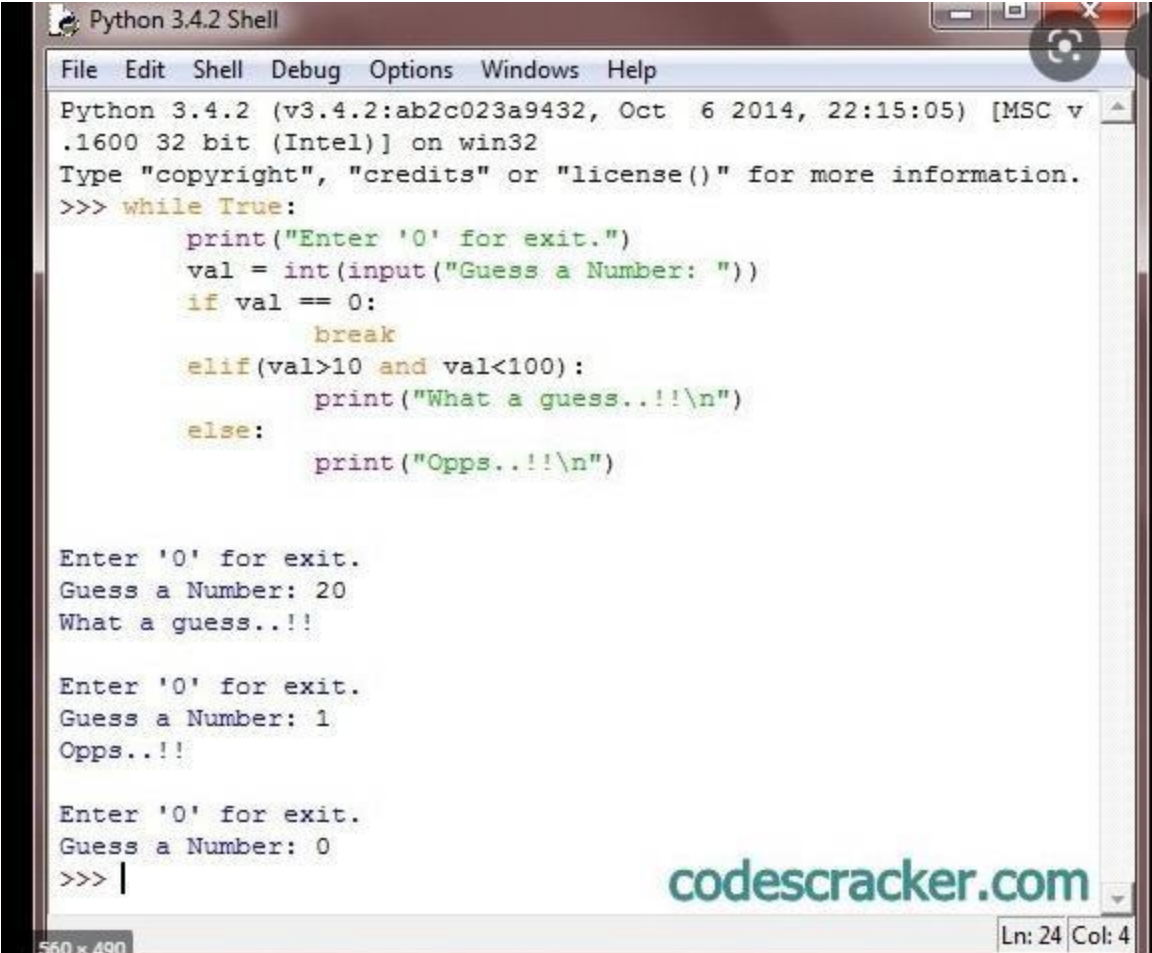


## ASSIGNMENT 03

TOPIC: Skill and Job Recommender

TEAM ID: PNT2022TMID29967

PRACTICE PYTHON IN IDLE:



