SUMMARY OF MODULE 1 (8 WEEKS)

WORLD QUANT MODULE 1

WEEK 1 – PROGRAMMING AND PYTHON FUNDAMENTALS, DATA STRUCTURES (PY_ProgramFlow, in, ip,)

UNIT 1.1.1: Basic Programming and Variables

- Computer Program: list of instructions that a computer carries out in order
- "Interpreter" serves as an intermediary
- Variables are stored in the computer memory
- Python automatically declares type of variable
- " { } ***.format( )": string formatting
- You cant remember everything: GOOGLE is your friend

UNIT 1.1.2: Functions

- Functions allow us to define a task we would like the computer to carry out based on input
- "def", best practice is to give functions and variables descriptive names
- "return": signals the end of a function
- We can use variables as function variables
- "assertions": True (then nothing happens), False (it raises an assertion error)
- Scope : part of the code where a variable has meaning
- Functions help organize code into hierarchy
- Modular Programming??? (related to functions)
- Syntax: structure of a Language
- Avoid using Keywords as Variable names

UNIT 1.1.3: Logic and Conditional Expressions

- Conditional statements take the form "if > then"
- Python uses whitespaces (indentations) to represent context (scope)
- Create a flow chart for conditional statement
- We can nest "if" statements
- Comparisons output "Boolean Values"
- Logical Operators combine Boolean values: and, or, not

UNIT 1.1.4: Iterations

- Iterations: repetitive loops to execute the same code many times
- Be careful of infinite loops
- "while" and "for" loops
- "range (*, *)"

- Aside (recursion): achieves repetition by calling itself
- List comprehension
- Arguments are place holders for inputs
- “Function Closure”
- Default values for arguments
- “*args”: to take in a list of arguments

UNIT 1.2.1: Data Structures (List)

- List: stores a collection of data
- Using list gives more readable code
- Collection: known as containers in Python
- In context of Data Science, we will often call a collection of data that logically belongs together a DATASET, and the type of variable we use to store it in python is a DATA STRUCTURE
- Lists are heterogeneous (can store different type of data)
- Index: List[*], List[*:*] (slicing)
- Use indexes to update or replace items in a list
- List are mutable
- “dir(object)”: to get methods that can be used on an object
- “.extend()”: to add multiple items
- “del(*)”, “.pop()”, “.sort()”
- Difference between variable and name (Variable_Name.jpg)

UNIT 1.2.2: Data Structures (Tuples)

- Tuple: it is immutable
- Can use index because it is ordered
- You can not use “del (*)”
- “unpacking syntax”: can be used on any container in python (????)
- Items / Object in the tuple are mutable

UNIT 1.2.3: Data Structures (Set)

- Sets: are mutable but unordered
- Cannot use indexing/ slicing
- Set give us flexibility about how the data is stored in memory
- “Problem solving with Algorithms and Data Structures using Python” (ebooks)
- Objects in a set must be unique
- “.intersection()”, “.union()”, “.difference()”, “.symmetric_difference()”
- Uses Hashing

UNIT 1.2.4: Data Structures (dict)

- Dict: Dictionary

- Key: Value pairs
- “zip()”: can be very useful for creating a dict
- Zip returns a generator like “range”
- We can cast to dict
- Uses Hashing, they are mutable
- “.update()”, “.del()”, “.pop()”, “.keys()”, “.values()”, “.items()”
- “in” keyword to search for data in our Data Structure
- Dict: searches by key
- “.get()”: Boolean value or None
- One of the best ways to iterate through a Dict is iterate through the items, then unpack it
- Comprehension: combining iteration with the creation of a Data Structure
- Comprehension: brings the code to plain language, makes them readable for simple tasks

UNIT 1.2.5: Data Structures (Collection Module)

