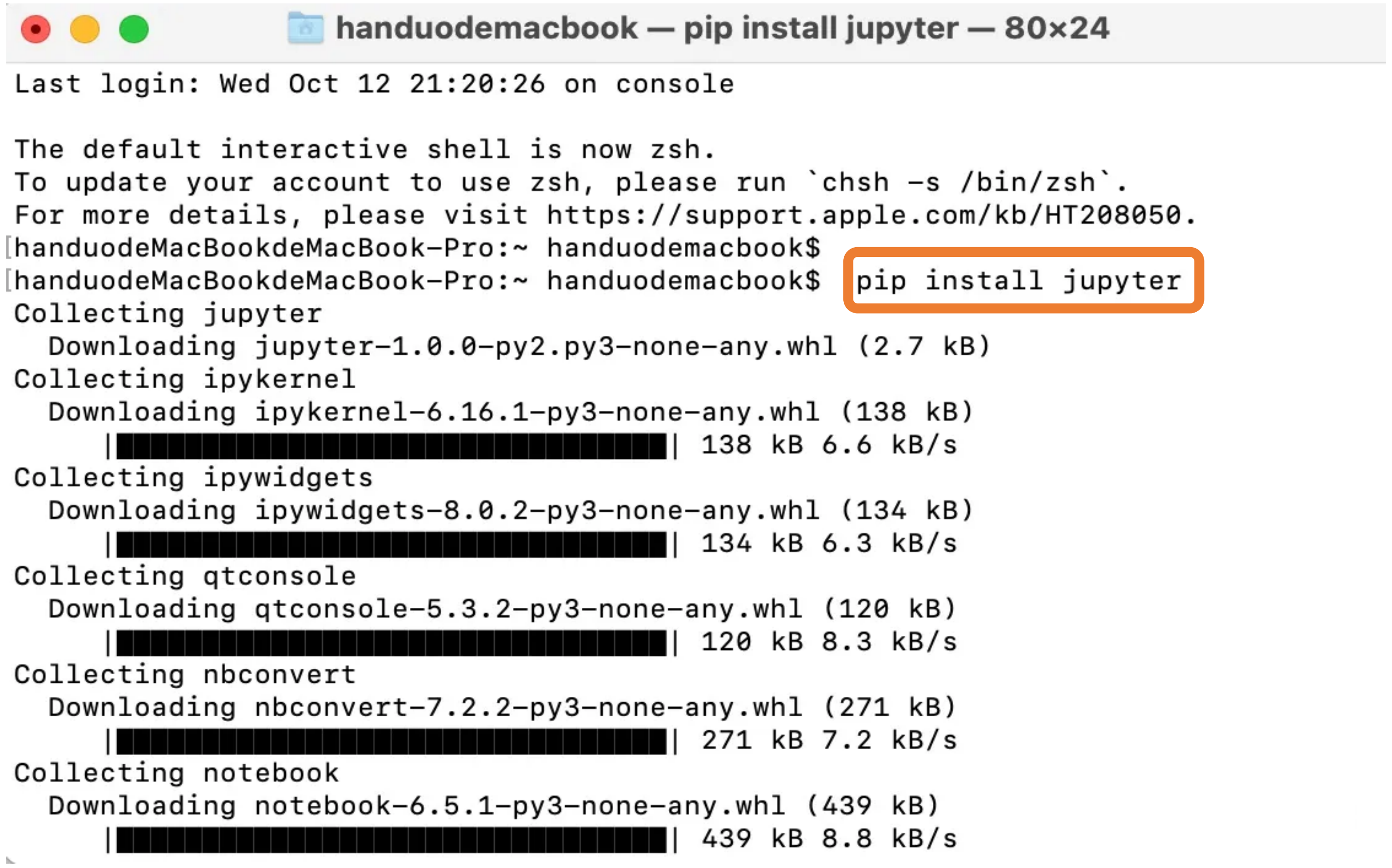
This cell has text formatted using the markdown language, which gets rendered like regular html. The next cell has some code:

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

```
0.10564822904
0.153941700348
0.518503128416
```

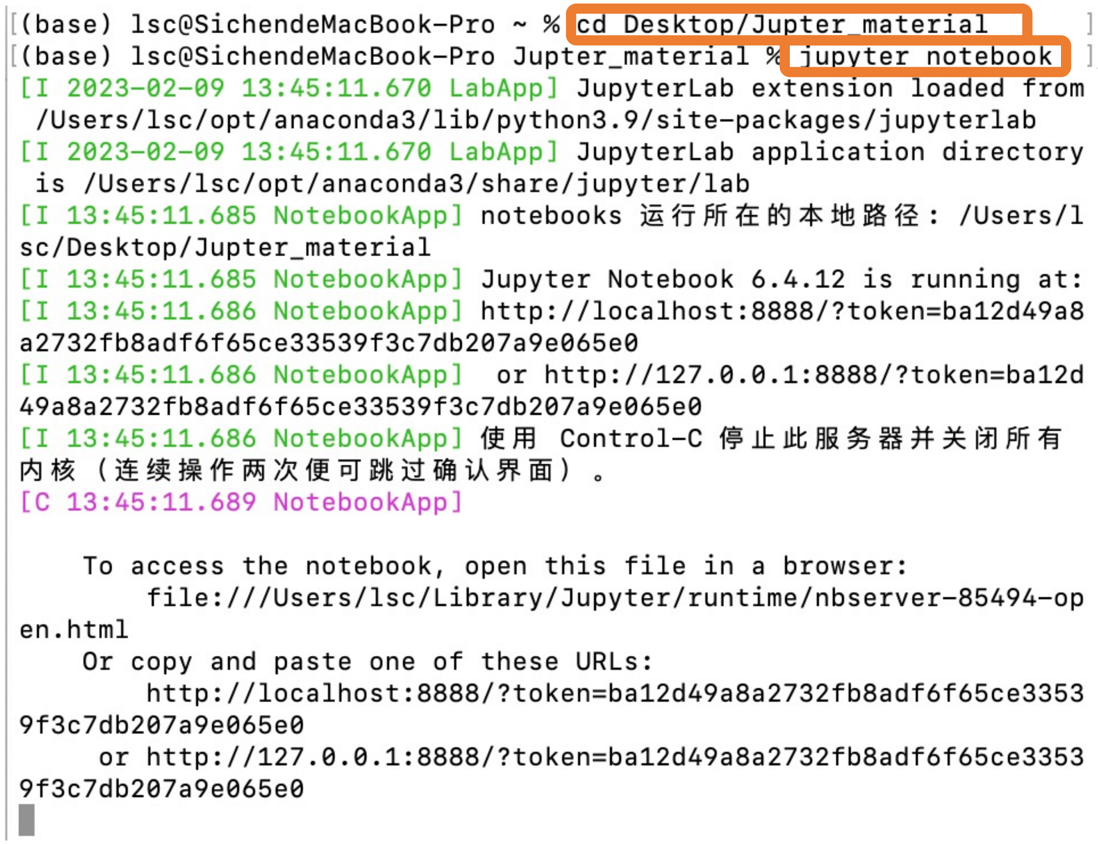
Here is another text cell, with some *formatting*.

1. Install Jupyter

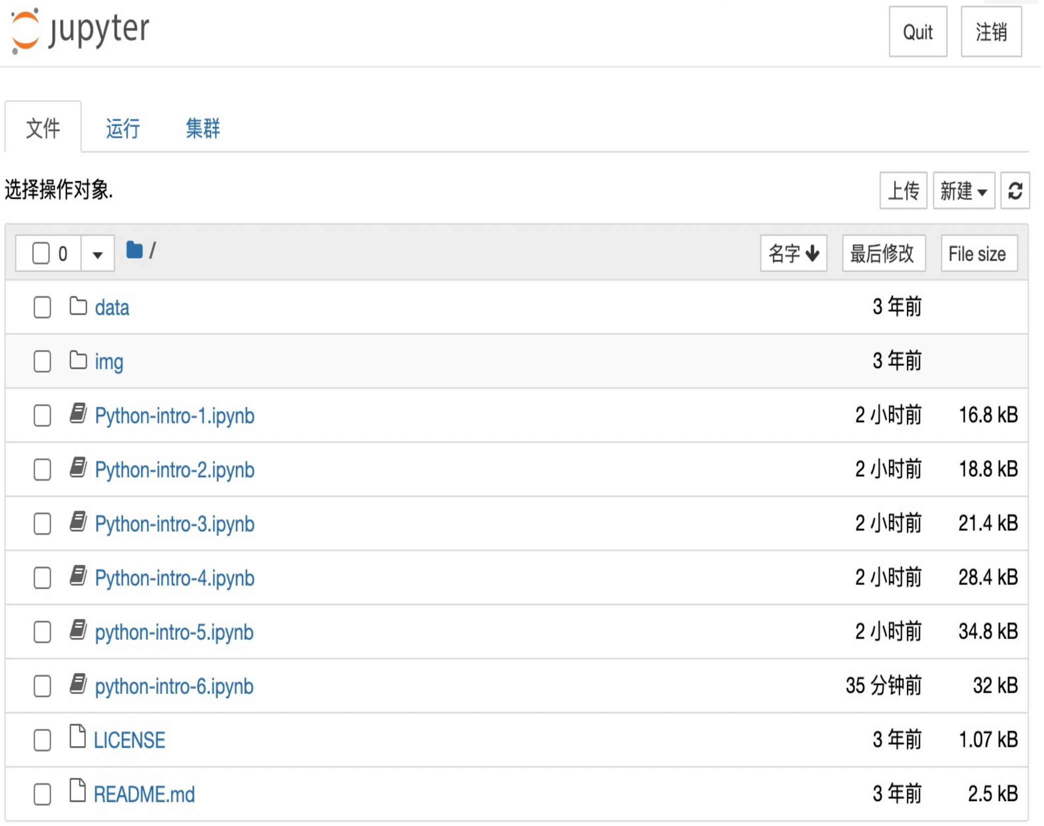


```
Last login: Wed Oct 12 21:20:26 on console

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
[handuodeMacBookdeMacBook-Pro:~ handuodemacbook$
[handuodeMacBookdeMacBook-Pro:~ handuodemacbook$ pip install jupyter
Collecting jupyter
  Downloading jupyter-1.0.0-py2.py3-none-any.whl (2.7 kB)
Collecting ipykernel
  Downloading ipykernel-6.16.1-py3-none-any.whl (138 kB)
     |████████████████████████████████| 138 kB 6.6 kB/s
Collecting ipywidgets
  Downloading ipywidgets-8.0.2-py3-none-any.whl (134 kB)
     |████████████████████████████████| 134 kB 6.3 kB/s
Collecting qtconsole
  Downloading qtconsole-5.3.2-py3-none-any.whl (120 kB)
     |████████████████████████████████| 120 kB 8.3 kB/s
Collecting nbconvert
  Downloading nbconvert-7.2.2-py3-none-any.whl (271 kB)
     |████████████████████████████████| 271 kB 7.2 kB/s
Collecting notebook
  Downloading notebook-6.5.1-py3-none-any.whl (439 kB)
     |████████████████████████████████| 439 kB 8.8 kB/s
```