The image shows a screenshot of a Python 3.4.2 Shell window. The window has a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Windows', and 'Help'. The title bar reads 'Python 3.4.2 Shell'. The main text area contains the following code and output:

```
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct 6 2014, 22:15:05) [MSC v
.1600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> while True:
    print("Enter '0' for exit.")
    val = int(input("Guess a Number: "))
    if val == 0:
        break
    elif (val>10 and val<100):
        print("What a guess...!!\n")
    else:
        print("Opps...!!\n")

Enter '0' for exit.
Guess a Number: 20
What a guess...!!

Enter '0' for exit.
Guess a Number: 1
Opps...!!

Enter '0' for exit.
Guess a Number: 0
>>> |
```

The output shows three iterations of the program. In the first, the user guesses 20, which is between 10 and 100, so it prints "What a guess...!!". In the second, the user guesses 1, which is not between 10 and 100, so it prints "Opps...!!". In the third, the user guesses 0, so the loop breaks. The status bar at the bottom left shows "560 x 490" and the bottom right shows "Ln: 24 Col: 4". A watermark "codescracker.com" is visible in the bottom right corner of the text area.

```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help

import time
def countdown(t):
    while t > 0:
        print(t)
        t -= 1
        time.sleep(1)
    print("BLAST OFF!")

print("How many seconds to count down? Enter an integer")
seconds = input()
while not seconds.isdigit():
    print("That wasn't an integer! Enter an integer:")
    seconds = input()
seconds = int(seconds)
countdown(seconds)
```

100 x 900

wiki How to Make a Countdown P

Python Shell

File Edit Shell Debug Options Windows Help

Python 3.2a3 (r32a3:85355, Oct 10 2010, 15:59:23) [MSC v.1500 64 bit (AMD64)] on win32

Type "copyright", "credits" or "license()" for more information.

```
>>> 2 ** 100
```

```
1267650600228229401496703205376
```

```
>>> 'blah! ' * 10
```

```
'blah! blah! blah! blah! blah! blah! blah! blah! blah! blah! '
```

```
>>> x = 'Python '
```

```
>>> x + 'IDLE'
```

```
'Python IDLE'
```

```
>>> import os
```

```
>>> os.getcwd()
```

```
'C:\\Python32'
```

```
>>> import sys
```

```
>>> sys.platform
```

```
'win32'
```

```
>>> sys.path
```

```
['C:\\Python32\\Lib\\idlelib', 'C:\\Windows\\system32\\python32.zip', 'C:\\Python32\\DLLs', 'C:\\Python32\\lib', 'C:\\Python32', 'C:\\Python32\\lib\\site-packages']
```

```
>>> help(bin)
```

```
Help on built-in function bin in module builtins:
```

```
bin(...)
```

```
    bin(number) -> string
```

```
    Return the binary representation of an integer or long integer.
```

```
>>>
```

Ln: 26 Col: 4

Python 3.6.4 Shell

```
Python 3.6.4 (v3.6.4:d48eeced5, Dec 18 2017, 21:07:28)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> WARNING: The version of Tcl/Tk (8.5.9) in use may be unstable.
Visit http://www.python.org/download/mac/tcltk/ for current information.
import pynput
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    import pynput
ModuleNotFoundError: No module named 'pynput'
>>> import sys
>>> sys.version
'3.6.4 (v3.6.4:d48eeced5, Dec 18 2017, 21:07:28) \n[GCC 4.2.1 (Apple Inc.
  5666) (dot 3)]'
>>> sys.path
['', '/Users/jwgsolitude/Documents', '/Library/Frameworks/Python.framework
  Versions/3.6/lib/python36.zip', '/Library/Frameworks/Python.framework/Versions
  3.6/lib/python36.zip', '/Library/Frameworks/Python.framework/Versions/3.6/lib/pyt
  h-3.6.zip', '/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6
  lib-dynload', '/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6
  site-packages']
>>> |
```

```
Python Shell
File Edit Shell Debug Options Windows Help
Python 3.2a3 (r32a3:85355, Oct 10 2010, 15:59:23) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 2 ** 100
1267650600228229401496703205376
>>> 'blah! ' * 10
'blah! blah! blah! blah! blah! blah! blah! blah! blah! '
>>> x = 'Python '
>>> x + 'IDLE'
'Python IDLE'
>>> import os
>>> os.getcwd()
'C:\\Python32'
>>> import sys
>>> sys.platform
'win32'
>>> sys.path
['C:\\Python32\\Lib\\idlelib', 'C:\\Windows\\system32\\python32.zip', 'C:\\Python32\\DLLs', 'C:\\Python32\\lib', 'C:\\Python32', 'C:\\Python32\\lib\\site-packages']
>>> help(bin)
Help on built-in function bin in module builtins:

bin(...)
    bin(number) -> string

    Return the binary representation of an integer or long integer.

>>>
```

