A brief start to python

Tomas Bylund

Linnaeus University,
Växjö, Sweden

January 28, 2020

Why Python?

From `Python.org`

> *Python is a programming language that lets you work more quickly and integrate your systems more effectively.*

This is true from a developer point of view: python is easy to learn, and there is an enormous library of packages that interface with mostly anything connected to computers.

Outline

Getting python

The most popular index of python packages has 215000 different projects with 1650000 different releases

- That's a lot of packages
- Not all package versions compatible with other packages at other versions

This leads to the following goals

- Don't mix different versions of same package in the same installation
- Don't resolve version conflict manually!
- Don't keep track of this manually!

Use a python *package manager* and *virtual environments*

Installing miniconda

Miniconda is the minimal distribution of Anaconda, a package manager and virtual environment manager.

Download it from
`docs.conda.io/en/latest/miniconda.html`

Then follow instructions for your OS found at
`docs.conda.io/projects/conda/en/latest/user-guide/install/`

Managing with conda

Once installed, you should be able to run conda in a terminal (miniconda terminal on windows) by calling:

```
conda --version
```

We can now use conda to create a virtual envirnoment to keep our python packages in using:

```
conda create --name kodkoll  python=3
```

This will take a bit, and install packages after you let it.
A virtual environment is when you mess around with environment variables to keep things separated, and it is best handled by code someone else wrote and already tested.

Managing with conda

Now activate this environment

```
conda activate kodkoll
```

We are now "inside" the virtual envirnoment, and your terminal prompt should be modified to indicate this.
Lets now install the rest of the packages we will try to use

```
conda install numpy matplotlib pandas requests
conda install ipython jupyter
```

As you see there are a lot of things required to get these six packages to run, best not to keep track of all that by hand

Running python

You can now run python (from a terminal) with

```
python
```

Write the following into a text file called `hello_world.py` that you save in your current directory

```
print("Hello World!")
```

you can then tell python to execute this script with

```
python hello_world.py
```

Try modifying `hello_world.py` so it reads

```
import os;print(f"Hello {os.environ['USER']!")
```

and then run it

Python

The following is a excerpts from a condensed version of `https://learnxinyminutes.com/docs/python3/`, called `LearnPythonBrief.py` found in the git repository

```
# Single line comments start with a number symbol.

"""
Multiline strings can be written
using three "s, and are often used
as documentation.
"""
```

Python

```
# Math is what you would expect
1 + 1   # => 2
10 * 2  # => 20

# The result of division is always a float
10.0 / 3  # => 3.3333333333333335

# Exponentiation (x**y, x to the yth power)
2**3  # => 8

# Enforce precedence with parentheses
1 + 3 * 2  # => 7
(1 + 3) * 2  # => 8
```

Python

```
# Strings are created with " or '
"This is a string."
'This is also a string.'

# A string can be treated like a list of characters
"This is a string"[0]  # => 'T'
# .format can be used to format strings, like this:
"{} can be {}".format("Strings", "interpolated")
# => "Strings can be interpolated"
# You can also format using f-strings or formatted string l
name = "Reiko"
f"She said her name is {name}."
# => "She said her name is Reiko"
```

Python

```
# Here is an if statement. Indentation is significant in Py
# Convention is to use four spaces, not tabs.
# This prints "some_var is smaller than 10"
if some_var > 10:
    print("some_var is totally bigger than 10.")
elif some_var < 10:    # This elif clause is optional.
    print("some_var is smaller than 10.")
else:                  # This is optional too.
print("some_var is indeed 10.")
```

Python

```
"""
For loops iterate over lists
prints:
    dog is a mammal
    cat is a mammal
    mouse is a mammal
"""
for animal in ["dog", "cat", "mouse"]:
    # You can use format() to interpolate formatted strings
    print("{} is a mammal".format(animal))
```

Python

```
# Use "def" to create new functions
def add(x, y):
    print("x is {} and y is {}".format(x, y))
    return x + y  # Return values with a return statement

# Calling functions with parameters
add(5, 6)  # => prints out "x is 5 and y is 6" and returns
```

Python

```
import math
print(math.sqrt(16))  # => 4.0

# You can get specific functions from a module
from math import ceil, floor
print(ceil(3.7))   # => 4.0
print(floor(3.7))  # => 3.0

# You can shorten module names
import math as m
math.sqrt(16) == m.sqrt(16)  # => True
```