Tutorial 0: Getting started

BS1819 Data Structures and Algorithms, Autumn 2018

Imperial College Business School

By the end of this tutorial you should have:

- **signed up for the OK code feedback service**
- **installed Python** on your laptop
- learned the basic usage of the **command line** on your operating system
- **tried out Python** in two different **programming environments**.

Part I: Setup

OK

In this course, we will use a program called `ok` for coding feedback. The program will be included with the assignments.

To use the program, you'll first need to register for the `ok` service.

- Visit the okpy website. Click on "Login" in the top right corner.
- You'll be redirected to a Google sign-in page. In order to log in, you will need a Google account. Here, you have two options.
    1. If you have an existing Google account, use it to log in. Simply log into your account, and it will be linked to `ok`.

2. If you don't have a Google account, there's a link to create one. On the `ok` login page, click on "More options", and then "Create account".

- After you've registered, the `ok` website will show that you're not enrolled to any course yet at the moment.
- **In order to enrol you to the right course, we need the email address of your Google account. You can submit this in the intro survey here. Please do this before the first tutorial.**

We will go over the use of `ok` in the first tutorial.

Python

Python is a popular, freely available programming language used for a broad range of applications. It is an essential tool across academia and industry for many data-intensive tasks, such as extracting information from large datasets, scraping the web, and automating repetitive computations. Python is an open-source language with an enormous community of users developing tools around it, meaning that for fields such as analytics, there are many useful and well-maintained tools and packages available. In practice, for any analytics task you can think of, chances are that someone has written a package to solve that problem.

We will introduce some of these packages later on in the module. To make sure we have them ready, we will install Python with a useful collection of packages (such collections are often referred to as *distributions*) called **Anaconda**.

> **Advanced**. If you're already comfortable with Python, you're welcome to use a different Python distribution. Otherwise, you should get Anaconda as it contains all packages we'll need (and more) in one convenient download.

Installing Anaconda

Please follow these instructions carefully - otherwise you may end up with the wrong version of Python and will need to start over.

- Download the Anaconda installer here. Scroll down to find "Download for Windows", "Download for macOS", or "Download for Linux" depending on your operating system.
- Make sure to select the option for **Python 3.6**, NOT Python 2.7. There are different versions of Python around, and we will be using the latest version 3.6.
- Run the installer and follow its instructions. It may ask you to make some choices; the default options should be fine. The installation will take a while.
- The installer may prompt you to also install Visual Studio Code; you don't need to do so.

You should now have Anaconda installed on your laptop. You can check this by running the Anaconda Navigator, as follows:

- On Windows, click on the Start button or hit the Windows button and start typing `Navigator`.
- On macOS, you can use Finder or Launchpad to find Anaconda Navigator.

The Navigator shows different ways of using Python on your machine. We will use three such *programming environments* in this module.

Programming environments

Programming environments are user interfaces for writing and running code. You can think of environments as follows. You can write text documents in both Notepad and Microsoft Word. These are "environments" for text documents. Think of a Python program as a document and environments as Notepad or Word.

We'll use three programming environments in this module.

- **The Python interpreter** is a text-based interface that allows us to type in Python-language commands to be executed line-by-line.
- **Spyder** is a graphical interface that combines a Python interpreter with a text editor and various enhancements to make coding easier.
- **Jupyter** is a browser-based interface, which allows us to create **Jupyter notebook** documents that combine code and its output with text commentary. This can be very useful for presenting and sharing your work with colleagues, and is common practice in analytics teams.

Part II: Running Python

You now have Anaconda installed, and ready to run Python in the first tutorial.

In the remainder of this tutorial we'll take Python out for a spin. We'll try out the Python interpreter and Spyder, and return to Jupyter later in the course. To run the Python interpreter, we'll use the **command line** on your machine. These are the tools that we'll use throughout the course to work with Python code. If you're not familiar with them, you should try them out before the first tutorial.

Using the command line

We normally interact with the computer with a mouse or a touch screen via a graphical user interface, clicking on icons, buttons, hyperlinks, and so on. The command line is an alternative **text-based** interface to interact with the computer and run programs. For some tasks, it is less convenient than the regular graphical interface we use with a mouse, but for others it gives us more flexibility in running programs.