Ln: 26 Col: 4

```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
Python 3.8.2 (default, Jul 16 2020, 14:00:26)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>> help
Type help() for interactive help, or help(object) for help about object.
>>> help()

Welcome to Python 3.8's help utility!

If this is your first time using Python, you should definitely check out
the tutorial on the Internet at https://docs.python.org/3.8/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules. To quit this help utility and
return to the interpreter, just type "quit".

To get a list of available modules, keywords, symbols, or topics, type
"modules", "keywords", "symbols", or "topics". Each module also comes
with a one-line summary of what it does; to list the modules whose name
or summary contain a given string such as "spam", type "modules spam".

help> |
```

```
Python 3.8.4rc1 Shell
File Edit Shell Debug Options Window Help
Python 3.8.4rc1 (tags/v3.8.4rc1:6c38841, Jun 30 2020, 15:17:30)
[MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("Hello World")
Hello World
>>> 10+30
40
>>> 50-40
10
>>> 60*3
180
>>> |
```



```
*myFile.py - /Users/mpneary/Documents/myFile.py
user_input = input("What is your name? ")
```

```
if user_input == "Python":
    print("Welcome to IDLE!")
else:
    print("Welcome to Python!")
```

```
print("This statement is an unsaved change!")
```

```
Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 03:13:28)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more informati
>>> x = 5
>>> print(x)
5
>>>
===== RESTART: Shell =====
>>> print(x)
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    print(x)
NameError: name 'x' is not defined
>>>
```

```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
Python 3.8.2 (default, Jul 16 2020, 14:00:26)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>> help
Type help() for interactive help, or help(object) for help about object.
>>> help()

Welcome to Python 3.8's help utility!

If this is your first time using Python, you should definitely check out
the tutorial on the Internet at https://docs.python.org/3.8/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules. To quit this help utility and
return to the interpreter, just type "quit".

To get a list of available modules, keywords, symbols, or topics, type
"modules", "keywords", "symbols", or "topics". Each module also comes
with a one-line summary of what it does; to list the modules whose name
or summary contain a given string such as "spam", type "modules spam".

help> |
```



## 1.practice python usi

The screenshot shows the Spyder IDE interface. The main editor displays a Python script with the following code:

```
1 # -*- coding: utf-8 -*-
2 """
3 Spyder Editor
4 This is a temporary script file.
5 """
6
7
8 from pandas import DataFrame
9
10 People_List = ['Jon', 'Mark', 'Maria', 'Jill', 'Jack']
11
12 df = DataFrame (People_List,columns=['First_Name'])
13 print (df)
```

The Variable explorer on the right shows the following variables:

Name	Type	Size	Value
df	DataFrame	(5, 1)	Column names: First_Name
People_List	list	5	['Jon', 'Mark', 'Maria', 'Jill', 'Jack']

An error dialog box is displayed in the foreground, stating:

**Error**

Spyder was unable to retrieve the value of this variable from the console.

The error message was:  
The kernel is dead.

Note: Please don't report this problem on Github, there's nothing to do about it.

OK

The IPython console at the bottom shows the following output:

```
IPython 7.19.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/User/.spyder-py3/temp.py', wdir='C:/Users/User/.spyder-py3')

First_Name
0      Jon
1      Mark
2      Maria
3      Jill
4      Jack

In [2]:
```

