L01_Jupyter_and_Python

August 21, 2024

1 Introduction to Jupyter and Python

Python is a programming language. The Jupyter Notebook is a browser-based graphical interface to the IPython shell, and builds on it a rich set of dynamic display capabilities. As well as executing Python statements, notebooks allow the user to include formatted text, static and dynamic visualizations, mathematical equations, and much more. Furthermore, these documents can be saved in a way that lets other people open them and execute the code on their own systems.

1.1 Introduction to Jupyter

- Keyboard shortcuts (see Help)
- Using markdown (basic syntax)
- Seeking help
- Magic functions (list of magic functions)
- Save and export Jupyter Notebook

1.1.1 Using markdown

Heading `#`, emphasis (bold `** **`, italic `* *`, bold + italic `* * *`), lists `-`, links `[]()`

Equation `$ $` and `$$ $$`

Bullet Point `*`

Link `[Link Text](URL)`

Math Text (LaTeX) `$` or `$$` (for centered equation on another line)

- Example 1: $ x_1 + x_2 x_3 $
- Example 2:

$$x_1 + x_2 \geq x_3$$

1.1.2 Seeking help

help `?`, access source code `??`, Tab completion of object contents `tab`

[ ]:
```
?len
```

```
Signature: len(obj,
/)
Docstring: Return the number of items in a container.
Type:      builtin_function_or_method
```

[ ]:
```
len??
```

```
Signature: len(obj,
/)
Docstring: Return the number of items in a container.
Type:      builtin_function_or_method
```

[ ]:
```
import numpy
```

[ ]:
```
numpy.random
```

[ ]:
```
<module 'numpy.random' from
'/Users/ivynangalia/Library/Python/3.12/lib/python/site-
packages/numpy/random/__init__.py'>
```

1.1.3 Magic functions

prefix by the % (line magic) and %% (cell magic)

[ ]:
```
%pip install numpy
# pip helps install packages(???)
```

```
Requirement already satisfied: numpy in
/Users/ivynangalia/Library/Python/3.12/lib/python/site-packages (1.26.3)
Note: you may need to restart the kernel to use updated packages.
```

[ ]:
```
%timeit L = [n ** 2 for n in range(1000)]
# timeit helps see how long running a certain line takes
```

```
34.2 µs ± 113 ns per loop (mean ± std. dev. of 7 runs, 10,000 loops each)
```

[ ]:
```
%%timeit
L = []
for n in range(1000):
    L.append(n**2)
```

```
25.1 µs ± 28.3 ns per loop (mean ± std. dev. of 7 runs, 10,000 loops each)
```

1.1.4 Save and export Jupyter Notebook'

E.g. download as an HTML

[ ]:
```

```

1.2 Introduction to Python

- Data types (boolean, int, float, complex, strings, None)
- Operators (arithmetic, assignment, comparison, logical, identity, membership, bitwise)

[ ]:
```
True, False
```

[ ]:
```
x = True
print(type(x))
```

[ ]:
```
1, 2, 3
```

[ ]:
```
x = 5
print(type(x))
```

[ ]:
```
x = 5.24
print(type(x))
```

[ ]:
```
x, y, z = 10, np.pi, False
print(x, y, z)
print(type(x), type(y), type(z))
```

[ ]:
```
1j
```

[ ]:
```
# type conversion
int(y)
```

[ ]:
```
None
```

[ ]:
```
x = "hello world!"
print(x)
```

[ ]:
```
x = "The paper says 'severe weather condition ...'"
print(x)
```

[ ]:
```
first_name = "Mickey"
last_name = "Mouse"
name = f"hello {first_name} {last_name}"
print(name)
```

[ ]:
```
name = "hello {} {}".format(first_name, last_name)
print(name)
```

[ ]:
```
name.lower(), name.upper()
```

[ ]:
```
print("Disney characters:\nMickey Mouse\nDonald Duck")
```

[ ]:
```
name = "Mickey Mouse      "
print(name)
name
```

[ ]:
```
name.rstrip()
```

1.2.1 Arithmetic operators

+, -, *, /, //, %, **

[ ]:

1.2.2 Assignment operators

=, +=, -=, *=, /=, ...

