From Snap! to Python (practice)

UBMS Snap! Programming Summer 2023

June 28, 2023

First Python program (online demo)

- We're going to code this Snap! program in Python
- Open colab.research.google.com in your browser
- If you're logged into your Google account: check GDrive
- All your colab notebooks are in one directory
- Always start with a TEXT block (CTRL + ALT + T)
- Continue with a CODE block (CTRL + ALT + I)
- Run code blocks with CTRL + ENTER or SHIFT + ENTER

You find the solution here as a GitHub code gist.

Getting started in Colab

1. Log into your (Lyon) Google account
2. Open `https://colab.research.google.com`
3. Select `New notebook`.
4. At the top, change `Untitled` in the title to `ColabDemo`.
5. You're on a free (virtual) Linux server (see sidebar).
6. Enter `CTRL + ALT + T` or select `+Text` for a text block.

7. This is *Markup*, a minimal layout language like HTML.

8. Create a headline: `# Import CSV data as DataFrame`

9. Enter `CTRL + ALT + I` or select `+Code` for a code block.

10. Text and code blocks can be moved up and down.

11. The code block is an *IPython* shell that sits on top of the Python shell, which is a terminal for the Python interpreter, which runs your commands right away.

12. IPython allows you to practice interactive computing. For example, you can import CSV files, print them as a table and get summary stats information:

```
import pandas as pd
df = pd.read_csv("./sample_data/california_housing_test.csv")
```

13. In Colab, run a code block with `CTRL + ENTER` or with `SHIFT + ENTER` (creates a new code block below it). This is just like in Snap! which is also an *interpreted* language (written in JavaScript and compiled to an HTML5 executable).

14. You can print the data frame as a table by typing its name:

```
df
```

15. You can see that there are 3000 rows or records and 9 columns or features describing the CA housing market. You can quickly get statistical information on this dataset:

```
df.describe()
```

16. You recognize the total count (number of entries or records or rows), the average, the minimum and maximum. Without more information about these data (units) this means little. It does however, not make sense to average over longitude and latitude.

17. This command limits the function to the columns 3 through 9 (excluding columns 1-2): the notation between the [ ] uses the 'slicing' operator to subset the rows and columns:

```
df.iloc[:,2:9].describe()
```

18. In fact there are three operators and two functions at work here:
    (a) The dot operator to extract methods/functions
    (b) The [] operator to index the data frame
    (c) The , operator to separate rows and columns
    (d) The : operator to slice off rows and columns
    (e) The `iloc[]` method to identify a data frame value based on index
    (f) The `describe()` method to compute a stats summary

19. In IPython, you can quickly make plots using the `matplotlib` library, which contains a module `pyplot`, which in turn contains plotting functions like `boxplot` or `scatter`:

```
import matplotlib.pyplot as plt
plt.boxplot(df[:100]['median_house_value'],vert=False)
plt.show()
```

20. In the last code block, we imported the plotting library `matplotlib` and the `pyplot` module in it, and told Python to plot a boxplot for one variable/column only, the median house value, and to restrict the plot to the first 100 records (i.e. locations).

21. The `plt.show()` command indicates that there's a difference, to Python, between making the plot and displaying it, or sending it to the standard output.

22. You can see in the box plot that there are five 'outliers', houses that are much more valuable than the rest, in this data, and that the middle magnitude (or median) is at around 180,000 USD (i.e. half the houses (except the outliers) are less, the other half more expensive.

23. Another useful plot is the scatterplot - that's what we did in the R programming language for the `mtcars` dataset and the variables `mpg` vs. `wt` (miles per gallon vs. car weight). Now, we plot the median income as a function of the house value - and we expect them to be *positively correlated*, that is to increase together:

```
plt.scatter(df.median_house_value,df.median_income)
plt.show()
```

24. It's hard to see anything in this plot (there are 3000 values here, one for each record) so let's reduce the number of (x,y)-values to 100 each:

```
plt.scatter(df.median_house_value[:100],df.median_income[:100])
plt.show()
```

25. We can customize this plot minimally by adding labels and a title:

```
plt.scatter(df.median_house_value[:100],df.median_income[:100])
plt.xlabel('Median Income')
plt.ylabel('Median House Value')
plt.title('Scatterplot: Median Income vs. Median House Value')
plt.show()
```

26. IPython/Colab has a lot more power, e.g. there are many 'magic commands' with additional information. For example, enter `%whos` now for a list of all the user-defined variables and functions that you created in this notebook session.

```
%whos
```

```
Variable   Type         Data/Info
----------------------------------
df         DataFrame    ... [3000 rows x 9 columns]
pd         module       <module 'pandas' ...
plt        module       <module 'matplotlib.pyplot' ...
```

This concludes a quick demonstration of Colab's IPython capabilities and some of Python's data and plotting abilities. See here for the complete Colab notebook as a GitHub gist: bit.ly/3NS6sbu.

The 'hello world' program

1. Open a `New notebook` in the `File` menu at the top.
2. Name the notebook `SnapToPython.ipynb`.
3. Enter `CTRL + ALT + T` or select `+Text` for a text block.
4. Create a headline: `# My first Python program`.

5. Open this Snap! file, `edit` the code and extract a picture of the code block: tinyurl.com/SnapToPython to your PC.

6. Create another text block and open the image symbol to `insert image`, then insert the PNG image that you downloaded.

7. Now leave the editor by clicking on the crossed out pen symbol at the top of the text block. You should see the image embedded in your notebook.

8. Create a new code block with `CTRL + ALT + I` (you can see all the keyboard short cuts in the sidebar at the bottom of the screen).

9. Enter the 'hello world' program and run it with `SHIFT + ENTER`:

   ```
   print('hello world')
   ```

10. In the next code block (that should already have appeared), enter another `print` statement:

6 ways to code in Python (online demo)

1. In the command line terminal (Python must be installed)
2. In Google Colaboratory (interactive Jupyter notebook)
3. In replit.com (REPL)
4. In the browser (extension)
5. In the Python IDLE (Integrated Development Learning Environment)
6. In an IDE (Integrated Development Environment): Emacs, VSCode etc.

References

- Birkenkrahe (2023). Introduction to programming in Python. URL: github.com/birkenkrahe/py
- Joshi (2021). Learn Python in a Snap! URL: abhayjoshi.net.
- Van Rossum, G., Drake, F. L. (2009). Python 3 Reference Manual. URL: `https://docs.python.org/3/reference/`.