2. Running Jupyter

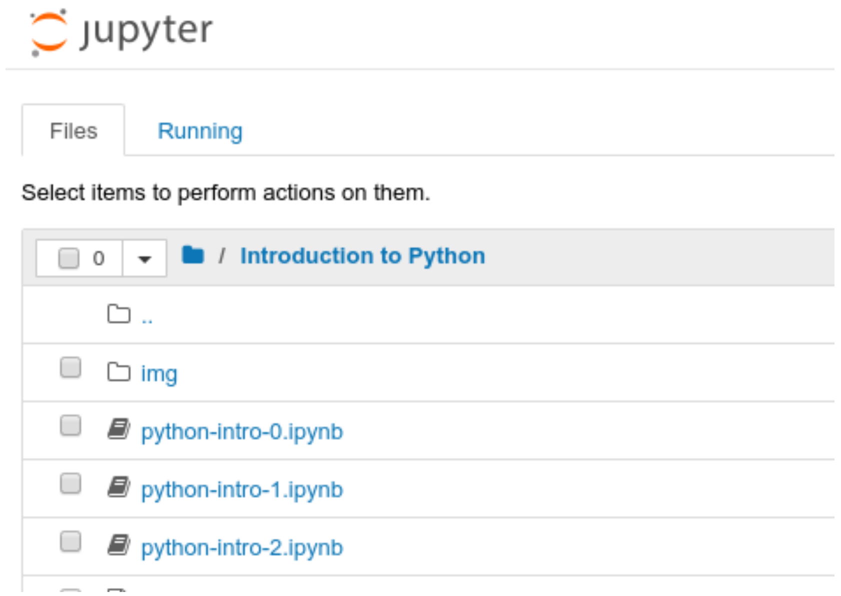


```
(base) lsc@SichendeMacBook-Pro ~ % cd Desktop/Jupter_material
(base) lsc@SichendeMacBook-Pro Jupter_material % jupyter notebook
[I 2023-02-09 13:45:11.670 LabApp] JupyterLab extension loaded from
 /Users/lsc/opt/anaconda3/lib/python3.9/site-packages/jupyterlab
[I 2023-02-09 13:45:11.670 LabApp] JupyterLab application directory
 is /Users/lsc/opt/anaconda3/share/jupyter/lab
[I 13:45:11.685 NotebookApp] notebooks 运行所在的本地路径: /Users/l
sc/Desktop/Jupter_material
[I 13:45:11.685 NotebookApp] Jupyter Notebook 6.4.12 is running at:
[I 13:45:11.686 NotebookApp] http://localhost:8888/?token=ba12d49a8
a2732fb8adf6f65ce33539f3c7db207a9e065e0
[I 13:45:11.686 NotebookApp]  or http://127.0.0.1:8888/?token=ba12d
49a8a2732fb8adf6f65ce33539f3c7db207a9e065e0
[I 13:45:11.686 NotebookApp] 使用 Control-C 停止此服务器并关闭所有
内核（连续操作两次便可跳过确认界面）。
[C 13:45:11.689 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///Users/lsc/Library/Jupyter/runtime/nbserver-85494-op
en.html
    Or copy and paste one of these URLs:
        http://localhost:8888/?token=ba12d49a8a2732fb8adf6f65ce3353
9f3c7db207a9e065e0
     or http://127.0.0.1:8888/?token=ba12d49a8a2732fb8adf6f65ce3353
9f3c7db207a9e065e0
```



data	3 年前	
img	3 年前	
Python-intro-1.ipynb	2 小时前	16.8 kB
Python-intro-2.ipynb	2 小时前	18.8 kB
Python-intro-3.ipynb	2 小时前	21.4 kB
Python-intro-4.ipynb	2 小时前	28.4 kB
python-intro-5.ipynb	2 小时前	34.8 kB
python-intro-6.ipynb	35 分钟前	32 kB
LICENSE	3 年前	1.07 kB
README.md	3 年前	2.5 kB

3. Starting

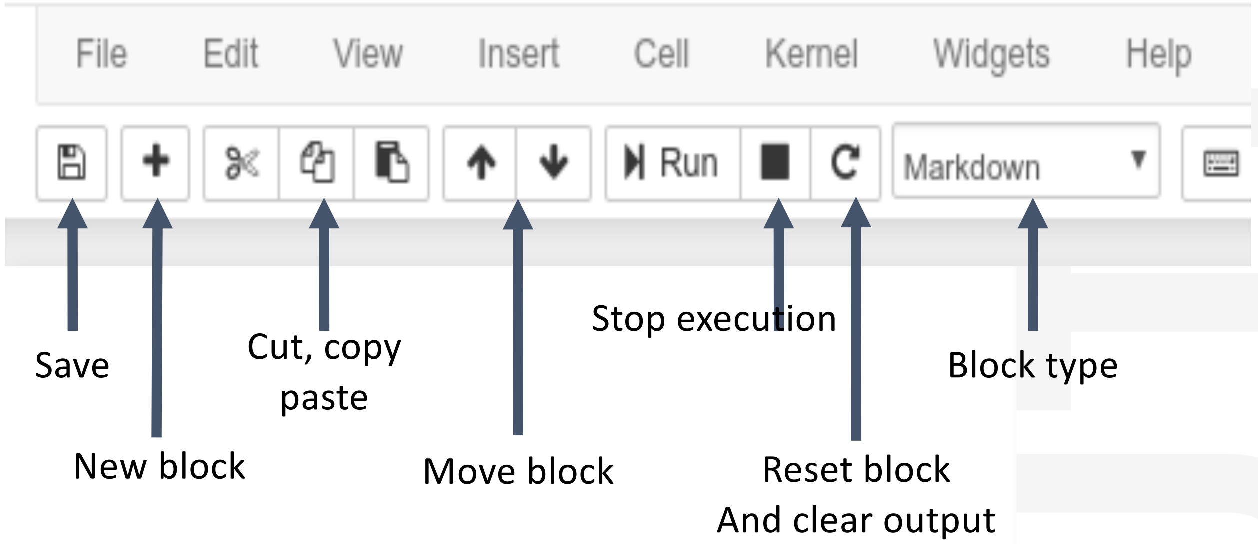


Select items to perform actions on them.