ng spyder

```
1 """
2 Plot a terrain model and a polar plot side by side.
3 """
4
5 # Third party imports
6 import numpy as np
7 import matplotlib.pyplot as plt
8 import matplotlib.cm
9 import matplotlib.colors
10 import mpl_toolkits.mplot3d # pylint: disable=unused-import
11
12 plt.style.use("dark_background")
13
14 def generate_polar_plot():
15     """Generate an example polar slice plot."""
16     # Compute pie slices
17     n_slices = 20
18     theta = np.linspace(0.0, 2 * np.pi, n_slices, endpoint=False)
19     radii = 10 * np.random.rand(n_slices)
20     width = np.pi / 4 * np.random.rand(n_slices)
21
22     fig, ax = plt.subplots(figsize=(8, 3))
23     fig.patch.set_facecolor('#395979')
24     ax1 = plt.subplot(1, 2, 2, projection='polar')
25     ax1.set_facecolor('#395979')
26     bars = ax1.bar(theta, radii, width=width, bottom=0.0)
27
28     # Use custom colors and opacity
29     for radius, plot_bar in zip(radii, bars):
30         plot_bar.set_facecolor(plt.cm.viridis(radius / 10.))
31         plot_bar.set_alpha(0.5)
32
33 def generate_dem_plot():
34     """Generate a 3D representation of a terrain DEM."""
35
36
91
92
93 _colors_full_map = {}
94 # Set by reverse priority order.
95 _colors_full_map.update(XXCD_COLORS)
96 _colors_full_map.update({k.replace('grey', 'gray'): v
97                          for k, v in XXCD_COLORS.items()
98                          if 'grey' in k})
99 _colors_full_map.update(CSS4_COLORS)
100 _colors_full_map.update(TABLEAU_COLORS)
101 _colors_full_map.update({k.replace('gray', 'grey'): v
102                          for k, v in TABLEAU_COLORS.items()
103                          if 'gray' in k})
104 _colors_full_map.update(BASE_COLORS)
105 _colors_full_map = _ColorMapping(_colors_full_map)
106
107
108 def get_named_colors_mapping():
109     """Return the global mapping of names to named colors."""
110     return _colors_full_map
111
112
113 def _sanitize_extrema(ex):
114     if ex is None:
115         return ex
116     try:
117         ret = ex.item()
118     except AttributeError:
119         ret = float(ex)
120     return ret
121
122
123 def _is_nth_color(c):
124     """Return whether *c* can be interpreted as an item in the c
125     return isinstance(c, str) and re.match(r"^[A-Z0-9]{1,2}$", c)
126
127
128 def is_color_like(c):
129     """Return whether *c* can be interpreted as an RGB(A) color.
130     # Special-case nth color syntax because it cannot be parsed
131     if _is_nth_color(c):
132         return True
133     try:
134         to_rgba(c)
135     except ValueError:
136         return False
```

