

# CLASS 10: INTRODUCTION TO PROGRAMMING IN PYTHON

ENGR 102 – Introduction to Engineering

2

# Programming

# What is Programming?

3

## □ Programming

- The implementation of ***algorithms*** in a particular computer ***programming language*** for execution on a ***computer***
- 

## □ Algorithm

- A step-by-step procedure for performing a computation, solving a problem, performing some action, etc. – a recipe
- ***Algorithm design*** is the meat of programming – the rest is just translation into a particular language

## □ Programming language

- We'll use Python. Others include C, C++, Java, MATLAB, etc.

## □ Computer

- May be a PC, or may be a microcontroller, FPGA, etc.

# Why Programming?

4

- I don't want to be a *software* engineer. Why do I need to learn to program?
  - **All** engineers will need to write computer code throughout their careers
    - Design and simulation
    - Numerical solution of mathematical problems
    - Data analysis – from measurements or simulation
    - Firmware for the control of mechatronic systems
  - More importantly: ***development of algorithmic thinking ability***
    - Learn to think like an engineer – single most important takeaway from your engineering education

# Python

5

- This a course in ***programming fundamentals*** and ***algorithmic thinking***
- The language we'll use to develop these concepts is ***Python*** (in the ***Spyder*** development environment)
  - ▣ Could just as well use another language, e.g., C, C++, Java, MATLAB, Fortran, ...
  - ▣ The important concepts are not language-specific
- ***Two goals*** of this course:
  - ▣ Learn to develop basic algorithms and to write structured computer code
  - ▣ Learn to use Python

# Introduction to Python & Spyder

The remainder of this section of notes is intended to provide a brief introduction to Python and the Spyder development environment.

# Python – What is It?

7

## □ ***A general-purpose programming language***



- Used for writing programs to describe procedures to be executed by computers
- *High-level*
  - Readable code – includes natural-language constructs
  - Makes use of extensive libraries of functions
  - Highly abstracted from the machine-level instructions that will ultimately be passed to the computer
- *Interpreted*
  - Translation to machine instructions happens at runtime
  - Not *compiled* – translations happens once, creating a separate executable file
- *Object oriented* – more on this later

# Python – How Do We Use it?

8

- Different ways to write and execute Python code
  - ***Text editor***
    - Simple editor for writing code
    - May include language specific formatting/coloring, etc.
    - E.g. Vi/Vim, Sublime Text, etc.
  - ***Integrated development environment (IDE)***
    - Software interface to facilitate code development
      - Code editor
      - Debugger
      - Console
      - Variable explorer
      - File browser,
      - Plotting support, etc.
    - E.g. Spyder, Pycharm, IDLE, Visual Studio, etc.



# Spyder – What is It?



9

- We will use the **Spyder** IDE
  - ▣ Scientific **PY**thon **D**evelopment **E**nviRonment
  - ▣ Designed for scientific, engineering, and data science applications

Name /	Type	Size	Value
f	int	1	4
fs	list	3	[1, 2, 4]
t	Array of float64	(200,)	[0.]
y	Array of float64	(200,)	[0.0000]

```
import numpy as np
import matplotlib.pyplot as plt

# List of frequencies, Hz
fs = [1, 2, 4]
t = np.linspace(0,1,200)

# for Loop to create and plot individual
sinusoids
for f in fs:
    y = np.sin(2*np.pi*f*t)
    plt.plot(t,y,label='{} Hz'.format(f))

# plot configuration
plt.legend()
plt.xlabel('time [sec]')
plt.ylabel('y(t)')
plt.title('Sinusoids')
```

```
In [3]: runfile('C:/Users/webbky/Box/KWebb/Classes/ENGR102_103/
Notes/Python/Section1/plotSines.py', wdir='C:/Users/webbky/Box/
KWebb/Classes/ENGR102_103/Notes/Python/Section1')

In [4]:
```

linspace

Return evenly spaced numbers over a specified interval.

Returns *num* evenly spaced samples, calculated over the interval [*start*, *stop*].

The endpoint of the interval can optionally be excluded.

Changed in version 1.16.0: Non-scalar *start* and *stop* are now supported.

# The Spyder Interface

10

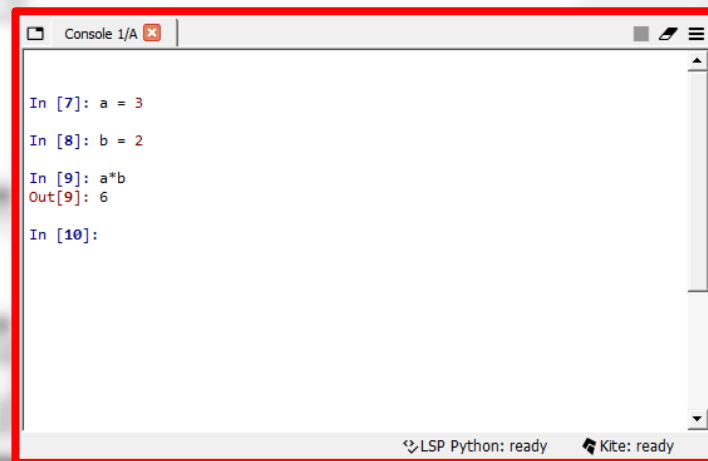
The image shows the Spyder Python IDE interface with several components highlighted by red boxes:

- Variable Explorer:** Located on the left side, it displays a table of variables. The table has columns for Name, Type, Size, and Value. The variables shown are: a (int, 1, 3), b (int, 1, 2), f (int, 1, 4), fs (list, 3, [1, 2, 4]), t (Array of float64, (200,), [0. ...]), and test\_dic... (dict, 2, {'a':1, 'b':2}).
- Plot Pane:** Located in the center, it displays a plot titled "Sinusoids". The plot shows three sine waves with different frequencies: 1 Hz (blue), 2 Hz (orange), and 4 Hz (green). The x-axis is labeled "time [sec]" and ranges from 0.0 to 1.0. The y-axis is labeled "y(t)" and ranges from -1.00 to 1.00.
- Editor:** Located on the right side, it displays the source code for the plotSines.py file. The code includes imports for numpy and matplotlib, a list of frequencies, and a loop to create and plot individual sinusoids.
- File Browser:** Located at the bottom left, it displays a list of files and folders, including plotSines.py, history.py, and Section1/plotSines.py.
- Console:** Located in the bottom center, it displays the output of the code execution, including the results of arithmetic operations: In [7]: a = 3, In [8]: b = 2, In [9]: a\*b, Out[9]: 6, and In [10]:.
- Command History:** Located at the bottom left, it displays a list of commands executed in the console, including the file path for plotSines.py.
- Help:** Located at the bottom right, it displays the documentation for the linspace function, including its description and usage.

# The Spyder Interface - Console

11

- Run Python commands interactively
- Behaves like a *calculator*
- Useful for:
  - ▣ Quick calculations
  - ▣ Simple debugging tasks



```
Console 1/A x
```

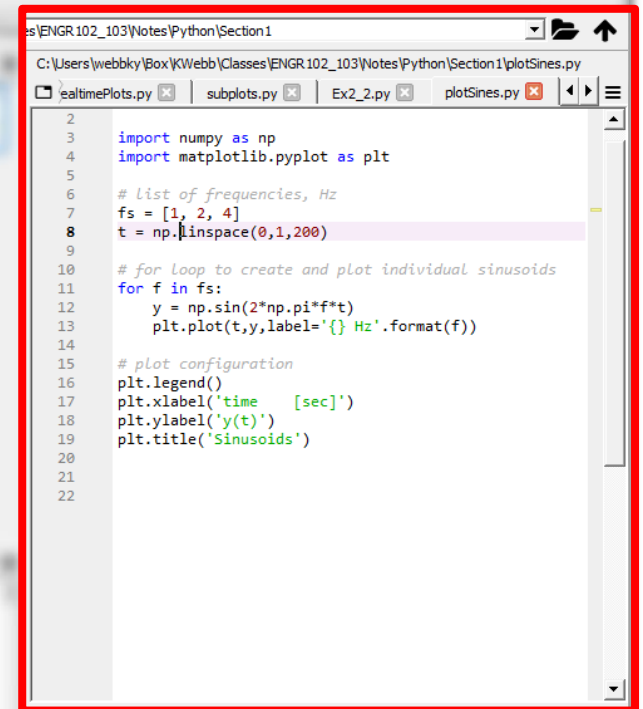
```
In [7]: a = 3
In [8]: b = 2
In [9]: a*b
Out[9]: 6
In [10]:
```

LSP Python: ready Kite: ready

# The Spyder Interface - Editor

12

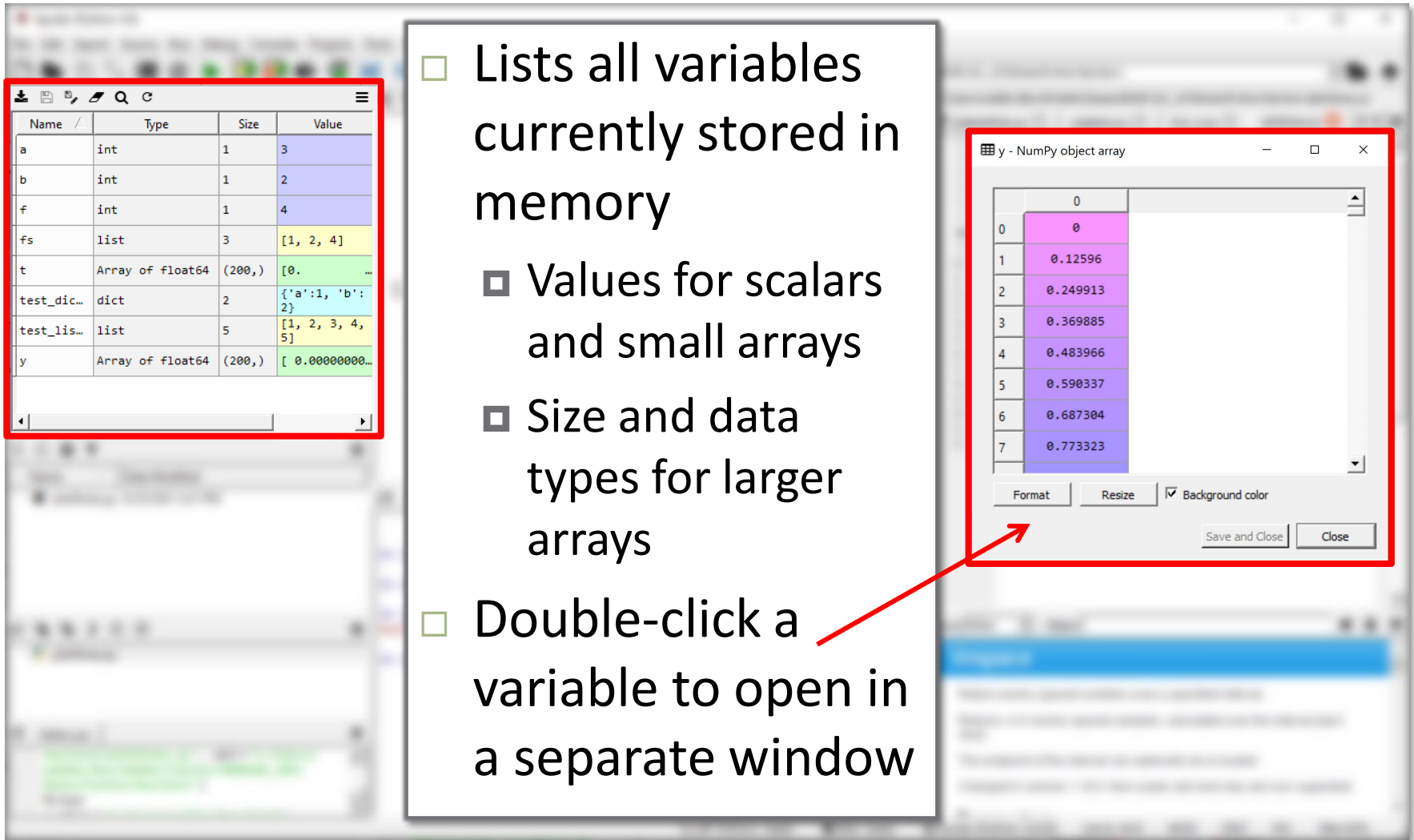
- Editor for Python scripts
- Write and execute our Python code here
- Auto formatting
  - ▣ Highlighting
  - ▣ Indenting
  - ▣ Code completion
- Built-in debugger
  - ▣ Set breakpoints
  - ▣ Step through code line-by-line or by section



```
2
3 import numpy as np
4 import matplotlib.pyplot as plt
5
6 # List of frequencies, Hz
7 fs = [1, 2, 4]
8 t = np.linspace(0,1,200)
9
10 # for loop to create and plot individual sinusoids
11 for f in fs:
12     y = np.sin(2*np.pi*f*t)
13     plt.plot(t,y,label='{} Hz'.format(f))
14
15 # plot configuration
16 plt.legend()
17 plt.xlabel('time [sec]')
18 plt.ylabel('y(t)')
19 plt.title('Sinusoids')
20
21
22
```

# The Spyder Interface – Variable Explorer

13



The image shows the Spyder Variable Explorer interface. On the left, a table lists variables with their names, types, sizes, and values. On the right, a separate window displays a detailed view of a NumPy array, showing its elements and various options like Format, Resize, and Background color.

Name	Type	Size	Value
a	int	1	3
b	int	1	2
f	int	1	4
fs	list	3	[1, 2, 4]
t	Array of float64	(200,)	[0. ...
test_dic...	dict	2	{'a':1, 'b':2}
test_lis...	list	5	[1, 2, 3, 4, 5]
y	Array of float64	(200,)	[ 0.00000000...

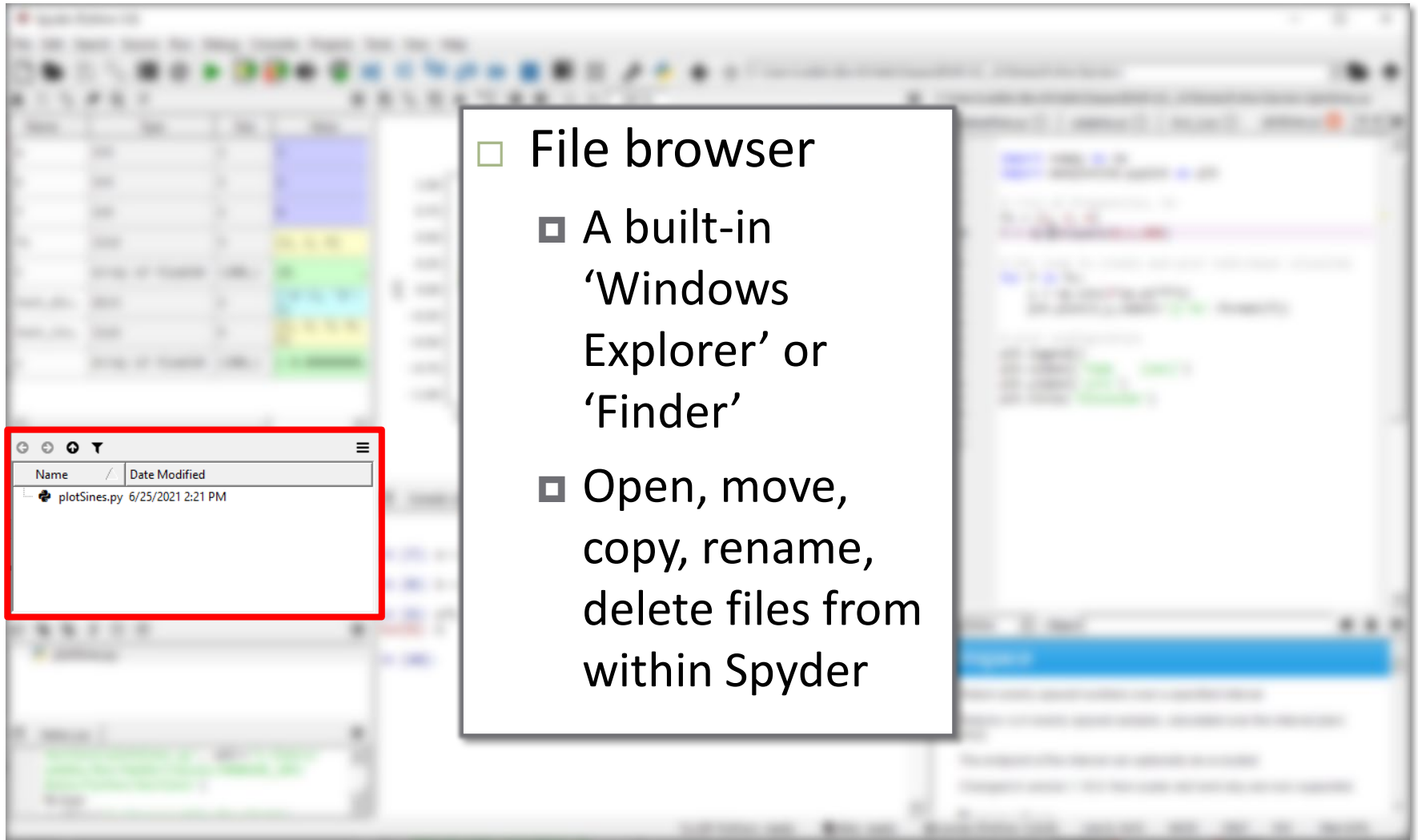
- Lists all variables currently stored in memory
  - Values for scalars and small arrays
  - Size and data types for larger arrays
- Double-click a variable to open in a separate window

The detailed view of the NumPy array 'y' shows the following values:

Index	Value
0	0
1	0.12596
2	0.249913
3	0.369885
4	0.483966
5	0.590337
6	0.687304
7	0.773323

# The Spyder Interface – File Browser

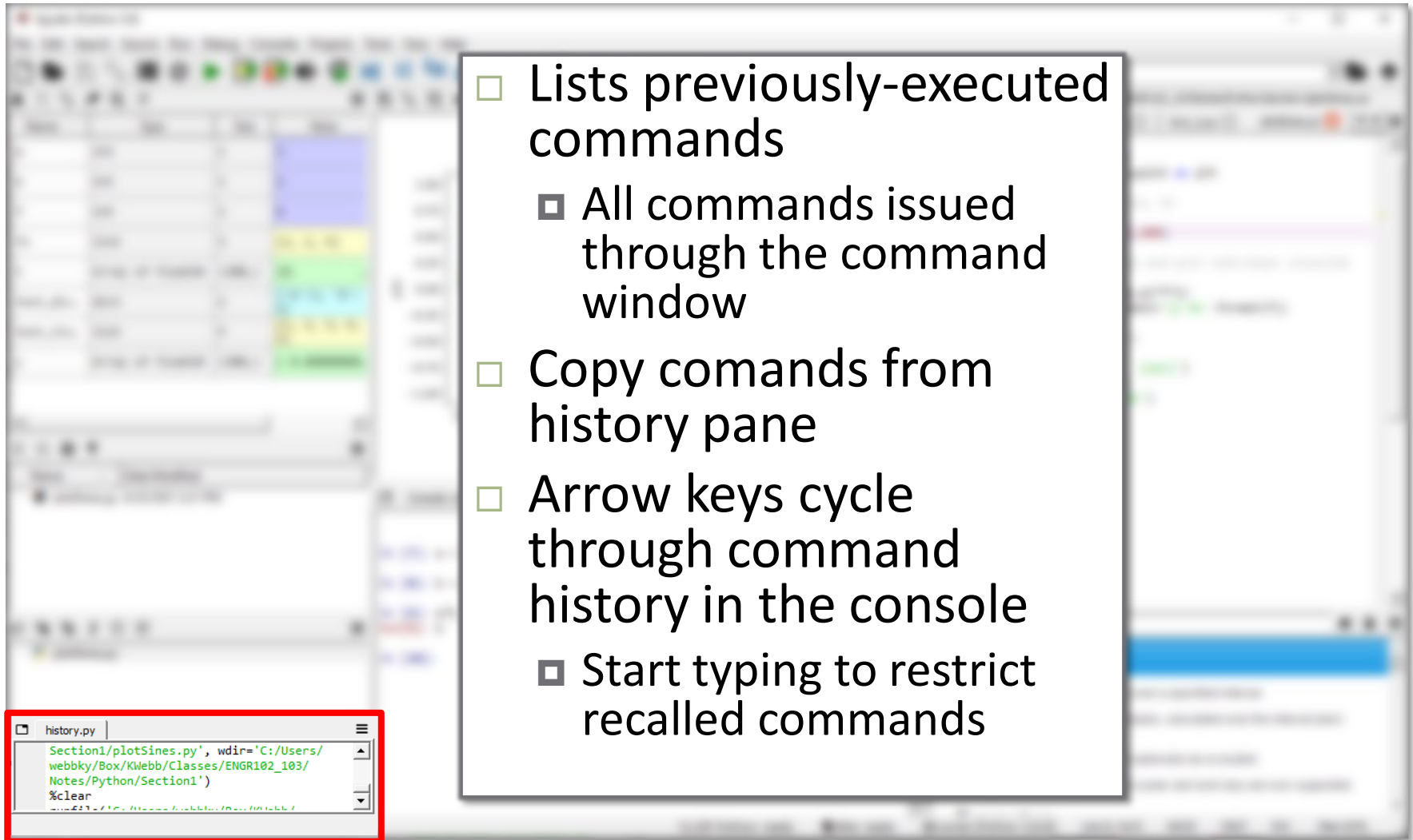
14



- File browser
  - ▣ A built-in 'Windows Explorer' or 'Finder'
  - ▣ Open, move, copy, rename, delete files from within Spyder

# The Spyder Interface – Command History

15



The image shows a screenshot of the Spyder IDE interface. A central white box contains a list of features for the Command History pane. To the left, a blurred view of the IDE's main interface is visible. At the bottom left, a code editor window titled 'history.py' is highlighted with a red border, showing Python code. The code includes a file path, a function call, and a clear command.

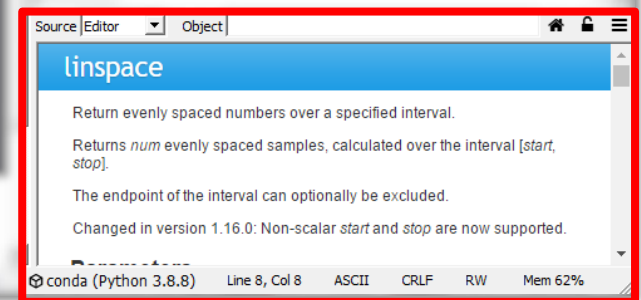
- Lists previously-executed commands
  - ▣ All commands issued through the command window
- Copy commands from history pane
- Arrow keys cycle through command history in the console
  - ▣ Start typing to restrict recalled commands

```
history.py  
Section1/plotSines.py', wdir='C:/Users/  
webbky/Box/KWebb/Classes/ENGR102_103/  
Notes/Python/Section1')  
%clear  
C:/Users/webbky/Box/KWebb/
```

# The Spyder Interface – Help Pane

16

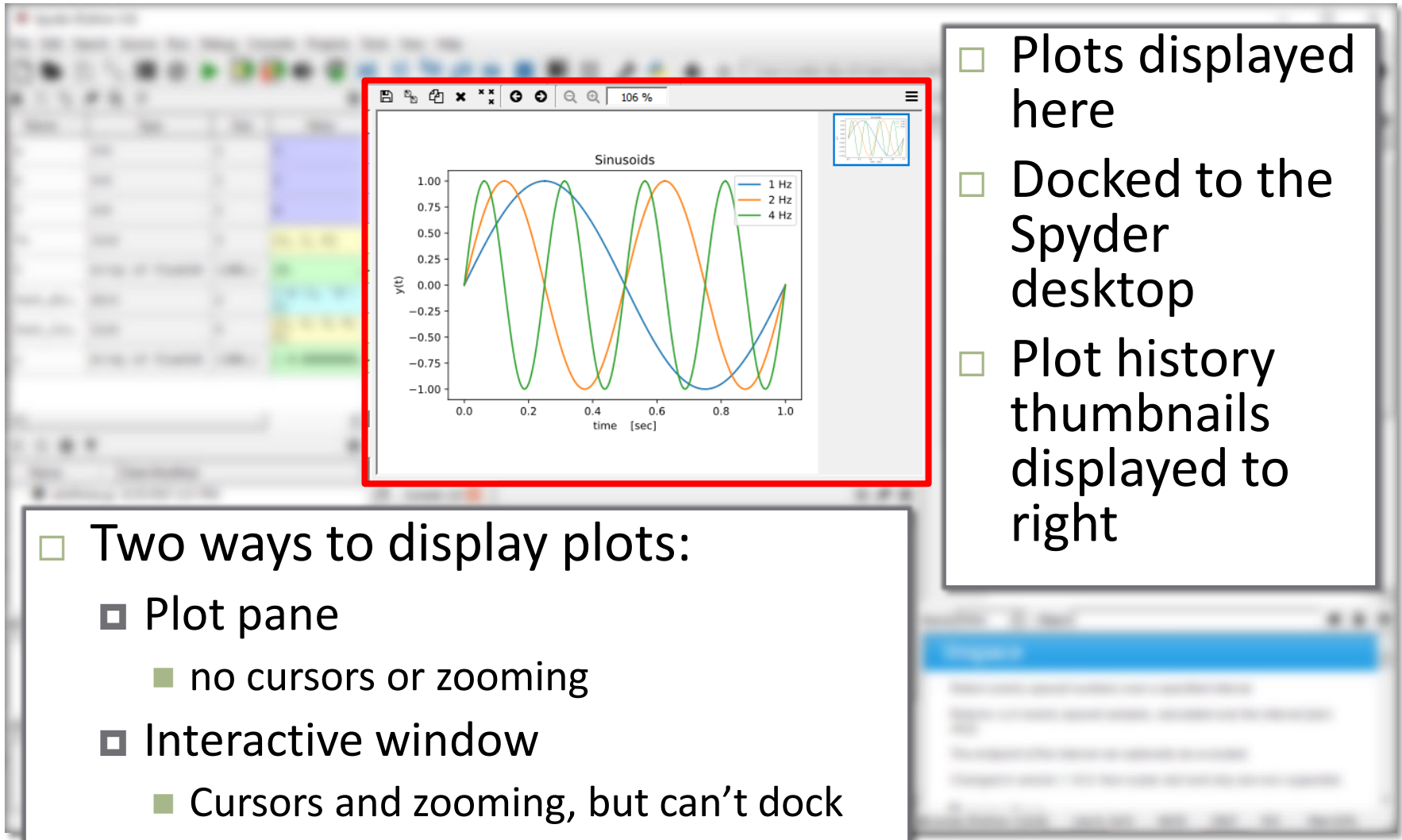
- Display help documentation for modules classes functions and methods
  - ▣ Enter object name directly in the 'Object' field
  - ▣ Place cursor on object in the editor window and type Ctrl-i





# The Spyder Interface – Plots Pane

17



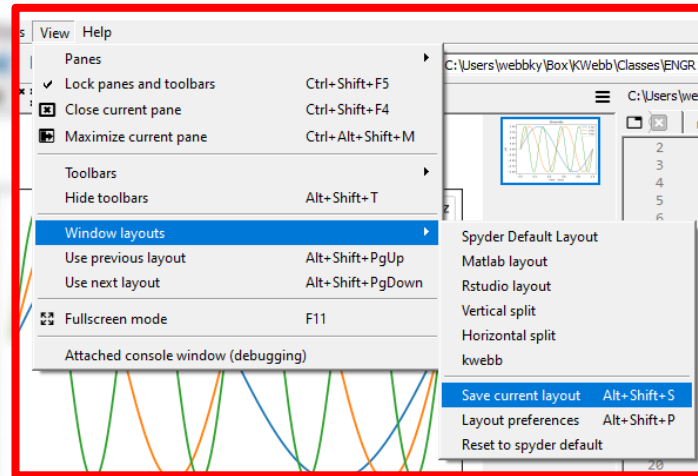
The image shows a screenshot of the Spyder software interface. A central plot pane is highlighted with a red border. The plot, titled "Sinusoids", displays three sine waves: a blue line for 1 Hz, an orange line for 2 Hz, and a green line for 4 Hz. The x-axis is labeled "time [sec]" and ranges from 0.0 to 1.0. The y-axis is labeled "y(t)" and ranges from -1.00 to 1.00. To the right of the plot, a small thumbnail shows a history of previous plots. The background shows the Spyder desktop environment with various toolbars and panels.

- Plots displayed here
- Docked to the Spyder desktop
- Plot history thumbnails displayed to right

- Two ways to display plots:
  - ▣ Plot pane
    - no cursors or zooming
  - ▣ Interactive window
    - Cursors and zooming, but can't dock

# The Spyder Interface – Saving Layouts

18



- Configure the panes in your Spyder desktop to suit your workflow
- Save one or more layouts to suit your preferences