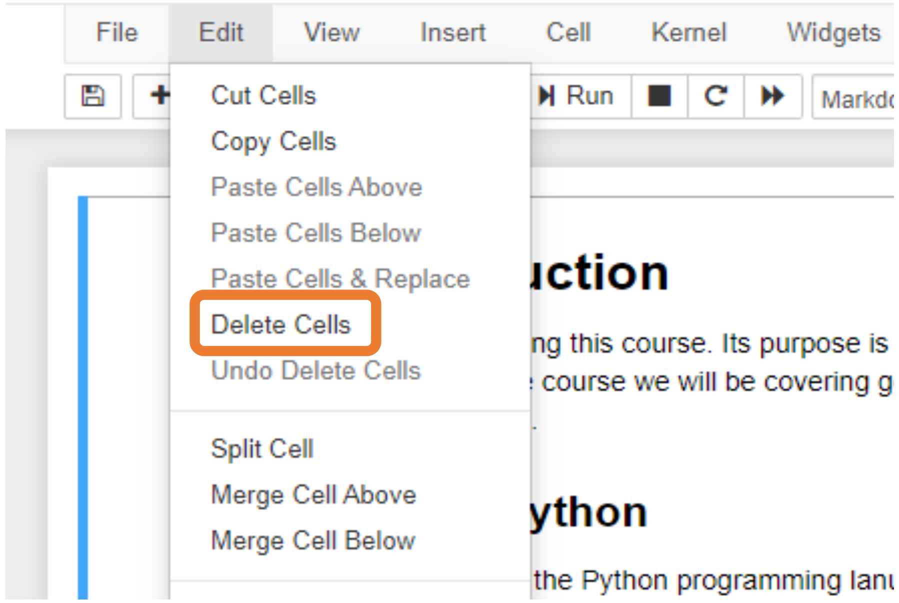
Introduction to Python

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

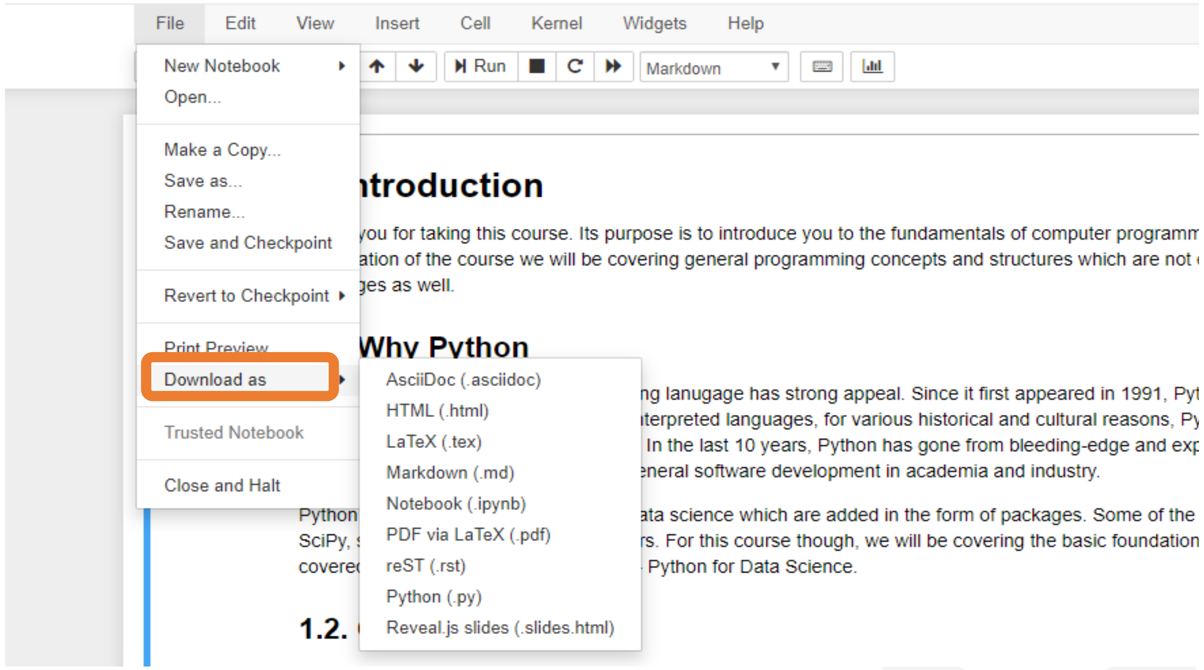
4. Toolbar



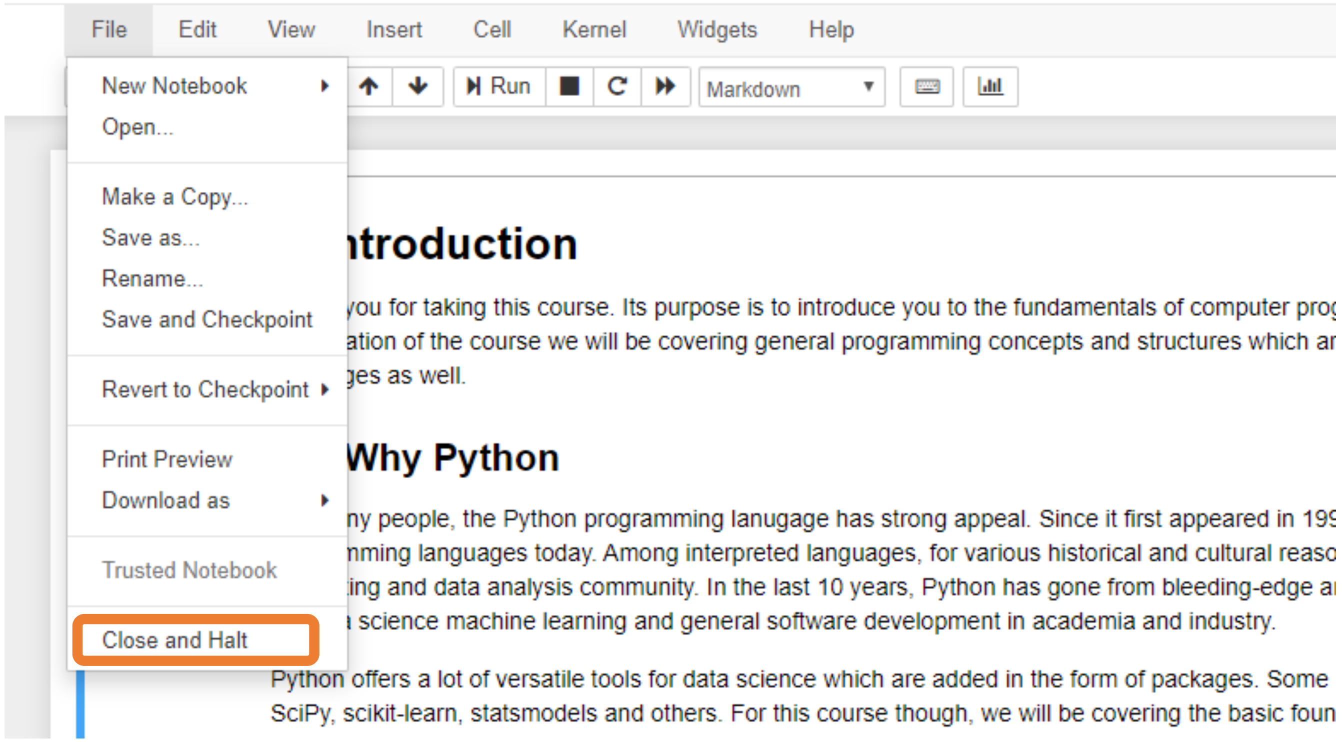
5. Edit/Delete Cell



6. Download files



7. File/ Close & Halt



ıtroduction

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

Why Python

ny people, the Python programming lanugage has strong appeal. Since it first appeared in 199
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 aı
ı 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

Running blocks

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

- 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
- Strings
- Exercises

Python as a calculator

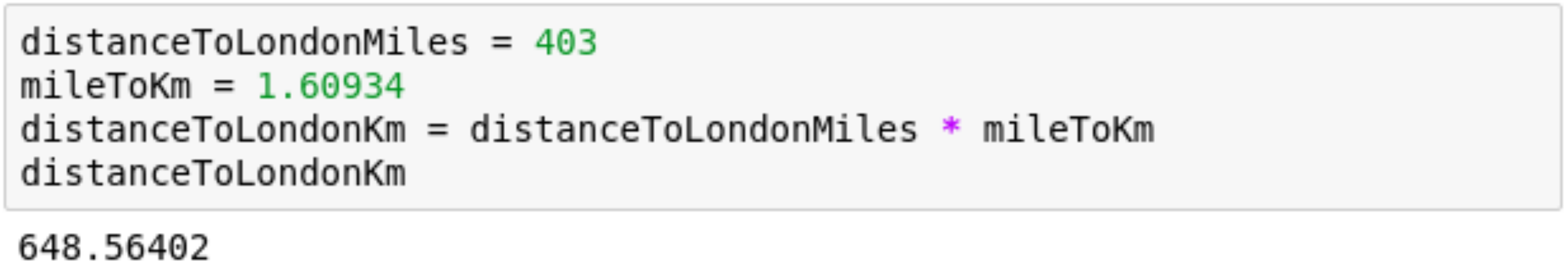
- Let us calculate the distance between Edinburgh and London in km

```
403 * 1.60934
```

```
648.56402
```

Variables

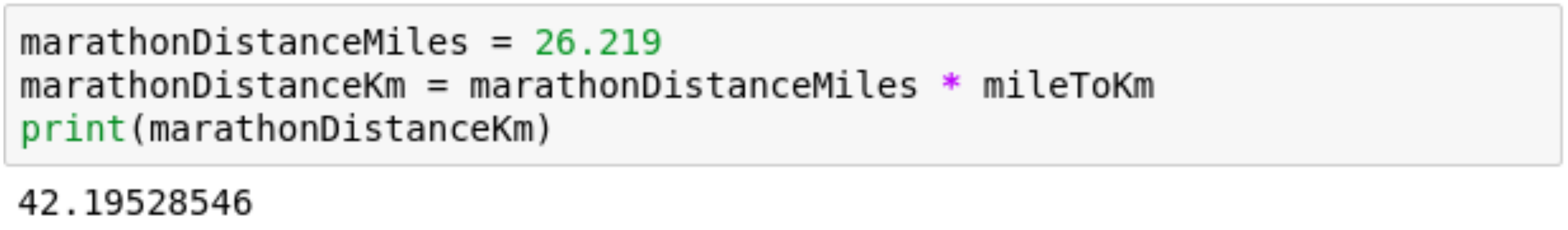
- Great calculator but how can we make it store values?
- Do this by defining variables
- Can later be called by the variable name
- Variable names are case sensitive and unique



```
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!



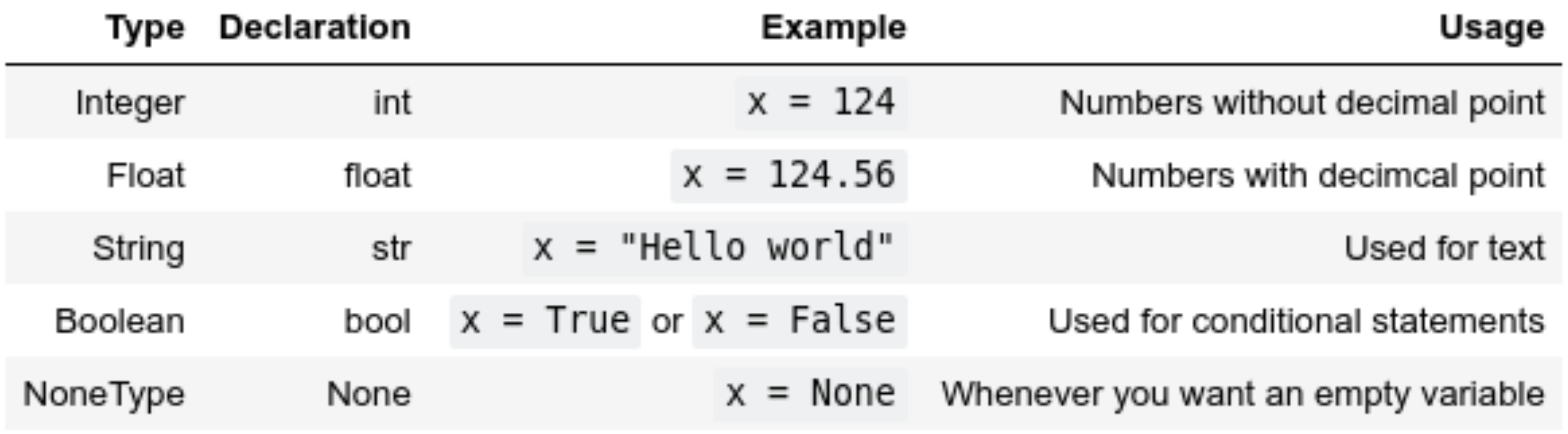
```
marathonDistanceMiles = 26.219
marathonDistanceKm = marathonDistanceMiles * mileToKm
print(marathonDistanceKm)
```

```
42.19528546
```

Types

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

The basic types are shown in this table:



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?

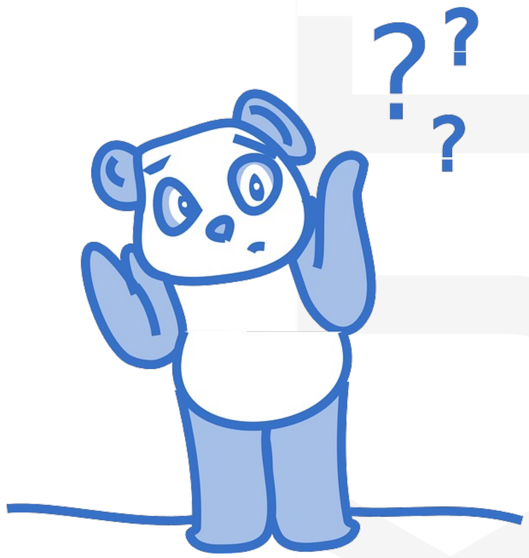
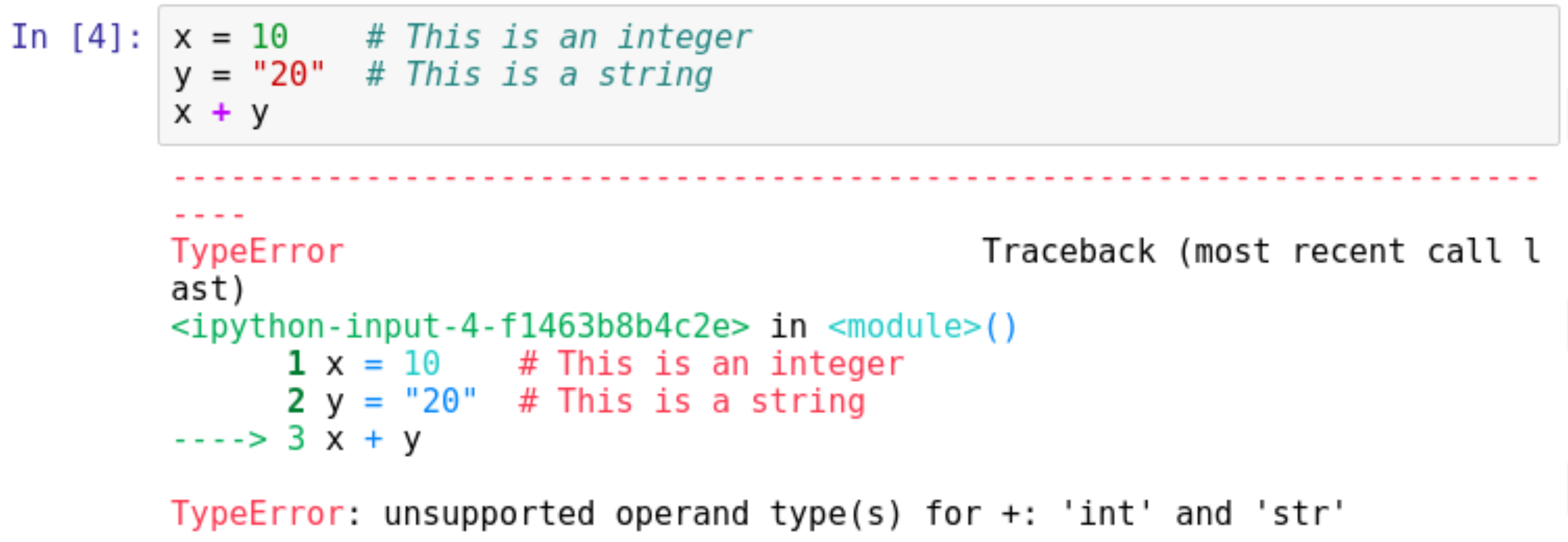


Image by Clker-Free-Vector-Images on Pixabay



```
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'
```

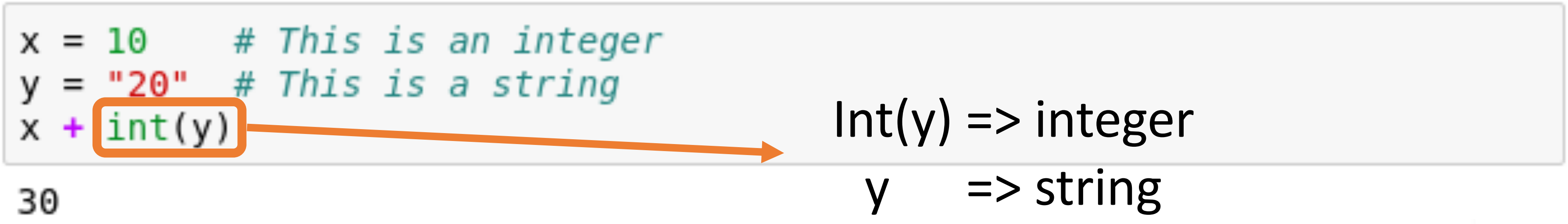
Important lesson to remember!
We can't do arithmetic operations on variables of different types. Therefore, make sure that you are always aware of your variables types!

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



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

```
30
```

Int(y) => integer

y => string

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

Quick quiz

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

What will be the result?

```
'1020'
```

Strings

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

Strings

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

```
'Python rocks'
```

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

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

Strings

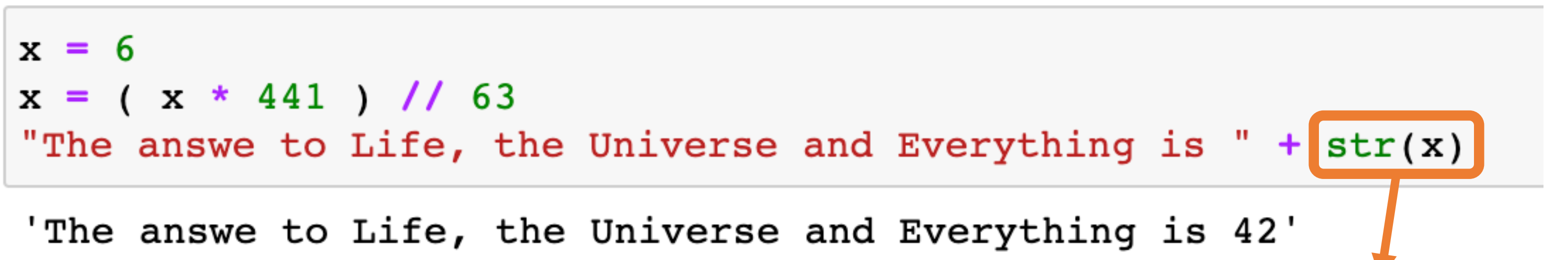
```
x = "Edinburgh"
x = x.upper()

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 in the output.
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.