Opening the command line

Windows

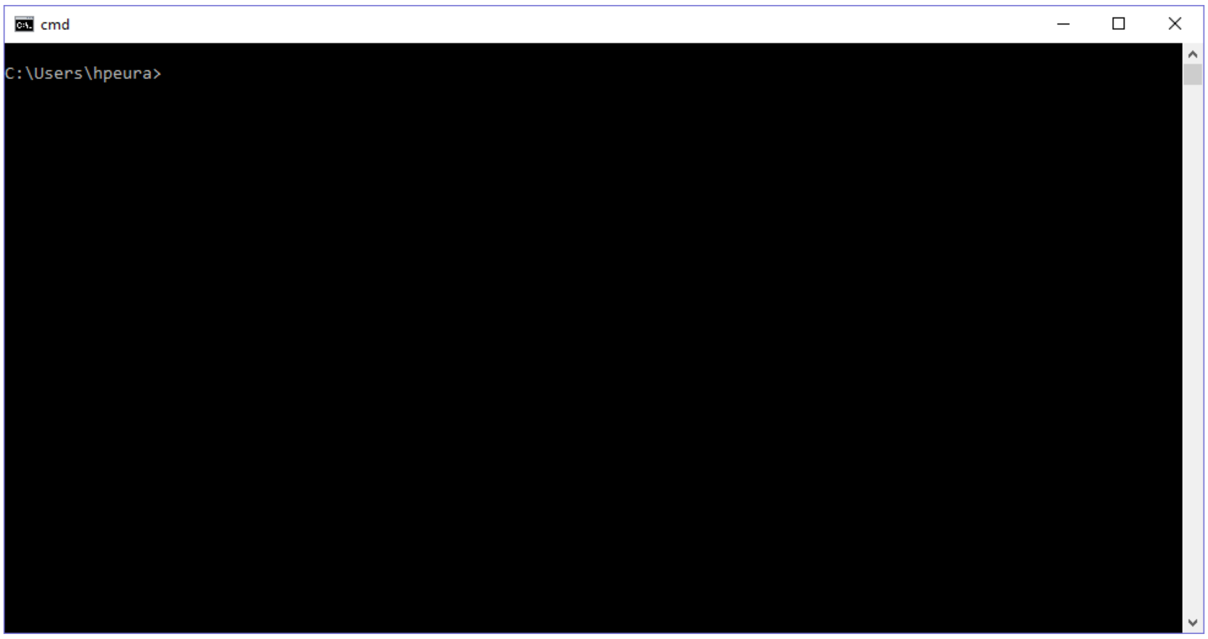
Press the Windows key and type `Anaconda Prompt`. This should open the start menu and find the Anaconda version of the Windows command line (called "Anaconda Prompt").

macOS

Use the Key combination `Cmd-Space`, type "terminal" and press Enter to open the macOS command line (called "Terminal").

Navigating to a folder

On Windows, the command prompt window will look something like this. The macOS view is slightly different. Whether you're on Windows or macOS, you can follow the steps below - we'll specify the differences as we go along.