[ ]:

1.2.3 Comparison operators

==, !=, <=, <, >=, >

[ ]:
```
1 == 3
```

[ ]:
```
x = 1
y = 3
x == y
```

[ ]:
```
x <= y
```

[ ]:

1.2.4 Logical operators: combine conditional statements

and, or, not

[ ]:

1.2.5 Identity operators: compare the objects, not if they are equal, but if they are actually the same object, with the same memory location

is, is not

[ ]:
```
x is not y
```

[ ]:
```
x = 1
y = 1.0
print(x is y, x==y)
```

[ ]:
```
x = [2,3]
y = [2,3]
x == y, x is y
```

[ ]:
```
id(x), id(y)
```

[ ]:
```
x = [2,3]
y = x
x == y, x is y
```

[ ]:
```
x[0] = -1
x, y
```

1.2.6 Membership operators: test if a sequence is presented in an object

`in`, `not in`

[ ]:
```
'a' in 'apple'
```

[ ]:
```
3 in [1,2,3,4,5]
```

[ ]:

[ ]:

1.2.7 Bitwise operators: compare (binary) numbers

`&`, `|`, `^`

[ ]:
```
format(10, '08b')
```

[ ]:
```
format(7, '08b')
```

[ ]:
```
x = 10 & 7
x, format(x, '08b')
```

[ ]:

1.2.8 Operator precedence: describe the order in which operations are performed

[ ]:
```
5*2**3/(4+8) - 1
```

[ ]:

[ ]:

1.2.9 More strings

1. Searching and indexing
2. Splitting and joining
3. Read and write text files

[ ]:
```
s = "hello world"
s
```

[ ]:
```
'hello world'
```

[ ]:
```
"hel" in s
```

[ ]: True

[ ]:
```
"loo" in s
```

[ ]: False

[ ]:
```
s.find("llo")
```

[ ]: 2

[ ]:
```
s.index("llo")
```

[ ]: 2

[ ]:
```
s.find("loo")
```

[ ]: -1

[ ]:
```
s.index("loo")
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-13-4641be587664> in <module>
----> 1 s.index("loo")

ValueError: substring not found
```

[ ]:
```
s.count("o")
```

[ ]: 2

[ ]:
```
s.count("oo")
```

[ ]: 0

[ ]:

[ ]:

[ ]:
```
s.split(" ")
```

[ ]: ['hello', 'world']

[ ]:
```
s.split("l")
```

[ ]:
```
['he', '', 'o wor', 'd']
```

[ ]:
```
'-'.join(s)
```

[ ]:
```
'h-e-l-l-o- -w-o-r-l-d'
```

[ ]:
```
"-".join(s.split(" "))
```

[ ]:
```
'hello-world'
```

[ ]:

[ ]:
```
import string
```

[ ]:
```
s = "Python is a high-level,*./?! general-purpose programming language.*./<>"
s.translate(str.maketrans('', '', string.punctuation))
```

[ ]:
```
'Python is a highlevel generalpurpose programming language'
```

[ ]:

[ ]:

[ ]:
```
%%file test.txt
Hello world!
This is a test file.
Good luck
```

```
Writing test.txt
```

[ ]:
```
with open("test.txt") as f:
    for line in f:
        print(line, end='')
```

```
Hello world!
This is a test file.
Good luck
```

[ ]:
```
with open("test.txt") as f:
    out = f.read()
out
```

[ ]:
```
'Hello world! \nThis is a test file. \nGood luck\n'
```

[ ]:
```
with open("test_alt.txt", "w") as f:
    f.write(out)
```

[ ]: