



Actividad 2

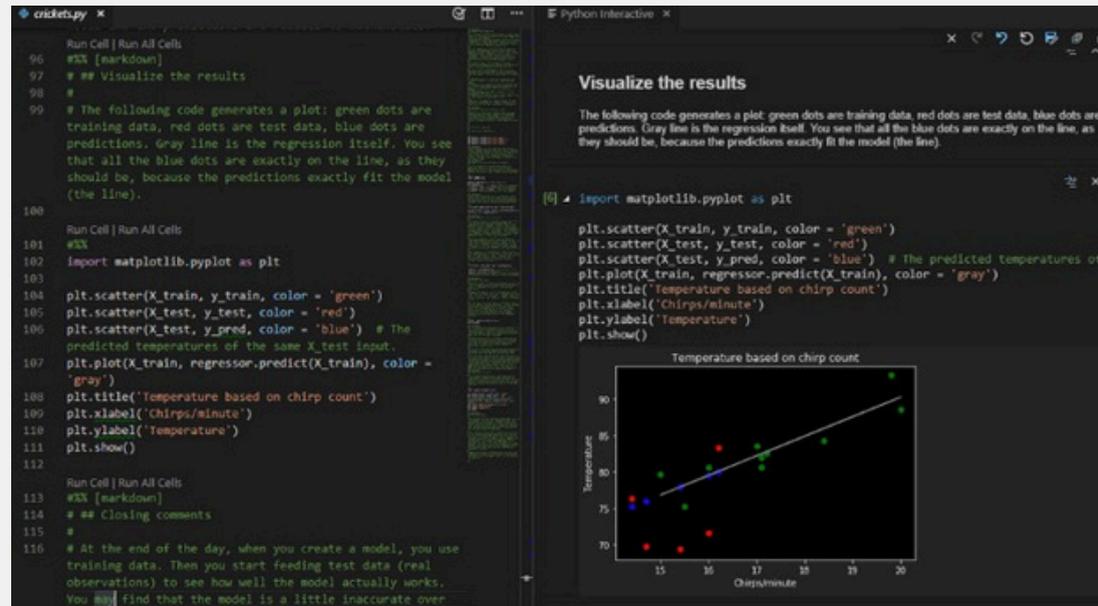
IDEs para Python

IDEs para Python

Hay varios entornos de desarrollo integrado (IDEs) populares para Python que son ampliamente utilizados por la comunidad de desarrolladores, algunos de los IDEs más destacados incluyen:

VSCoDe (Visual Studio Code):

VSCoDe es un editor de código fuente desarrollado por Microsoft que se ha vuelto muy popular en la comunidad de desarrolladores de Python. Ofrece extensiones para Python que proporcionan funcionalidades como depuración, autocompletado y soporte para entornos virtuales.



The image shows a screenshot of the Visual Studio Code editor. On the left, a Python file named 'crickets.py' is open, displaying code for data visualization. The code includes comments and uses the matplotlib library to create a scatter plot with a regression line. On the right, the 'Python Interactive' window shows the execution of the code, including a text explanation of the plot and the actual plot itself. The plot is titled 'Temperature based on chirp count' and shows a positive correlation between chirps per minute and temperature.

```
crickets.py
Run Cell | Run All Cells
96 #%% [markdown]
97 ## Visualize the results
98 #
99 # The following code generates a plot: green dots are
100 # training data, red dots are test data, blue dots are
101 # predictions. Gray line is the regression itself. You see
102 # that all the blue dots are exactly on the line, as they
103 # should be, because the predictions exactly fit the model
104 # (the line).
105
106 Run Cell | Run All Cells
107 #%%
108 import matplotlib.pyplot as plt
109
110 plt.scatter(X_train, y_train, color = 'green')
111 plt.scatter(X_test, y_test, color = 'red')
112 plt.scatter(X_test, y_pred, color = 'blue') # The predicted temperatures of the
113 # predicted temperatures of the same X_test input.
114 plt.plot(X_train, regressor.predict(X_train), color =
115 # 'gray')
116 plt.title('Temperature based on chirp count')
117 plt.xlabel('Chirps/minute')
118 plt.ylabel('Temperature')
119 plt.show()
120
121 Run Cell | Run All Cells
122 #%% [markdown]
123 ## Closing comments
124 #
125 #
126 # At the end of the day, when you create a model, you use
127 # training data. Then you start feeding test data (real
128 # observations) to see how well the model actually works.
129 # You will find that the model is a little inaccurate over
```