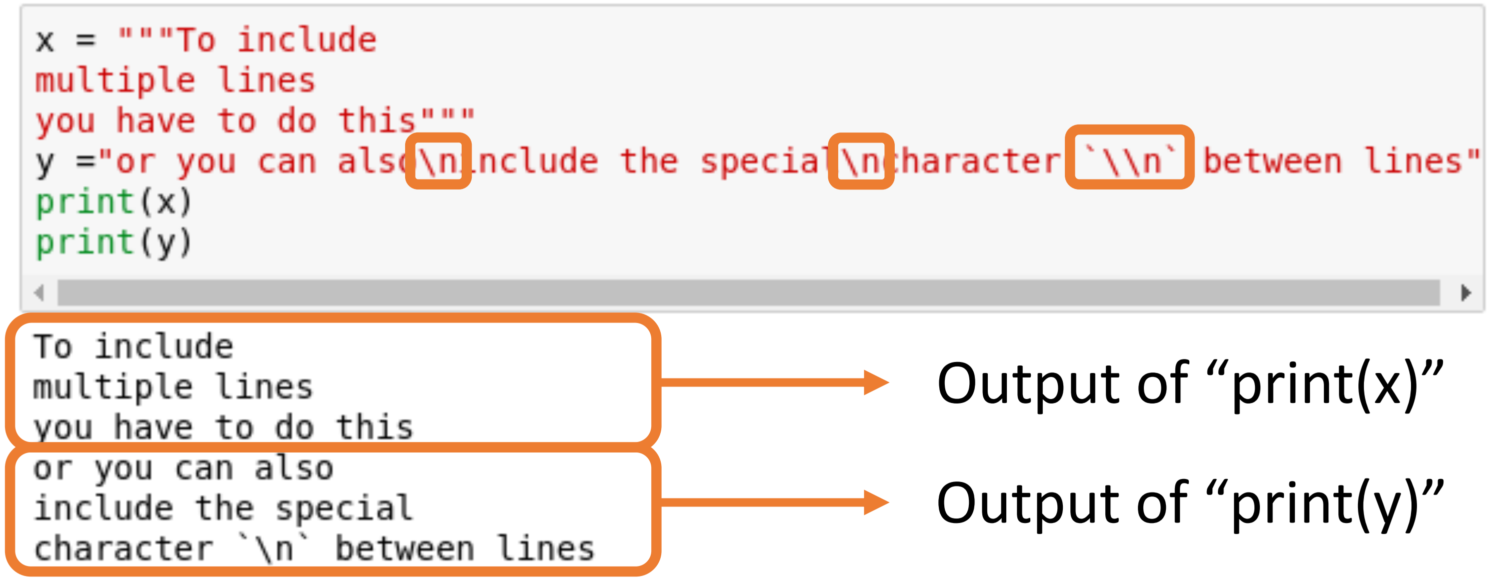


```
x = 6
x = ( x * 441 ) // 63
"The answe to Life, the Universe and Everything is " + str(x)
```

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

str(x) => string
x => integer

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
```

Exercise

Install **Anaconda** (windows/mac pc)

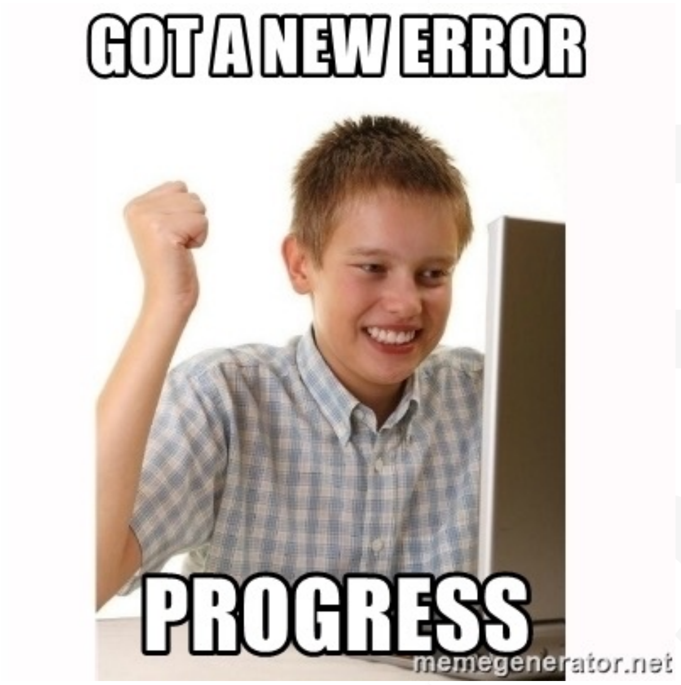
Simple and fun exercises.(Notebooks 1)

Failure is progress!

Ask us anything.

Ask among yourselves as well.

Google is your best friend when coding.



Let us start

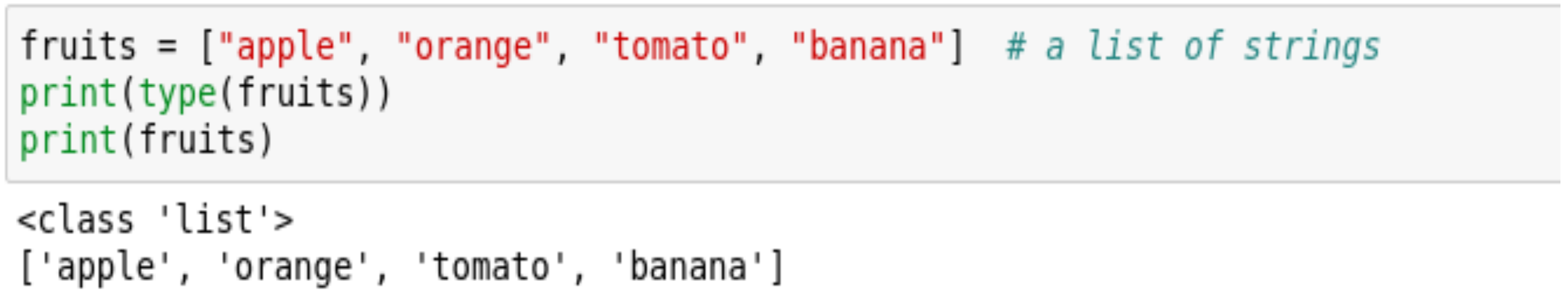
- If you like to follow along, you can open your own notebook. But please try to keep up with my presentation, as you still have time for exercises after the teaching.

Agenda

- Lists
- Tuples
- Sets
- Dictionaries
- Exercises

Lists

- One of the most useful concepts
- Group multiple variables together (a kind of **container**!)



```python
fruits = ["apple", "orange", "tomato", "banana"]  # a list of strings
print(type(fruits))
print(fruits)
```

```
<class 'list'>
['apple', 'orange', 'tomato', 'banana']
```

Indexing a list

- Indexing – accessing items within a data structure

```
fruits = ["apple", "orange", "tomato", "banana"]
fruits[2]
```

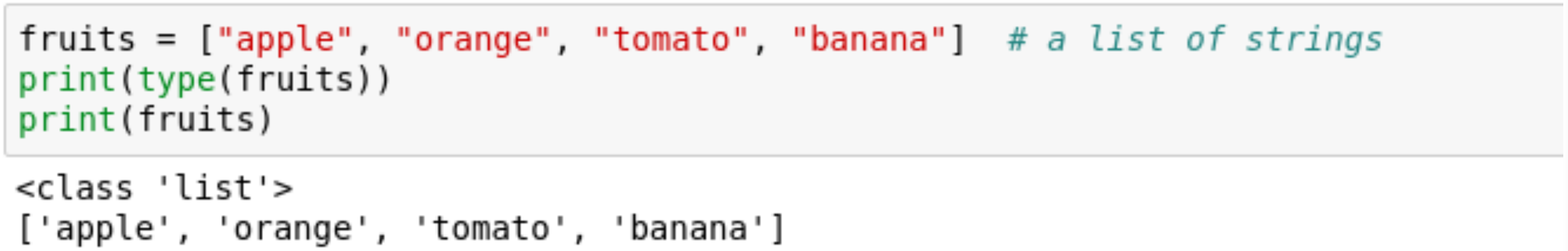
```
'tomato'
```

- Indexing a list is not very intuitive...
- The first element of a list has an index 0

Index:	0	1	2	3
List:	apple	orange	tomato	banana

Quick quiz

What will **fruits[3]** return?



```python
fruits = ["apple", "orange", "tomato", "banana"]  # a list of strings
print(type(fruits))
print(fruits)
```

```
<class 'list'>
['apple', 'orange', 'tomato', 'banana']
```

Quick quiz

What will this return?