signal1.py

nextpow2

val\_abs

exponent

exponent

exponent

data

fig

ax1

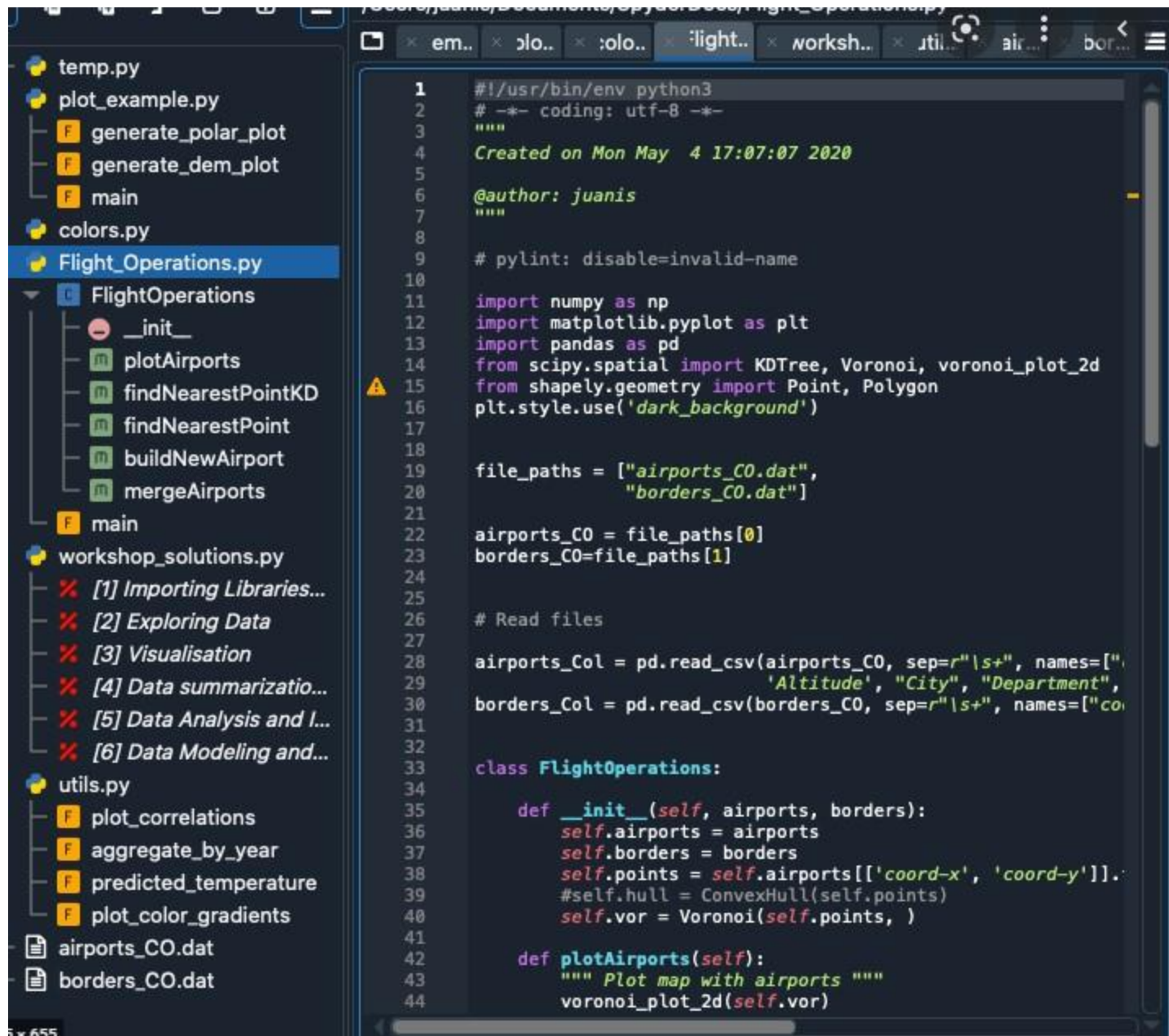
data

fig

ax1

```
1  # -*- coding: utf-8 -*-
2  """Test outline."""
3  # Third party imports
4  from matplotlib.pyplot import figure
5  from numpy import linspace
6  from numpy.core.umath import log2
7
8
9  def nextpow2(val):
10     r"""."""
11     val_abs = abs(val)
12
13     exponent = 1
14
15     while exponent < val_abs:
16         exponent *= 2
17
18     exponent = log2(exponent)
19
20     return exponent
21
22
23 if __name__ == '__main__':
24     data = linspace(1, 2, 3)
25
26     fig = figure()
27     ax1 = fig.add_subplot(1, 1, 1)
28     ax1.plot(data)
29
30     data = linspace(1, 3, 3)
31     fig = figure()
32     ax1 = fig.add_subplot(1, 1, 1)
33     ax1.plot(data)
```