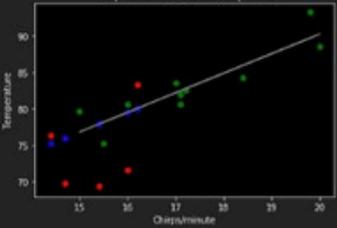
Python Interactive

Visualize the results

The following code generates a plot: green dots are training data, red dots are test data, blue dots are predictions. Gray line is the regression itself. You see that all the blue dots are exactly on the line, as they should be, because the predictions exactly fit the model (the line).

```
import matplotlib.pyplot as plt
plt.scatter(X_train, y_train, color = 'green')
plt.scatter(X_test, y_test, color = 'red')
plt.scatter(X_test, y_pred, color = 'blue') # The predicted temperatures of t
plt.plot(X_train, regressor.predict(X_train), color = 'gray')
plt.title('Temperature based on chirp count')
plt.xlabel('Chirps/minute')
plt.ylabel('Temperature')
plt.show()
```

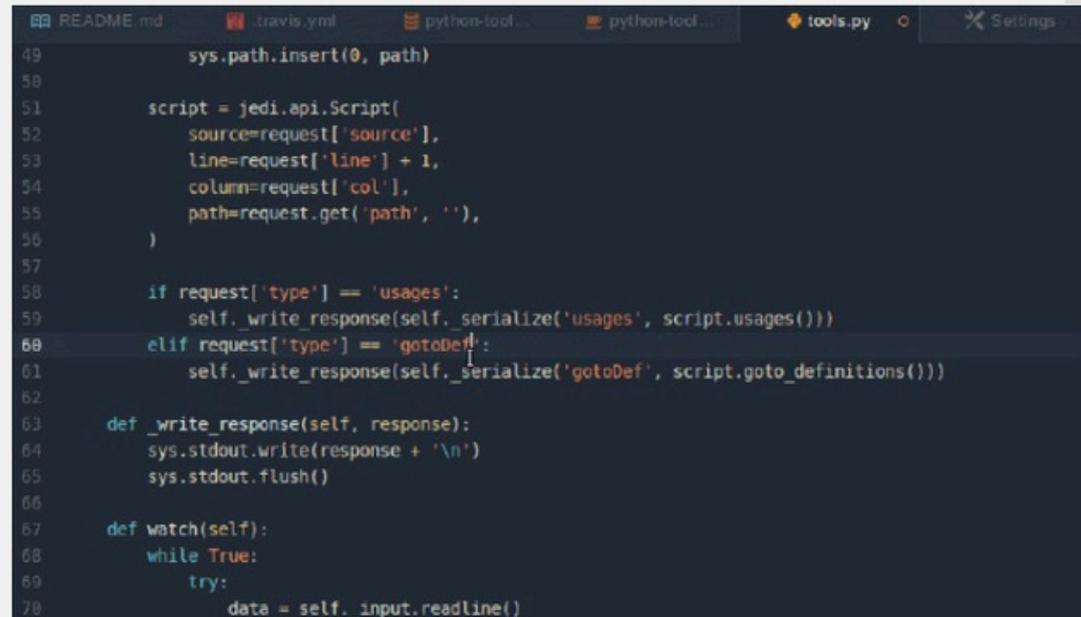
Temperature based on chirp count



Chirps/minute	Temperature
15	75
16	78
17	82
18	85
19	88
20	90

Atom

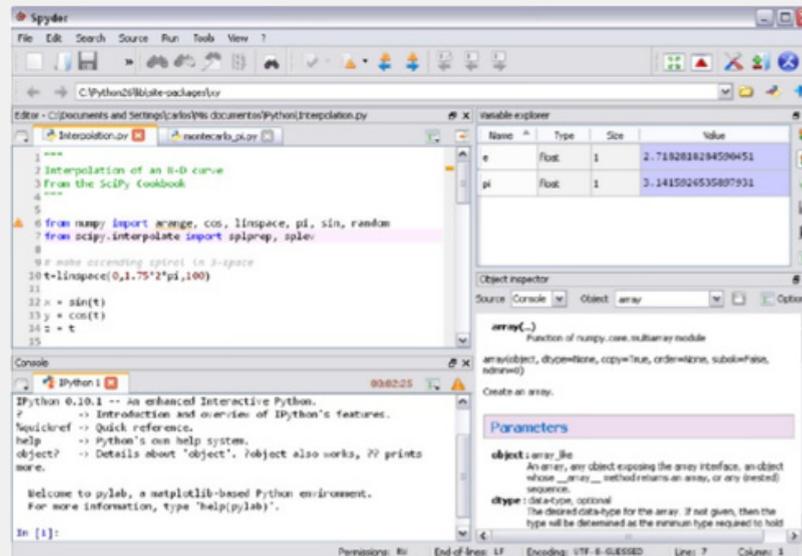
Atom es un editor de texto desarrollado por GitHub y es conocido por su exhibilidad y personalización. Dispone de extensiones para Python que permiten la integración con herramientas de desarrollo.

