Introduction to Spyder

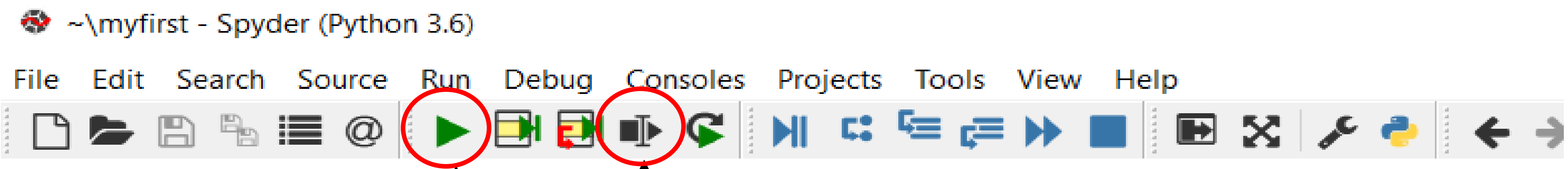
GITAA
Transforming careers

In this lecture

- How to execute a Python file?
- How to execute pieces of code - Run?
- How to add comments?
- How to reset and clear console

File execution

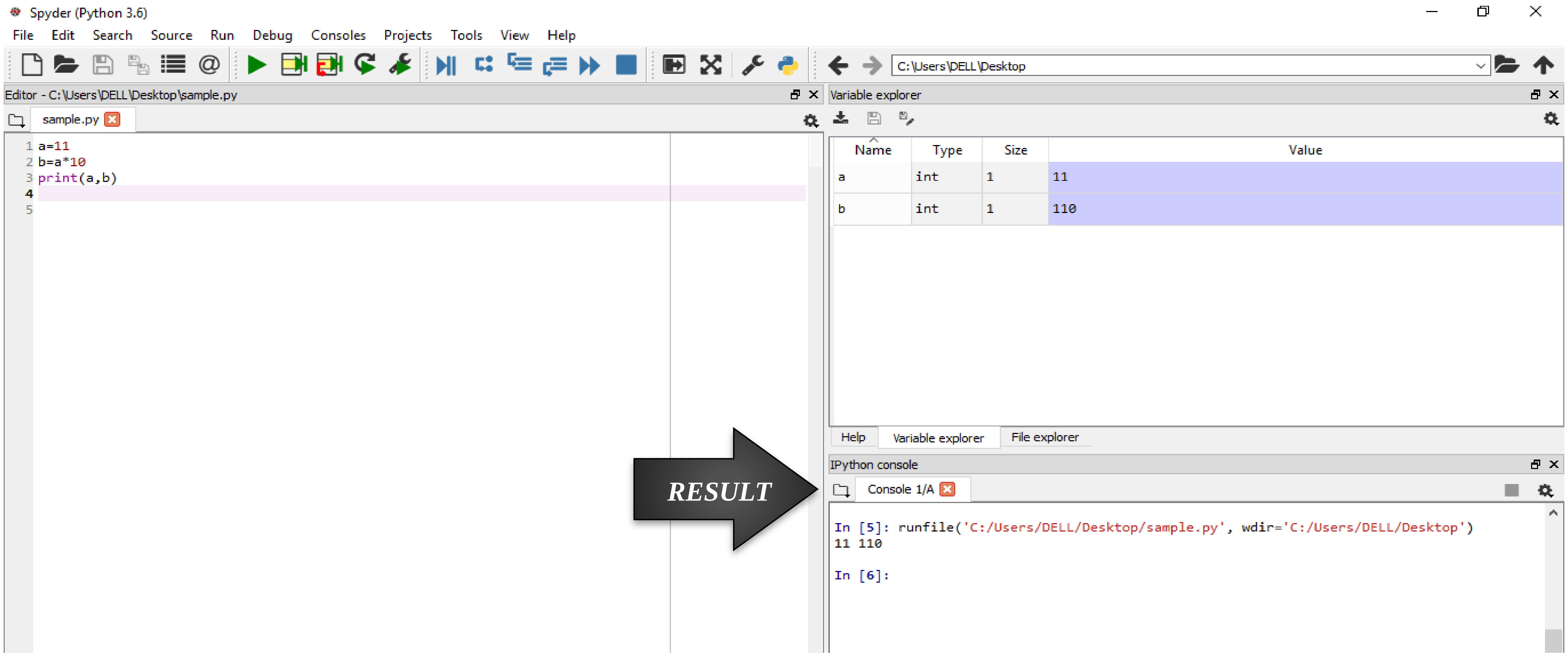
Executing script files


~\myfirst - Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help

To run full code:-
1. Press ‘*Run file*’ from icon bar
2. F5 to run full code

To run chosen line, select the line and
1. Press *‘Run selection’* from icon bar
2. Press *Ctrl+Enter or F9*

Executing script files using Run file/F5



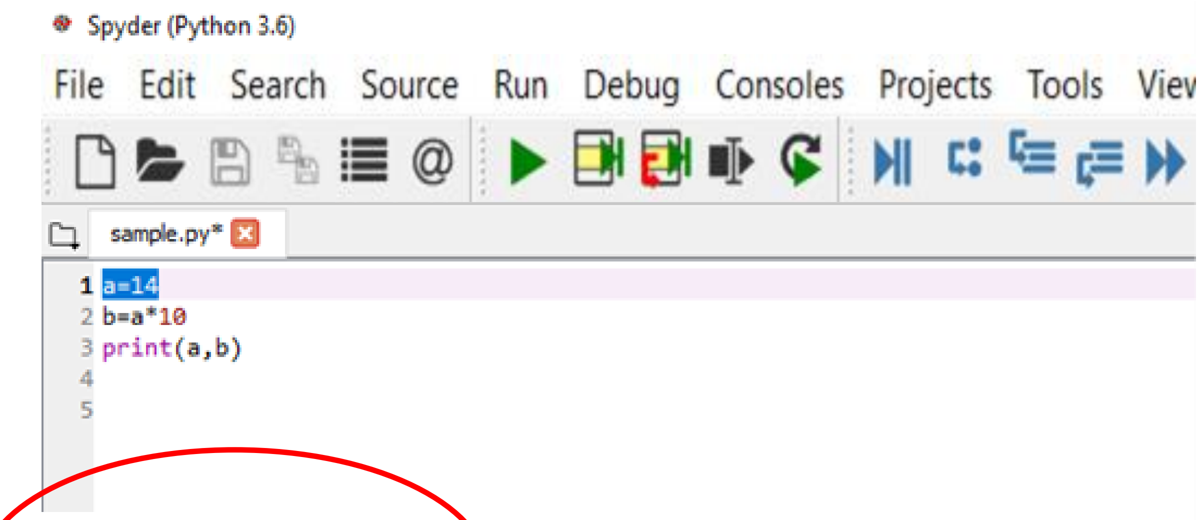
```
a=11
b=a*10
print(a,b)
```

Name	Type	Size	Value
a	int	1	11
b	int	1	110

```
In [5]: runfile('C:/Users/DELL/Desktop/sample.py', wdir='C:/Users/DELL/Desktop')
11 110

In [6]:
```

Executing script files using Run selection/F9

Step 1: Assign a new value of 14 to 'a' in the script and press F9


Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View
sample.py*

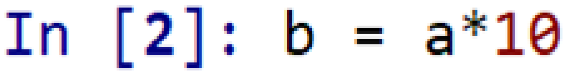
```
a=14
b=a*10
print(a,b)
```

```
In [1]: a = 14
```

Console output

Executing script files using Run selection/F9

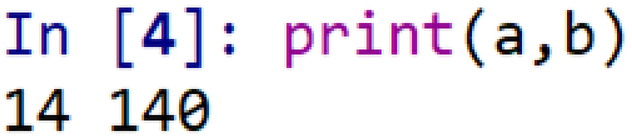
Step 2: Select line 2 and press F9



```
In [2]: b = a*10
```

→ **Console output**

Step 3: Select line 3 and press F9



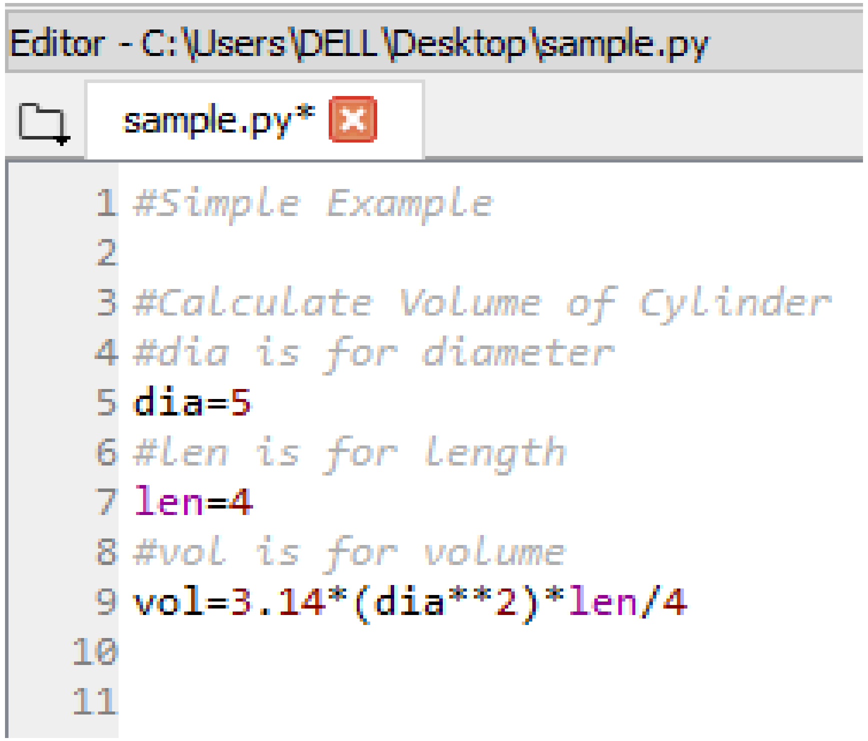
```
In [4]: print(a,b)
14 140
```

→ **Console output**

Commenting script files

Commenting lines of codes

- Adding comments will help in understanding algorithms used while developing codes
- In practice, commented statements will be added before the code and begin with a '#'
- Multiple lines can also be commented



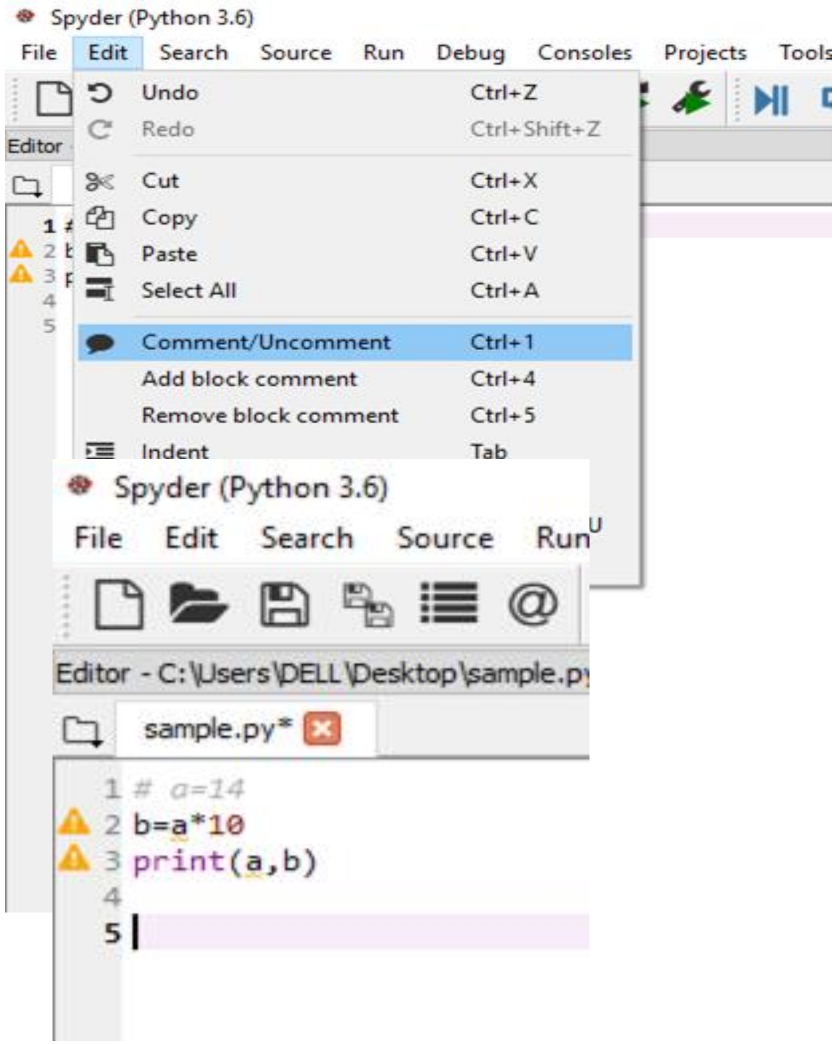
```
#Simple Example

#Calculate Volume of Cylinder
#dia is for diameter
dia=5
#len is for length
len=4
#vol is for volume
vol=3.14*(dia**2)*len/4
```

Commenting multiple lines

- Select lines that have to be commented and then press "Ctrl + 1"

- Select "Edit" in menu and select "Comment/Uncomment"

- Uses - to add description, render lines of code inert during testing

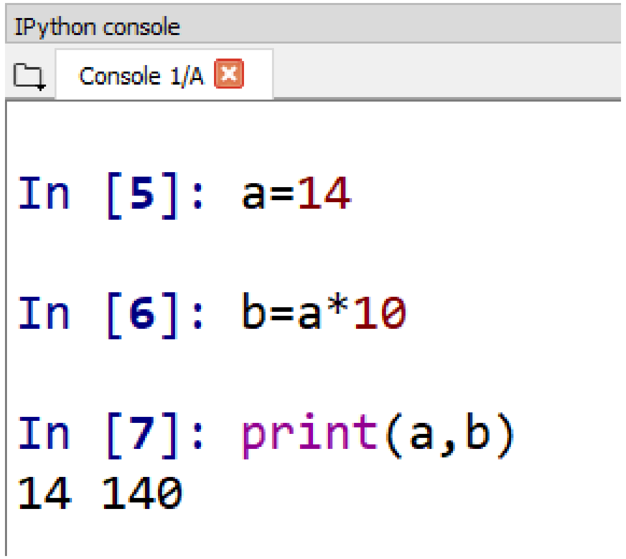


```
# a=14
b=a*10
print(a,b)
```

Clearing console and environment

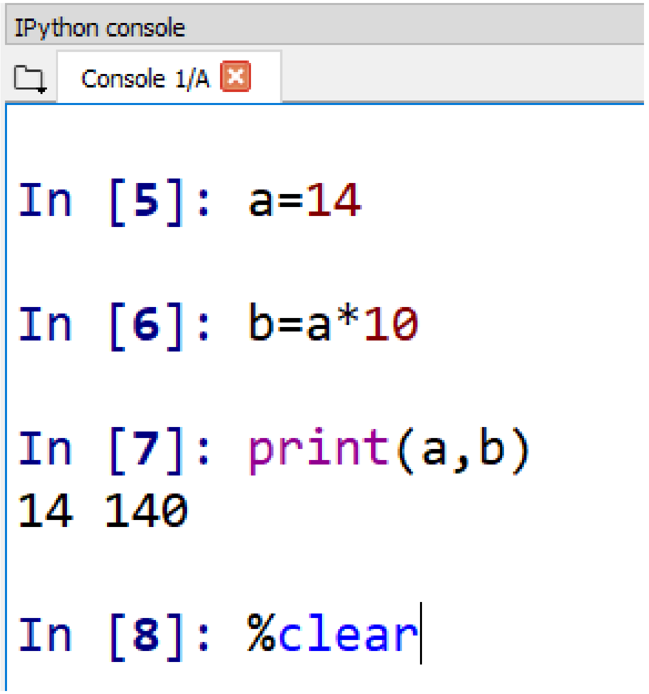
Clearing an overpopulated console

Console



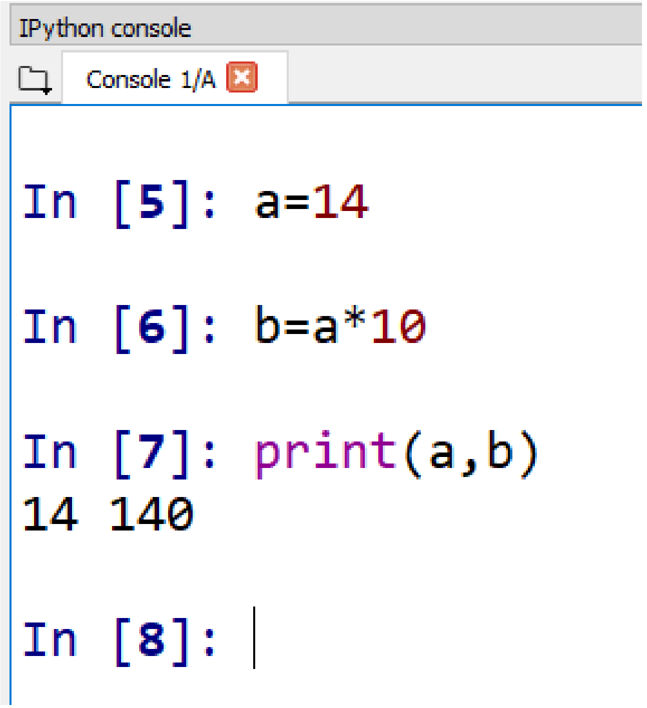
```
In [5]: a=14

In [6]: b=a*10

In [7]: print(a,b)
14 140
```

Type *%clear* in console



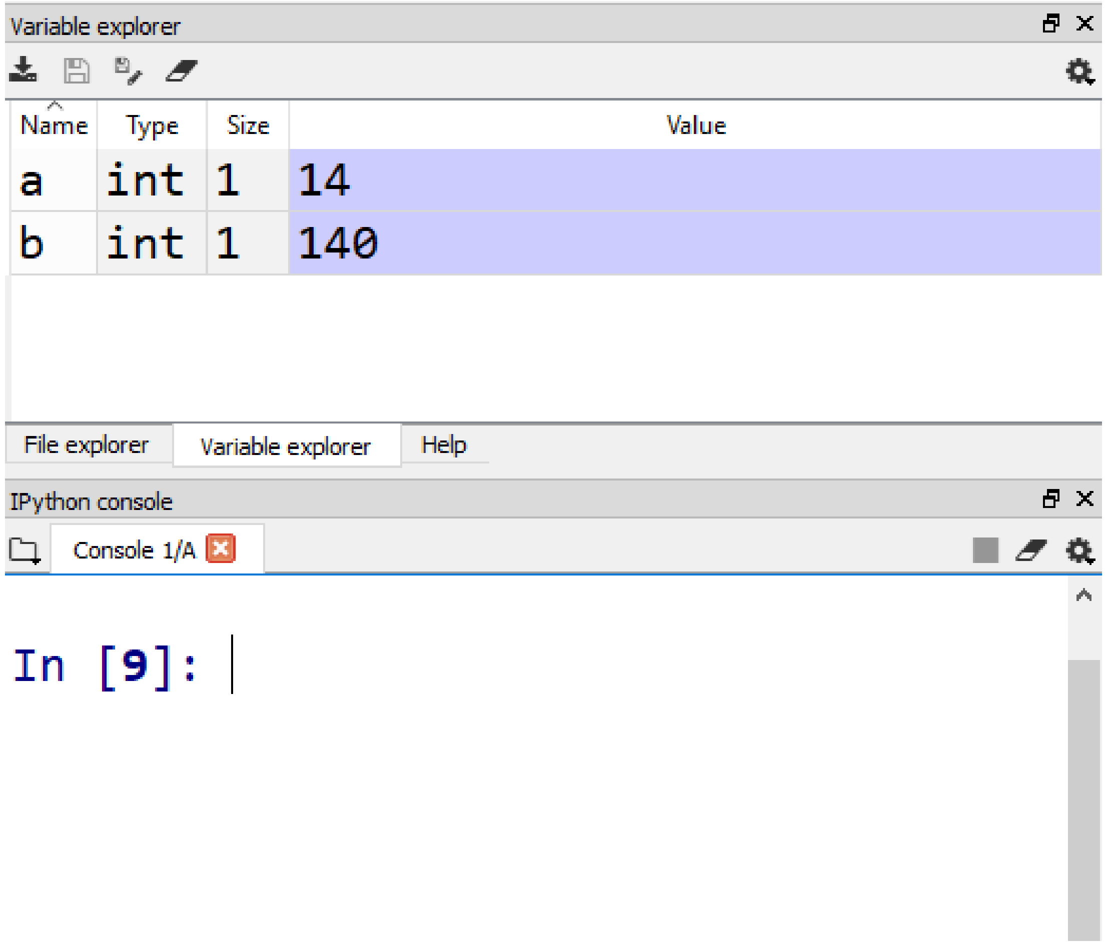
```
In [5]: a=14

In [6]: b=a*10

In [7]: print(a,b)
14 140

In [8]: %clear
```

Place cursor on console and press *Ctrl+L*



```
In [5]: a=14

In [6]: b=a*10

In [7]: print(a,b)
14 140

In [8]:
```

After clearing an overpopulated console

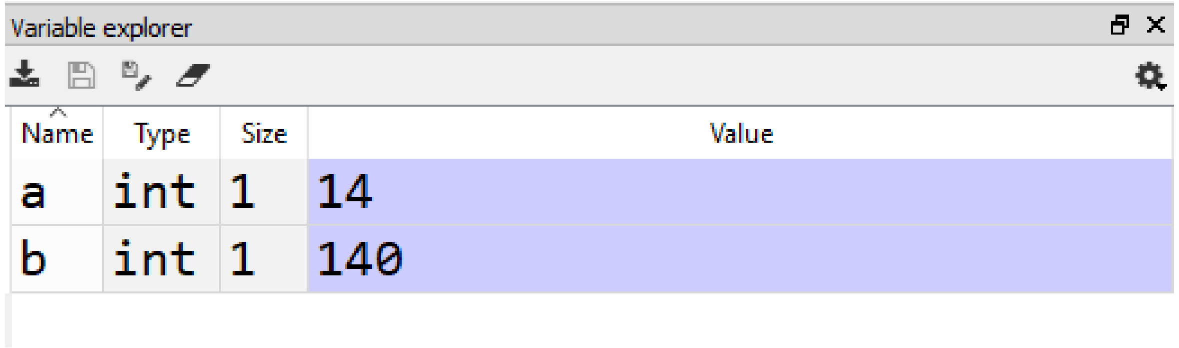


Name	Type	Size	Value
a	int	1	14
b	int	1	140

```
In [9]:
```

Removing/deleting variable(s)

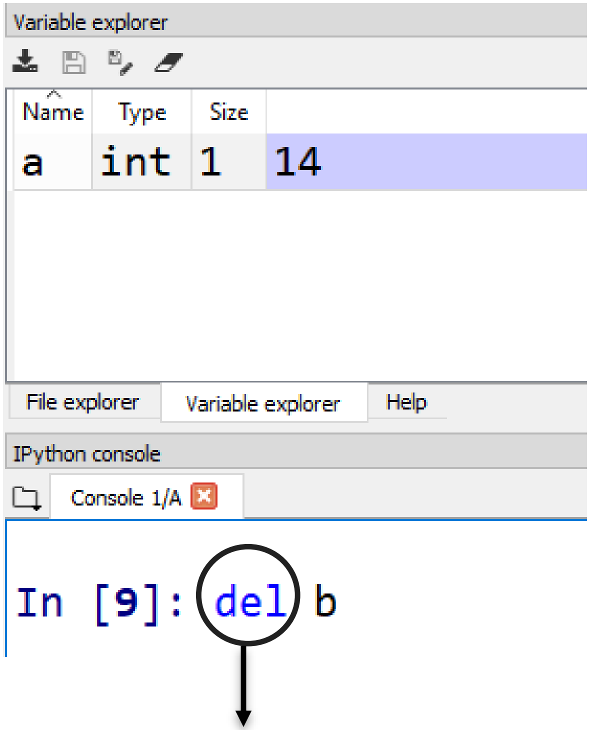
Environment



Name	Type	Size	Value
a	int	1	14
b	int	1	140

Removing/deleting variable(s)

Removing single variable

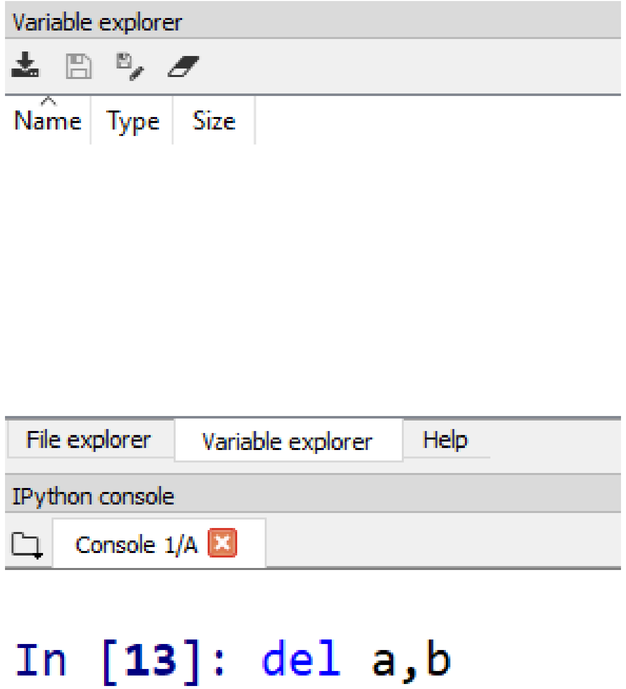


Name	Type	Size	
a	int	1	14

```
In [9]: del b
```

Using del followed by variable name

Removing multiple variables



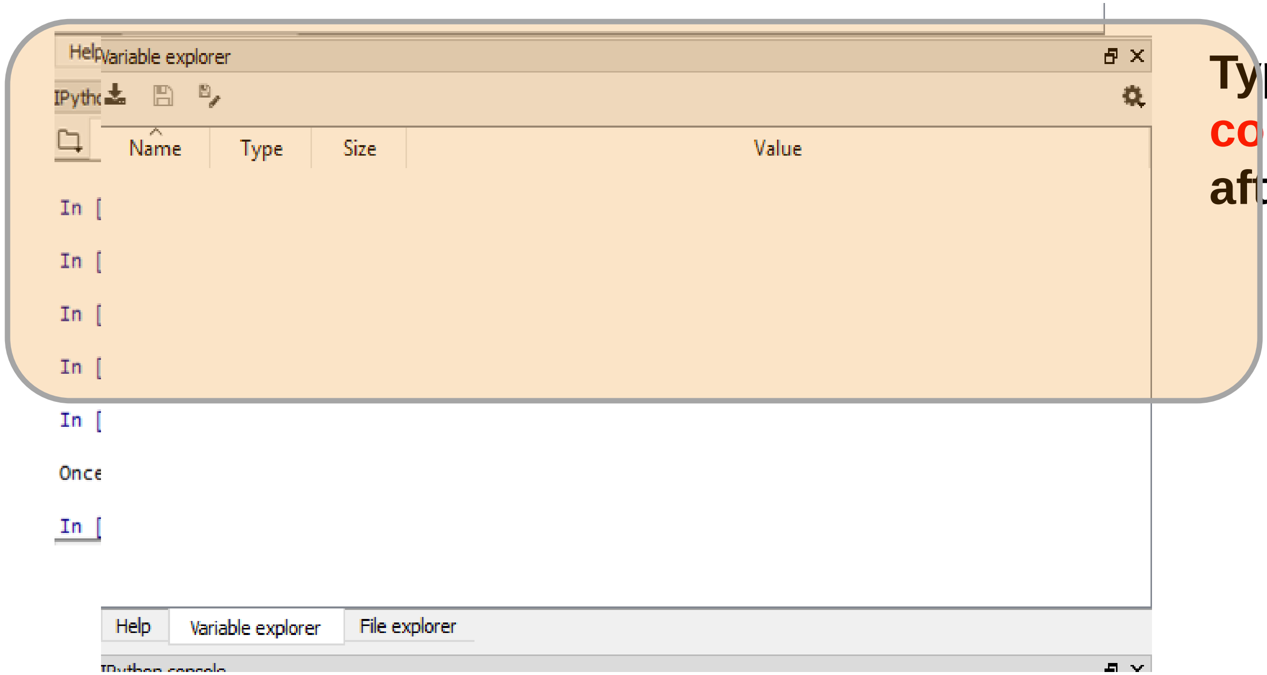
Name	Type	Size

```
In [13]: del a,b
```

Clearing the entire environment at once

- There are two ways to clear the environment

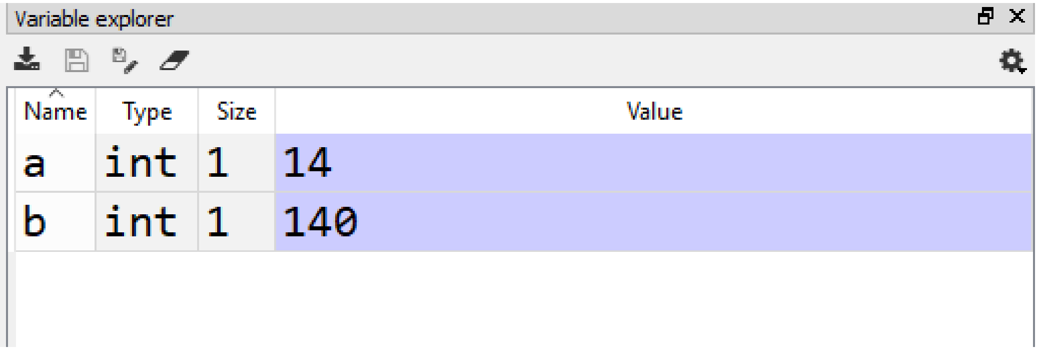
Method 1



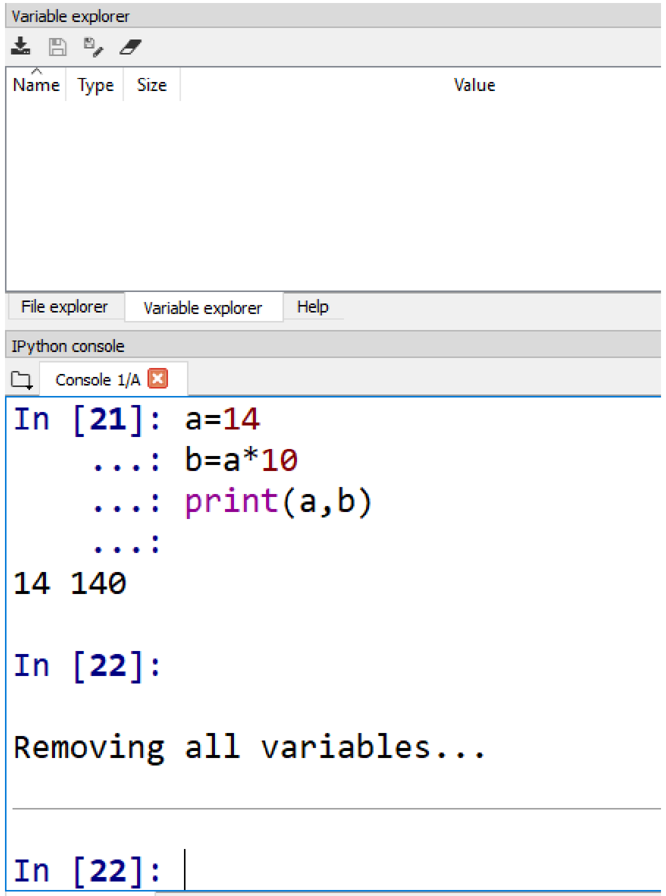
Type %reset in console and type 'y' after the prompt

Clearing the entire environment at once

Method 2



Name	Type	Size	Value
a	int	1	14
b	int	1	140



```
In [21]: a=14
    ...: b=a*10
    ...: print(a,b)
    ...:
14 140

In [22]:

Removing all variables...

In [22]:
```

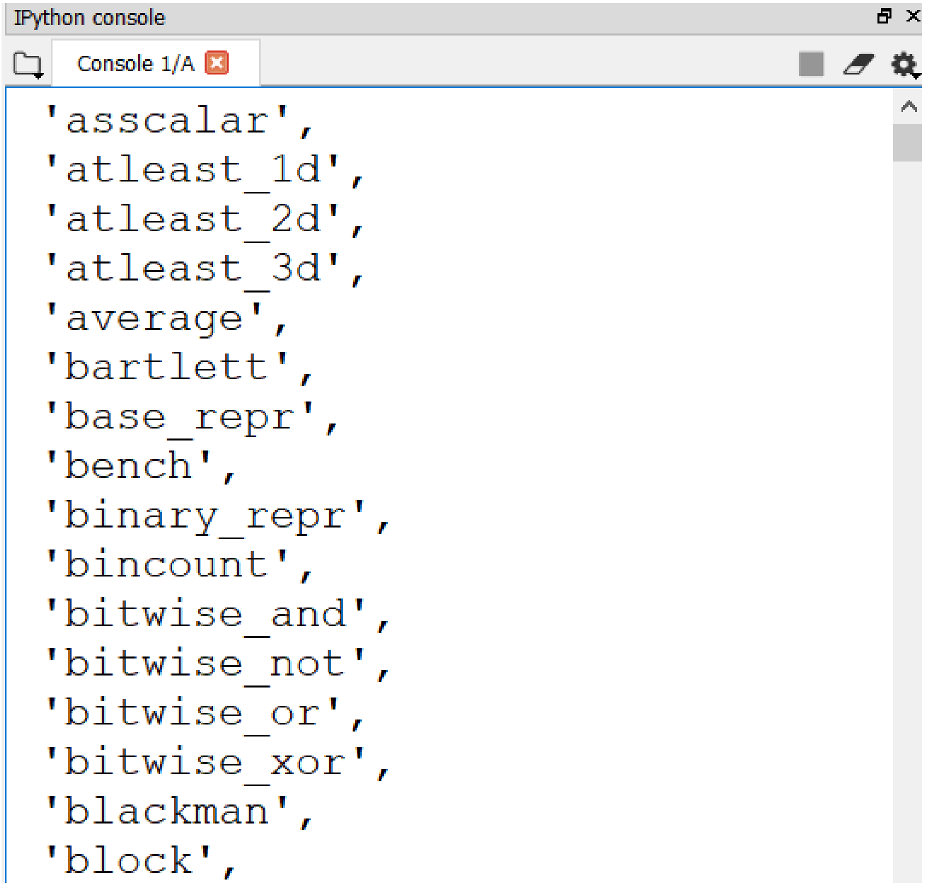
Click the symbol to remove variables in environment

Basic libraries in Python

Basic libraries in Python

- Basic libraries
  - NumPy – Numerical Python
  - Pandas – Dataframe Python
  - Matplotlib - Visualization
  - Sklearn – Machine Learning
- Modules within a library. E.g.-