- Collections: module in Python Standard Library
- It contains Containers which are extremely useful
- “import”: importing packages, modules, libraries
- “from Collections import namedTuple”: good for creating self-documenting code with almost no memory cost
- “from Collections import deque”: a double ended queue, optimized for removing/add from the end of the structure
- “%%timeit”: magic command to time our code
- “from Collections import counter”: counts element in some iterable and returns a dictionary-like structure containing the count of each element
- “.most_common(2)”
- “from Collections import defaultdict”
- Are Strings immutable: YES
- Are Strings ordered: YES
- Are Strings sliceable: heaven YES

WEEK 2 – ALGORITHMS, OOP (PY_Algorithms, vc, PY_OOP)

UNIT 2.1.1: Algorithms and Complexity Analysis

- Algorithm: a procedure for solving a problem
- Algorithms and Implementation have 3 major bottlenecks:
    1. Computational Complexity (how many instructions are executed)
    2. Memory Needs
    3. I/O (reads and writes to disk, network requests)
- “Big O”: to quantify the complexity of a particular algorithm
- Linear Growth {O(N)} or Quadratic Growth {$O(N^2)$}

UNIT 2.1.2: Computational/ Time Complexity

- Optimization of functions
- Logarithm complexity is better than linear complexity

UNIT 2.1.3: Recursions and Memoization

- Memoization related to dynamic programming, using data structure to our advantage for sequence problems
- Tradeoff between Memory and Computational Cost
- No repetition in Factorial so Memoization will not improve it

UNIT 2.1.4: Memory Complexity and Data Structures

- Nested loops: O(N*)
- List in a List: O(N*)
- Composite Keys in a Dict: O(N*)
- Get the book "Problem solving with Algorithm and Data Structures"
- Choice of Algorithm is tied to the type of Data Structure

UNIT 2.2.1: What is an Object

- Object: anything we can store in a variable (basic definition)
- "type(*)": expresses the class of *
- Class: is a blueprint of an Object
- Attributes: variables, Methods: functions
- Object Attributes: are the internal variables that we used to store information
- Object Methods: are the internal functions that implement different capabilities
- We will interact with an Object method more than its attributes
- Use "dir()" to know the attributes and methods of an Object or Class
- We can also check out Documentations

UNIT 2.2.2: Introduction to Classes

- Class: blueprint for how an Object should behave
- "__init__": for initialization
- "__repr__": for representation
- Class name is usually Capitalized (Camel case)
- Functions and variables names: lowercase
- Self: object itself
- "conventions" give clarity
- "__*__":dunder methods, when the class hooks into the fundamental elements of Python
- Dunder methods are not called by the Object directly
- Dunder variables: a user is not expected to use directly but intend for use by other parts of our code
- All methods are public in PYTHON

UNIT 2.2.3: Defining Useful Classes

- We have to tell Python to implement mathematical operators
- "isinstance()"
- Dunder method calls the "__mul__" of the method (element) on the left
- Use "__rmul__" to correct it

UNIT 2.2.4: OOP and Inheritance

- Classes help us organize our code into conceptual blocks
- Almost everything in Python is the invocation of Object or creation of Object
- Classes build on each other
- Often classes we define in Python will build off an existing idea in other classes (Inheritance)
- Built Objects: green highlights in notebook
- Sub-class and Super-class
- "isinstance(*, class)": to check if it inherits from Class

UNIT 2.3.1: VC Mini Project Intro and Hackerrank Coding Challenges

- Vc: vector class
- Website: "hackerrank"

WEEK 3 – PYTHONIC STYLE, READING AND WRITING DATA (PY_Pythonic, DS_IO, pw)

UNIT 3.1.1: Pythonic Programming

- Pythonic: written in good python style
- Python uses whitespaces as a marker making it more readable
- Namespaces are important

UNIT 3.1.2: Python Modules and Imports

- Packages are stored in "site-packages"
- Standard alias: Pandas as pd
- Global scope: highest level
- Local scope
- Namespaces are a way of organizing scope of names, domain over which a name has meaning
- We can import part of packages
- Python files: Modules (import module), Script (run from command line)