A screenshot of the Atom text editor interface. The window title bar shows several tabs: 'README.md', '.travis.yml', 'python-tool...', 'python-tool...', 'tools.py', and 'Settings'. The main editor area displays Python code with line numbers from 49 to 70. The code includes a function call to 'sys.path.insert', a 'jedi.api.Script' object creation, conditional logic for 'usages' and 'gotoDef', and a 'watch' method with a 'while True' loop. The code is syntax-highlighted in a dark theme.

```
49     sys.path.insert(0, path)
50
51     script = jedi.api.Script(
52         source=request['source'],
53         line=request['line'] + 1,
54         column=request['col'],
55         path=request.get('path', ''),
56     )
57
58     if request['type'] == 'usages':
59         self._write_response(self._serialize('usages', script.usages()))
60     elif request['type'] == 'gotoDef':
61         self._write_response(self._serialize('gotoDef', script.goto_definitions()))
62
63     def _write_response(self, response):
64         sys.stdout.write(response + '\n')
65         sys.stdout.flush()
66
67     def watch(self):
68         while True:
69             try:
70                 data = self._input.readline()
```

Spyder

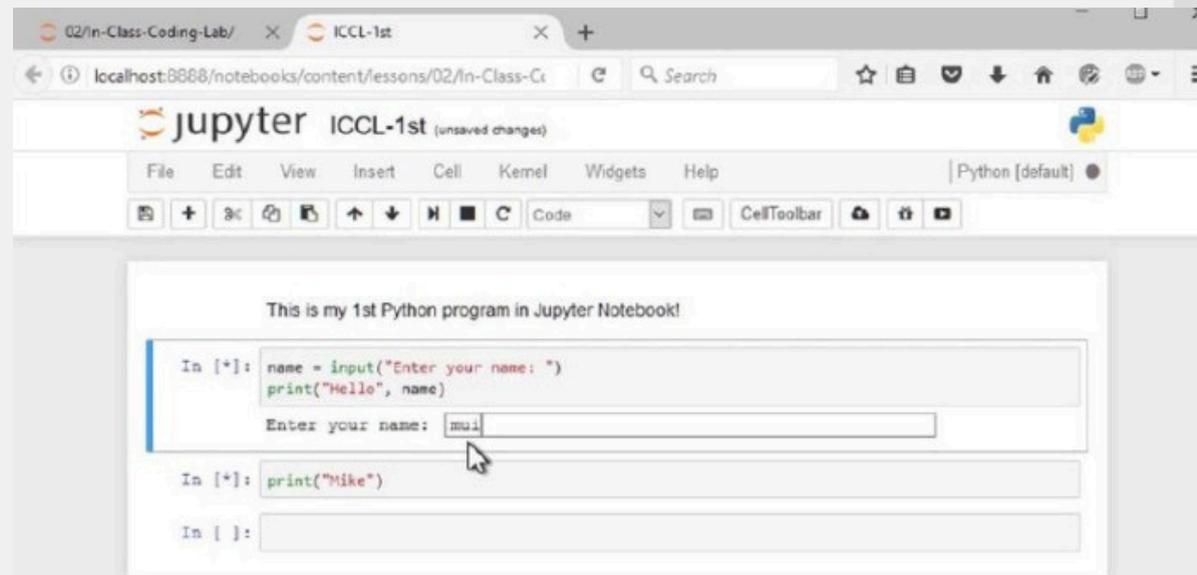
Spyder es un IDE diseñado específicamente para científicos de datos e ingenieros que trabajan con Python. Proporciona un entorno interactivo con herramientas integradas para análisis de datos, visualización y desarrollo de código.



Jupyter Notebooks

Jupyter es una plataforma interactiva que permite crear y compartir documentos que contienen códigos, visualizaciones y texto narrativo.

Es ampliamente utilizado en entornos científicos y de investigación para la exploración de datos y la presentación de resultados.



Estos son solo algunos ejemplos, y la elección del IDE depende en gran medida de las preferencias personales y los requisitos específicos del proyecto. Cada uno de estos IDEs tiene sus propias fortalezas y características que pueden adaptarse mejor o diferentes necesidades de desarrollo