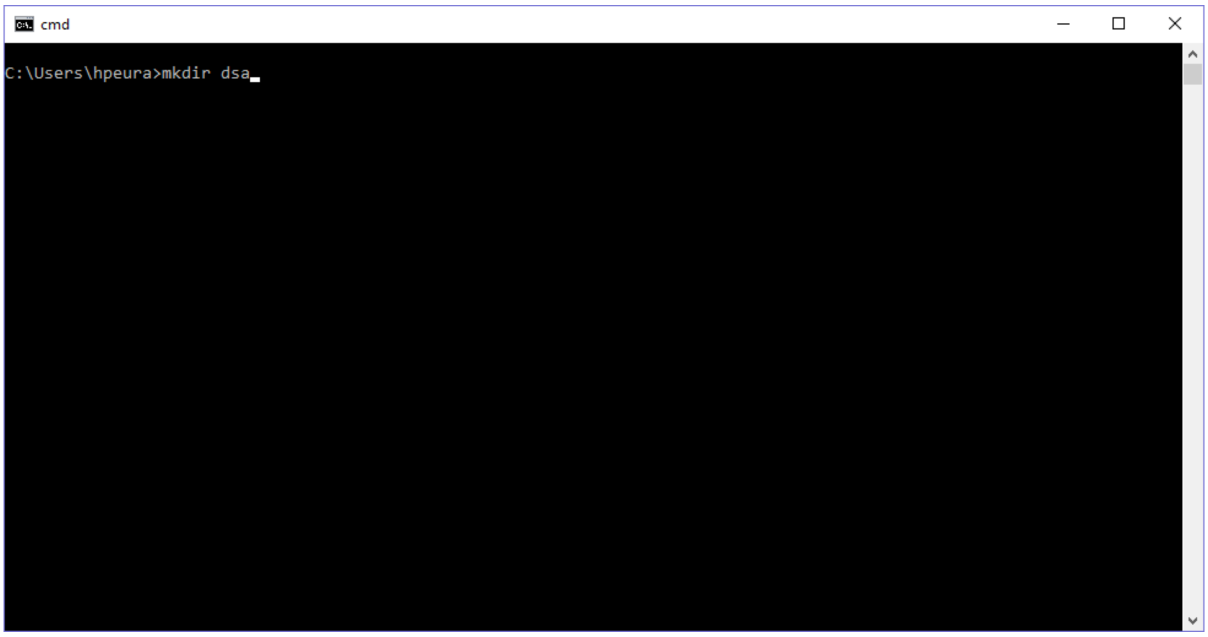


```
C:\Users\hpeura>
```

The interface shows us where we are in the computer's file system: here, in the directory `C:\Users\hpeura`. You can view the contents of the directory by typing `dir` and pressing enter in Windows and `ls` in macOS. You should see the same files that you see when navigating to this folder in the graphical file explorer interface. (In Windows, you can launch the graphical file explorer with the command `explorer`.)

Now, let's create a new directory for our coursework. Type in the following command and press Enter. (`mkdir` is for "make directory".)

```
mkdir dsa
```

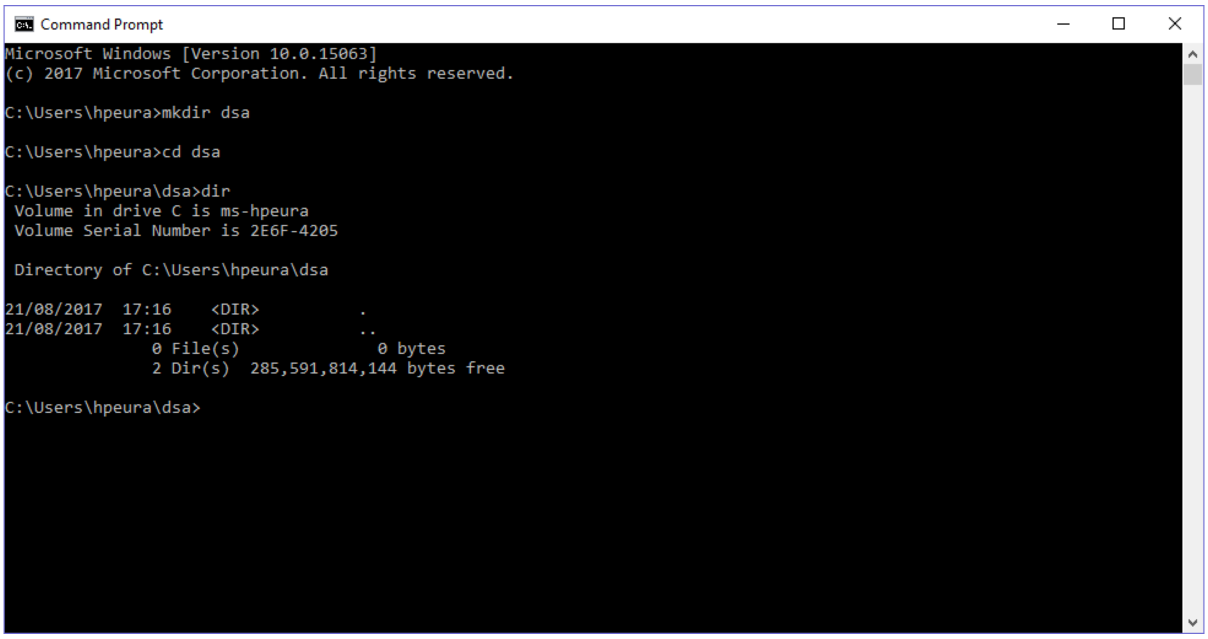


```
C:\Users\hpeura>mkdir dsa
```

To get to the new folder, we can type in the command (`cd` is for "change directory")

```
cd dsa
```

This moves us to the folder in the directory tree. We can again see the contents of the current folder with the command `dir` on Windows, and `ls` in macOS. The screenshot below shows that currently there are no files in the folder. You can also check that the folder now appears in the regular file explorer.



```
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\hpeura>mkdir dsa

C:\Users\hpeura>cd dsa

C:\Users\hpeura\dsa>dir
 Volume in drive C is ms-hpeura
 Volume Serial Number is 2E6F-4205

 Directory of C:\Users\hpeura\dsa

21/08/2017  17:16    <DIR>          .
21/08/2017  17:16    <DIR>          ..
               0 File(s)              0 bytes
               2 Dir(s)  285,591,814,144 bytes free

C:\Users\hpeura\dsa>
```

If we wish to navigate back to the parent folder, we can use the command

```
cd ..
```

Navigating folders on the command line will come in handy in the first tutorial. We recommend that you keep all the course materials in the same folder for easy access.