719 x 662





```
plugin.py
#
#-----
# get_name
# get_description
# get_icon
# register
# unregister
#
#-----
# current_widget
# add_shellwidget
# remove_shellwidget
# set_shellwidget
chart_plot_example.py
% Plot final terrain model
# generate_polar_plot
# generate_dem_plot
# main
plugin.py
IPythonConsole
# _init_
# -- SpyderPluginMain API
# update_font
# _apply_gui_plugin_settings
# _apply_mp_plugin_setting
# _apply_advanced_plugin_s
# _apply_pdb_plugin_setting
# apply_plugin_settings_to_c
# apply_plugin_settings
# toggle_view
# -- SpyderPluginWidget AP
# get_plugin_title
# get_plugin_icon
# get_focus_widget
# closing_plugin
# refresh_plugin
# get_plugin_actions
# register_plugin

/Users/juanitagomez/Local/Dev-Spyder/spyder/spyder/plugins/plots/plugin.py
X plugin.py - plots X chart_plot_example.py X plugin.py - ipythonconsole
1 # coding: utf-8
2 #
3 # Copyright © Spyder Project Contributors
4 # Licensed under the terms of the MIT License
5 # (see spyder/_init_.py for details)
6
7 """
8 Plots Plugin.
9 """
10
11 # Third party imports
12 from qtpy.QtCore import Signal
13
14 # Local imports
15 from spyder.api.plugins import Plugins, SpyderDockablePlugin
16 from spyder.api.translations import get_translation
17 from spyder.plugins.plots.widgets.main_widget import PlotsWidget
18
19 # Localization
20 _ = get_translation('spyder')
21
22
23 class Plots(SpyderDockablePlugin):
24     """
25     Plots plugin.
26     """
27
28     NAME = 'plots'
29     REQUIRES = [Plugins.IPythonConsole]
30     TABIFY = [Plugins.VariableExplorer, Plugins.Help]
31     WIDGET_CLASS = PlotsWidget
32     CONF_SECTION = NAME
33     CONF_FILE = False
34     DISABLE_ACTIONS_WHEN_HIDDEN = False
35
36     # -- SpyderDockablePlugin API
37
38     def get_name(self):
39         return _('Plots')
40
41     def get_description(self):
42         return _('Display, explore and save console generated plots.')
43
44     def get_icon(self):
45         return self.create_icon('hist')
46
47     def register(self):
48         # Plugins
49         ipyconsole = self.get_plugin(Plugins.IPythonConsole)
50
51         # Signals
52         ipyconsole.sig_shellwidget_changed.connect(self.set_shellwidget)
53         ipyconsole.sig_shellwidget_process_started.connect(
54             self.add_shellwidget)
55         ipyconsole.sig_shellwidget_process_finished.connect(
56             self.remove_shellwidget)
```

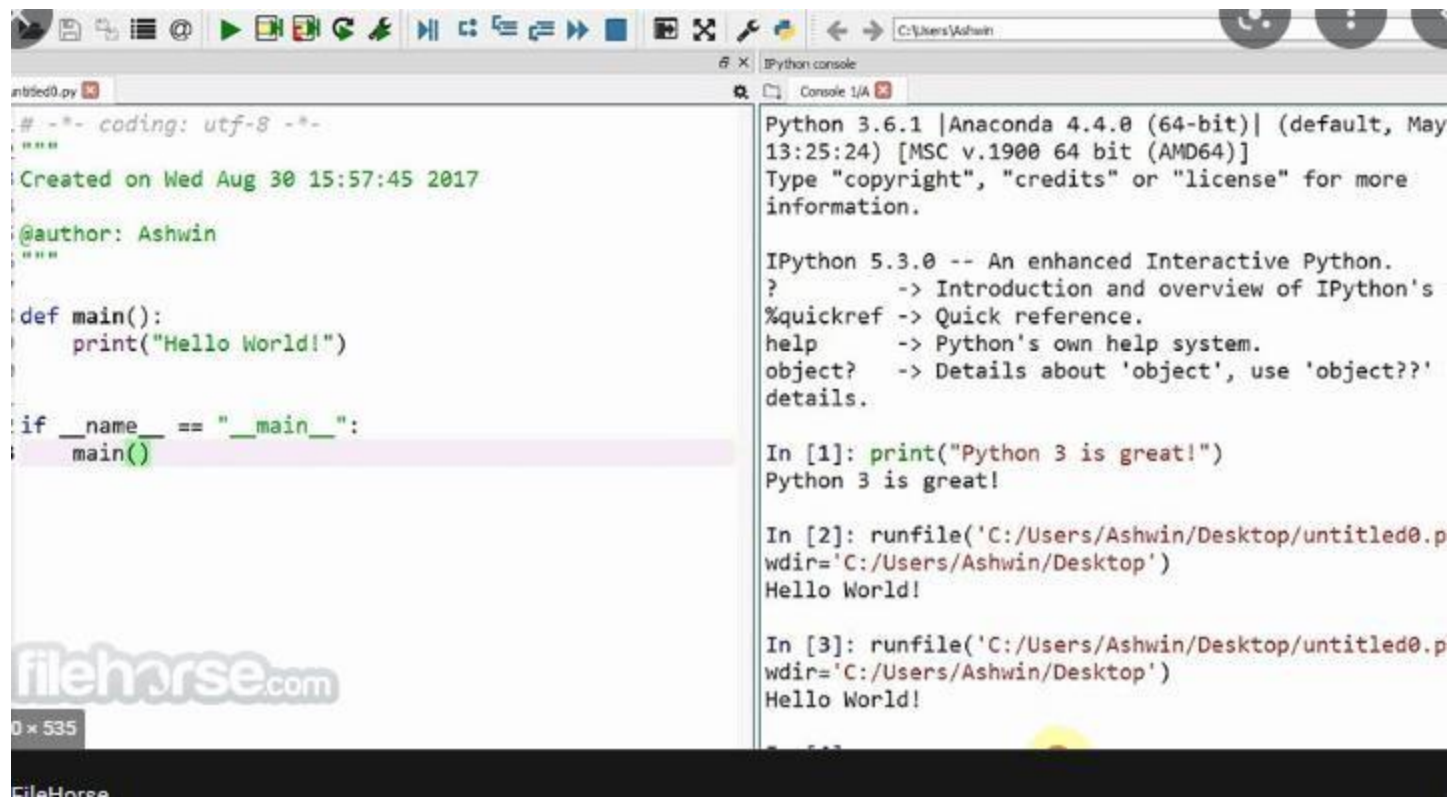
```
IPython console
IP: Console 6742/A x
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://binstar.org
? -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help -> Python's own help system.
object? -> Details about 'object', use 'object??' for extra details.
%gui? -> A brief reference about the graphical user interface.

In [1]: print("Hello, World")
Hello, World

In [2]: 7035 * 0.15
Out[2]: 1055.25

In [3]: |

Console History log IPython console
```





