PBQ-01 Lecture FAQ

QuantInsti Quantitative Learning Pvt Ltd

1. How can we improve the performance of the python program?

There are various which we can use to increase the program performance. Some of them are listed below:

- Use vectorized statements that use broadcasting to perform the operations.
- Use alternative implementations of Python language like `pypy` and `cpython`.
- Try to use built-in functions wherever possible.
- Though Python manages memory internally, try releasing memory whenever possible.

2. Examples of where to use List and Tuples?

List and tuples are sequential data structures. Their uses differ from application to application. Generally,

- We use lists whenever we are unsure about the number of elements that goes into them.
- We use tuples when we are sure that we do not need to change once created. For example, when working with a complex project (probably using the OOP), we often need to return the values from one class to another. To avoid any mutability, we can use tuples.

3. How to convert a code cell to a markup type and vice versa?

We can change a cell type using the drop-down menu (below the menu bar) in the Jupyter notebook. Alternatively, we can use the keyboard shortcuts to do so. They are as follows:

- Change the cell to command mode by pressing the `Esc` (escape) key on the keyboard. On Jupyter notebook, when a cell has Blue colored border, it is said to be in the command mode.
- Then, press the following keyboard shortcuts to convert the cell type:
  - Press `m` to convert a code cell to a markdown cell.
  - Press `y` to convert a markdown cell to a code cell.

4. How to open an IPYNB extension file? Or how to open a Jupyter notebook file?

There are various ways to run or open a Jupyter notebook. We can run it either locally or on the cloud.

- To run it locally:
  - Open Anaconda, and then use applications like `Jupyter Notebook` or `Jupyter Lab`.
- To run on the cloud:
  - One can use cloud hosted architectures like Google Colab, Kaggle Notebooks, Microsoft Azure Notebooks, or Binder.

5. How to parse data when it contains dates and time?

While reading such data in pandas using the `.read_csv()`, we can provide `True` to the `parse_dates` argument. Doing so, pandas will try to parse the dates in the file that we are reading. It usually parses most of the date formats; however, there can be instances when the parsing fails. In such cases, we need to provide a custom date parser — Read this for more information.

6. How to assign a value in a dataframe cell and how to access it?

Value assignment:

```
df['col_name'].iloc[row_index] = value
df['col_name'].loc['index_value'] = value
```

Value Access :

```
value = df['col_name'].iloc[row_index]
value = df['col_name'].loc['index_value']
```

7. How to fetch all files from a folder?

We can use the `glob` library as shown below:

```
path = '/home'

for fname in glob.glob(path + '/*.csv'):

    print('Filename:', fname)
    df = pd.read_csv(fname)

    # Print head of the file
    df.head()
```

8. In the `pd.read_csv()` what does the `parse_dates=True` do?

It tries to parse dates in the dataset we are reading as `dates`. Suppose we provide `False` to `parse_dates`. In that case, dates will be imported as strings. We will not be able to perform any dates or time-related operations like filtering dataset based on dates.

9. Is there a thumb rule when to use single quotes and double quotes?

Not really. In Python, we can use either. In other words, Python supports both, single quotes `'` or double quotes `"`. However, be consistent with whatever you use.

10. How can we add multiple elements in a list or tuple?

We can use the `extend()` function on a list to extend it. For example:

```
stocks = ['FB', 'AAPL', 'BAC']
new_stocks = ['HP', 'TSLA', 'GOOG']

print(stocks)

stocks.extend(new_stocks)
```

```
print(stocks)
```

Tuples, by definition are immutable, so we can not alter them once created.

11. What are the various IDEs and their features?

There are various IDEs available for programming in Python. Some of them are listed below:

- - Atom
- - Sublime Text
- - Spyder
- - PyCharm
- - Microsoft Visual Studio
- - Jupyter

Each of these software has a rich set of features, some common, some unique. Every developer has their preference in terms of their usage. Mostly, developers try their hands on multiple editors before deciding which to continue with based on their comfort levels.

12. Is there a quick guide for pandas library?

Yes, there is. A 10 minutes to pandas tutorial is the one. If you wish to explore it more, read cookbook.

13. What is the difference between `[]` and `()`? And when to use them?

When we want to extract elements from sequential data structures like list, tuples, or pandas series, we use `[]`. When we define methods or call them, we use `()`. The usage of brackets would remain uniform across the data structures we use, be it lists, tuples, NumPy arrays or pandas dataframe. More on it will be covered in the PBQ-01 tutorial session.

14. What is the difference between 0 and null?

In Python, null values are represented by `nan`, `NaN` or `np.nan`. Whereas 0 is used as an integer. Unlike other programming languages, there is no particular meaning of 0 in Python.

15. What is the difference between `conda` and `pip`?

`pip` installs Python packages, whereas `conda` installs packages that may contain software written in any language. Another key difference between the two tools is that `conda` can create isolated environments containing different versions of Python and/or the packages installed in them. Refer to this for more information.

16. Provide a list of useful python packages/libraries?

A non-comprehensive list of Python packages that we will use is as follows:

- Data Fetching:
  - yfinance
  - nsepy
  - quandl
- Data Analysis

  - pandas
  - numpy
  - pyFlux
  - fbprophet
  - statsmodels
- Technical Indicators
  - Talib
- Data Visualization
  - matplotlib
  - seaborn
  - plotly
- Portfolio Analysis
  - pyfolio
- Machine Learning
  - sklearn
  - Keras
- NLP
  - Spacy