We can run various programs from the command line just like we can using the graphical interface.

For now, let's run Python.

The Python interpreter

On the command line, start the Python interpreter by typing `python` and pressing Enter (Python will work in any directory).

You should be greeted by something like this:

```
Python 3.6.6 |Anaconda custom (64-bit)| (default, Jun 28 2018, 11:27:44) [MSC v.1900 64
bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Opening the interpreter prints out basic information, such as the version numbers of Python and Anaconda.

The `>>>` indicates the Python prompt, which you use to communicate with Python. This is the Python interpreter: Python is waiting for you to type in something for it to interpret.

Let's type in a command:

```
print('Hello, world!')
```

The result should look like this:

```
>>> print('Hello, world!')
Hello, world!
>>>
```

This means that Python has interpreted you command, resulting in the display of the text string 'Hello, world!' on your screen, and has also displayed a new prompt for you to type another command.

Let's try a different command:

```
>>> 2 + 2
```

This should of course print out 4.

You can exit the Python interpreter by typing

```
>>> exit()
```

You then return to the terminal or command prompt.

Spyder

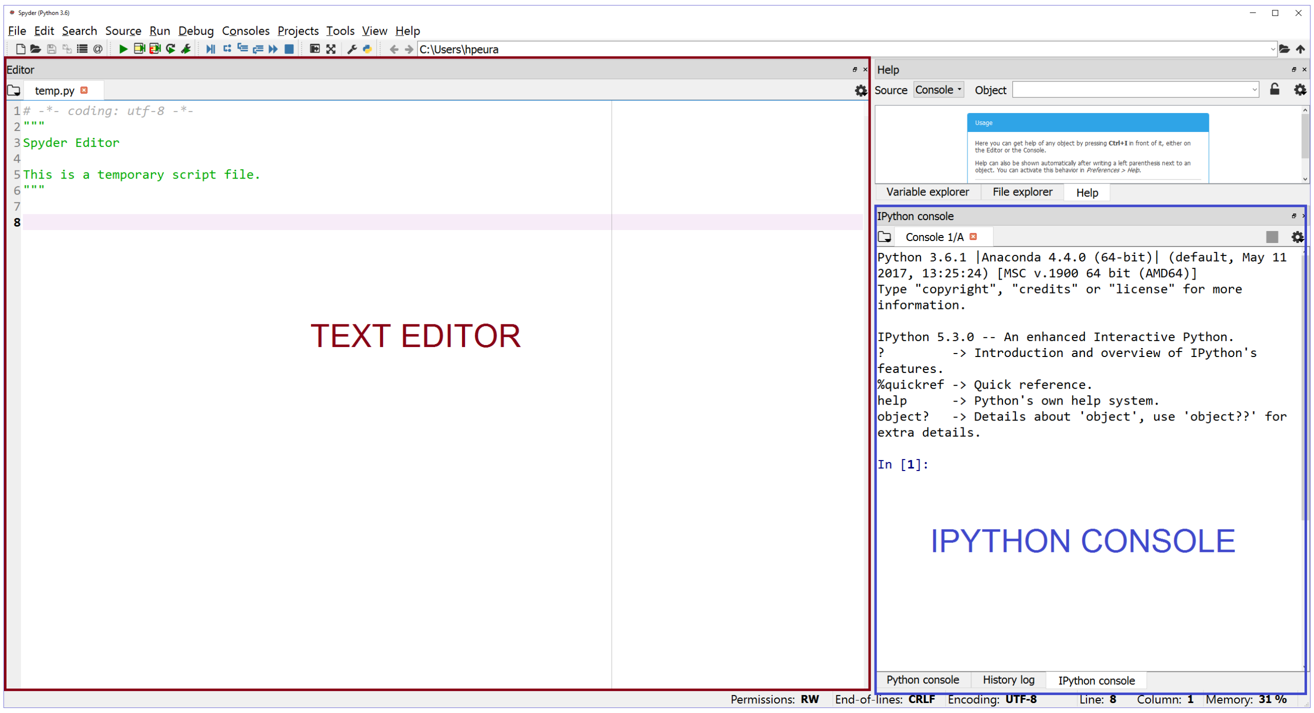
The above Python commands are one-line snippets that are quick and easy to run in the Python interpreter. But when we start building more complex programs, this mode of running code becomes inconvenient. A better way is to store our commands in a text file, and send the entire file to the interpreter. The interpreter will then read the file line by line and execute the commands.

These files are often called *scripts*. Scripts are a convenient way to save our code for future use. We can write them using any text editor, but many programmers use editors that are part of an *integrated development environment* (IDE), which aim to make programming easier and more fun in various ways. Our Anaconda installation comes with a widely-used one called **Spyder**.

Having installed Anaconda, you should find Spyder on your laptop. Open the Anaconda Navigator or search for Spyder directly on your computer:

- In Windows, press the Windows key and start typing `spyder`.
- in macOS, use Finder or Launchpad to find Spyder.

Go ahead and open Spyder. It may take a while to load, but eventually you should see an interface like this:


Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\hpeura
Editor
temp.py
TEXT EDITOR
Help
Source Console Object
Usage
Variable explorer File explorer Help
IPython console
Console 1/A
IPYTHON CONSOLE
Python console History log IPython console
Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 8 Column: 1 Memory: 31 %