```
fruits[4]
```

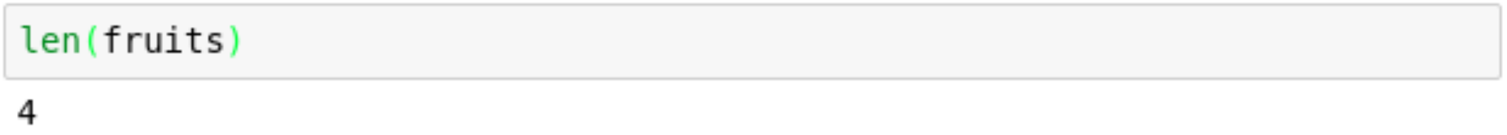
```
---------------------------------------------------------------------------
----
IndexError                                Traceback (most recent call l
ast)
<ipython-input-14-b8c91da6ba3a> in <module>()
----> 1 fruits[4]

IndexError: list index out of range
```

Data structure sizes

Make sure you are always aware of the sizes of each variable!

This can easily be done using the **len()** length function.

It returns the length/size of any data structure



```
len(fruits)
```

```
4
```

Is a tomato really a fruit?

```
fruits[2] = "peach"
print(fruits)
```

```
['apple', 'orange', 'peach', 'banana']
```

Furthermore, we can modify lists in various ways

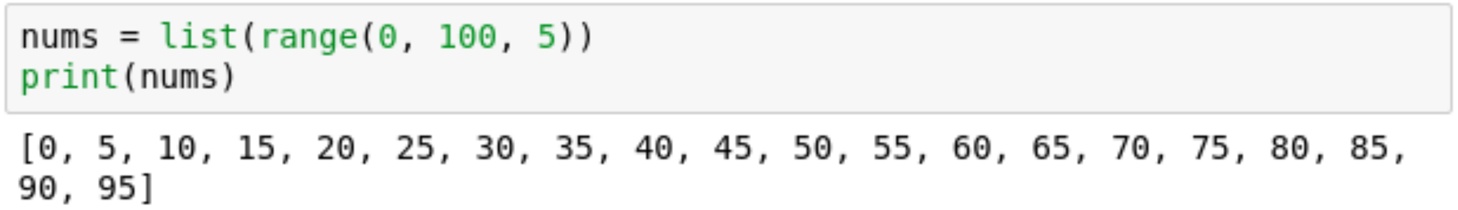
```
fruits.append("cherry")     # add new item to list
print(fruits)
fruits.remove("orange")   # remove orange from list
print(fruits)
```

```
['apple', 'orange', 'peach', 'banana', 'cherry']
['apple', 'peach', 'banana', 'cherry']
```

Lists with integers

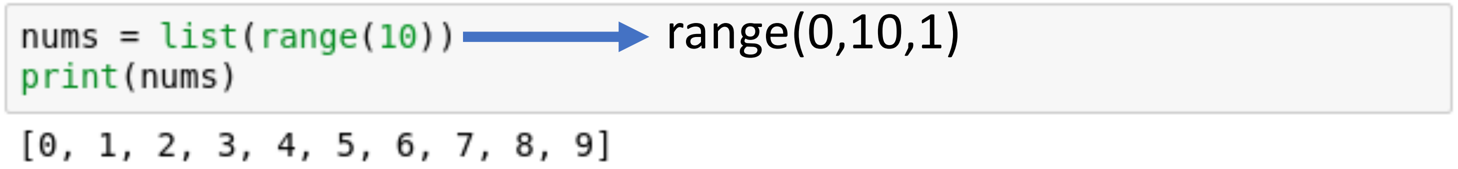
range() - a function that generates a sequence of numbers as a list

Syntax : range(start point, end point, step size) ➔ End point is not included



```
nums = list(range(0, 100, 5))
print(nums)
```

```
[0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85,
90, 95]
```



```
nums = list(range(10))
print(nums)
```

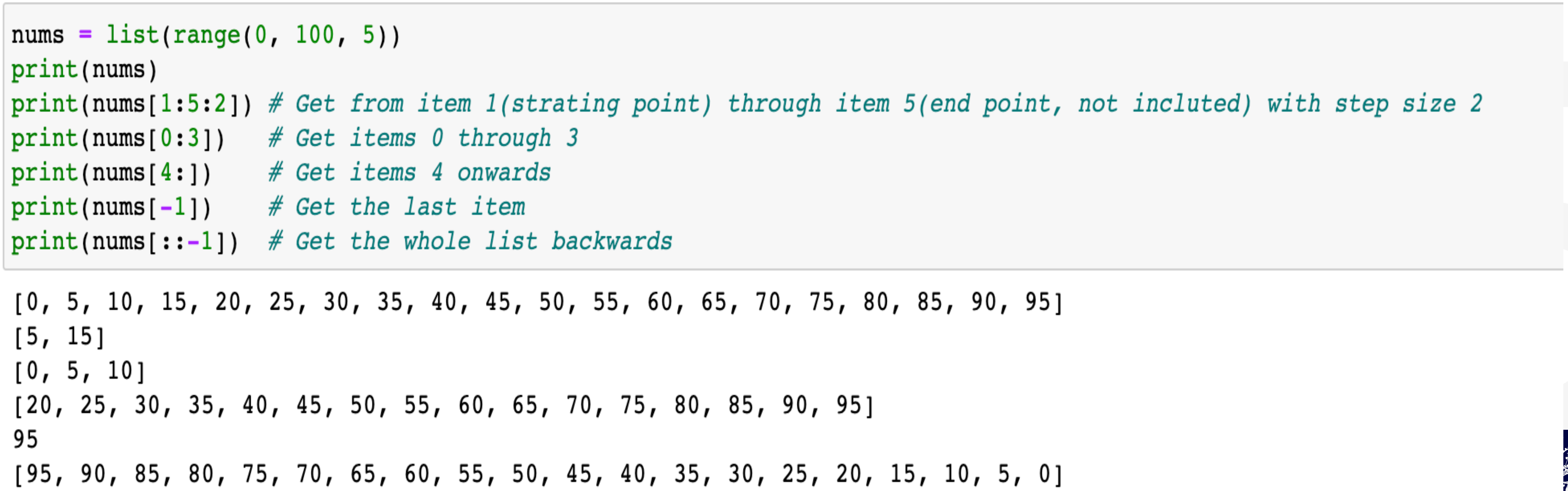
$\longrightarrow$ range(0,10,1)

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Slicing lists

- Slicing – obtain a particular set of sub-elements from a data structure.
- Very useful and flexible.
- Syntax: List[start point : end point : step size]

Colon

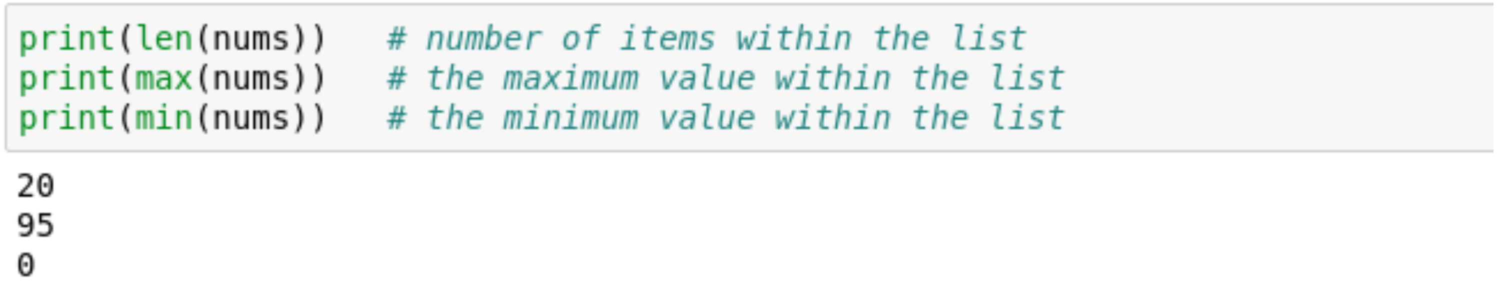


```
nums = list(range(0, 100, 5))
print(nums)
print(nums[1:5:2]) # Get from item 1(strating point) through item 5(end point, not incluted) with step size 2
print(nums[0:3])   # Get items 0 through 3
print(nums[4:])    # Get items 4 onwards
print(nums[-1])    # Get the last item
print(nums[::-1])  # Get the whole list backwards
```

```
[0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95]
[5, 15]
[0, 5, 10]
[20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95]
95
[95, 90, 85, 80, 75, 70, 65, 60, 55, 50, 45, 40, 35, 30, 25, 20, 15, 10, 5, 0]
```

Lists – helpful functions

- Makes them extremely useful and versatile

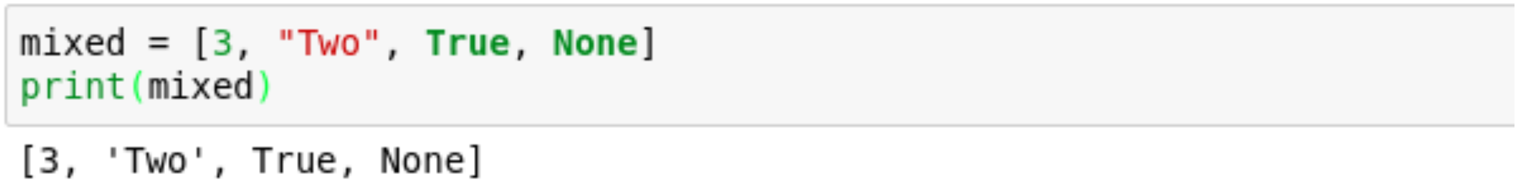


```
print(len(nums))   # number of items within the list
print(max(nums))   # the maximum value within the list
print(min(nums))   # the minimum value within the list
```