UNIT 3.1.3: Exceptions and Debugging

- Exception: something that deviates from the expected behavior

- “try” and “except” keywords (ask for forgiveness instead of permission, pythonic)
- Error messages are not so bad
- We should be specific about what kind of error we are catching
- “isinstance”: look before you leap, not pythonic
- Error message: Trace back

UNIT 3.2.1: Basic Python IO

- “.open()”, “.read()”, “.close()”
- “with open()”: no need to manually close the file (it is automatically closed)
- “.readline/s()”: returns a list
- “.write()”, “r”>read mode, “w” > write mode
- We cannot open a file for reading if does not exist
- “r+” und “w+” allow us to write and read at the same time (might be confusing)

UNIT 3.2.2: File System Operators in Python

- “OS”: module for navigating computer file systems
- “OS.listdir(“.”)”: files in current directory
- “OS.walk()”
- More info on “OS module” on the documentation pages

UNIT 3.2.3: CSV Files

- Most widely used format for tabular data
- CSV: Comma Separated Values
- “pd.read_csv()”, “to_csv()”
- Inspect the data structure after passing it to CSV

UNIT 3.2.4: JSON Files

- JSON: JavaScript Object Notation
- JSON files are popular for web
- JSON: nested Dicts and lists
- JSON does not have to conform to a tabular structure
- “json.dump()”: to write
- “jscon.load/s()”: to read
- “pd.read_json()”: will try to parse it into a tabular form (does not always work out)
- Use “pd.read_json()”: to scrap data from the web

UNIT 3.2.5: Compression and GZip Files

- Compression used to save storage and network resources
- “.gz”: data often contains patterns, uses the deflate algorithm
- Compressed files are binary files
- “wb”: write binary files

- Using “w” could damage the file

UNIT 3.2.6: Serialization and Pickle Files

- “pickle.dump()”
- Serialization: a protocol of representing bytes in memory as bytes on disks
- Pickle: capable of representing arbitrary Objects in Python
- “.pkl”: file extension
- We can pickle almost anything in Python
- “!***”: Bash commands

UNIT 3.3.1: PW Mini Project Intro

WEEK 4: PYTHON DATA SCIENCE PACKAGES (DS_Basic_DS_Modules)

UNIT 4.1.1: Intro to Numpy

- Numpy and Scipy are the oldest of the bunch
- Numpy features (similar to list) are 3 folds:
    1. Math Functions
    2. Random Sub module
    3. Numpy (ndarray) Object
- Numpy not part of Python Standard Libraries
- “np.array()”: has a shape
- Np.array must be rectangular (same no of elements for each rows and columns)
- Np.array: homogenous Data Structure

UNIT 4.1.2: Creating Numpy Arrays

- “np.linspace(*(lower bound), *(upper bound), *(no of elements))”
- “np.arrange()”: array(range(*, *, *))
- Develop the habit of checking or reading Documentations
- “np.logspace()”, “np.zeros()”, “np.ones()”, “np.diag()”, “np.eye()”
- Numpy is often used for linear algebra
- “astype(*)”: to cast to a type

UNIT 4.1.3: Advantages of Numpy

- Every element in a numpy array takes a uniform amount of memory (block of memory)
- Uniform interpretation of mathematical operations (np.array), making it fast
- Numpy has better syntax
- Basis of other Python DS packages

UNIT 4.1.4: Computation with Numpy

- In numpy array the default operation is “elementwise”

UNIT 4.1.5: Mutating Numpy Arrays

- Np.arrays are mutable
- ".ravel()": ravels array to one dimension

UNIT 4.1.6: Plotting with Matplotlib

- Matplotlib is the most popular Python plotting library
- "plt.figure" <-> "plt.plot()", hooks to the nearest (most recent) plt.figure
- ".subtitle()", "plt.subplots"
- Create plots > plot data > label plot (Objected Oriented Approach)
- Many ways to create visualizations

UNIT 4.1.7: Intro to Pandas

- A Data Frame is a table
- Each columns represents a different type of data (field), each row represents a different record/ entity
- Make Data Frame using "zip ()"
- Column names are like keys to the Data Frame
- Use ".loc[**]" to retrieve a row in a Data Frame
- "inplace = True": to make change permanent