## X Utility Functions

```
defmacro :-message-without-echo (str &rest args)
  "Wrap 'message' passing in STR and ARGS, without showing in the echo area."
  `
  (let ((inhibit-message t))
    (message ,str ,@args)))

-----
; Undo List Make Linear
;
; Note that this only works for 'buffer-undo-list', not 'pending-undo-list'.

(defun :-linear-undo-list (undo-list equiv-table)
  "Collapse UNDO-LIST using EQUIV-TABLE making it linear.

  This gives the same behavior as running 'undo-only',
  ignoring all branches that aren't included in the current undo state."
  (let ((linear-list nil))
    (while
      ;; Collapse all redo branches (giving the same results as if running 'undo-only')
      (let ((undo-list-next nil))
        (while (setq undo-list-next (gethash undo-list equiv-table))
          (setq undo-list undo-list-next)
          (and undo-list (not (eq t undo-list)))))

      ;; Pop all steps until the next boundary 'nil'.
      (let ((undo-elt t))
        (while undo-elt
          (setq undo-elt (pop undo-list))
          (push undo-elt linear-list))))

    ;; Pass through 'nil', when there is no undo information.
    ;; Also convert '(list nil)' to 'nil', since this is no undo info too.
    ;;
    ;; Note that we use 'nil' as this is what 'buffer-undo-list' is set
  )
  linear-list)
```

undo-fu-session.el <N> [-] (undo-fu-session--linear-undo-list)

```

62 @bot.message_handler(commands=['all'])
63 def send_to_all(message):
64     if message.chat.id != 64634999:
65         return
66     #Дописать for для отправки сообщения всем
67
68 @bot.message_handler(commands=['start'])
69 def handle_start(message):
70     info(f'START: {message.chat.id} начинает работу с ботом')
71     print('\nStart ', message.chat.id, datetime.now())
72     if opendb().find_usr(message) != None:
73         info(f'{message.chat.id} пытался зарегистрироваться повторно. В
74         bot.send_message(message.chat.id, config.startagain)
75         handle_help(message)
76         return
77
78     msg = bot.send_message(message.chat.id, config.start)
79     bot.register_next_step_handler(msg, registration)
80
81 def registration(message):
82     info(f'START: {message.chat.id} ввел группу {message.text}')
83     all_groups = get_grp_list()
84     if not (message.text in all_groups):
85         info(f'START: {message.chat.id} ввел не существующую группу {me
86
87         opendb().ins_id(message)
88         gr_failture = bot.send_message(message.chat.id, config.complete
89
90         bot.register_next_step_handler(gr_failture, change_gr)
91
92     else:
93         opendb().ins_all(message)
94         bot.send_message(message.chat.id, config.completet)
95         current_func = ''
96         info(f'START: {message.chat.id} прошел регистрацию. Открываю ме
97         handle_help(message)