```
20
95
0
```

Lists can be of different types

- Not very useful, but possible



```python
mixed = [3, "Two", True, None]
print(mixed)
```

```
[3, 'Two', True, None]
```

Mutability

Mutable object – can be changed after creation.

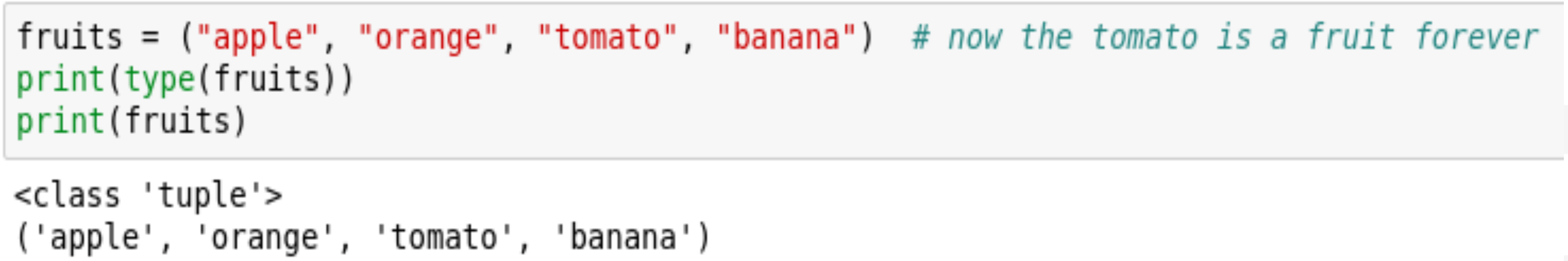
Immutable object - can **NOT** be changed after creation.

Quick quiz

- Are lists mutable?

Tuples

- Effectively lists that are immutable (I.e. can't be changed)

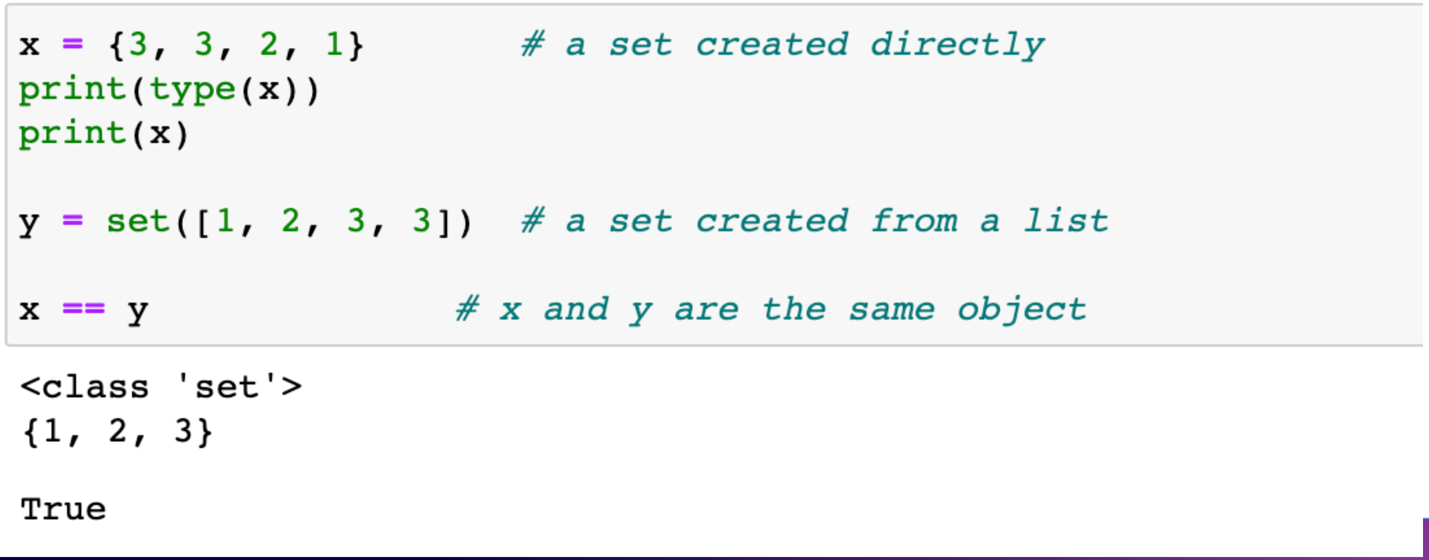


```
fruits = ("apple", "orange", "tomato", "banana")  # now the tomato is a fruit forever
print(type(fruits))
print(fruits)
```

```
<class 'tuple'>
('apple', 'orange', 'tomato', 'banana')
```

Sets

- Effectively lists that can't contain duplicate items
- Similar functionality to lists
- Can't be indexed or sliced, doesn’t have order.
- Can be created with **{}** or you can convert a list to a set

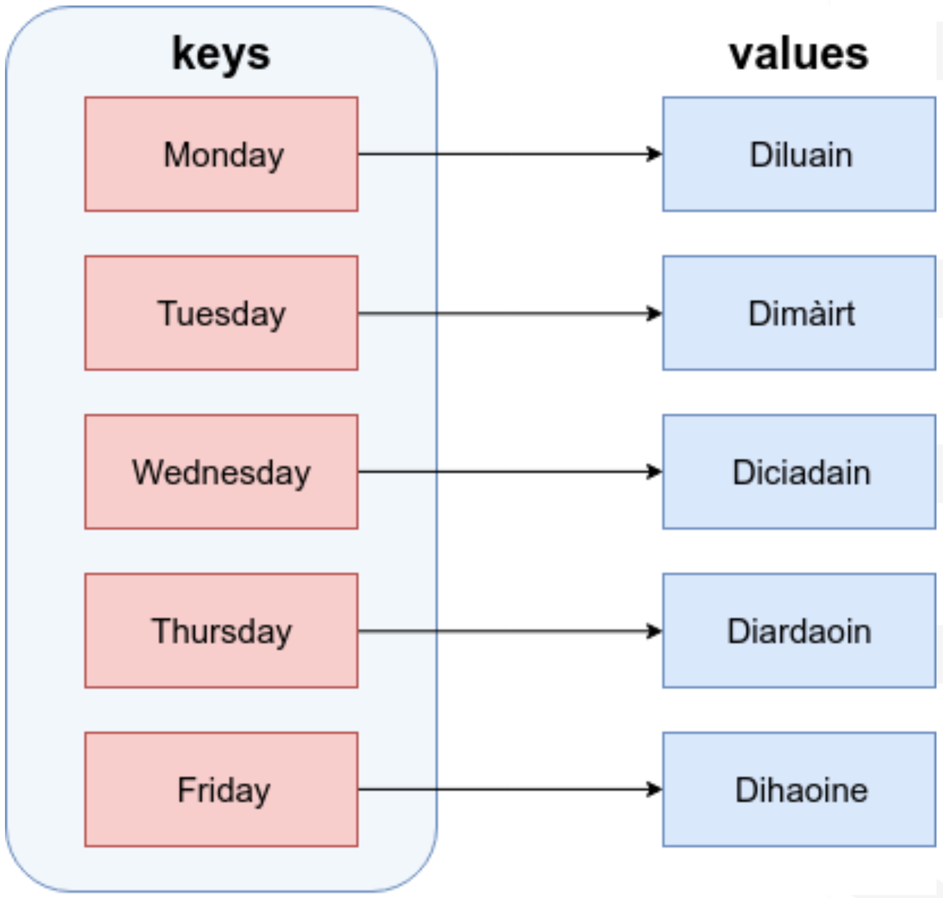


```python
x = {3, 3, 2, 1}       # a set created directly
print(type(x))
print(x)

y = set([1, 2, 3, 3])  # a set created from a list

x == y              # x and y are the same object
```

```
<class 'set'>
{1, 2, 3}

True
```

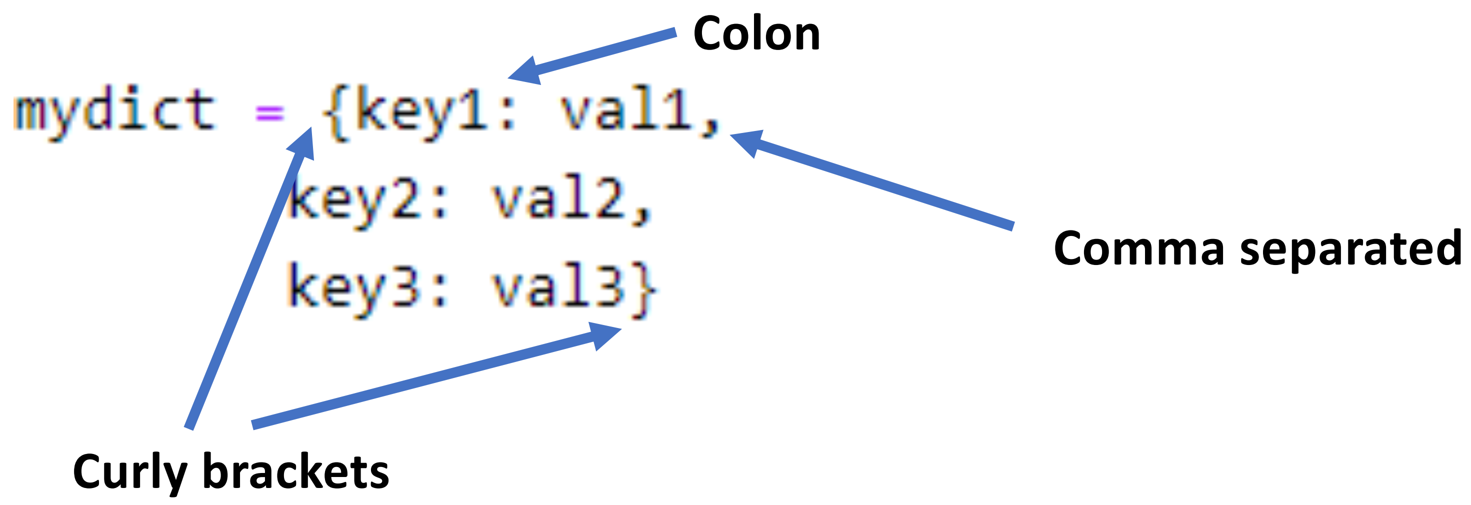
Dictionaries

- Similar to actual dictionaries
- They are effectively way to combine 2 lists – keys and values
- We use the **keys** to access the values instead of indexing them like a list
- Each value is mapped to a **unique** key



Dictionary definition

Defined as comma separated **key : value** pairs:



```
mydict = {key1: val1,
          key2: val2,
          key3: val3}
```

Dictionary properties

- Values are mapped to a key
- Values are accessed by their key
- Key are unique and are immutable
- Values cannot exist without a key

Dictionaries

Let us define a dictionary

```
days = {"Monday": "1",
        "Tuesday": "2",
        "Wednesday": "3",
        "Thursday": "4",
        "Friday": "5"}
print(type(days))
print(days)
```

```
<class 'dict'>
{'Monday': '1', 'Tuesday': '2', 'Wednesday': '3', 'Thursday': '4', 'Friday': '5'}
```

Accessing a dictionary

Values are accessed by their keys (just like a dictionary)

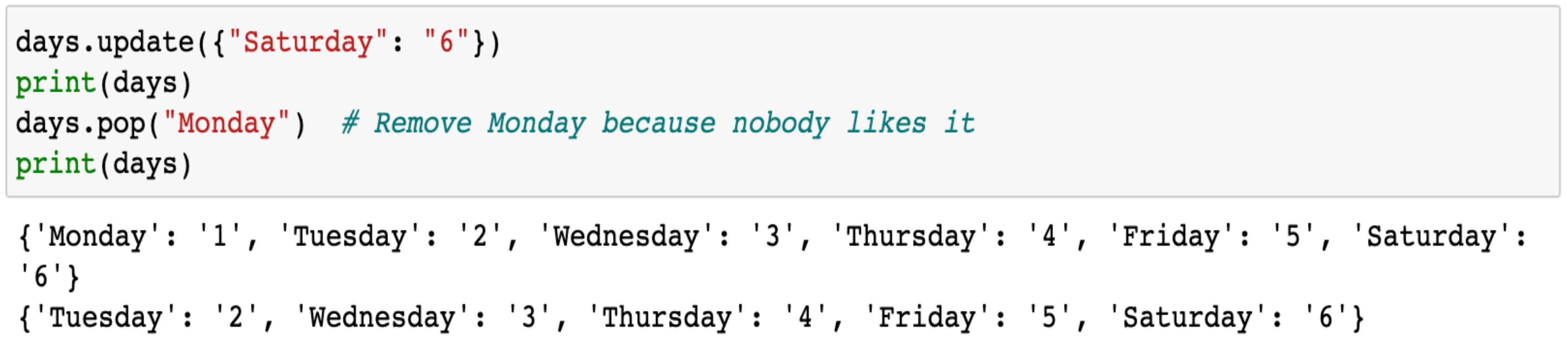
```
days["Friday"]
```

```
'5'
```

Note that they can't be indexed like a list

Altering a dictionary

Can be done via the dictionary methods



```python
days.update({"Saturday": "6"})
print(days)
days.pop("Monday")  # Remove Monday because nobody likes it
print(days)
```

```
{'Monday': '1', 'Tuesday': '2', 'Wednesday': '3', 'Thursday': '4', 'Friday': '5', 'Saturday':
'6'}
{'Tuesday': '2', 'Wednesday': '3', 'Thursday': '4', 'Friday': '5', 'Saturday': '6'}
```

Keys and Values

It is possible to obtain only the keys or values of a dictionary.

```
print(days.keys())   # get only the keys of the dictionary
print(days.values()) # get only the values of the dictionary
```

```
dict_keys(['Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday'])
dict_values(['2', '3', '4', '5', '6'])
```

This is useful for iteration.

Exercise time

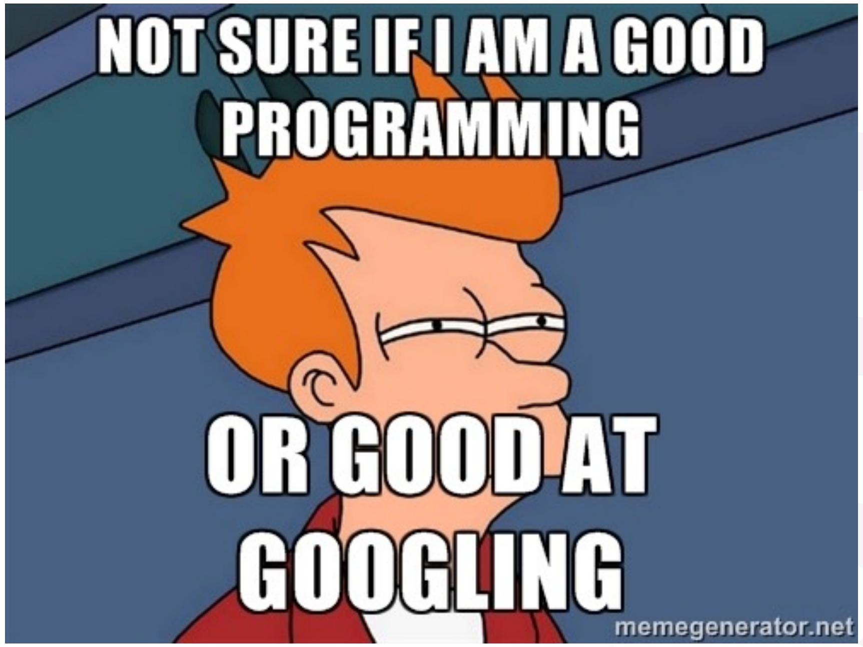
Simple and fun exercises.(Notebooks 2)

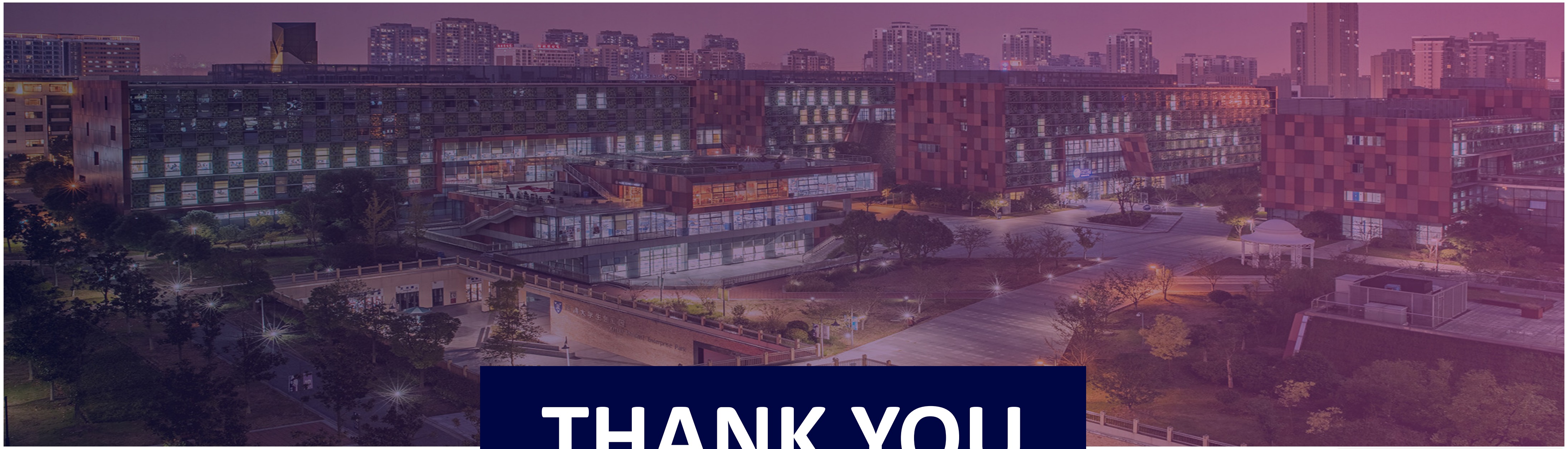
Failure is progress!

Ask us anything.

Ask among yourselves as well.

Google is your best friend when coding.





THANK YOU

VISIT US

WWW.XJTLU.EDU.CN

FOLLOW US

@XJTLU

Xi'an Jiaotong-Liverpool University
西交利物浦大学

XJTLU | SCHOOL OF FILM AND TV ARTS