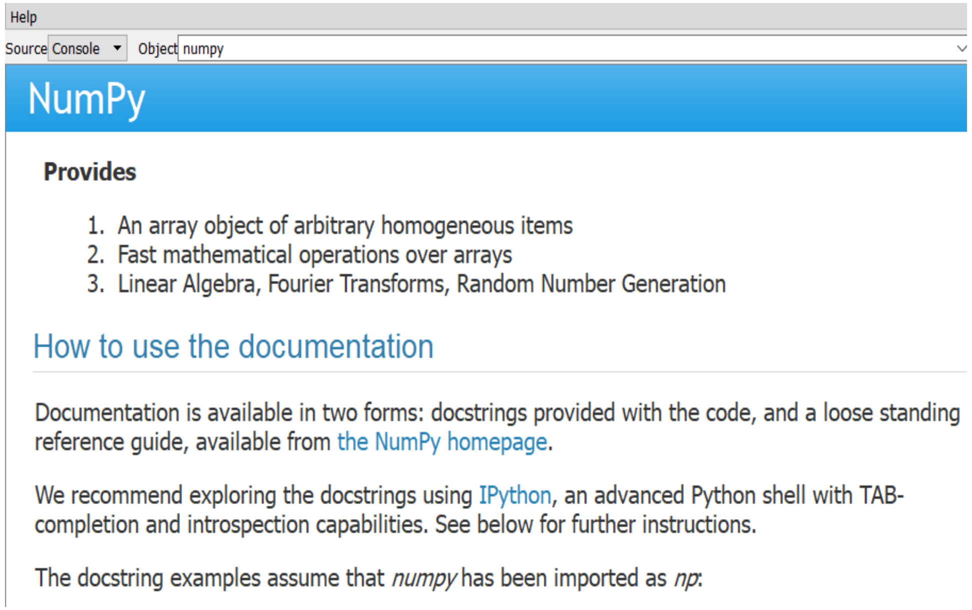
```python
import numpy
content  = dir(numpy)
print(content)
```



```
'asscalar',
'atleast_1d',
'atleast_2d',
'atleast_3d',
'average',
'bartlett',
'base_repr',
'bench',
'binary_repr',
'bincount',
'bitwise_and',
'bitwise_not',
'bitwise_or',
'bitwise_xor',
'blackman',
'block',
```

Help in Python

Type the name of the library in 'Object'



Source Console Object numpy

NumPy

Provides

1. An array object of arbitrary homogeneous items
2. Fast mathematical operations over arrays
3. Linear Algebra, Fourier Transforms, Random Number Generation

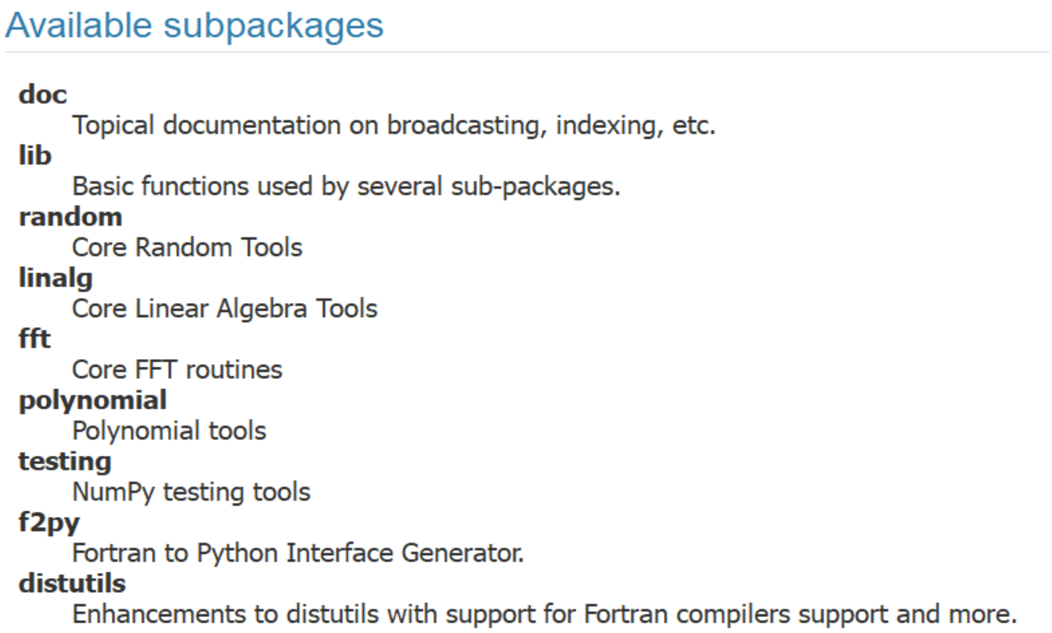
How to use the documentation

Documentation is available in two forms: docstrings provided with the code, and a loose standing reference guide, available from the NumPy homepage.

We recommend exploring the docstrings using IPython, an advanced Python shell with TAB-completion and introspection capabilities. See below for further instructions.

The docstring examples assume that *numpy* has been imported as *np*:

The following are the sub libraries



Available subpackages

doc
Topical documentation on broadcasting, indexing, etc.
lib
Basic functions used by several sub-packages.
random
Core Random Tools
linalg
Core Linear Algebra Tools
fft
Core FFT routines
polynomial
Polynomial tools
testing
NumPy testing tools
f2py
Fortran to Python Interface Generator.
distutils
Enhancements to distutils with support for Fortran compilers support and more.

Note: You can click the details of the sublibraries by typing ***libraryname.sublibraryname*** under object
Eg- **numpy.lib** in object

Summary

- Execute Python script file
- Commenting lines of code
- Clearing console and environment
- Basic libraries in Python

```
rror_object
operation == "MIRROR_X":
mirror_mod.use_x = True
mirror_mod.use_y = False
mirror_mod.use_z = False
_operation == "MIRROR_Y":
mirror_mod.use_x = False
mirror_mod.use_y = True
mirror_mod.use_z = False
_operation == "MIRROR_Z":
mirror_mod.use_x = False
mirror_mod.use_y = False
mirror_mod.use_z = True

#selection at the end -add
_ob.select= 1
ier_ob.select=1
ntext.scene.objects.active
("Selected" + str(modifier
mirror_ob.select = 0
= bpy.context.selected_ob
data.objects[one.name].se

print("please select exactly
```

THANK YOU