```

temp.py

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 """
4 Created on Thu May 16 17:02:53 2019
5
6 @author: gpena-castellanos
7 """
8
9 # Third party imports
10 import numpy as np
11 import matplotlib.pyplot as plt
12
13
14 # %% This is a cell!
15 x1 = np.linspace(0.0, 5.0)
16 x2 = np.linspace(0.0, 2.0)
17
18 # %% This is another cell!
19 y1 = np.cos(2 * np.pi * x1) * np.exp(-x1)
20 y2 = np.cos(2 * np.pi * x2)
21
22 plt.subplot(2, 1, 1)
23 plt.plot(x1, y1, 'o-')
24 plt.title('A tale of 2 subplots')
25 plt.ylabel('Damped oscillation')
26
27 plt.subplot(2, 1, 2)
28 plt.plot(x2, y2, '-o')
29 plt.xlabel('time (s)')
30 plt.ylabel('Undamped')
31
32 plt.show()
33
```

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:

```
""" import numpy as np
```

Code snippets are indicated by three greater-than signs:

Variable explorer | Help | Plots | Files | Find

Console 2/A

Python 3.7.3 | packaged by conda-forge | (default, Mar 27 2019, 15:43:19)  
Type "copyright", "credits" or "license()" for more information.

IPython 7.4.0 -- An enhanced Interactive Python.

In [1]:

```
1 # -*- coding: utf-8 -*-
2 #
3 # Copyright © Spyder Project Contributors
4 # Licensed under the terms of the MIT License
5 """Workshop main flow."""
6
7
8 # %% Importing Libraries and Data
9
10 # Third-party imports
11 import matplotlib.pyplot as plt
12 import pandas as pd
13 from sklearn.model_selection import train_test_split
14 from sklearn import linear_model
15 from sklearn.metrics import explained_variance_score
16
17 # Local imports
18 from utils import (
19     plot_correlations, plot_color_gradients, aggregate_by_year,
20     predicted_temperature)
21
22 # %% Exploring Data
23
24 # Reading data
25 weather_data = pd.read_csv('data/weatherHistory.csv')
26
27 # Print size of data
28 print(len(weather_data))
29 # Print first 3 rows of DataFrame
30 print(weather_data.head(3))
31
32 # TO DO: Print the last 3 rows of the DataFrame
33 print(weather_data.tail(3))
34
35
36 # %% Visualisation
37
38 # Order rows according to date
39 weather_data['Formatted Date'] = pd.to_datetime(
40     weather_data['Formatted Date'], format='%Y-%m-%d %H:%M:%S')
41 weather_data_ordered = weather_data.sort_values(by='Formatted Date')
42 # Order Index according to date
43 weather_data_ordered = weather_data_ordered.reset_index(drop=True)
44 # Drop categorical columns
```

plot\_example.py  
colors.py  
Flight\_Operations.py  
workshop\_solutions.py  
Importing Libraries and ...  
Exploring Data  
Visualisation  
Data summarization and...  
Data Analysis and Interp...  
Data Modeling and Predi...

utils.py

- plot\_correlations
- aggregate\_by\_year
- predicted\_temperature
- plot\_color\_gradients

airports\_CO.dat  
borders\_CO.dat

## 2.webpage creation using python

```
blog
  -templates
    -blog
      -index.html
```

Add the following code in **index.html**.

```
01
02  <!DOCTYPE html>
03
04  <html lang="en">
05
06  <head>
07      <meta charset="utf-8" />
08      <link rel="stylesheet" href="css/style.css">
09      <link href="images/favicon.ico" rel="shortcut icon">
10      <title>First Blog</title>
11  </head>
12
13  <body>
14      <div class="container">
15          <h1>First Blog</h1>
16          <h2>Title</h2>
17          <h3>Posted on date by author</h3>
18          <p>Body Text</p>
19      </div>
20
21  </body>
22
23  </html>
24
```

Now, we'll create our blog URLs. Create the file **urls.py** in the blog directory and write the URL path for serving the index page.

```
01  from django.urls import path
02
03  from . import views
04
05  urlpatterns = [
```

```
06
07     path('', views.home),
08
09
10 ]
```