WEEK 5 – PANDAS, INTRO TO STATISTICS (DS_Pandas, DS_Intro_Statistics, dw)

UNIT 5.1.1: Pandas Overview

- Pandas best for working with tabular data
- Each column is called a Pandas Series
- DataFrame is a dictionary of data/pandas series
- Each series of a DataFrame has a data type assigned to it
- ".loc[**row, **columns=]", need to specify a row but no a columns

UNIT 5.1.2: Pandas IO

- Natural way to construct a DataFrame is to use a Dict or series like objects
- The "index" in a DataFrame is not a column
- ".index"
- ".csv": one of the most popular format for representing data
- Check out "pandas.read_csv, read_json (orient, compression)" documentations
- "delimiter": alias for separator
- "**__version__": check version of **

UNIT 5.1.3: Filtering DataFrame

- "&" > and, "~" > not, "|" > or

- “str.contains(***)”

UNIT 5.1.4: Data Transformation

- “np.log”: one to one operation transformation
- “.mean()”: many to one operation aggregation
- “.apply()”: apply function to entire dataset
- “.fillna()”
- Lambda: anonymous function (lambda arg: return statement)

UNIT 5.1.5: Data Aggregation

- Groupby: “split”, “apply”, “combine”
- Checkout Pandas Groupby documentation
- “.agg()”: to aggregate groups
- Instead of “for loop” on Pandas Object you can use “.apply()” or ”.agg()” {takes in function as arguments}

UNIT 5.1.6: Joins and Multiple DataFrame

- “sorted_values()”
- “.merge()”
- Types of Joins:
    1. Inner join: deletes if there is not match
    2. Outer join: keeps even if there is no match
    3. Left join: prioritize one DF over the other
    4. Right join:
- Pandas by default performs “inner join”, “how” (keyword for specifying type of join)
- “concat()”: stack on each other, “axis= 1” (refers to column wise)

UNIT 5.1.7: Time Series Data

- Time series data to represent user data over time
- “.set_index”
- Downsample (decreasing the frequency of data) or Upsample our data using “.resample()” (similar to a Groupby)
- Python “Timestamp” class similar and compatible to “datetime” class
- Check out Pandas “Plotting” documentation

UNIT 5.2.1: Intro to Statistics

- Probability is the study of randomness while Statistics is the study of how random variables behave in Aggregate
- Statistical Inference often fall into: Hypothesis Testing and Parameter Estimation

UNIT 5.2.2: Estimating Means

- Scipy is very related to Numpy, mostly used for its Statistical and computational tools
- Most times the higher the sample the smaller the difference between true value and sample value.
- Difference between Sample mean and True mean is called the Standard Error of the mean

UNIT 5.2.3: Standard Error of the Mean

- Standard Deviation is the average difference between two plots (how much the individual salary varies from the average salary)
- As sample size increases (we get a better estimate) our error decreases (law of large numbers)
- Central Limit Theorem
- Normal distribution is "bell-shaped"
- Check out videos on "stats tutorials" YouTube videos (Extra Statistics Videos)
- https://www.youtube.com/playlist?list=PLVNgHN883b8FuKC-Fpnh72thE4m7J1nNO

UNIT 5.2.4: Hypothesis Testing & Z-Scores

- Z-score and p-value
- Calculate the probability that the difference between sample mean and true mean is due only to random choice (p-value)

UNIT 5.2.5: Parameter Estimation and Confidence Intervals

WEEK 6 – DATA MUNGING (DS_Data_Munging)

UNIT 6.1.1: Data Munging and Web Scraping

- Munging is the process of cleaning and giving structure to raw data
- "pd.read_html": pandas reads tables easily from webpages
- Web pages carry out a request and response cycle
- Request Library: "response = requests.get(url)" > "response.reason"
- Beautiful Soup Library