```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""

```

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

```
Python 3.6.1 |Anaconda 4.4.0 (64-bit)| (default, May 11
2017, 13:25:24) [MSC v.1900 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more
information.

IPython 5.3.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's
features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for
extra details.

In [1]:
```

The interface can look complicated at first but you will soon get used to it. Look first at the bottom right panel. This is an **IPython** console, which is an enhanced version of the Python interpreter we were just using on the command line. The main visible difference is that instead of `>>>` it prompts you for commands with `In [1]`. You can run the above commands here just like before.

```
In [1]: 2 - 1
Out[1]: 1
```

Next look at the top right panel. It displays helpful information on your code, variables, and functions, but do not worry about this too much for now.

The left panel is a text editor, where we can write and edit code scripts and save them to our hard drive. But different from the console, this code will not run immediately. Instead, we'll have to invoke Spyder's commands to run our code or parts of it.

When we create a new file with Spyder, it comes with a prefix, something like:

```
# -*- coding: utf-8 -*-
"""
Created on Wed Aug 23 15:06:38 2017

@author: hpeura
"""
```

This is a comment area that allows us to spell out what our script does, either for our own future reference or for others who will use our code. To allow these kind of comments, the Python interpreter will not run lines starting with `#` or anything between triple quotations `"""`.

Let's create a simple script. Copy the following code into the editor, below the comment.