UNIT 6.1.2: Cleaning and Structuring Data

- Fortune 500 companies
- Mapping: if data have slightly different names
- ".lstrip()", ".rstrip()"

UNIT 6.1.3: Dealing with Multiple Data Sources

- Fortune 500 companies
- ".split()"

UNIT 6.1.4: Data Munging in Pandas

- Fortune 500 companies
- DataFrame approach much better when dealing with tabular data
- Pandas is great for Data Munging

WEEK 7 – SQL AND OBJECT RELATION MAPPING (DS_SQL)

UNIT 7.1.1: SQL & Relational Databases

- SQL: Structured Query Language, standardized for accessing and manipulating relational databases
- Relational data model: corresponds with our intuitive notion of a TABLE, each row is a RELATION, each column is an ATTRIBUTE
- SQL ipython extension
- Database Normalization to reduce data redundancies in SQL

UNIT 7.1.2: Basic SQL Syntax

- "SQLite": basic SQL database manager that is useful for small analysis and instructional purposes
- SQLite databases are stored in a file
- No matter the Database you start with a connection
- "--": used to comment in SQL
- "DROP TABLE": to delete a table
- "CREATE TABLE": to create a table
- "PRIMARY KEY": similar to index in Pandas but must be unique
- Each line of code is terminated with a " ; " (semicolon)
- "SELECT": To inspect a table
- "INSERT": To insert data into our tables
- SQL commands are in UPPERCASE while names are in LOWERCASES
- "LIMIT": To limit number of rows printed to screen (good idea when inspecting tables)
- "AUTOINCREMENT": to auto increase the index value with each row added

UNIT 7.1.3: Filtering & Sorting Data in SQL

- "import sqlite3": Python SQLite module
- "%%sql": magic command to allow run SQL command in Notebook
- "sqlite3.connect": To connect to Database
- "***(panda_DF).to_sql": csv files to SQL Database
- "WHERE": Used to accomplish filter in SQL
- We use "=" instead of "==" for comparison
- "LIKE": for pattern matching, similar to string contain method
- Check out your Pattern Matching keyword for your particular flavor of SQL

- “IN”: similar to Pandas isin() method
- “ORDER BY”: To sort, it is done in Ascending order by default

UNIT 7.1.4: Joins and Aggregation in SQL

- “AVG”, “MAX”, “”
- SQL is a declarative query language
- “AS”: to give an alias
- Aggregation keywords available depend on your Flavor of SQL
- “JOIN”: To join tables, we can perform all kinds of Joins
- “HAVING”: A sort of conditional/ filtering keyword
- Subqueries:
- Hacker rank practice coding

UNIT 7.1.5: Comparison of SQL Flavors

- SQLite is stored in a portable file on your Disk
- Your decision might be based on your company’s choice
- PostgreSQL is free and open source
- Most SQL except SQLite can handle large databases

WEEK 8 – OBJECT RELATION MAPPING (ORM, DS_Classes_and_ORM)

(very confusing…don’t verstehen)

UNIT 8.1.1: Introduction to Object Relation Mapping

- SQL is a declarative language
- “Mixin”: used to define methods you would like to introduce to broad range of Objects, Classes
- Validation useful for validating user entered data

UNIT 8.1.2: Using Classes for Applications

- Class inheritance

UNIT 8.1.3: Translating Classes into SQL with ORM

- Database don’t disappear if the computer restarts
- Translating each class into a table in the Database
- ORM: Python Object (Instance of Class) < > Relation (Table)
- Checkout the SQLAlchemy Documentation
- “@hybrid_property”:
- Database mostly used to store static data

UNIT 8.1.4: Pushing Objects to SQL & Making Queries

- Session is similar to a python interpreter (handles interaction between Python and SQLAlchemy)

UNIT 8.2.1: Course Conclusion

- Download as zip
- Hacker Rank
- EBook on “Problem Solving with Algorithms and Data Structures using Python”
- Code Academy free python course (not sure if there are still free courses)
- Google Python Class