```python
print('Hello world') # Everything after # is a comment.
# The above line will print out "Hello world"
print(3 + 3)
print('3 + 3')
print('It works!')
```

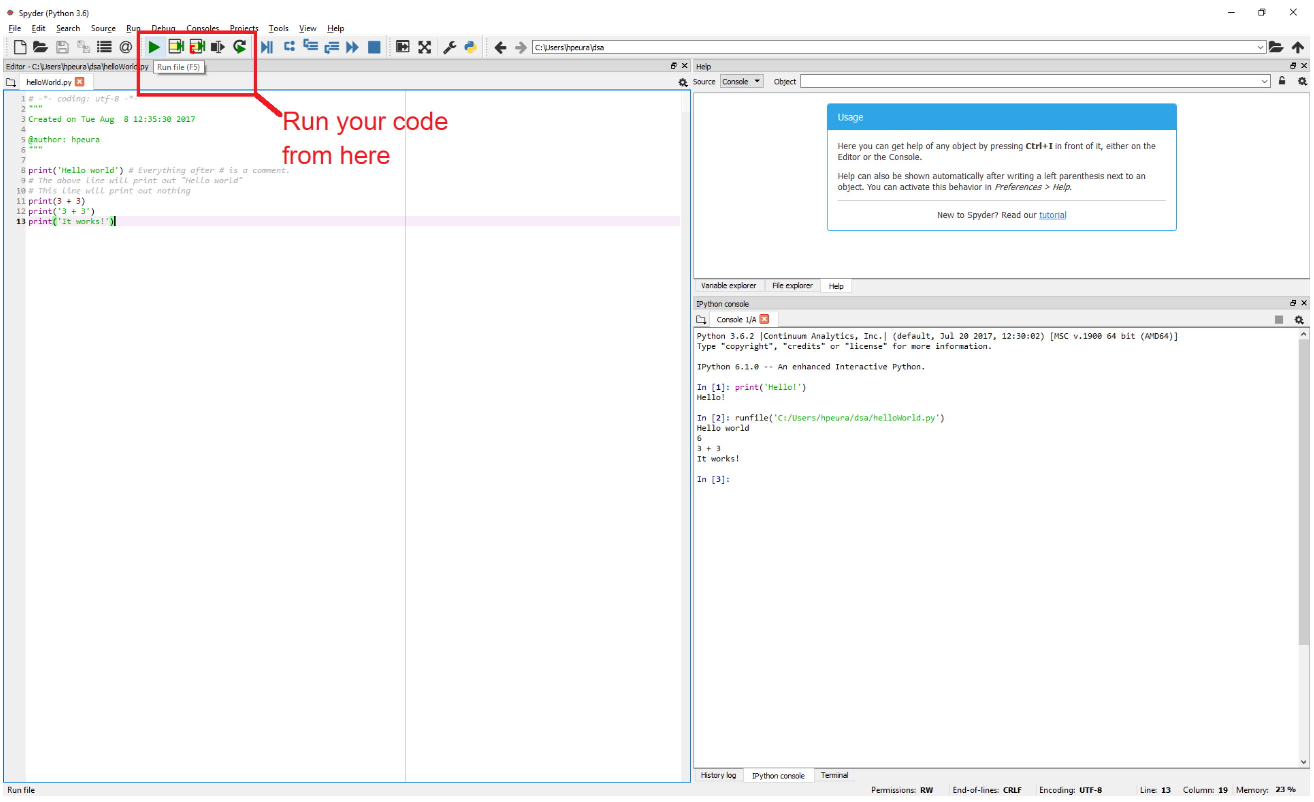
Let's save the file. Find the **File** menu on the top left corner of the screen, and **Save As** in the menu. Then navigate to a suitable directory, for example the `dsa` directory that you've just created on the command line. Give your script a filename, for instance `helloWorld.py`, and save it in the directory. (The file extension for Python scripts is `.py`.)

We have now written several lines of code in the text editor. The difference of working in the editor compared with the Python interpreter or the IPython console is that this code does not automatically run as we type it, which is convenient for longer pieces of code. So how do we execute it when we're ready?

We can do this in a couple of different ways in Spyder. To execute the entire file, select **Run->Run** from the menu. You can also do this by pressing the green "play" button, or hit the `F5`-key.

Go ahead and run the program. The output will appear in the IPython console window on the bottom right: it first indicates that you sent the entire script file to the interpreter, and then the printed output of the script. Some of the output may look surprising - we will go through how these and many other Python commands work in the first tutorial.

Often we do not want to run the entire script, but just try out a few lines. To do this, select the code you wish to run, for example by clicking and dragging over the lines. Then select **Run->Selection or Current Line** from the menu, or hit the `F9`-key. The output of the selection should appear in the console.



Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\hpeura\dsa
Editor - C:\Users\hpeura\dsa\helloWorld.py
Run file (F5)
helloWorld.py
Run your code from here
Help
Source Console Object
Variable explorer File explorer Help
IPython console
Console 1/A
History log IPython console Terminal
Run file
Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 13 Column: 19 Memory: 23 %

```python
# -*- coding: utf-8 -*-
"""
Created on Tue Aug  8 12:35:30 2017

@author: hpeura
"""

print('Hello world') # Everything after # is a comment.
# The above line will print out "Hello world"
# This line will print out nothing
print(3 + 3)
print('3 + 3')
print('It works!')
```

Usage

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

```
Python 3.6.2 |Continuum Analytics, Inc.| (default, Jul 20 2017, 12:30:02) [MSC v.1900 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 6.1.0 -- An enhanced Interactive Python.

In [1]: print('Hello!')
Hello!

In [2]: runfile('C:/Users/hpeura/dsa/helloWorld.py')
Hello world
6
3 + 3
It works!

In [3]:
```

We will use Spyder throughout the module to work on tutorials and assignments.

All done!

We're now ready to write and run Python programs in both the Python interpreter and Spyder. We will start working on Python basics in tutorial one.

Review questions

- What are the different ways we run Python code on our computer?
- In Spyder, identify the editor and the IPython console. What's the difference between writing Python commands in the two?