Engineering for everyday world

AceEngineer

PYTHON

Development Document

30th August 2020

30-Aug-2020	01	New Issue	VA		
DATE	**REV**	**DESCRIPTION**	**ORIG**	**CHK**	**APPR**

DOCUMENT CONTROL NO	**Project**	**Type**	**Area**	**Client**	**-**	**-**	**Sequence**	**Revision**
	0026	**PY**	**-**	**-**	**-**	**-**	**0001**	**01**

Revision History:

REV	DATE	DESCRIPTION	ORIG	CHK	APPR
01	30th Apr 2018	New Issue	VA		

Change Log

REV	SECTION	CHANGE DESCRIPTION

Document Holds

Hold	DESCRIPTION
HOLD 01	

CONTENTS

1 INTRODUCTION

Python programming language is a very simple and high-level object-oriented language. It can be easily used by any person and has the versatility to interface with the computer systems as well as existing programs. It helps in easy customization, optimization and automation of repeated tasks, popular software actions, websites actions etc.

This document contains the following:

- A good starting guidance for Python developer
- Information to begin Python coding
- The key features and version evolution of python to help understand the differences between 2.x and 3.x.
- A high level document summary is also given in Section 1.3 as cheatsheet for quick reference.

1.1 Python Unique Features

The key Python features worth noting are:

- A high level programming language. Easy to understand.
- The program is highly readable by common audience. Naming the variables to be more intuitive will enhance this readability further.
- Compiler/Interpreter is not required. The program interprets the commands directly
- Program is case sensitive
- Python stores and references data using pointers, [1]
- Open source with lot of reliable modules to accomplish a project. The available resources can be overwhelming at times.

1.2 Python 2.X vs. 3.X

Python 3.x implements key updates, [2]. Therefore there will be syntax and feature differences between 2.x and 3.x. Some of the key differences are highlighted below:

- Syntax and handling is different
  - Print (with brackets in 3)
    - Print(“ He is a good boy”) for python 3+ versions
    - Print “he is a good boy’ for python 2+ versions
  - Integer division handling (more correct in 3)
  - Unicode exists in python 3
  - Converted “xrange” to “range” in python 3
  - _contains_ method is added in python 3 for range objects
- If a program is already existing in 2.x, upgrading from 2.x to 3.x is difficult due to the following reasons:
  - Library availability. If libraries are used in 2.x program, less popular libraries may not be readily available in 3.

    - Porting to use 2.x codes or libraries
        - A well written code is easy to port
        - Some features will never come through
            - Non-local variable declaration
        - If porting is hard, can the feature be dropped? (or) can the feature re-developed in 3?
        - If absolutely required, start with 2.x

- Downfalls of choosing to stay with 2.x:
    - No continued support
    - Continued development may be in 3 by other project members or stakeholders
    - More libraries will be available in 3 and helps in easy future updates
    - 3 is more advanced and is considered the future of Python
    - Python 2.x support ends in 2020.

A decision making summary flowchart for deciding the development way forward is given below:

- New project : python 3.x
- Existing 2.x project libraries:
    - Port (rewrite) to 3.x if not too difficult
    - Hybrid solution : Utilize 3.x for further development and use existing 2.x libraries.
    - Run in 2.x if absolutely no solution with 3.x is useful or enhances the application
    - Strongly recommend to run 2.x in virtual environment. Moreover, any new project should run in a virtual environment.

1.3 Document Summary

To help utilize this document effectively, a summary is given in this section. The beginner python tools can still be used efficiently by advanced users.

Area	**Beginner Python**	**Intermediate Python**	**Section**
Python Distribution	Anaconda	Any	Section 2 Appendix 1.0
IDE	Visual Studio Code Spyder	PyCharm	
Console	IPython		
Program Management	Anaconda Navigator		
Module Installer	Conda or Pip	PIP	
Command Prompt	Anaconda Prompt	IDLE	

1.4 References

S.No.	Link	Description
[1]	http://scottlobdell.me/2013/08/understanding-python-variables-as-pointers/	Understanding Python and Pointers
[2]	https://lwn.net/Articles/650904/	Key updates in Python 3.5
[3]	https://automatetheboringstuff.com/#toc Automate the boring stuff http://www.javatpoint.com/python-tutorial http://www.tutorialspoint.com/python/index.htm	Good reference books.
[4]	https://www.codementor.io/sheena/python-path-virtualenv-import-for-beginners-du107r3o1 https://docs.python.org/2/tutorial/modules.html	Python strength features for beginners
[5]	http://www.openbookproject.net/thinkcs/python/english3e/	How to think like a Computer Scientist
[6]	https://f0nzie.github.io/yongks-python-rmarkdown-book/	Practical Python Resource

2 BEGINNERS PYTHON

Starting to learn python can be intimidating due to the buzz and other people opinions. This section makes the python learning process a bit smoother especially for beginners. Python starters should note that Python is an open source programming language. This is good for development as program inputs and standard code snippets can be obtained from various communities/sources/websites. However, this also makes initial learning difficult. Multiple sources for learning and information and various distributions/flavours/methods available to solve your unique problem.

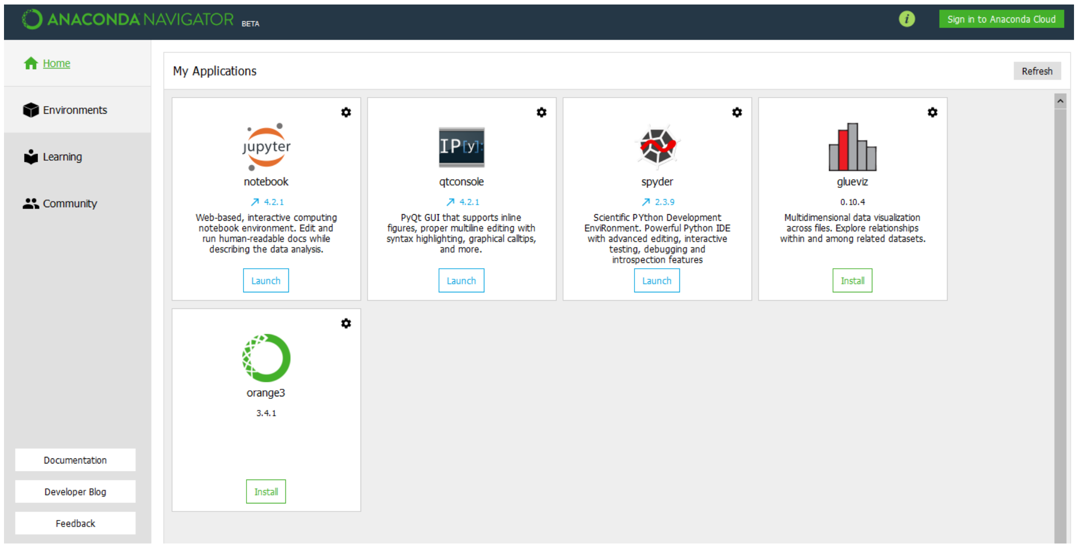
2.1 Anaconda Installation and IDE to Use

There are many python installations that are available for use. For data analysis or data science or for beginners, Anaconda python is the easiest to start with.

The installation of Anaconda can be found below:
https://anaconda.org/anaconda/python
Installation instructions can be found below:
https://medium.com/@GalarnykMichael/install-python-on-windows-anaconda-c63c7c3d1444

The key Anaconda features are summarized below:

- The IDE Spyder is easy to run. Spyder is an IDE for scientific computing in Python Language.
  - by selecting it directly from the windows program.
  - Spyder can be run by typing spyder in Anaconda prompt (or)
- IPython is an advanced console
  - Can look at live functions. This is very similar to Matlab's run-time environment.
- Jupyter QT Console and Notebook: Advanced in-line processing to help write comments, visualize plots as you code along.
  - Interactive console for programming
  - Great for starters to explore
  - A great tool when teaching any concept in detail to audience
  - The pages/program can also be shared in cloud for demonstration & easy execution
- Anaconda Navigator helps easy management of modules. It give access to administrative actions such as update, downgrade or remove modules that are incompatible.
- See Appendix 1.1 for further installation details. The common errors encountered using Anaconda IDE are also given in Appendix 1.0.



Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

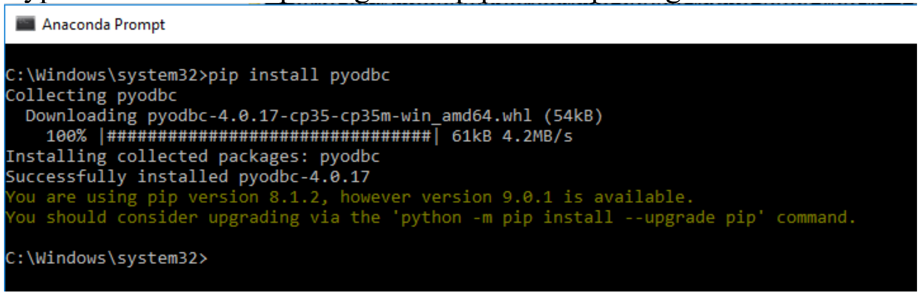
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

Multidimensional data visualization across files. Explore relationships within and among related datasets.

Common Actions

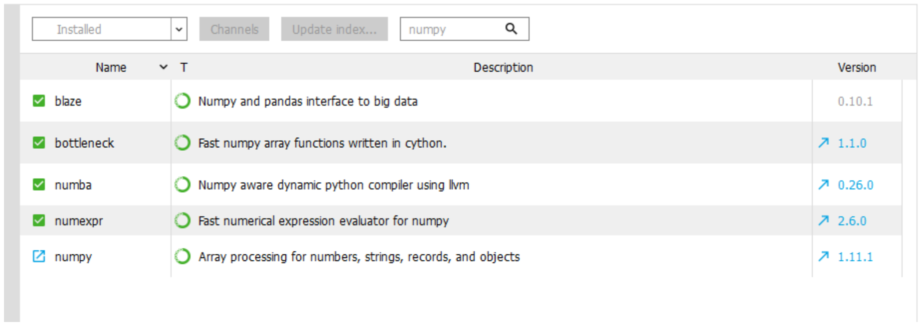
- Adding a Module
  - Open Anaconda Prompt
  - Type : "conda install package" or "pip install package"



```
C:\Windows\system32>pip install pyodbc
Collecting pyodbc
  Downloading pyodbc-4.0.17-cp35-cp35m-win_amd64.whl (54kB)
    100% |################################| 61kB 4.2MB/s
Installing collected packages: pyodbc
Successfully installed pyodbc-4.0.17
You are using pip version 8.1.2, however version 9.0.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

C:\Windows\system32>
```

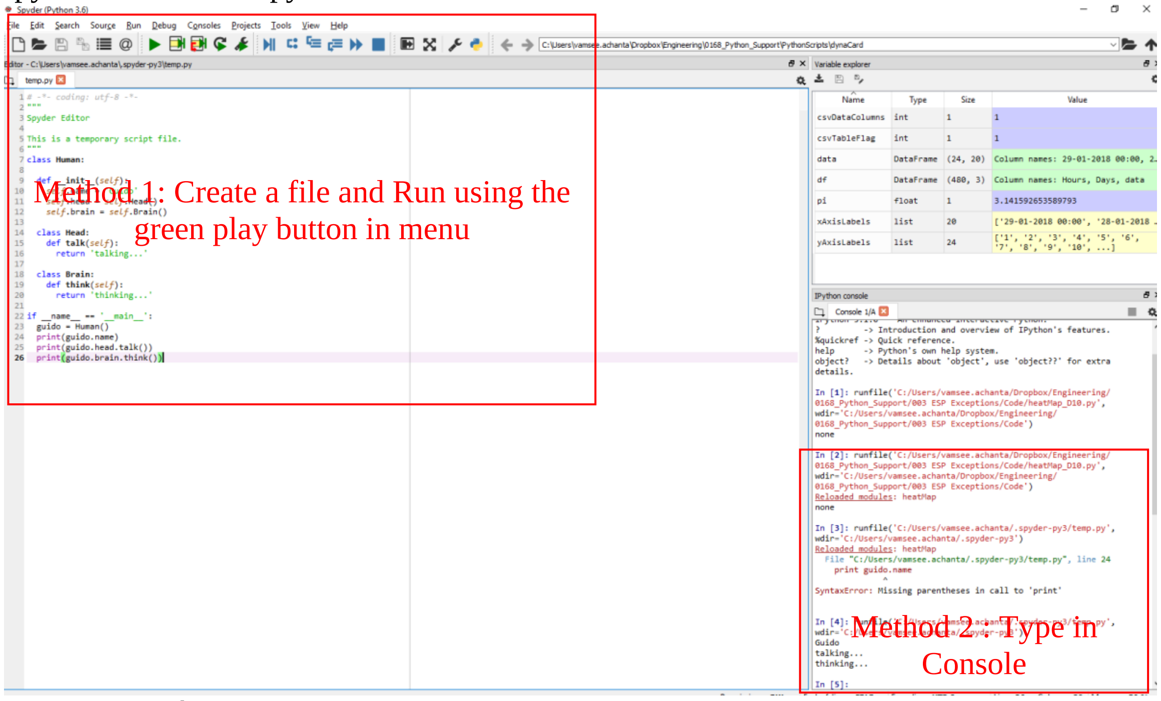
  -
- Updating a module using Anaconda Navigator
  - Click on "Environments" on the left tab
  - Search for required package



	Name	T	Description	Version
✓	blaze	○	Numpy and pandas interface to big data	0.10.1
✓	bottleneck	○	Fast numpy array functions written in cython.	↗ 1.1.0
✓	numba	○	Numpy aware dynamic python compiler using llvm	↗ 0.26.0
✓	numexpr	○	Fast numerical expression evaluator for numpy	↗ 2.6.0
↗	numpy	○	Array processing for numbers, strings, records, and objects	↗ 1.11.1

  -
  - Mark for upgrade by right clicking on package name on left column.
  - Appy to ugrade package
  - https://stackoverflow.com/questions/45197777/how-do-i-update-anaconda
  - Updating a module using Anaconda Prompt. See secton

2.2 Running Files

- Type simple command in :
  - python command prompt (or)
  - Ipython console of Spyder. See section 1.1.2.
  - 

Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\vamsee.achanta\Dropbox\Engineering\0168_Python_Support\PythonScripts\dynaCard
Editor - C:\Users\vamsee.achanta\.spyder-py3\temp.py
temp.py
Variable explorer
IPython console
Console 1/A
Method 1: Create a file and Run using the green play button in menu
Method 2 : Type in Console

```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""
class Human:

  def __init__(self):
    self.name = 'Guido'
    self.head = self.Head()
    self.brain = self.Brain()

  class Head:
    def talk(self):
      return 'talking...'

  class Brain:
    def think(self):
      return 'thinking...'

if __name__ == '__main__':
  guido = Human()
  print(guido.name)
  print(guido.head.talk())
  print(guido.brain.think())
```

Name	Type	Size	Value
csvDataColumns	int	1	1
csvTableFlag	int	1	1
data	DataFrame	(24, 20)	Column names: 29-01-2018 00:00, 2…
df	DataFrame	(480, 3)	Column names: Hours, Days, data
pi	float	1	3.141592653589793
xAxisLabels	list	20	['29-01-2018 00:00', '28-01-2018 …
yAxisLabels	list	24	['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', ...]

```
?         -> 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.

In [1]: runfile('C:/Users/vamsee.achanta/Dropbox/Engineering/
0168_Python_Support/003 ESP Exceptions/Code/heatMap_D10.py',
wdir='C:/Users/vamsee.achanta/Dropbox/Engineering/
0168_Python_Support/003 ESP Exceptions/Code')
none

In [2]: runfile('C:/Users/vamsee.achanta/Dropbox/Engineering/
0168_Python_Support/003 ESP Exceptions/Code/heatMap_D10.py',
wdir='C:/Users/vamsee.achanta/Dropbox/Engineering/
0168_Python_Support/003 ESP Exceptions/Code')
Reloaded modules: heatMap
none

In [3]: runfile('C:/Users/vamsee.achanta/.spyder-py3/temp.py',
wdir='C:/Users/vamsee.achanta/.spyder-py3')
Reloaded modules: heatMap
  File "C:/Users/vamsee.achanta/.spyder-py3/temp.py", line 24
    print guido.name
              ^
SyntaxError: Missing parentheses in call to 'print'


In [4]: runfile('C:/Users/vamsee.achanta/.spyder-py3/temp.py',
wdir='C:/Users/vamsee.achanta/.spyder-py3')
Guido
talking...
thinking...

In [5]:
```

- As you write your 2nd program
  - Write the code in a .py file
  - Execute it in command prompt (or) IDE
  - This will help write large programs
  - These files will contain codes that can be reused
- If you are writing a line of code twice:
  - Think of writing a function
  - Writing functions is further described in Section 5.2.

2.3 Simple Programs

Program	**Code**	**Notes**
For Print	*Print(" He is a good boy") for python 3+ versions* *Print "he is a good boy' for python 2+ versions*	

For creating and writing to text file	*Method 1: When writing to single file in one single go:*
	with open('RiserStackUp_Code.txt','w') as stackup:
	print("import OrcFxAPI as orca", file=stackup)
	print("model = orca.Model()", file=stackup)
	stackup.close()
		"w" is used for writing, "a" is used for appending, and "r" is used for reading

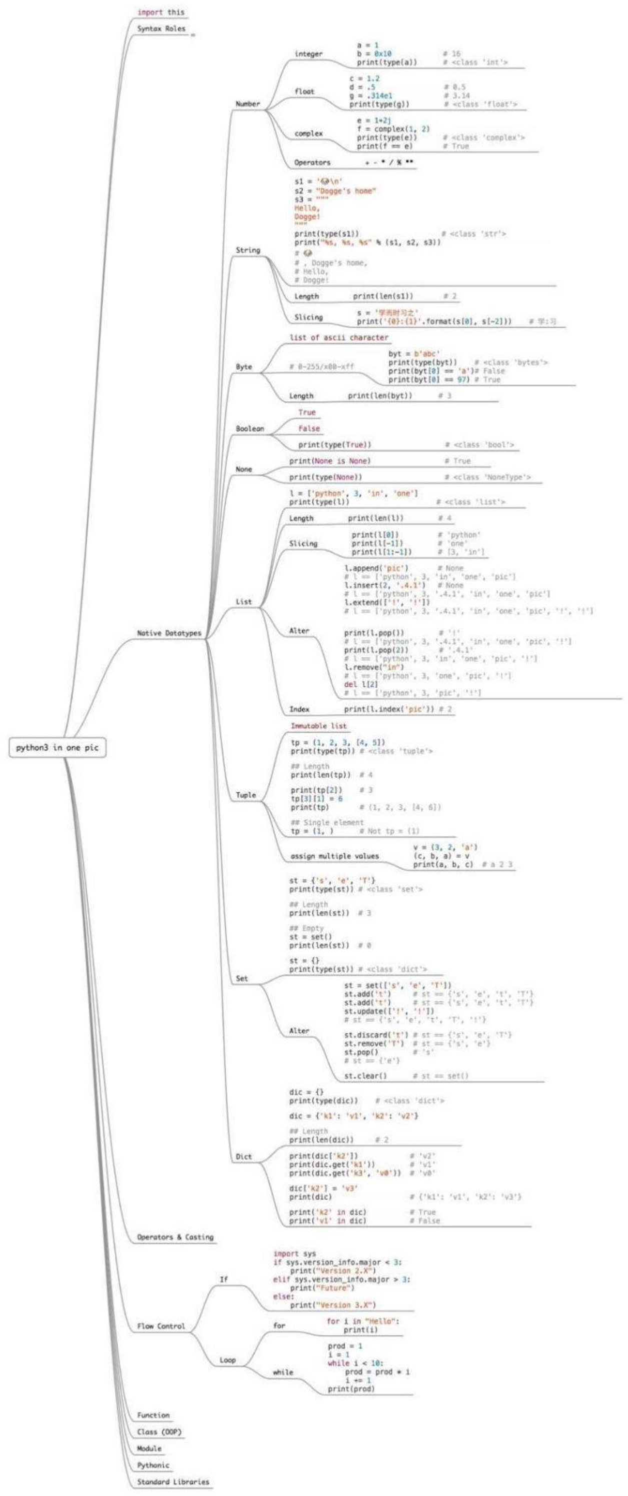
	Method 2: If writing to same file but at various locations in code: *f = open('myfile', 'w')* *f.write('hi there\n') # python will convert \n to os.linesep* *f.close()*	
List/sort by File type	*from glob import glob* *textFiles = glob('*.txt')*	This will need import glob. Learning to import modules in python is essential here.
Remove extensions while using file names	*from glob import glob* *textFiles = glob('*.txt')* *NameList = [item[:-4] for item in textFiles]*	Parses the folder for any text file and removes".txt" while saving the text file names in an array

Draw 2 simple rectangles	*import pygame, sys*
	*from pygame.locals import **
	pygame.init()
	white=(255,255,255)
	Red=(255,0,0)
	pygame.draw.rect(DISPLAY,red,(20,70,560,20))
	pygame.draw.rect(DISPLAY,white,(255,143,70,30))
	pygame.image.save(DISPLAY, "2 Rectangles.png")
		Installation and Import of pygame module is required.

open a work book and changing a particular cell.	*import openpyxl* *workbook = openpyxl.load_workbook('InterventionRiser (MP1).xlsm', keep_vba=True)* *sheet = workbook.get_sheet_by_name('KeyRiser')* *sheet['B9'] = '72.0'* *workbook.save('InterventionRiser (MP1).xlsm')*	Will need to install openpyxl module and import it If the work book is macrto enabled, we need to use keep_vba = True
To read data from excel file as arrays	*import xlrd* *wb = xlrd.open_workbook('environmentLoading.xlsx') # Open Excel file* *sh1 = wb.sheet_by_name(u'Sheet1') # Read Sheet* *waveType = sh1.col_values(0) # Defining wave type in array* *waveDirection = sh1.col_values(1) # Defining wave direction in array* *waveHeight = sh1.col_values(2) # Defining wave height in array* *timePeriod = sh1.col_values(3) # Defining wave period in array*	Another module to handle excel files. However, note that usage of module is based on usecase.
To read data from excel file as dataFrames	*import pandas* *df = pandas.read_excel(open('your_xls_xlsx_filename','rb'), sheetname='Sheet 1')*	Another module to read data from excel. DataFrames is a powerful way to data analysis and handle large amounts of data. Pandas imports data from excel as dataframes. DataFrames are

		very powerful way of storing data for data science applications. https://stackoverflow.com/questions/17063458/reading-an-excel-file-in-python-using-pandas
Search other simple programs on internet		

2.4 High-level Python 3 Fundamentals



```
a = 1
b = 0x10              # 16
print(type(a))        # <class 'int'>

c = 1.2
d = .5                # 0.5
g = .314e1            # 3.14
print(type(g))        # <class 'float'>

e = 1+2j
f = complex(1, 2)
print(type(e))        # <class 'complex'>
print(f == e)         # True

+ - * / % **
```

```
s1 = '🐶\n'
s2 = "Dogge's home"
s3 = """
Hello,
Dogge!
"""
print(type(s1))                         # <class 'str'>
print("%s, %s, %s" % (s1, s2, s3))
# 🐶
# , Dogge's home,
# Hello,
# Dogge!

print(len(s1))          # 2

s = '学而时习之'
print('{0}:{1}'.format(s[0], s[-2]))    # 学:习
```

```
# 0-255/x00-xff
byt = b'abc'
print(type(byt))      # <class 'bytes'>
print(byt[0] == 'a')# False
print(byt[0] == 97) # True

print(len(byt))       # 3
```

```
True
False
print(type(True))                 # <class 'bool'>

print(None is None)               # True
print(type(None))                 # <class 'NoneType'>
```

```
l = ['python', 3, 'in', 'one']
print(type(l))                   # <class 'list'>

print(len(l))         # 4

print(l[0])           # 'python'
print(l[-1])          # 'one'
print(l[1:-1])        # [3, 'in']

l.append('pic')       # None
# l == ['python', 3, 'in', 'one', 'pic']
l.insert(2, '.4.1')   # None
# l == ['python', 3, '.4.1', 'in', 'one', 'pic']
l.extend(['!', '!'])
# l == ['python', 3, '.4.1', 'in', 'one', 'pic', '!', '!']

print(l.pop())        # '!'
# l == ['python', 3, '.4.1', 'in', 'one', 'pic', '!']
print(l.pop(2))       # '.4.1'
# l == ['python', 3, 'in', 'one', 'pic', '!']
l.remove("in")
# l == ['python', 3, 'one', 'pic', '!']
del l[2]
# l == ['python', 3, 'pic', '!']

print(l.index('pic')) # 2
```

```
tp = (1, 2, 3, [4, 5])
print(type(tp)) # <class 'tuple'>

## Length
print(len(tp))  # 4

print(tp[2])    # 3
tp[3][1] = 6
print(tp)       # (1, 2, 3, [4, 6])

## Single element
tp = (1, )      # Not tp = (1)

v = (3, 2, 'a')
(c, b, a) = v
print(a, b, c)  # a 2 3
```

```
st = {'s', 'e', 'T'}
print(type(st)) # <class 'set'>

## Length
print(len(st))  # 3

## Empty
st = set()
print(len(st))  # 0

st = {}
print(type(st)) # <class 'dict'>

st = set(['s', 'e', 'T'])
st.add('t')      # st == {'s', 'e', 't', 'T'}
st.add('t')      # st == {'s', 'e', 't', 'T'}
st.update(['!', '!'])
# st == {'s', 'e', 't', 'T', '!'}

st.discard('t') # st == {'s', 'e', 'T'}
st.remove('T')  # st == {'s', 'e'}
st.pop()        # 's'
# st == {'e'}

st.clear()      # st == set()
```

```
dic = {}
print(type(dic))    # <class 'dict'>

dic = {'k1': 'v1', 'k2': 'v2'}

## Length
print(len(dic))     # 2

print(dic['k2'])               # 'v2'
print(dic.get('k1'))           # 'v1'
print(dic.get('k3', 'v0'))     # 'v0'

dic['k2'] = 'v3'
print(dic)                     # {'k1': 'v1', 'k2': 'v3'}

print('k2' in dic)             # True
print('v1' in dic)             # False
```

```
import sys
if sys.version_info.major < 3:
    print("Version 2.X")
elif sys.version_info.major > 3:
    print("Future")
else:
    print("Version 3.X")
```

```
for i in "Hello":
    print(i)
```

```
prod = 1
i = 1
while i < 10:
    prod = prod * i
    i += 1
print(prod)
```

2.5 Writing Own Code and When Stuck

- Build your own project. A simple example project can be:
    - Read a text file with data
    - Perform simple mathematical operations
    - Write data output into files
- Start a simple implementation
- Expand around it
- When errors happen, google and overstackflow will definitely have a solution. Searching right words/terms is an art and one will learn over time.
- Understanding Code:
    - Pythontutor.com is a great resource to understand how python works.
    - See below example:
    - http://pythontutor.com/visualize.html#code=%0Adogs%20%3D%20%5B%5D%20%0Adogs.append%28'willie'%29%20%0Adogs.append%28'hootz'%29%20%0Adogs.append%28'peso'%29%20%0Adogs.append%28'goblin'%29%20%0Afor%20dog%20in%20dogs%3A%20%0A%20%20%20%20print%28%22Hello%20%22%20%2B%20dog%20%2B%20%22!%22%29%20%0A%20%20%20%20print%28%22I%20love%20these%20dogs!%22%29%20%0A%20%20%20%20print%28%22%5CnThese%20were%20my%20first%20two%20dogs%3A%22%29%20%0Aold_dogs%20%3D%20dogs%5B%3A2%5D%20%0A%0Afor%20old_dog%20in%20old_dogs%3A%20%0A%20%20%20%20print%28old_dog%29%20%0A%0Adel%20dogs%5B0%5D%20%0Adogs.remove%28'peso'%29%20%0Aprint%28dogs%29%0A&cumulative=false&curInstr=31&heapPrimitives=nevernest&mode=display&origin=opt-frontend.js&py=3&rawInputLstJSON=%5B%5D&textReferences=false

3 GOOD PRACTICES

This section summarizes the good practices to be followed by typical developers

https://www.oreilly.com/learning/20-python-libraries-you-arent-using-but-should

Modules	Purpose		
collections			
Contextlib	Context manager		
Concurrent.futures			
logging			
sched			
colorama	For coloring the python command line outputs		
colorlog	To color logging messages		
begins			
Argparse			
Pyqtgraph	Vs. matplotlib		
Pywebview	To get webpages quickly		
Psutil	Utilities		
watchdog	Utilities, to track changed files		
Ptpython	Utilities to hget hel for basic user interface		
hug	Web APIs similar to Flask and Django		
arrow	An enhanced datetime module		
parsedatetime	A datetime parser from		

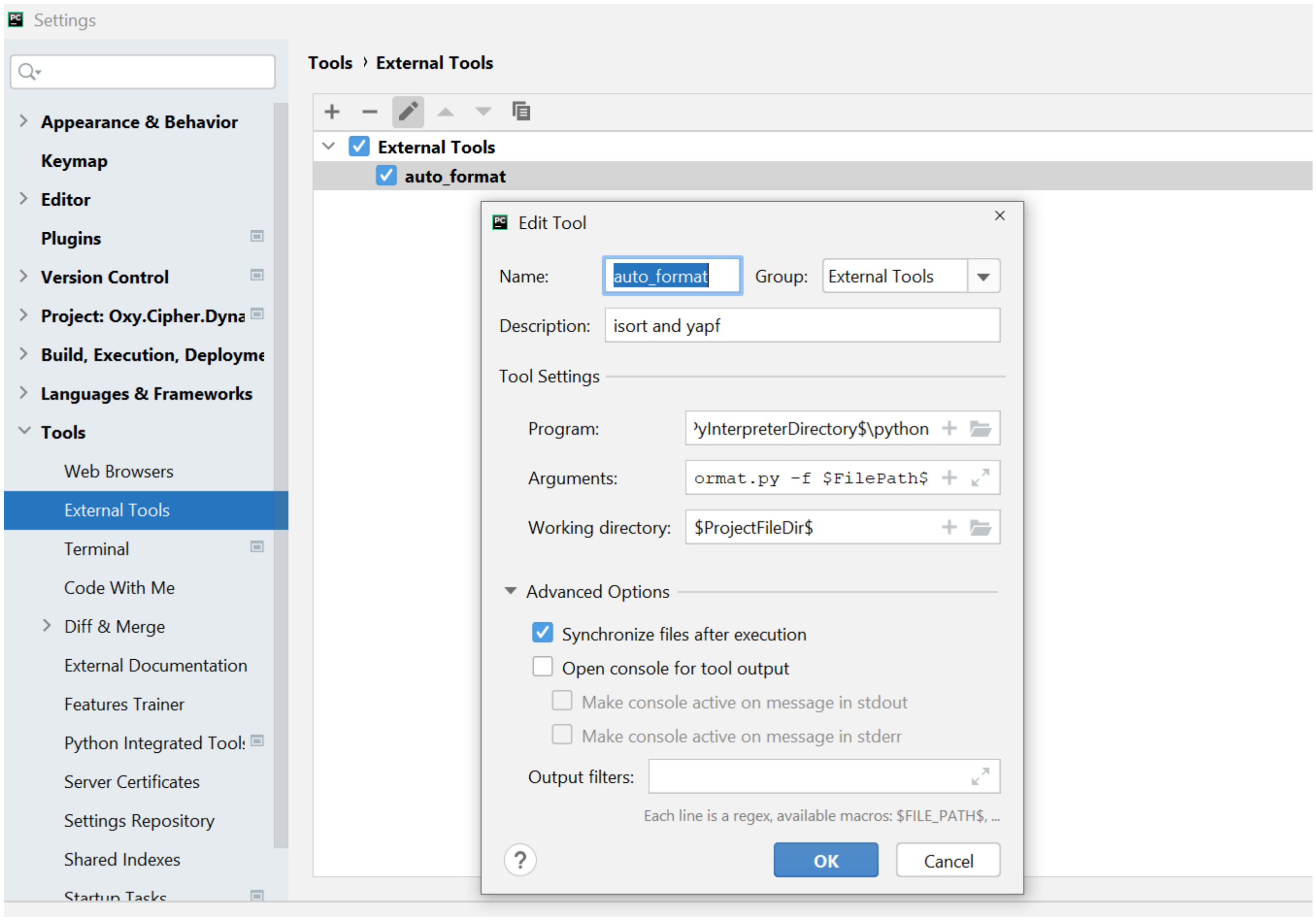
	text to datetime		
Boltons			
Boltons.cacheutils			
atexit	Standard library to do things at exit		
Boltons.fileutils			
Boltons.debugutils			
Boltons.strutils	Slugify Ordinalize Cardinalize Sigularize Pluralize Bytes2humans		
cython	To speed up ordinary slow loops or functions		

3.1 Style Guide : PEP8

https://google.github.io/styleguide/pyguide.html
https://www.python.org/dev/peps/pep-0008
https://pep8.org/

3.1.1 Autoformatters

yapf is a good starting choice. Setting up in Pycharm:



Settings

Tools › External Tools

External Tools
auto_format

Edit Tool

Name: auto_format
Group: External Tools
Description: isort and yapf

Tool Settings

Program: ›yInterpreterDirectory$\python
Arguments: `ormat.py -f $FilePath$`
Working directory: $ProjectFileDir$

Advanced Options

[✓] Synchronize files after execution
[ ] Open console for tool output
[ ] Make console active on message in stdout
[ ] Make console active on message in stderr

Output filters:

Each line is a regex, available macros: $FILE_PATH$, ...

Install packages and use blow:

Reference: https://developer.mantidproject.org/Standards/PythonStandards.html

3.1.2 Variable Naming

- Python recommends:
    - UpperCamelCase for class names
    - CAPITALIZED_WITH_UNDERSCORES for constants
    - lowercase_separated_by_underscores for other names.
- Variable naming:
    - upperCamelCase for variable names starting with lower capital for the first word.
    - This will help read the code by common man and easy to understand the logic
- Use 4 spaces per indentation level
- Maximum line length

    - Limit all lines to a maximum of 79 characters. This will make code comparison side by side easy to visualize.
    - For flowing long blocks of text with fewer structural restrictions (docstrings or comments), the line length should be limited to 72 characters.
- Blank lines
    - Surround top-level function and class definitions with two blank lines.
    - Method definitions inside a class are surrounded by a single blank line.
    - Extra blank lines may be used (sparingly) to separate groups of related functions. Blank lines may be omitted between a bunch of related one-liners (e.g. a set of implementations).
    - Use blank lines in functions, sparingly, to indicate logical sections.
- Imports
    - Imports should usually be on separate lines, e.g.:
        - Recommended way:
            - import os
            - import sys
        - Not recommended way:
            - import sys, os
    - However, it is okay to import multiple functions from a single file or module as below:
        - from subprocess import Popen, PIPE
        - Note the key difference between importing an entire module versus selective submodules. Such selective import will reduce the memory footprint for the program. This is specially advantageous when running large data, memory intensive and repetitive tasks
    - Imports should be grouped in the following order:
        - standard library imports
        - related third party imports
        - local application/library specific imports
    - Recommend to put a blank line between each group of imports.
- Block Comments
- Inline Comments:
    - Utilize in-line comments appropriately to convey a new user on what is happening in the code.
    - Typically there are not enough comments in a good program
- Document strings
- Designing for Inheritance
    - Class methods and instance variables should be public or non-public
    - If in doubt, choose non-public. It is easier to make non-public public than other wise

3.1.3 Indents and Loops

Some aspects of Python coding are given below:

- The length of all loops in Python is determined by the indentation level. Very important to note and understand this.
- Do not mix spaces and tabs in your program code for indentation, this produces bugs that are not easy to identify.

References:
https://www.python.org/dev/peps/pep-0008/
https://realpython.com/python-pep8/

3.2 Errors and Exceptions

https://docs.python.org/3/tutorial/errors.html

https://www.startertutorials.com/blog/exception-handling-python.html

3.2.1 Syntax Errors

3.2.2 Try and Except

Always utilize Try and Catch

- "Try" and "catch" are keywords that represent the handling of exceptions due to data or coding errors during program execution.
- A try block is the block of code in which exceptions occur.
- A catch block catches and handles try block exceptions.
- Other try implementations:
    - Try, except and raise
    - Try and finally

```
import traceback
 try:
   int('k')
 except:
   var = traceback.format_exc()
 print var
```

```
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
ValueError: invalid literal for int() with base 10: 'k'
```

https://docs.python.org/3/library/traceback.html
https://stackoverflow.com/questions/8238360/how-to-save-traceback-sys-exc-info-values-in-a-variable

https://www.youtube.com/watch?v=nqGhjLUhyDc

```
try:
        f = open('testfile', 'r')
        f.write('Test this file')
except IOError:
        print('Error: Could not find the file or write data')
else:
        print('No Error in Code')
finally:
        print('Executing finally clause.')
```

3.3 if name main

Always use the if name main function to help identify the main lines of code that are to be compulsorily executed.

if __name__ == '__main__':
https://stackoverflow.com/questions/419163/what-does-if-name-main-do

3.4 __init__py

Utilize init file. This will ensure development is easy to implement.

- The __init__.py files are required to make Python treat the directories as containing packages;
- This helps prevent directories with a common name, such as string, from unintentionally hiding valid modules that occur later (deeper) on the module search path.
- In the simplest case, __init__.py can just be an empty file, but it can also execute initialization code for the package or set the __all__ variable, described later.
- For further information, please refer to link below:

https://stackoverflow.com/questions/448271/what-is-init-py-for

3.5 Choice of IDE

- As beginner, Anaconda with Spyder IDE is good start.
- After a few projects, visual studio code (VS Code) or pycharm are good options to handle deep programming complexities of Python.

3.6 Configuration

A program requires some level of configuration to help provide variables to the source code.

Example config.yml File:

```
material:
    resistivity: 100
    resistance: 50
other:
    preprocessing_queue:
        - preprocessing.scale_and_center
        - preprocessing.dot_reduction
        - preprocessing.connect_lines
    use_anonymous: yes
```

Reading config.yml file. import config.yml using code below:

```
with open("config.yml", 'r') as ymlfile:
    cfg = yaml.load(ymlfile)

for section in cfg:
    print(section)

print("Print contents of Material section")
print(cfg['material'])
print("Print contents resistiity in Meterial SEction")
print(cfg['material']['resistivity'])

print("Print contents of Other section")
print(cfg['other'])
```

https://stackoverflow.com/questions/8525765/load-parameters-from-a-file-in-python
https://docs.python.org/3/library/configparser.html
https://martin-thoma.com/configuration-files-in-python/

3.7 Working with Arguments

Create ArgumentParser Object. This will hold all necessary information to parse the command line into data types
parser = argparse.ArgumentParser(description='Process some integers.')

add_argument() method will fill ArgumentParser with information about program arguments

Example 1:

```
import argparse

parser = argparse.ArgumentParser(description='Process some integers.')
# parser.add_argument('integers', metavar='N', type=int, nargs='+',
#                     help='an integer for the accumulator')
# parser.add_argument('--sum', dest='accumulate', action='store_const',
#                     const=sum, default=max,
#                     help='sum the integers (default: find the max)')
```

```
parser.add_argument('--foo', nargs=2)
parser.add_argument('bar', nargs=1)

parsed = parser.parse_args('c --foo a b'.split())
print(parsed )
```

Example 2:

```
import argparse

parser = argparse.ArgumentParser(description='Process some integers.')
# parser.add_argument('integers', metavar='N', type=int, nargs='+',
#                     help='an integer for the accumulator')
# parser.add_argument('--sum', dest='accumulate', action='store_const',
#                     const=sum, default=max,
#                     help='sum the integers (default: find the max)')
parser.add_argument('--database', nargs=3)

args = parser.parse_args('--database ID 3 5'.split())
print(args.database[0])
print(args.database[1])
print(args.database[2])
```

https://docs.python.org/3/library/argparse.html

"—" makes it a keyword argument
Can have keyword argument code

3.7.1 Output (Advanced)

Outputs can also be obtained from the python program into the system/OS environment to pass into another program. This features will need to be studies further.

https://stackoverflow.com/questions/26005583/return-value-of-x-os-system Return value in python OS System i.e. running in command mode
https://docs.python.org/2/library/commands.html

3.8 Logging

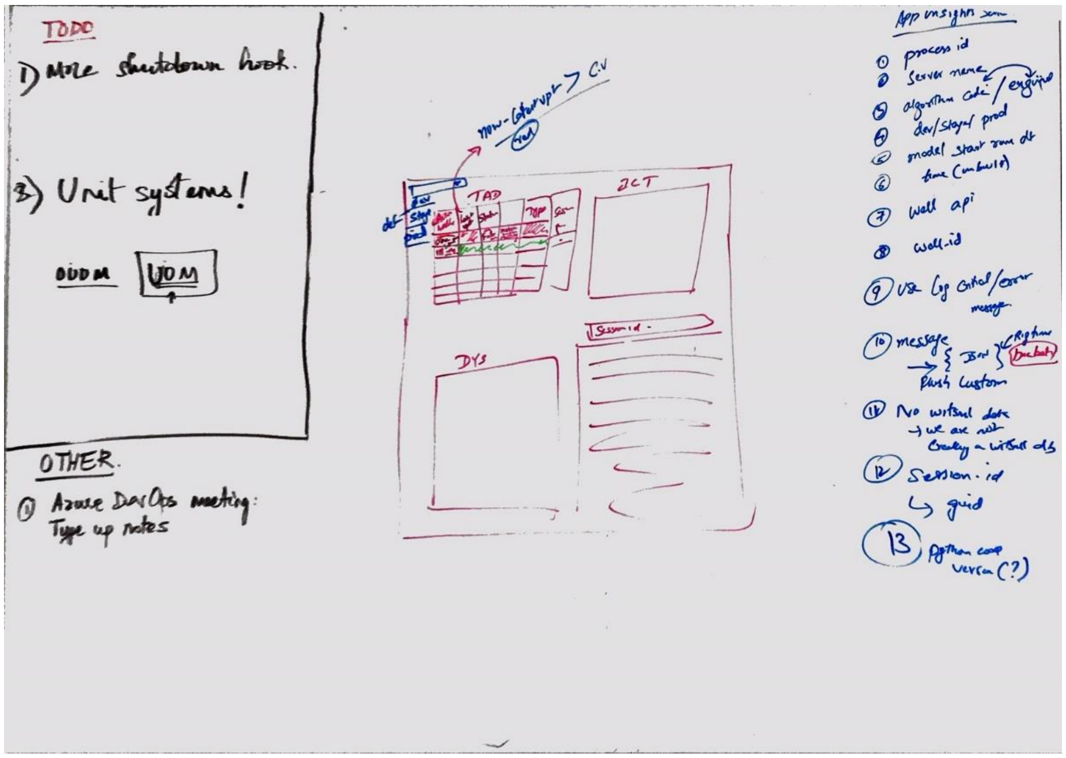
Level	Numeric Value	Function	Used to
CRITICAL	50	logging.critical()	Show a serious error, the program may be unable to continue running
ERROR	40	logging.error()	Show a more serious problem
WARNING	30	logging.warning()	Indicate something unexpected happened, or could happen
INFO	20	logging.info()	Confirm that things are working as expected
DEBUG	10	logging.debug()	Diagnose problems, show detailed information

- Logging levels can be set to increase the show messages to user as required
    - Debug level setting will show majority of the messages. Example 1 below shows a DEBUG level
    - Critical level setting will ONLY show critical messages. Example 2 below shows a CRITICAL level
- When a level is set the first time, this setting persists until the program is executed from start again. The level setting cannot be changed mid-way.
- Example 1: DEBUG (Level = 10). Code is given below
    - *import logging*
    - *# Set logging settings*
    - *logging.basicConfig(level=10)*
    - *logging.basicConfig(format='%(asctime)s %(message)s', datefmt='%m/%d/%Y %I:%M:%S %p')*
    -
    - *logging.info('This is Information. I told you so') # will not print anything*
    - *logging.error('This is error. I told you so') # will not print anything*
    - *logging.debug('This is Debug Mode. I told you so') # will not print anything*
    - Example 1 Output:
        - *INFO:root:This is Information. I told you so*
        - *ERROR:root:This is error. I told you so*
        - *DEBUG:root:This is Debug Mode. I told you so*
- Example 2: CRITICAL (Level = 50). Code is given below
    - *import logging*
    - *# Set logging settings*
    - *logging.basicConfig(level=50)*
    - *logging.basicConfig(format='%(asctime)s %(message)s', datefmt='%m/%d/%Y %I:%M:%S %p')*

-
- *logging.info('This is Information. I told you so') # will not print anything*
- *logging.error('This is error. I told you so') # will not print anything*
- *logging.debug('This is Debug Mode. I told you so') # will not print anything*
- *logging.critical('This is a critical message I wanted to show you.') # will not print anything*
- Output:
  - *CRITICAL:root:This is a critical message I wanted to show you.*

https://stackoverflow.com/questions/20780608/how-to-log-two-variables-values-in-a-single-logging-in-python Python format strings
https://fangpenlin.com/posts/2012/08/26/good-logging-practice-in-python/ Good logging practice in Python

3.8.1 Logging for Data Science



3.9 CI and CD

Jenkins is a good continuous integration (CI)/continuous deployment (CD) tool. Explore further.

http://www.alexconrad.org/2011/10/jenkins-and-python.html

3.10 Why Virtual Environments (Advanced)

Updating or downgrading python modules is easy to do but dependencies can be rendered incompatible.

Upgrading some dependencies will sometimes remove or uninstall other required basic dependencies. Therefore, it is recommended to work in virtual environments to avoid such problems. See example in Section 13.

If the programmer is required to work on multiple projects which require different modules then it is highly recommended to develop, run and deploy in each application in a dedicated virtual environment.

3.11 Pythonic Ways

https://www.toptal.com/python/python-parameterized-design-patterns
Decorator functions etc. etc.

3.12 Versioning

https://semver.org/ Sematic Versioning
https://github.com/python-versioneer/python-versioneer
https://github.com/python-versioneer/python-versioneer/blob/master/INSTALL.md

3.13 Jupyter Notebook

3.13.1 On premises

A very easy way to visualize what you are doing step by step. Can use markdown language to put notes and track what is being done. Great way to share learnings and work among peers.

3.13.2 In cloud

As of 2018, Google is providing a free notebook to run machine learning in the cloud. See below:
https://colab.research.google.com/ Google Colabs for GPUs

3.13.3 Case

CamelCase vs. LowerCase vs Pascal Case : Know the differences

3.14 General – Write Clean Code

3.15 References

S.No.	Link	Description
[7]	https://wiki.python.org/moin/HandlingExceptions	Handling Exceptions in Python
[8]	https://code.tutsplus.com/tutorials/professional-error-handling-with-python--cms-25950	Good guide to exceptions
[9]	https://www.digitalocean.com/community/tutorials/how-to-use-logging-in-python-3 https://docs.python.org/3/library/logging.html https://docs.python.org/3.6/howto/logging.html https://stackoverflow.com/questions/6386698/using-the-logging-python-class-to-write-to-a-file	Logging fundamentals and examples

4 LOOPS AND CONDITIONS

4.1 For Loop

Example1:

```
for x in range(0, 3):
        statement1
        statement2

for offsetCount, offset in enumerate(range(-10,12,2)):
```

- In this example, seaStateCount, counts no of sea states and saves value
- Enumetate function is used to count
- Range, the range between 2 numbers

Example2:

```
for seastateCount, (inputwaveType, inputwaveDirection, inputwaveheight, inputtimePeriod) in enumerate(zip(waveType, waveDirection, waveHeight, timePeriod)):
```

- In this example, seaStateCount, counts no of sea states and saves value
- Again, enumerate function is used to count
- Zip function is used when multiple strings or arrays handling

4.2 If else condition

Sample code of "IF ELSE" loop

```
if componentLength < 8:
        print("The length of component is less than 8")
else:
        print("The length of component is greater than 8")
```

4.3 Directory

Specifying a directory in code:
Note the double backslash (\\) below for directories:
'C:\\Users\\vamsee.achanta\\Dropbox (STA SEWOL)\\Engineering\\0119 Programming\\Py\\PDFReader'

Change directory: Os.chdir(*newDirectory*)

Find current directory: os.getcwd()

4.4 Working with Files

- To read a file in the default folder, no modules are required.
- If working with other files or create new directories in the computer, OS module is required to be imported

4.5 Print Using Variables

Passing variables and printing a text interspersed with values is very intuitive. All variable arrays can be stacked towards the end of the line. See below example

Print('The name is %s and the age is %f', (var_name,var_age))

4.6 References

https://www.youtube.com/watch?v=cIH6TjK0H5s&list=PLqaJe1KXgdP0UyO_JptIwn8GccfL9QANl

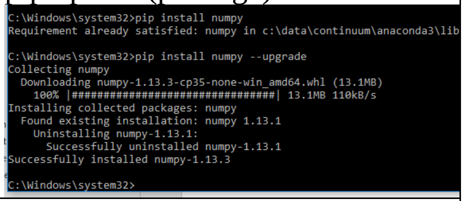
5 MODULES, FUNCTIONS AND CLASSES

- Modules are libraries of codes that are available as standard support through python support groups. Modules help perform generic yet clearly defined tasks easily
  - Can be internal or external in nature
  - Can be imported/loaded as required to perform a task
- Functions are block of code which can be parametrized with inputs and outputs
- Classes are fundamental building block of object oriented programming with properties, inheritance, encapsulation etc.

5.1 Modules or Libraries

Modules or libraries help simplify development work.

Help ("Modules") will show all available modules currently existing in Python on a machine. Some modules of interest are:
Math
Pydocs
Etc.
Etc.

Action	**Anaconda Prompt**	**All Python Prompts**
Install	conda install (package)	pip install (package) Pip install (whlfile)
Update	conda update (package) conda env update –f environment.yaml	pip update (package) 
version	Python -c "import pymssql; print(pymssql.__version__)" (or) Python3 -c "import pymssql; print(pymssql.__version__)" (in a docker container)	

```
C:\Windows\system32>pip install numpy
Requirement already satisfied: numpy in c:\data\continuum\anaconda3\lib

C:\Windows\system32>pip install numpy --upgrade
Collecting numpy
  Downloading numpy-1.13.3-cp35-none-win_amd64.whl (13.1MB)
    100% |################################| 13.1MB 110kB/s
Installing collected packages: numpy
  Found existing installation: numpy 1.13.1
    Uninstalling numpy-1.13.1:
      Successfully uninstalled numpy-1.13.1
Successfully installed numpy-1.13.3

C:\Windows\system32>
```

Table 5.1 –Modules Installation/Update Summary

How to object-orient maintain many virtual environments

https://stackoverflow.com/questions/56232524/should-the-conda-base-environment-be-kept-up-to-date/56240425

5.1.1 Module Summary

Name	Description	Key features
Pandas	Data structures & analysis	Manipulate tables and time series
NumPy	Base N-dimensional array package http://stackoverflow.com/questions/568962/how-do-i-create-an-empty-array-matrix-in-numpy https://docs.scipy.org/doc/numpy/reference/arrays.ndarray.html	Scientific Computing , multidimensional array object and matrices for fast operations Transpose : matrix_a.T Dot product: matrix_a.dot.matrix_b
Scipy	Fundamental library for scientific computing	
PyBrain	Machine Learning Library	
Matplotlib	Comprehensive 2D and 3D Plotting	Charts, histograms, bar, error, scatter plots, power spectra etc
IPython	Enhanced Interactive Console	
Sympy	Symbolic mathematics	
Pylab	Core parts of Numpy, Scipy, and Matplotlib	
Globe	Finds pathnames matching a specified pattern	https://docs.python.org/2/library/glob.html
OS	Operating system dependent functionality	Working with system information https://docs.python.org/2/library/os.html
Datatime	Manipulating dates and times	
Win32com	Windows application runner	
Xlrd	Xlsx, xls reader	
Xlwt	Xls writer	
Openpyxl	Xlsx, xls reader and writer	
Xlsxwriter	Xlsx writer	
Pygame	Drawing of 2d and 3d graphics; SDL multimedia library	
Scikit Fuzzy		Classification Fuzzy matrix
Numerical python		
plotly	Interactive visualizations	
bokeh	Interactive visualizations	

PyBrain		Blackbox optimizations Supervised classification
Pygame	https://www.pygame.org/docs/ Pygame is a Python wrapper module for the SDL multimedia library. It contains python functions and classes that will allow you to use SDL's support for playing cdroms, audio and video output, and keyboard, mouse and joystick input.	Engineering Graphics

5.1.2 Installation of Modules

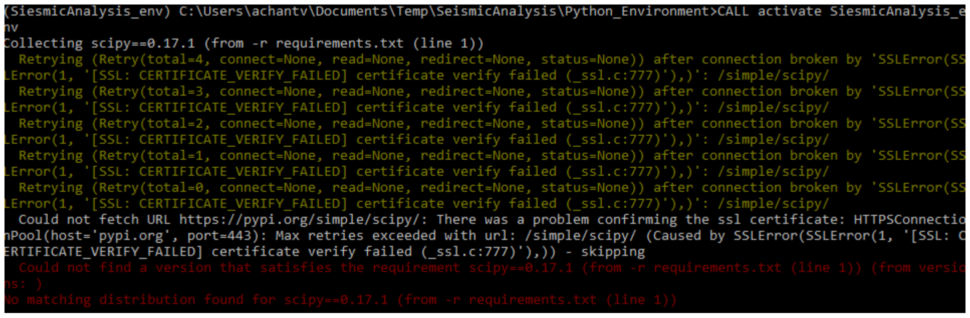
The installing of modules is described in this section. The installation and the key requirements are summarized below:

- It is recommended to work in virtual environments when installing additional required modules.
  - This is a must when installing non-standard modules and the main python installation can potentially breakdown due to incompatible dependences.
- Modules to be installed can be hosted in the following locations:
  - Packages hosted on official python websites. Some examples are:
    - Wheel files
    - Executed zipped files
- Codes can be installed in Python using python Command prompt:
  - Example Anaconda Prompt in Anaconda
  - Python prompt in other python programs
- The code in the command prompt, the directory must be changed to python installed location, otherwise the modules installation is not possible. Typically Anaconda prompt will find the python installation folder automatically.
- To specify a particular version of package, see instructions below:
  - conda install cx_Oracle=6.0b2
- If generic Microsoft DOS window or command prompt is used:
  - The pip or conda needs to be accessible via path variable
  - Alternatively, the python folder where this code is available should be accessible
- Some example packages and their installation is given below.

Package	**Installation Command**			**Comments**
	Command – Part 1		**Command – Part2**	
	In Anaconda Prompt	**Generic Python Prompt**		
Numpy	Conda	pip	install numpy	Arrays for data analysis
Scipy	Conda	pip	install scipy	Data analysis
Matplotlib	Conda	pip	install matplotlib	Plotting
Pandas			… pandas	For data analysis

jsonpickle			… jsonpickle	For json encoding and decoding

- SSL Certificate Errors during installation of packages
    - Note the "Could not fetch URL" in screenshot below:



```
(SiesmicAnalysis_env) C:\Users\achantv\Documents\Temp\SeismicAnalysis\Python_Environment>CALL activate SiesmicAnalysis_e
nv
Collecting scipy==0.17.1 (from -r requirements.txt (line 1))
  Retrying (Retry(total=4, connect=None, read=None, redirect=None, status=None)) after connection broken by 'SSLError(SS
LError(1, '[SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed (_ssl.c:777)'),)': /simple/scipy/
  Retrying (Retry(total=3, connect=None, read=None, redirect=None, status=None)) after connection broken by 'SSLError(SS
LError(1, '[SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed (_ssl.c:777)'),)': /simple/scipy/
  Retrying (Retry(total=2, connect=None, read=None, redirect=None, status=None)) after connection broken by 'SSLError(SS
LError(1, '[SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed (_ssl.c:777)'),)': /simple/scipy/
  Retrying (Retry(total=1, connect=None, read=None, redirect=None, status=None)) after connection broken by 'SSLError(SS
LError(1, '[SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed (_ssl.c:777)'),)': /simple/scipy/
  Retrying (Retry(total=0, connect=None, read=None, redirect=None, status=None)) after connection broken by 'SSLError(SS
LError(1, '[SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed (_ssl.c:777)'),)': /simple/scipy/
  Could not fetch URL https://pypi.org/simple/scipy/: There was a problem confirming the ssl certificate: HTTPSConnectio
nPool(host='pypi.org', port=443): Max retries exceeded with url: /simple/scipy/ (Caused by SSLError(SSLError(1, '[SSL: C
ERTIFICATE_VERIFY_FAILED] certificate verify failed (_ssl.c:777)'),)) - skipping
  Could not find a version that satisfies the requirement scipy==0.17.1 (from -r requirements.txt (line 1)) (from versio
ns: )
No matching distribution found for scipy==0.17.1 (from -r requirements.txt (line 1))
```

    -
    -
    - Typically occur due to security reasons. Can be from windows security restrictions or anti-virus requirements.
    - A corporate restriction may also cause this.
- To help bypass this:
    - Create conda.rc file and add below:
        - conda config --set ssl_verify false
    - Create pip.ini
    - Open %appdata% in explorer. This will take the relevant folder.
    - Create a pip folder and place pip.ini file with below contents
        - *[global]*
        - *trusted-host = pypi.python.org*
        - *pypi.org*
        - *files.pythonhosted.org*
- Install package through local repository
    - Conda install ciphercommon -c s/temp/cipher.zip
- Install package through local/other http server
    - conda install -c https://ohywnxu3-d/ ciphercommon

5.1.3 Importing modules

To utilize the modules in
matplotlib - import matplotlib
import os
PDF reader - import PyPDF2
Pygame - import pygame
Pylab - import pylab
import globe

5.1.4 Importing Modules - Advanced

- Use classes & functions defined in another file. A Python module is a file with the same name (plus the .py extension),
- There are three formats of the calling functions/file modules:
  import filename
  from filename import *
  from filename import className
- import filename
  • Everything in filename.py gets imported.
  • To refer to something in the file(function, class,..), append the text “filename.” to the front of its name:
  filename.className.method(“abc”)
  filename.myFunction()
- from filename import *
  • Everything in filename.py gets imported
  • To refer to anything in the module, just use its name. Everything in the module is now in the current namespace.
  • This import command can easily overwrite the definition of an existing function or variable.
  className.method(“abc”)
  myFunction()
- from filename import className
  • Only the item className in filename.py gets imported.
  • After importing className, you can just use it without a module prefix. It’s brought into the current namespace.
  • Overwrites the definition of this name if already defined in the current namespace.
  className.method(“abc”) ← imported
  myAnotherFunction() ← Not imported

5.1.5 Finding Version of Modules

import pip
print (pip.__version__)

- This may not work for all modules as there are modules that have NO attribute __version__.
- Version can also be found by issuing the installation command. The python available on computer is searched prior to installation.

Finding version of python module from command prompt where python is recognized.

- python -c "import Cython; print(Cython.__version__)"
- python -c "import scipy; print(scipy.__version__)"

5.1.6 Find Environment Modules

This section describes a way to find the modules installed in a given environment.

As project evolves, it will need to be hosted in another computer of shared with peers. The key commands to know the packages are:

- Pip freeze (or)
  - *(base) C:\Users\vamsee.achanta>pip freeze*
    - *alabaster==0.7.9*
    - *anaconda-client==1.6.0*
    - *anaconda-navigator==1.4.3*
    - *arrow==0.12.1*
    - *astroid==1.4.9*
    - *astropy==1.3*
    - *Babel==2.3.4*
    - *backports.shutil-get-terminal-size==1.0.0*
    - *beautifulsoup4==4.5.3*
    - *binaryornot==0.4.4*
    - *bitarray==0.8.1*
    - *bkcharts==0.2*
- Conda list
  - *(base) C:\Users\vamsee.achanta>conda list*

```
▪ # packages in environment at C:\Program Files\Anaconda3:
▪ #
▪ # Name                Version              Build  Channel
▪ _license              1.1                  py36_1
▪ alabaster             0.7.9                py36_0
▪ anaconda              custom               py36h363777c_0
▪ anaconda-client       1.6.0                py36_0
▪ anaconda-navigator    1.4.3                py36_0
▪ arrow                 0.12.1               <pip>
▪ astroid               1.4.9                py36_0
▪ astropy               1.3                  np111py36_0
▪ babel                 2.3.4                py36_0
▪ backports             1.0                  py36_0
▪ beautifulsoup4        4.5.3                py36_0
▪ binaryornot           0.4.4                <pip>
▪ bitarray              0.8.1                py36_1
▪ bkcharts              0.2                  py36_0
▪ blaze                 0.10.1               py36_0
▪ bokeh                 0.12.7               py36_0
▪ boto                  2.45.0               py36_0
▪ bottleneck            1.2.0                np111py36_0
▪ bzip2                 1.0.6                vc14_3  [vc14]
▪ certifi               2018.4.16            <pip>
▪ cffi                  1.9.1                py36_0
```

- Note the following:
  - If work is performed in base python environment, all key modules (along with versioning) will need to be listed.

- In base environment will list all python modules and will need to hand-picked
  - If working in a virtual environment, the trail modules unnecessary for final project will need to be removed from the list

5.1.7 Conda vs. PIP

https://jakevdp.github.io/blog/2016/08/25/conda-myths-and-misconceptions/
http://technicaldiscovery.blogspot.com/2013/12/why-i-promote-conda.html

Command Summary

	PIP	CONDA	
Ignore ssl errors	trustedsource	No verify SSL	

5.1.8 QR Codes

https://pypi.org/project/qrcode/
https://pypi.org/project/PyQRCode/

5.1.9 Custom Modules

Utilize cookiecutter to prepare template folder.

To install ciphercommon into an existing conda environment:
```
conda install ciphercommon -c https://ohywnxu3-d/
```

To update:
```
conda env update -f environment.yml
```

Installing a package through a channel
```
conda install jsonpickle -c conda-forge
```
(or) add the channel to the environment.yml file

5.2 External Functions

- The two methods to import and call external functions are given in this section
- If methods are located in another folder, it is recommended to utilize __init__ python packaging structure to manage files/folders. See section 3.4 for further details.

Method 1:
from externalFile import function1
function1() # call function

Method 2: This method is very suitable if externalFile has multiple functions (function1, function 2 etc.).
import pythonFunction
pythonFunction.function1 # Call function1
pythonFunction.function2 # Call function2

5.3 Inner Functions

https://realpython.com/inner-functions-what-are-they-good-for/

5.4 Classes

5.4.1 Class

5.4.2 Inner Class

Example 1: Simple Example

```
class Human:

  def __init__(self):
    self.name = 'Guido'
    self.head = self.Head()
    self.brain = self.Brain()

  class Head:
    def talk(self):
      return 'talking...'

  class Brain:
    def think(self):
      return 'thinking...'

if __name__ == '__main__':
  guido = Human()
  print guido.name
  print guido.head.talk()
  print guido.brain.think()
```

Example 2: Practical example to pass values and construct class objects

```
class AllOutPut:
  def __init__(self, df, du, CalculationMethod, CalculationMethodReason, ErrorMessage, displacement,
fluidLevel):
    self.DownholeCard = self.DownholeCard(df, du, CalculationMethod, CalculationMethodReason,
ErrorMessage)
    self.DownholeCardAnalysis = self.DownholeCardAnalysis(displacement, fluidLevel)

  class DownholeCard:
    def __init__(self, df, du, CalculationMethod, CalculationMethodReason, ErrorMessage):
      self.Load = df
      self.Position = du
      self.CalculationMethod = CalculationMethod
      self.CalculationMethodReason = CalculationMethodReason
      self.ErrorMessage = ErrorMessage

  class DownholeCardAnalysis:
    def __init__(self, displacement, fluidLevel):
      self.Displacement = displacement
      self.FluidLevel = fluidLevel
```

5.5 Functions and Summary

A summary of working with function is given in this section.

Name	Description	Example Code
Built-in	https://www.tutorialspoint.com/python/python_classes_objects.htm	Can be accessed by Dir (__builtins__)
User defined with multiple values	http://stackoverflow.com/questions/354883/how-do-you-return-multiple-values-in-python	*def f(x):* *y0 = x + 1* *y1 = x * 3* *y2 = y0 ** y3* *return (y0,y1,y2)*
Add folder with modules or files		Note: These commands work for Anaconda. *import sys* *sys.path.insert(0, "/path/to/your/package_or_module")* *sys.path.insert (0,"/BucklingDetection_SuckerRod")* *conda develop (folder)*

		BucklingDetection_SuckerRod
Lamda	Lamda function is a powerful which can save lot of programming effort and code. These are essentially inline functions that can be used to perform an operation and is very helpful for data science. For further information,	*downholeU = list(map(lambda x: x*0.0254, op.du))* *downholeF = list(map(lambda x: x*0.ŕ6*9.81, op.df))*

5.6 Managing modules (Administration)

Manually managing modules for deployment to another computer difficult. A starting guidance is given in link below.
https://stackoverflow.com/questions/18966564/pip-freeze-vs-pip-list Pip Freeze vs. Pip List

5.7 References

S.No.	**Link**	**Description**
[10]	https://pythonspot.com/inner-classes/	How to create inner classes
[11]	http://stackoverflow.com/questions/7701646/how-to-call-a-function-from-another-file	How to call external functions

6 PYTHON TESTING

6.1 Introduction

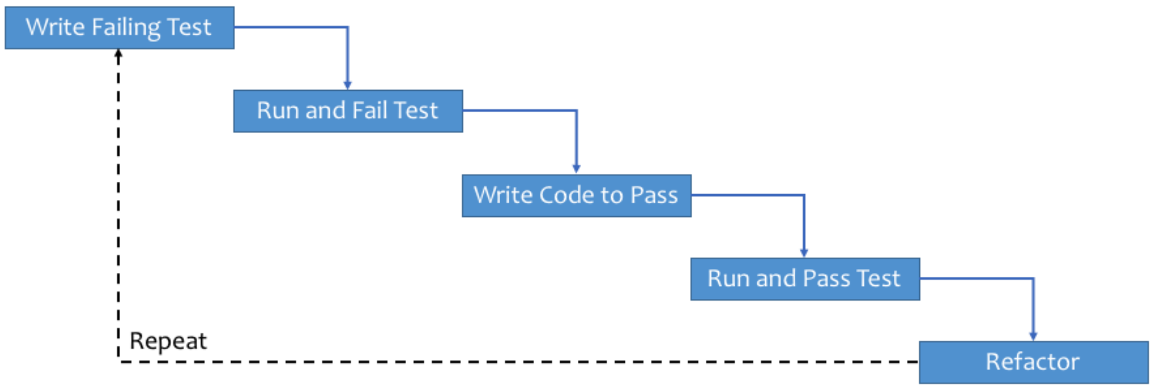


Figure 6.1 – Workflow of Test Driven Development (TDD)

6.2 Testing Guidance

Good tests:

- Are written with knowledge of test scope and type
- Are written with clear step by step procedures prior to programming
- A single test should focus on a single thing
- Use minimal setup, config or inputs as necessary
- Logic should be re-used from shared library
- Functional tests must be deterministic
- Leave no trace – use safe setup and cleanup

Automated testing is encouraged for the following reasons:

- Requirement to test the code immediately or in the near future
- Help check if new code changes breaks existing features
- Automated tests
    - can re-run easily and quickly
    - Scheduled to run anytime
- Other features that can be included are:
    - Show code coverage
    - Display test results in custom format
- Formalize the test process

- Automated tests typically don't make mistakes

6.3 Python Test Packages

Unitest is the standard python unit test framework. The guiding principles of the Unittest:

- Tests are written as a TestCase subclass
    - Contains
        - setUp,
        - tearDown
        - assertion
    - This forces to use class inheritance (good for starters)
    - This class does not support parametrized test cases.
- Methods prefixed with "test_" are identified as test cases
- Utilize unittest-xml-reporting to generate XML reports

Pytest is the fully matured python testing tool with the following features:

- Tests can be written as function or classes methods
- Provides following features
    - Setup
    - Cleanup
    - Temp dir
    - Args (tests can be parametrized)
    - Assert statements have advanced introspection
- Plugins can extend pytest

6.4 References

S.No.	Link	Description
[12]	https://medium.freecodecamp.org/learning-to-test-with-python-997ace2d8abe	Introduction to python testing
[13]	https://docs.python-guide.org/writing/tests/	

7 DATA ANALYSIS MODULES

The noteworthy data analysis modules are summarized on section 7.1 and explained in greater detail in the following subsections.

7.1 Summary

Module	**Purpose**
Matplotlib	Plotting
Pandas	Data science and handling large data. Dataframes, data reader etc.
Datetime	Handling date and time especially for data
OS	Handling system variables
FileCompare	For file compare on a high level file statistics
Sympy	For symbolic mathematical equation solver. Very basic.

7.2 Matplotlib

2D plot example:

```
import matplotlib.pyplot as pyplot
x = (20, 70, 50, 85)
y = (0.25, 0.35, 0.125, 0.458)
pyplot.plot(x, y)
pyplot.show()
pyplot.savefig(‘Trail.png’, dpi = 200)
```

3D plot example:

```
import xlrd
import matplotlib.pyplot as pyplot
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
wb = xlrd.open_workbook('Heave_RAO.xlsx')
sh1 = wb.sheet_by_name(u'Sheet1')

timePeriod = sh1.col_values(0)
amplitude = sh1.col_values(1)
waveHeading = sh1.col_values(2)

X=np.array(timePeriod)
Y=np.array(amplitude)
z=np.array(waveHeading)
Z = np.vstack((X, z))
fig = pyplot.figure()
```

```
ax = pyplot.axes(projection='3d')

ax.plot_surface(X, Z, Y, cmap=pyplot.cm.rainbow, rstride=1, cstride=1, linewidth=0, antialiased=False)
ax.set_xlabel('Time Period (s)')
ax.set_ylabel('Wave Heading')
ax.set_zlabel('Amplitude(m/m)')
heavePlot = pyplot.gcf()
pyplot.show()
pyplot.draw()
heavePlot.savefig("Heave.svg")
heavePlot.savefig("Heave.png", dpi=400)
```

Plotting dotted lines:
plt.plot(X_test, y_3, '--', color="blue", label="max_depth=3", linewidth=2)

Good plot practices:

- To keep all things pertaining to a plot element together, define everything element by element eg. Plot label for legend.
- Etc.

7.3 Pandas

- A cross platform distribution for data analysis and scientific computing.
- It offers data structures and operations for manipulating numerical tables and time series.
- The following code is used to read the data from CSV file.

```
import pandas as pd
import matplotlib.pyplot as plt
import pylab as pl

df = pd.read_csv("A1.csv",index_col='Time')  # pandas to open CV file and to read the file.
   for i in itertools.islice(df,0,50):

      df[['Differential Pressure','Fuel','Gas Flow','Heat Rate',
         'Power','Temperature']].plot()  #To Plot the data from CSV
      plt.title('Compressor Data') #Add Title to graph
      plt.legend((df),scatterpoints=1,loc='upper right',ncol=3,fontsize=8)  #To Lable the graph
      pl.xticks(rotation = 13)      #To  rotate the time date stamp plotting on x-axis

      plt.margins(1)
      plt.show()
```

Pandas library, pivot table, stack and unstack explained:
https://nikgrozev.com/2015/07/01/reshaping-in-pandas-pivot-pivot-table-stack-and-unstack-explained-with-pictures/

7.4 WebScraping

7.4.1 Beautiful Soup

https://www.datacamp.com/community/tutorials/web-scraping-using-python

7.4.2 Pandas Datareader

Pandas datareader is a remote access data module:

- Helps get data from standard websites easily. Examples include google finance, yahoo finance, etc.
- Reliable as long as the api design of the data provider is not drastically affected.

Example Code:

```
data_source = "morningstar"

import pandas_datareader.data as web
import datetime

start = datetime.datetime(2010, 1, 1)
end = datetime.datetime(2013, 1, 27)
df = web.DataReader('F', data_source, start, end)
```

7.5 datetime

import datetime

Date and time format - import datetime

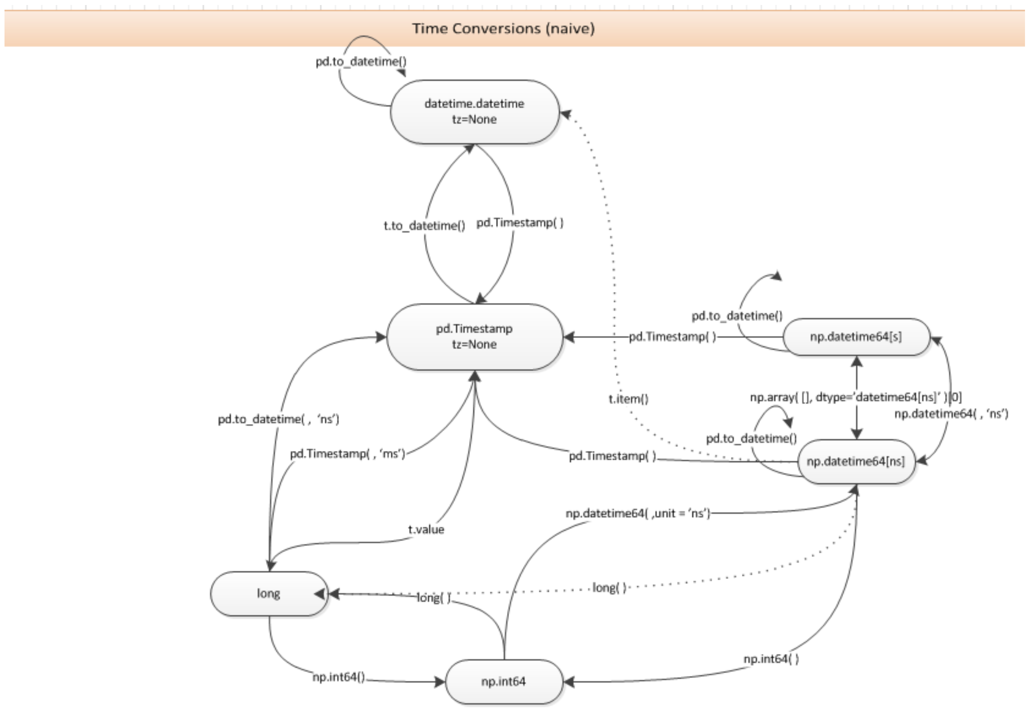


Figure 7.1 – Various Time Formats

7.5.1 Formatting DateTime

Date Format:

```
date_string = '2009-11-29 03:17 PM'
dateFormat = '%Y-%m-%d %I:%M %p'
my_date = datetime.datetime.strptime(date_string, dateFormat)

date_string2 = '6/28/2017 1:03:08 AM'
dateFormat2 = '%m/%d/%Y %I:%M:%S %p'
my_date2 = datetime.datetime.strptime(date_string2, dateFormat2)

    For current system time
print(datetime.datetime.now())

    For current system time and add timedelta to it.
print(datetime.datetime.now() + datetime.timedelta(days=-1))
```

Mathematical operations can be performed on 2 date formats as long as they can be recognized by Python.
A very picky thing to be careful about

To manipulate the date or datetime:

date.replace(year=self.year, month=self.month, day=self.day)

datetime.replace(year=self.year, month=self.month, day=self.day, hour=self.hour, minute=self.minute, second=self.second, microsecond=self.microsecond, tzinfo=self.tzinfo, * fold=0)

7.5.2 Getting only the date

7.5.3 Getting only the day

Reference:
https://stackoverflow.com/questions/13703720/converting-between-datetime-timestamp-and-datetime64
https://docs.python.org/3/library/datetime.html

Reference:

https://docs.python.org/2/library/datetime.html

7.5.4 Usecases

Usecase 1: UTC timezone is recommended for all data when the following happens:
- Assets or systems are in various locations in various timezones and countries
- Very important for real-time alarms and realtime data processing
- Any averages, maximum, etc. should also be evaluated using UTC/GMT as reference

7.6 OS

from os import path
resources_dir = path.join(path.dirname(__file__), 'lib/')

7.7 FileCompare

File compare can be handy when doing automated testing to understand file changes.

https://docs.python.org/2/library/filecmp.html FileCompare for automated tests.

7.8 sympy

A mathematical equation solver for simple cases outlined as follows:

- polynomial
- transcendental
- piecewise combinations of the above
- systems of linear and polynomial equations
- systems containing relational expressions

http://docs.sympy.org/latest/modules/solvers/solvers.html Python equation solver with following capabilities

7.9 Cookiecutter

Usage:

To install cookiecutter on Anaconda:
https://anaconda.org/conda-forge/cookiecutter
conda install -c conda-forge cookiecutter
Note that cookiecutter typically uses git to pull in the template packages. Installing git may be required. Ensure git path is added to the path environment variables.

CookieCutter is a standard way of creating templates for starting a project.
https://cookiecutter.readthedocs.io/en/latest/
http://cookiecutter.readthedocs.io/en/latest/first_steps.html

8 WORKING WITH DATA

Data reading, handling, manipulation and writing will be required in all projects. Based on the task, these functions need to be accomplished as required.

8.1 Basics

8.1.1 Data Type

Typical data types are:

- Int
- string
- Dictionary
- Array or list
- dataframes

- Checking if variable is of type:
    - *isinstance(<var>, int)*
- https://www.analyticsvidhya.com/blog/2017/03/read-commonly-used-formats-using-python/

8.2 Data Interface

Passing data from one function to another or one program to another needs to be designed carefully.

8.2.1 Yaml

https://stackoverflow.com/questions/55677397/why-does-pyyaml-5-1-raise-yamlloadwarning-when-the-default-loader-has-been-made

https://stackoverflow.com/questions/54803496/python-replacing-a-string-in-a-yaml-file

8.2.2 Yaml vs. JSON

https://stackoverflow.com/questions/1726802/what-is-the-difference-between-yaml-and-json-when-to-prefer-one-over-the-other

Remove alias in pyYaml file dump (eg: &id001)
http://signal0.com/2013/02/06/disabling_aliases_in_pyyaml.html

8.3 Data validation

Conversation got me curious about existing libraries that help with data validation. Have any of you worked with a library and found it useful? Here are a couple of examples that I think could be worth looking into. Any thoughts?

https://github.com/daveoncode/pyvaru

https://julien.danjou.info/python-schema-validation-voluptuous/

https://github.com/pyeve/cerberus

cerberus and voluptuous appear more popular. Looks like Pyvaru is not as lightweight as Cerberus and not popular either. Personally I am going to checkout Cerberus which seems very simple to use. There is a lot of value in storing validation rules as json and possibly storing them in DB.

https://www.yeahhub.com/7-best-python-libraries-validating-data/

8.4 Dictionaries

http://databio.org/posts/python_AttributeDict.html

https://stackoverflow.com/questions/14692690/access-nested-dictionary-items-via-a-list-of-keys How to access nested dictionary items with keys

8.5 Read ASCII File

An example script to read a given .dat ASCII file to csv file is given below:

- *import csv*
- *with open('20160812HR0015.DAT') as input_file:*
- *newLines = []   # indicates the data to be entered in it*
- *for line in input_file:*
- *newLine = [x.strip() for x in line.split(',')]*
- *if len(newLine) == 8 and newLine[0] and newLine[0]:*
- *newLines.append(newLine)   # append means adding data to new lines.*
-
- *with open('20160812HR0015.csv', 'w',newline='') as output_file:*

# The command newline=’’ is used to eliminate space between rows in csv file.

- *file_writer = csv.writer(output_file)*

# The command csv.writer enters the data in required file.

- *file_writer.writerows(newLines)*

8.6 Lists

List is the simplest of data structure. Lists are equivalent to Arrays. Lists are improved in Python 3. They are equivalent to the operations that can be performed in MatLab.

- For accessing List elements in python. If list name is AssetID, just follow below rule where i is an arbitrary variable definition.
  - *for i in AssetID:*
  - *print(i)*

https://docs.python.org/3/tutorial/introduction.html#lists

Checking if a list is NOT empty
```
if myList:
  print("List is NOT empty")
```

Checking if a list is empty
```
if not myList:
  print("List is empty")
```

8.6.1 Accessing data from 2 lists simultaneously

```
[a + b for a, b in zip(list1, list2)]
```

```
all_surveys_dictionary=all_surveys_md_filtered.to_dict('records')
all_surveys_attribute_dictionary=[AttributeDict(x)forxinall_surveys_dictionary]
all_surveys_as_positionpoint=[PositionPoint((x.srv_md,x.srv_inc,x.srv_azi),x.srv_code)forxinall_surveys_attribute_dictionary]
all_surveys_as_positionpoint_updated=[PositionPoint.update_calculated_kpi(PositionPoint((x.srv_md,x.srv_inc,x.srv_azi),x.srv_code),(x.srv_
north,x.srv_east,x.srv_tvd,x.srv_vs))forp,xinzip(all_surveys_as_p
```

8.6.2 Arrays

Python Form Arrays of Zero
```
   ServiceFactor = [0] * (len(jsonObject))
```

```
# format for initializing a matrix is = [[0 for x in range(columns)] for y in range(rows)]
```

Finding number of rows
```
print(len(exceptionIDMatrix))
```

Finding number of columns
```
print(len(exceptionIDMatrix[0]))
```

Numpy arrays may be serialized into json objects.
https://stackoverflow.com/questions/26646362/numpy-array-is-not-json-serializable Method to convert an array to JSON

8.7 Working with Excel

8.7.1 Module Installation : openpyxl

The module installation steps are given below:

- Download the package
- https://pypi.python.org/pypi/openpyxl
- Unzip into a folder.
- Change directory to the folder containing the "setup"
- Run the following:
    - python setup.py install
- No errors indicate successful installation

8.7.2 Read file

- For reading the csv or excel or text files, we need to define a variable for reading the file. Ex: df = csv.reader(open('filename.csv',index_col = 'column name'))

8.7.3 Create Workbook

https://openpyxl.readthedocs.io/en/stable/tutorial.html#create-a-workbook

http://xlsxwriter.readthedocs.io/working_with_pandas.html

8.7.4 Editing Workbook Data

Openpyxl will provide a method to:

- Open workbook
- Select sheet
- Edit values in a sheet

https://www.datacamp.com/community/tutorials/python-excel-tutorial

8.7.5 Write File

- For writing an CSV file we need to import CSV module.

Import csv

- Define a variable for creating a file and writing.
  File = open ('Path/filename.csv ','w')
- To write the data,
  Csvwrite = csv.writer(File)

8.7.6 Resources

http://www.python-excel.org/
http://www.simplistix.co.uk/presentations/python-excel.pdf
https://automatetheboringstuff.com/chapter12/
http://stackoverflow.com/questions/2141967/using-python-to-program-ms-office-macros
http://timgolden.me.uk/pywin32-docs/html/com/win32com/HTML/QuickStartServerCom.html
http://ironpython.net/ .NET framework IDE
https://www.datacamp.com/community/tutorials/power-spreadsheets-python

8.8 Databases

A summary of the python module used and typical instructions is given in Table 8.1.

Command\ Database	**MongoDB**	**Cassandra**	**MSSQL**	**Oracle SQL**	**MySQL**
Module	import pymongo from pymongo import Connection	import cql	Import pymssql import pyodbc	import cx_Oracle	import MySQLdb import peewee import mysqlconnector
Define Database					
Define Table/Collection					
Make connection	connection = Connection() connection = Connection('localhost', 27017) db = connection.testdb db = client['test-database']	con= cql.connect(host="127.0.0.1",port=9160, keyspace="testKS")	cnxn = pyodbc.connect("Driver={SQL Server Native Client 11.0};" "Server=server_name;" "Database=db_name;" "Trusted_Connection=yes;")	connection = cx_Oracle.connect('sde/sde@orcl')	db = MySQLdb.connect(host="localhost", user="john", passwd="megajonhy", db="jonhydb")
Define cursor parameter for cursor method	collection = db['test-collection']	cur=con.cursor() result=cur.execute("select * from TestCF")	cursor = cnxn.cursor()	cursor = connection.cursor()	cur = db.cursor()

Define query String	collection = db.testcollection			querystring = "select * from Parcels"	cur.execute("SELECT * FROM YOUR_TABLE_NAME")
Pass query			cursor.execute('SELECT * FROM Table')	cursor.execute(querystring)	
Write to Database	posts = db.posts post_id = posts.insert_one(post).inserted_id post_id ObjectId('...')				db.commit()
Print	for post in collection.find(): print post	result.fetchone() result.fetch()	for row in cursor: print('row = %r' % (row,))		for row in cur.fetchall(): print row[0]
Close Connection					db.close()

Table 8.1 – Working with Databases

How to reduce memory footprint if reading bigdata.
https://stackoverflow.com/questions/17861152/cursor-fetchall-vs-listcursor-in-python

8.8.1 cx_Oracle

- Finding Oracle Client Version
    - import cx_Oracle
    - print(cx_Oracle.clientversion())
-

https://oracle.github.io/python-cx_Oracle/
http://cx-oracle.readthedocs.io/en/latest/installation.html

https://github.com/oracle/python-cx_Oracle/issues/84 Possible fix.
Possible reasons: Visual studio may help run the cx_Oracle.
See below: https://oracle.github.io/odpi/doc/installation.html#windows

Sometimes restarting the computer installing Oracle client helps.

https://stackoverflow.com/questions/44728343/problems-using-pandas-read-sql-with-a-connection-using-cx-oracle-6-0b2 latest Oracle Wheel needs to be installed to get a working solution.

Common Problem:
ora-12154 tns could not resolve the connect identifier

https://docs.oracle.com/cd/B19306_01/server.102/b14219/net12150.htm
Possible Solution: Specify the TNS admin path in Python connection.
The server alias name is not resolved.

Cannot locate a 64-bit Oracle Client library
https://www.oracle.com/database/technologies/instant-client/downloads.html

8.8.2 Database Connection Errors and Exceptions

Errors:

_mssql.MSSQLDriverException: Connection to the database failed for an unknown reason.
pymssql.InterfaceError: Connection to the database failed for an unknown reason.

8.8.3 MSSQL good practices

Pretty much the road I’ve been going down also… with a smaller chunk size.

Interesting you were having problems with a newer version. I was having trouble getting it working on a pretty big dataframe and pretty much gave up in favor of dynamic SQL. Here’s my use case:

- Dataframe sources are text files (LAS).
- Each file can have different channels and some channels are even repeated even though the LAS standard is pretty clear that you’re not supposed to do that.
- Read_fwf is giving me 98% OK dataframes, but occasionally the widths seem to be a character off. I try to build some logic to clean out extraneous characters, like colons.
- My target table started off with a set of columns based on a sample of LAS files I read in. However, since it’s likely that other files will introduce channels that don’t exist in my table, I also wrote some code that dynamically alters the table to add the new column(s) as needed prior to an insert.

Once I have the dataframe cleaned up and column names assigned so that they are consistent with the target table, I iterate rows and execute an insert statement for each index. This is slow, but it’s working well.

Going with df.to_sql, I was getting a SQL Alchemy error preventing the command from even hitting the db, so it wasn't showing up in profiler. I set breakpoints in SQL Alchemy's db modules, but was going down too many rabbit holes.

Anyway, that's my long story. I'd really appreciate it if you'd be interested in looking at some of the ideas I put into the script. There are no doubt much better ways to get this done.

Thanks,
Mike

From: George, Dennis (Sparkhound) <Dennis_George@oxy.com>
Sent: Wednesday, June 12, 2019 3:11 PM
To: Achanta, Vamsee S <Vamsee_Achanta@oxy.com>; Suchoff, Michael <Michael_Suchoff@oxy.com>
Subject: RE: Data Library

Sorry another note: I am using pyodbc with "ODBC Driver 13 for SQL Server" 64-bit, and the 4.0.25 release of pyodbc. I had some issues with newer version of pyodbc, but those might be resolved in whatever is latest. I haven't tried to see.

From: George, Dennis (Sparkhound)
Sent: Wednesday, June 12, 2019 3:06 PM
To: Achanta, Vamsee S <Vamsee_Achanta@oxy.com>; Suchoff, Michael <Michael_Suchoff@oxy.com>
Subject: RE: Data Library

I am using dataframe.to_sql with pyodbc also, but with a chunksize of 1000. I also use a cursor execute hook to force a fast_executemany mode:

```
from sqlalchemy import create_engine
from urllib.parse import quote_plus

parameters = {
   "Driver": self.driver,
   "Server": self.serverName,
   "Database": self.databaseName,
   "Uid": self.userName,
   "Pwd": self.userPassword
```

```
    }
    connection_string = "mssql+pyodbc:///?odbc_connect=" +
quote_plus(";".join(map("=".join, parameters.items())))
    engine = create_engine(connection_string, encoding='utf-8', echo=False)

    @event.listens_for(engine, 'before_cursor_execute')
    def receive_before_cursor_execute(conn, cursor, statement, params, context,
executemany):
        if executemany:
            # performance hack to force underlying pyodbc engine to use fast_executemany
method
            cursor.fast_executemany = True

    dataframe.to_sql(tablename, engine)
```

From: Achanta, Vamsee S <Vamsee_Achanta@oxy.com>
Sent: Wednesday, June 12, 2019 10:04 AM
To: Suchoff, Michael <Michael_Suchoff@oxy.com>
Cc: George, Dennis (Sparkhound) <Dennis_George@oxy.com>
Subject: RE: Data Library

That is a Dennis (and KG) question.

He was leveraging sqlalchemy to get it to 1000 chuck size and he did comprehensive performance tests.

From: Suchoff, Michael <Michael_Suchoff@oxy.com>
Sent: Wednesday, June 12, 2019 10:02 AM
To: Achanta, Vamsee S <Vamsee_Achanta@oxy.com>
Subject: Data Library

Hi Vamsee,
What approach are you all taking now for dumping fairly big pandas dataframes into sql server? I have dataframes that are around 50K rows by 25 columns. I've been using dataframe.to_sql with pyodbc. Chunksize is set to 100. Performance isn't exactly fantastic. I had been iterating dataframes and executing inserts on the fly with similar performance.

8.9 Data and Units

Handling units is an age old problem in engineering and engineering data visualization. Dictionaries can handle data and units well.

For a high level architecture development, allow python to track the units inherently:

- Utilize @use_unit
    - See https://realpython.com/primer-on-python-decorators/
- For unit conversions, utilize pint
    -

References:
https://pypi.python.org/pypi/numericalunits
https://pint.readthedocs.io/en/latest/
https://stackoverflow.com/questions/2125076/unit-conversion-in-python
https://www.youtube.com/watch?v=6gjYvgIpDo8 A high school tutorial example of pint

8.10 DataFrames

DataFrames are easily managed.

8.10.1 Simple Operations

- Getting a data value by matrix index (m by n) method. To get data from 2nd row and 6th column
    - surveys_df.iloc[2,6]
- to Select a (or multiple) rowNumber:
    - df.loc[[rowNumber]]
    - df.loc[[rowNumber1, rowNumber2]]
    - Note that the double brackets to required to select a row number (or) multiple row numbers as a new dataframe.
- to Select a (or multiple) columnName:
    - df.loc[[rowNumber],[columnName]]
    - df.loc[[rowNumber1, rowNumber2], [columnName1, columnName2]]
- Finding if dataframe is empty
    - if df.empty:
- Finding if dataframe is not empty
    - if not df.empty:
- Filtering a dataFrame
    - 1 Condition:
        - *surveys_df[surveys_df.year == 2002]*
    - 2 Conditions
        - *temp_df = df[(df.TableName == table_name) & (df.StatisticsClassId == 2)].copy()*

  - Filtering and setcopy warning:
  - if linking to previous dataframe is not required, ensure to use .copy() at the end of an fitered or equated dataframe
  - https://www.dataquest.io/blog/settingwithcopywarning/
- To get array of dataframe index
  - df.index.values()
  - https://stackoverflow.com/questions/17241004/pandas-how-to-get-the-data-frame-index-as-an-array
- References
  - https://www.shanelynn.ie/select-pandas-dataframe-rows-and-columns-using-iloc-loc-and-ix/
  - https://pythonhow.com/accessing-dataframe-columns-rows-and-cells/

8.10.2 Assigning Values

8.10.2.1 use .iloc

df.iloc[row_number, df.columns.get_loc('COL_NAME')] = x

.ix is superseded (example: *df.ix[df_row_index, 'NominalWeight'] = 16*)

Assign after filtering a dataframe
https://stackoverflow.com/questions/38876816/change-value-of-a-dataframe-column-based-on-a-filter

df.iloc[df ['fe_filename'] == file_name, 'RunStatus']='SimulationStopped'

8.10.2.2 Mask

df['RunStatus'].mask(df['fe_filename'] == file_name, RunStatus, inplace=True)

This is giving a warning

Explore more to avoid this warning as it masks other errors:
https://kanoki.org/2019/07/17/pandas-how-to-replace-values-based-on-conditions/

https://www.dataquest.io/blog/settingwithcopywarning/

Can void warning and reset it using the following commands (Still not working):
pd.set_option('mode.chained_assignment', None)
df['RunStatus'].mask(df['fe_filename'] == file_name, RunStatus, inplace=True)

pd.reset_option('mode.chained_assignment')

More info:
https://stackoverflow.com/questions/20625582/how-to-deal-with-settingwithcopywarning-in-pandas

8.10.2.3 Using Lamda

```
df['NPTPercent_clean'] = df['NPTPercent'].apply(lambda x: 100 if x > 100 else
(0 if x < 0 else x))
```

Working Solution:

Common Class:

```
class PandasChainedAssignent:
    def __init__(self, chained=None):
        acceptable = [None, 'warn', 'raise']
        assert chained in acceptable, "chained must be in " + str(acceptable)
        self.swcw = chained

    def __enter__(self):
        import pandas as pd
        self.saved_swcw = pd.options.mode.chained_assignment
        pd.options.mode.chained_assignment = self.swcw
        return self

    def __exit__(self, *args):
        import pandas as pd
        pd.options.mode.chained_assignment = self.saved_swcw
```

Usage:

```
  from common.data import PandasChainedAssignent
  with PandasChainedAssignent():
        Perform Pandas chained Operations
```

8.10.3 Advanced Operations (Filter)

- When filtering a dataframe into a smaller dataframe, the index is NOT reset and can cause difficulty to access values. Reset of index is sometimes required.
- Reset the index of a dataframe after filtering to avoid value access errors. See example below:

```
PIDataValuesSBShutdownTimeDF = PIDataValuesDF[
```

PIDataValuesDF.AttributeName == 'SBShutdownTime']
PIDataValuesSBShutdownTimeDF.reset_index(inplace = True, drop=True)

- Filter by multiple values
    - *column_list = [1, 2, 3]*
    - *df_filtered = self.df[self.df.API12.isin(column_list)]*
- Writing Dataframe to csv
    - *surfaceCardDF.to_csv('surfaceCardDF.csv')*
- Drive the dataframe
- df.reset_index().values.ravel().view(dtype=[('index', int), ('A', float), ('B', float), ('C', float)])

8.10.4 Construct Arrays in to Matrix

This is a step before constructing a data frame from multiple arrays

taperMatrix = np.stack((Taper_Sequence, invOD,Number_of_Joints,Joint_Length, ServiceFactor, Dev, Drag, Upstroke_damping_factor, Downstroke_damping_factor, Rod_Guide, Strokes_Per_Minute, Modulus_of_Elasticity), axis=-1)

8.10.5 DataFrame To JSON Format

Saving a dataframe:
df.to_json(orient = ‘index’)

Saving a row
Save a DF row as a unique .jSON file
https://stackoverflow.com/questions/36051134/pandas-row-to-json
```
for i in df.index:
    df.loc[i].to_json("row{}.json".format(i))
```

Convert a row into json format and pass on to a variable.
```
for i in df1.index:
    result = df1.loc[i].to_json(orient = 'index')
```

8.10.6 Transpose

https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.transpose.html
- df.T (working)
- df.transpose (Not working and needs further tinkering?)

8.10.7 Interpolation

- Original DataFrame:
  - *MD Azimuth Inclination*
  - *0 -17.5 358.809540 0.00*
  - *1 82.5 293.579529 0.29*
  - *2 182.5 350.269531 0.14*

- Code to input zero:
  - *surveyDF.loc[-1, 'MD'] = 0*
  - *surveyDF = surveyDF.sort_values('MD').reset_index(drop=True)*
- Interpolate using index as 'MD':
  - *surveyDF = surveyDF.set_index('MD').squeeze()*
  - *surveyDF = surveyDF.interpolate(method ='index')*
- Reset the “MD” as column
  - *surveyDF.reset_index(level=0, inplace=True)*
- New dataframe:
  - *0 -17.5 358.809540 0.00*
  - *1 0 347.394288 0.05075*
  - *2 82.5 293.579529 0.29*
  - *3 182.5 350.269531 0.14*
- References:
  - https://stackoverflow.com/questions/41854578/using-pandas-dataframe-interpolate-to-add-rows-to-dataframe dataframe interploation
  - https://chrisalbon.com/python/data_wrangling/pandas_missing_data/

8.10.8 Replace Value

Replace a value in dataFrame:

- Utilize lamda function to run through all rows. Best way to replace values.
- Example lamda is below
  - *mask = DF.applymap(lambda x: x is None)*
  - *cols = DF.columns[(mask).any()]*
  - *for col in DF[cols]:*
  - *DF.loc[mask[col], col] = ''*
  - *logging.debug("Replaced None(s) with empty string, " ")*
- To fill nans with a value, below function is equally effective
  - pandas.DataFrame.fillna
- Following function is not effective in replacing values efficiently:
  - pandas.DataFrame.replace¶
  - May need further study.

8.10.9 Reshape

https://stackoverflow.com/questions/42928911/reshape-a-pandas-dataframe Reshaping of data frames. to give lines of arrays etc.

8.10.10 Grouping

https://www.shanelynn.ie/summarising-aggregation-and-grouping-data-in-python-pandas/
https://chrisalbon.com/python/data_wrangling/pandas_apply_operations_to_groups/

8.10.11 Add Row Array to DataFrame

```
df.loc[len(df)] = [1,2,3]
```

8.10.12 Adding Column to DataFrame

Add a column to dataframe using an array
```
df['col'] = [1,2,3]
```

8.10.13 Merge (Similar to SQL Join)

https://datacarpentry.org/python-ecology-lesson/05-merging-data/

8.11 JSON

8.11.1 Json Module

json.dumps

- JSON module can be used to handle json objects
    - Json.dumps is typically used. See [19], [21], [22] for further information
    - An example code is below
    - *Json.dumps(class.__dict__)*
- Takes an object and produces a json string
- For complex objects, it is recommended to use jsonpickle.

Writing JSON file to a file in Python
```
with open('MSSQL1stRecord.json', 'w') as outfile:
    json.dump(jsonData, outfile, sort_keys = True, indent = 4,
            ensure_ascii = False)
```

Converting Data Frame to a json and writing 2 .JSON objects to a file
```
    jsonData1 = surfaceCardDF.to_json(orient = 'split')
    jsonData2 = taperDF.to_json(orient = 'records')
    with open('MSSQL1stRecord.json', 'w') as outfile:
        outfile.write(jsonData1)
        outfile.write(jsonData2)
```

8.11.2 Jsonpickle

Use JSONPickle to handle class data easily

The use of jsonpickle module is explained in this section:

- If an object consists of class and inner classes then simple JSON module will not be sufficient. use JSON pickle. See reference [20] for further information.
- An example code and reference are given below:

encoding JSON example

```
concatenatedOutput = AllOutPut(None, None, CalculationMethod, CalculationMethodReason, ErrorMessage, displacement, fluidLevel)
print(jsonpickle.encode(concatenatedOutput, unpicklable=False))


class AllOutPut:
  def __init__(self, df, du, CalculationMethod, CalculationMethodReason, ErrorMessage, displacement, fluidLevel):
    self.DownholeCard = self.DownholeCard(df, du, CalculationMethod, CalculationMethodReason, ErrorMessage)
    self.DownholeCardAnalysis = self.DownholeCardAnalysis(displacement, fluidLevel)

  class DownholeCard:
    def __init__(self, df, du, CalculationMethod, CalculationMethodReason, ErrorMessage):
      self.Load = df
      self.Position = du
      self.CalculationMethod = CalculationMethod
      self.CalculationMethodReason = CalculationMethodReason
      self.ErrorMessage = ErrorMessage

  class DownholeCardAnalysis:
    def __init__(self, displacement, fluidLevel):
      self.Displacement = displacement
      self.FluidLevel = fluidLevel
```

Example JSON with sort order

```
import jsonpickle

class OutputResultClass(object):
  def __init__(self, array1,array2,array3,density):
    self.array1= array1
    self.array2= array2
    self.array3= array3
    self.density = density


array2 = 30
array1 =50
```

```
array3 = 300

density = 1000


outputResult = OutputResultClass(array1,array2,array3,density)

#jsonpickle.set_encoder_options('simplejson', sort_keys=False)
jsonpickle.set_preferred_backend('simplejson')
jsonpickle.set_encoder_options('simplejson', sort_keys=True)
jsonpickle.set_encoder_options('json', sort_keys=True)
jsonpickle.set_encoder_options('demjson', sort_keys=True)
jsonObject_jsonpickle = jsonpickle.encode(outputResult, unpicklable=False)
print(jsonObject_jsonpickle)
```

Decoding JSON example:
TBA

8.11.3 Convert JSON into Objects

8.11.4 Common Errors

Common error: If you get backslash in the json file output. It means that the json operation is being applied again. Check the code thoroughly.

8.11.5 Large Data

Large data can be compressed and stored in database if required.
https://stackoverflow.com/questions/26753147/how-to-gzip-a-bytearray-in-python?noredirect=1&lq=1 Guess, one of the option could be to use ByteArray (inPython 3) and use zlib module to compress/decompress the array. But we still need to know whether 8 MB blob column will be sufficient even after compression or not.

8.11.6 Json Data to Python Object

Converting a python dictionary or JSON file into a python object for easy use with other programs. An easy way to parse json data into python objects will drastically reduce the data input/parse effort.

Bunch.bunchify in python 2.x
Munch.munchify in python 3.x

References:
https://pypi.org/project/bunch/

https://changelog.com/posts/bunch-lets-use-python-dict-like-object
https://pypi.python.org/pypi/bunch For python 2x
https://github.com/Infinidat/munch For python 3x
https://stackoverflow.com/questions/1305532/convert-python-dict-to-object

8.11.7 py-ubjson

https://github.com/Iotic-Labs/py-ubjson Another JSON conversion file

8.11.8 To Dataframes

- Example code below:
  - df = pd.DataFrame.from_dict(json_normalize(data), orient='columns')
- https://stackoverflow.com/questions/41168558/python-how-to-convert-json-file-to-dataframe/41168691
- https://github.com/vi3k6i5/pandas_basics/blob/master/1_a_create_a_dataframe_from_dictonary.ipynb

8.11.9 References

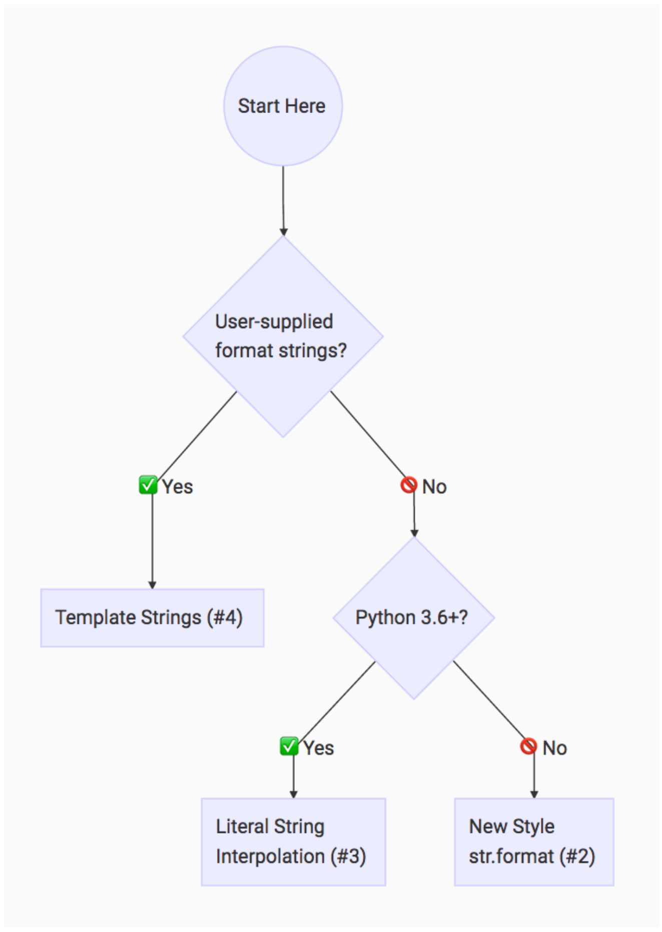
https://stackoverflow.com/questions/7127053/python-find-location-of-data-within-json-object-parse-the-corresponding-data
https://stackoverflow.com/questions/11241583/python-accessing-data-in-json-object
https://stackoverflow.com/questions/2835559/parsing-values-from-a-json-file

8.12 Strings

The various string formatting methods available are summarized below:

- Before .format Style (Superseded)
  - *'Hello, %s' % name*
  - *"Hello, Bob"*
- .format style
  - *'Hello, {0}, We are located in {1}'.format(name, place)*
  - *'Hello, Bob, We are located in Houston'*
- String interpolation or f-strings (Python 3.6+)
  - *f'Hello, {name}!'*
  - *'Hello, Bob!'*
- Template Strings (more secure for user interactive applications)
  - *from string import Template*
  - *t = Template('Hey, $name!')*

- *t.substitute(name=name)*
- *Output: 'Hey, Bob!'*


Start Here
User-supplied format strings?
Yes
No
Template Strings (#4)
Python 3.6+?
Yes
No
Literal String Interpolation (#3)
New Style str.format (#2)

https://realpython.com/python-string-formatting/
https://realpython.com/python-f-strings/

8.12.1 Regular expressions

https://stackoverflow.com/questions/4703390/how-to-extract-a-floating-number-from-a-string

https://docs.python.org/3/library/re.html

Specify a token and parse them as required. Tokens can also be directly specified in NLKT

8.12.2 NLKT (Natural Language Tool Kit)

8.12.3 Parsing

Parsimonious package is a good start for using and identifying regular expressions.

References:

https://www.youtube.com/watch?v=tCUdeLIj4hE&feature=youtu.be
https://stackoverflow.com/questions/47982949/how-to-parse-complex-text-files-using-python/47984221#47984221
https://stackoverflow.com/questions/47982949/how-to-parse-complex-text-files-using-python
https://www.vipinajayakumar.com/parsing-text-with-python/
https://dzone.com/articles/a-guide-to-parsing-algorithms-and-technology-part-8

8.13 Streams

Research Streams
Research light-weight packages available
Messaging with handshake?

https://docs.python.org/3/library/io.html

https://pypi.org/project/stream-python/
https://faust.readthedocs.io/en/latest/

https://pypi.org/project/ctds/
https://readthedocs.org/projects/ctds/
https://www.freetds.org/
https://github.com/zillow/ctds/blob/master/doc/install.rst

8.14 Numbers

8.14.1 Integers, No padding

- Old
    - '%d' % (42,)
- New
    - '{:d}'.format(42)

- Output
- 42

- Padding

- Old
  - '%4d' % (42,)
- New
  - '{:4d}'.format(42)
  - Output
  - 42

- Old
- '%f' % (3.141592653589793,)
- New
- '{:f}'.format(3.141592653589793)
- Output
- 3.141593

- Old
  - '%06.2f' % (3.141592653589793,)
- New
  - '{:06.2f}'.format(3.141592653589793)
  - Output
  - 003.14

- New
  - '{:.2E}'.format(206916857833.72)
  - Output
  - 2.07E+11

- https://pyformat.info/

8.15 Working with Word

Description	Value

Module	python-docx
import	docx

Table 8.2 – Working With Word Overview

8.15.1 Setup

Installation

- Download latest version from below link:
https://pypi.python.org/pypi/python-docx
- Install module as per instructions provided in Section 5.1.

8.15.2 References

http://python-docx.readthedocs.org/en/latest/user/documents.html
https://automatetheboringstuff.com/chapter13/

Dependencies
https://pypi.python.org/pypi/lxml/3.6.0#downloads

8.16 Working with PDF

TBA

8.17 Special Values

8.17.1 None

Equate the variable to none. See example code below
variable1 = None

Equivalent of C# null or nil

References:
https://stackoverflow.com/questions/25498810/what-is-closer-to-python-none-nil-or-null
https://stackoverflow.com/questions/22359737/how-to-assign-a-null-value-to-a-pointer-in-python
https://www.pythoncentral.io/python-null-equivalent-none/
https://www.ibm.com/support/knowledgecenter/en/SSFPJS_8.5.6/com.ibm.wbpm.wid.integ.doc/topics/rjsonnullunsempprops.html
https://stackoverflow.com/questions/21120999/representing-null-in-json
https://stackoverflow.com/questions/34157359/how-to-handle-variables-that-is-null-in-json

8.17.2 Pickle Files

Pickle helps in sending python data over TCP or storing program data so that it can continue where it left. This is a good way of restarting a program if it crashes in the midst of a long running duration.

http://www.diveintopython3.net/serializing.html Serializing unknown objects in Python (Pickle & JSON)
http://robotfantastic.org/serializing-python-data-to-json-some-edge-cases.html

Shelving

https://www.thoughtco.com/using-shelve-to-save-objects-2813668 Valid in only 1 session?

Pickling
https://ianlondon.github.io/blog/pickling-basics/

http://hoa-lsgtudhar.blogspot.com/2014/03/how-to-save-ctypes-objects-containing.html
http://hoa-lsgtudhar.blogspot.com/2014/03/how-to-save-ctypes-objects-containing.html

Pickling is not possible when objects are linked to other ctype objects. Will need a thorough linking process or mechanism. There is no automatic way of saving these objects in python.

8.17.3 Download Files

http://stackabuse.com/download-files-with-python/

8.17.4 Files or directories

Check if file or directory exists

- os.isdir()
- os.path.isfile()

8.18 References

S.No.	Link	Description
[14]	https://www.datacamp.com/community/tutorials/python-dictionary-tutorial	Data Format – Dictionaries
[15]	https://stackoverflow.com/questions/20383647/pandas-selecting-by-label-sometimes-return-series-sometimes-returns-dataframe	Inconsistent DataFrame behavior?

[16]	https://pypi.python.org/pypi/numericalunits	Numerical Units
[17]	https://stackoverflow.com/questions/32557920/what-are-type-hints-in-python-3-5	Python Type Hints.
[18]	https://www.youtube.com/watch?v=2wDvzy6Hgxg	Data Types explained in PyCon
[19]	https://stackoverflow.com/questions/26244323/convert-pandas-dataframe-to-json-as-element-of-larger-data-structure	Create simple JSON object
[20]	http://jsonpickle.github.io/	For enconding and decoding complex JSON objects
[21]	https://docs.python.org/3.2/library/json.html	Python JSON format library.
[22]	http://gowrishankarnath.com/read-write-json-python/	JSON read and write help

https://www.youtube.com/watch?v=u8g9scXeAcI Nice looping video
Tuple Unpacking help have functions such as enumerate(), sort(), reversed(), range(), zip()
https://www.youtube.com/watch?v=NdObDUbLjdg Python Debuggers (tracefunc())
https://www.youtube.com/watch?v=HTLu2DFOdTg Classes by Raymond Hittinger (Finish this)
https://www.youtube.com/watch?v=wf-BqAjZb8M Python best practices
http://docs.python-guide.org/en/latest/scenarios/xml/ XML
https://pypi.python.org/pypi/xmltodict XML
http://code.activestate.com/recipes/577267/ XML
http://code.activestate.com/recipes/577266-xml-to-python-data-structure-de-serialization/ XML

9 TYPICAL APPLICATIONS

9.1 Simple Scripts

- Typically saved as .py files
- Can be packaged into modules as well
- Execute on desktop or server
- Can be Execute on cloud
- Can be executed inline within databases as scripts

9.2 Desktop Application Programming

Desktop applications can also be developed using Python.
GUI programming with
- Buttons
- Textboxes

- Other calculations
- Menubar using “Menu” class
- Etc.

9.3 Web Platform

Set up a live platform or Django?
https://www.djangoproject.com/

Prepare set up for a general project
Prepare set up for a specific analysis project (eg. drilling riser analysis)

10 OUTPUT

10.1 Saving Data

CSV without header and footer
JSON format (with header and footer)
Understand the plotting tools before choosing data format and minimize parsing this after writing the data.

References
http://www.inquidia.com/news-and-info/hadoop-file-formats-its-not-just-csv-anymore

10.2 Printing Objects

10.2.1 Pretty Print Dictionary

```
import pprint
pp = pprint.PrettyPrinter(indent=4)
pp.pprint(mydict)
```

10.3 Accessing Objects (Advanced)

The last print on screen can also be accessed by external programs such as C# or webAPIs etc.

Example C# code:
methodUsed with type string

```
public class Downholecard
   {
       public float[] df { get; set; } //Load
       public float[] du { get; set; } //Position
   }
```

10.4 Logging

A decorator can be written to format dictionaries to logfile. See below.
https://pymotw.com/3/pprint/

11 VISUALIZATION

- If the plot data is from csv file, we need to give the heading in the column and data is taken for plotting.
  Ex: df [[‘column1’,’column2’,’column3’]].plt()
  plt.show ()

References:
https://plot.ly/python/
https://d3js.org/
Plotting DateTime on x-axis

- For Plotting Date Time Stamp on x-axis, the date time data from the csv file is read by the code and plotted on x-axis where the plotting is overlapped by each value.
- For avoiding the overlapping we use plt.xticks()
- Actually xticks is the interval for your x axis ticks or measurement, so as your level of measurement is in hours so it is better to tick for each hour in a week (i.e. 7 days * 24 hours) for the week's in the data set, and the second list comprehension put's the label's for that one week interval (week 0, week 1 .....),

11.1 General

https://python-graph-gallery.com/
Visualization Rules
https://www.data-to-viz.com/caveats.html

https://towardsdatascience.com/pythons-one-liner-graph-creation-library-with-animations-hans-rosling-style-f2cb50490396
https://towardsdatascience.com/@rahul_agarwal

https://www.linkedin.com/feed/update/urn:li:ugcPost:6535790071075958784/

https://www.dataquest.io/blog/python-data-visualization-libraries/

11.2 D3JS (External)

An introduction to D3JS
https://www.youtube.com/watch?v=8jvoTV54nXw

https://github.com/curran/screencasts/tree/gh-pages/introToD3

Setup the first chart

11.3 Subplots

Subplots are good feature to help interpret large amount of data from multiple sources in a serial and methodical manner. This subsection outlines the following details of subplotting:

- Install the various python packages used for handling the large data and its plotting
- Sub plotting method using python script.

11.4 Matplotlib

The sub plotting is a process of joining images as a subplot under the main image plotted.

Function	Command	Potential Errors

Import Extensions	import pandas as pb
	import numpy as np
	import matplotlib.pyplot as plt
	from matplotlib import style
	style.use("ggplot")	

Create random numbers of M by N matrix	Numpy.random.randn (M, N)	
Concatenate arrays		
Split arrays	Array_split hsplit vsplit dsplit	
Stack or join data	Stack Hstack Vstack dstack	
A good way to find bugs and test data	assert	

The file to be accessed and python script to be saved as.py file should save in same folder otherwise the below error occurs while running the script

- OSError: File b'20160811HR0017.csv does not exist

The python script starts with importing the extensions in the python shell as follows.

- import pandas as pb
- import numpy as np
- import matplotlib.pyplot as plt
- from matplotlib import style
- style.use("ggplot")

To plot the graph in various styles in matplotlib following command is used.

- from matplotlib import style
- style.use("ggplot")

The syntax to handle large amount of data using python pandas for any type of format files is

- H17(variable)=pb.read_csv('file name')

The following syntaxes is used to access the data from row and columns in the file read above.

- xsec=H17.ix[[“Row name”:,[‘column name”]]
- xsec=H17.ix[0:,['"sec"']]

# In above syntax we required data from column named “sec” only so, row is indicated as zero.

The minimum and maximum values from rows and columns are obtained by following command.

- Maxvalue=variable. max()
- Minvalue=variable. min()

To plot the main figure by defining data and subplots in it from matplotlib are shown below with the syntax.

- ax1.plot(xaxis(data),yaxis(data),color='r') # plots graph for given data in x and y axis.
- fig = plt.figure()   # indicates as main figure
- ax1(variable)=fig.add_subplot(511)

# Here (511) first number (5) indicates there are totally 5 subplots in figure and
Second number (1) indicates all subplots are plotted in single column
Third number (1) indicates the place of row

- plt.axis([xmin,xmax,ymin,ymax])

# To define range for x and y axis in a subplot above syntax is used.

The term patches are imported from matplotlib to write a name in a box with required colour and dimensions by defining it in a syntax. For, example to plot a legend in a subplot the syntax is used as below.

- import matplotlib.patches as mpatches
- box = dict(facecolor='Blue', pad=5, alpha=0.2)
- x5(Variable)= mpatches.Patch(color='c', label='legend name')
- plt.legend(handles=[x5])

To set title of figure, legends, labels of x and y axis with colour boxes in a subplot these syntaxes are used.

- ax1.set_title('Sub Plot', bbox=box)   #box indicates colour with dimensions
- ax1.set_xlabel('axis name',bbox=box)
- ax1.set_ylabel(‘axis name', bbox=box)

The tick labels () command is used in matplolib to share the common x axis or y axis between the subplots and make the shared axis visibility is false to hide the axis.

- plt.setp(ax1.get_xticklabels(), visible=False)

# Here, we share the x axis as common for all subplots by making it invisible.

To adjust the space between subplots the tight layout command is used as follows

- plt.tight_layout(pad=0.4, w_pad=0.5, h_pad=0.5)

The syntax for adding date to subplot by defining the axis range and position of the date is below.

- plt.annotate('DATE:31-08-2016', (0,-0.1), (0, -20), xycoords='axes fraction', textcoords='offset points', va='top')

The syntax for showing a subplot and saving the subplot with required is indicated below.

- plt.show()
- plt.savefig(‘subplot name',dpi=800)

ImportError: No module named 'PyQt4'. This is utilized by MatplotLib. Upgrading MatplotLib fixed the error.

TypeError: bad operand type for unary -: 'list'
Possible that some of the variables are strings and mathematical operations can not be performed.
Use int and float functions to convert string to number
http://www.jquery-az.com/convert-a-python-string-to-int-and-float-by-using-int-and-float-class/

`Study colors and Options to add linestypes further.`
https://stackoverflow.com/questions/8389636/creating-over-20-unique-legend-colors-using-matplotlib

11.5 Bokeh

Is a module that will help prepare crisp interactive plots for the web. This is detailed further in visualization guide.

11.6 Dash

https://plot.ly/products/dash/
https://medium.com/@plotlygraphs/introducing-dash-5ecf7191b503

11.7 Graphics

11.8 Pygal

https://www.pluralsight.com/guides/creating-interactive-charts-with-python-pygal

11.8.1 Pygame

11.8.2 Python Turtle Module (PTM)

https://docs.python.org/3.3/library/turtle.html?highlight=turtle

Can this be embedded into PDF as static image (yes)
Can this be embedded into interactive dynamic image to resize as necessary?

11.9 Interactive Charts in PPT

Web add in. Insert web pages only here
https://medium.com/@Infogram/add-interactive-infogram-charts-to-powerpoint-online-in-5-easy-steps-5b70d85e6ae

Need to create https pages?

Or utilize html code?
https://www.techwalla.com/articles/how-to-add-an-html-object-to-powerpoint

https://superuser.com/questions/1221880/how-to-embed-an-html-file-into-a-powerpoint-presentation

12 ADVANCED REQUIREMENTS

12.1 Profiling

cProfile and profile provide a deterministic profiling for python programs. A profile is a set of statistics that describe how often and how long various parts of the program executed. These stats can be formatted into reports via the pstats module.

https://docs.python.org/2/library/profile.html

How to Clock the process?
import cProfile
lIn = [random.random() for i in range(100000)]
cProfile.run('f1(lIn)')
cProfile.run('f2(lIn)')
cProfile.run('f3(lIn)')

PyChecker

Help in documentation. Limiting PycallGraph Levels:
https://pycallgraph.readthedocs.io/en/master/guide/filtering.html#maximum-depth

12.2 Scheduler or Chron

https://docs.python.org/3/library/sched.html
https://stackoverflow.com/questions/373335/how-do-i-get-a-cron-like-scheduler-in-python

12.3 VC Compiler

Can install a VC compiler in an environment to enable running models etc or other programs

- Install mingw32 in python using the following commands:
    -
- https://pystan.readthedocs.io/en/latest/windows.html
- https://media.readthedocs.org/pdf/pystan/latest/pystan.pdf
- Instructions to install in an environment:
    - conda install libpython==2.1
    - conda install -c msys2 m2w64-toolchain (Should specify channel?)
    - conda install m2w64-toolchain=5.3.0

12.4 Symbolic Math

More symbolic and natural to understand:
http://www.scipy-lectures.org/advanced/sympy.html

12.5 Polynomials

https://pythonhosted.org/pypol_/roots.html
An example function is given below:

```
import numpy as np

# ACTION : Convert to class function
def solvePolynomialEquation(coefficients):
    # coefficients = [3.2, 2, 1]
    solution =  np.roots(coefficients )
    return solution
```

12.6 Reporting

A combination of Latex and Python may be a solution.

12.7 Getters and Setters

Getter and Setters
https://www.python-course.eu/python3_properties.php
https://stackoverflow.com/questions/2627002/whats-the-pythonic-way-to-use-getters-and-setters

12.8 Matlab

```
import matlab.engine
eng = matlab.engine.start_matlab()
x, y, z = 3, 5, 8
r = eng.compute(x, y, z)
```

From <https://stackoverflow.com/questions/32393542/run-matlab-code-in-python>

12.9 Lamda or inline Functions

```
Lambda (or) inline functions:
```

12.10 Decorators

Boiler plate code:

Use cases for decorators:

- Timer functions
- Calling a function multiple times with different arguments etc.
- For web applications:
    - `Create template of 2 decorator functions. Will need for web authentications, checkings , use for logging, use for profiler, etc.`

https://realpython.com/primer-on-python-decorators/

12.11 SubProcess

subprocess
https://stackoverflow.com/questions/1685157/specify-working-directory-for-popen

https://realpython.com/python-exceptions/

12.12 args vs. kwargs

*args – Takes arguments without keywords
The args are ordered. It is defined as a list and immutable.

**kwargs – Takes arguments with keywords
The kwargs essentially removes the order dependency as the order is tracked. It is defined as a set and unordered.

12.13 Team Collaboration

The following package allows to Microsoft Teams using webhooks
https://pypi.org/project/pymsteams/

13 VIRTUAL ENVIRONMENTS

13.1 Introduction

Virtualization helps create isolated environments and help in transferring the application to another computer or scaling applications to a larger scale such as a cluster or cloud computers. Dockers is a system level virtualization while python virtual environment is program level virtualization. A high level rough estimate comparison is given in Table 13.1.

Feature	Virtual Machine	Docker/ Container	Program Environment
Level	System	System	Program
Installation Time	>5 mins	<5 mins	1-2 mins
Start-up Time	>5 mins	5-10 s	2 - 3 s
Configuration	OS, programs	OS, programs	program
Operating System		Linux preferred. Unstable in windows	

Table 13.1 – Virtualization of Environments

Virtual environments help isolate the development and (also) deployment of code. The virtual environments help with a stable and consistent environment.

A virtual environment helps in support and maintenance of reliable code when working on multiple projects:

- Updating or downgrading python modules is easy to do but dependencies can be rendered incompatible.
- Upgrading some dependencies will sometimes remove or uninstall other required basic dependencies. Therefore, it is recommended to work in virtual environments to avoid this problem. See example in Section
- If the programmer is required to work on multiple projects which require different modules then it is highly recommended to develop, run and deploy in each application in a dedicated virtual environment.

Virtual environments also help run legacy codes with ease:

- Legacy codes may not be supported in latest python installations. Therefore, it may be difficult to maintain a reliable code with 1 python installation.

Virtual Environments and why? https://www.python.org/dev/peps/pep-0405/

13.2 Python

Use below command to create and manage virtual environments.
venv

Conda has more refined way of managing environments and the associated dependencies as described in Section 13.4.

13.2.1 Python.exe vs pythonw.exe

- Python.exe is CLI
- Pythonw.exe is for:
  - launching GUI/no-UI-at all applications
  - standard streams (sys.stdin, sys.stdout, sys.stderr) are not available.
  - Unhandled exceptions cause script to abort sliently

https://stackoverflow.com/questions/9705982/pythonw-exe-or-python-exe

13.3 Pip

13.3.1 Create venv

- go to directory where the virtual environment needs to be installed. creates a virtual environment named "pipflask"
  - *python -m venv pipflask*
- Activate environment to ensure the packages are installed in appropriate environment
  - *activate pipflask*
- Use below command to install. Requirements full path required if in separate directory
  - *pip install -r K:\digitaltwinfeed\requirements.txt*
- to check all packages and associated versions.
  - *pip list*

13.4 Conda (Anaconda/Miniconda)

13.4.1 Environment.yml

- Define environment.yml file with an environment name and dependencies.
  - Example contents of environment.yml:

```
name: <environment_name>
channels:
  - defaults
  - https://ohywnxu3-d/
dependencies:
  - ciphercommon
  - <list any additional conda packages here (you can include =, or >= version numbers also)>
  - pip:
      - <list any additional pip packages here (you can include =, or >= version numbers also)>
```

  - Example with pip packages

```
name: API579
channels:
  - defaults
  - conda-forge
dependencies:
  - python=3.6
  - pyyaml
  - yaml
  - oyaml
  - matplotlib
  - scipy
  - pandas
  - python-docx
  - pip:
    - imgkit
    - openpyxl
    - xlrd
```

- Create Environment:

```
conda env create -f environment.yml
```

or use below and Program will search for default *environment.yml* in current folder and then create the environment.

```
conda env create
```

- Removing Environment:

```
conda env remove -n FOO
```

- Update Environment:

```
conda env update -n statistical_assessment –f statistical_assessment.yml
```

- Example 1 : environment.yml:
  - *name:* ***CipherCommon***
  - *channels:*
  - *- defaults*
  - *dependencies:*

- ○ *- cx_oracle=6.3.1=py35h2fa13f4_0*
- ○ *- numpy=1.14.5=py35h9fa60d3_0*
- ○ *- pandas=0.23.1=py35h830ac7b_0*
- ○ *- pyodbc=4.0.22=py35h6538335_0*
- ○ *- python=3.5.5=h0c2934d_2*
- ○ *- sqlalchemy=1.2.8=py35hfa6e2cd_0*

- Example 2 : environment.yml:
  - *name : **finance***
  - *channels:*
  - *- defaults*
  - *dependencies:*
  - *- python=3.5*
  - *- pandas=0.22.0*
  - *- pandas-datareader=0.6.0*

13.4.2 Detailed Explanation

Install Anaconda. Ensure conda is installed prior to creating the below virtual directory.

- Create a virtual box
  - *(echo y) | conda create -n dynaCard_env python=3.5*
- Enter the virtual box
  - *activate dynaCard_env*
- Install all required packages
  - *(echo y) | conda install numpy*
  - *(echo y) | conda install pandas*
  - *(echo y) | conda install pymssql*
  - *(echo y) | conda install spyder*
  - *(echo y) | conda install matplotlib*
- Run the python file
  - *python dynaCard.py --filename 30015201550000MibA1_lf_rpc1_Hprd6_Card_Items_Lastshutdown_SurfaceCardinput.json*
- Exit the environment
  - *deactivate dynaCard_env*
- ANother EXAMPLE
  - %windir%\system32\cmd.exe "/K" c:\data\Continuum\Anaconda3\Scripts\activate.bat c:\data\Continuum\Anaconda3
  - C:\ProgramData\Anaconda3\pythonw.exe C:\ProgramData\Anaconda3\cwp.py C:\ProgramData\Anaconda3
  - %windir%\system32\cmd.exe "/K" C:\ProgramData\Anaconda3\Scripts\activate.bat C:\ProgramData\Anaconda3
  - Working Line for Env
  - CALL %windir%\system32\cmd.exe "/K" c:\data\Continuum\Anaconda3\Scripts\activate.bat ESPExceptions_env

conda env list REM Display all environments in Conda

13.4.3 Building A Package

- Add following package to the base environment
  - Install conda-buiild
  - Conda install conda-build
- Utilize the below to build the package in current path. A specific path can also be specified.
  - Conda develop .
  - https://docs.conda.io/projects/conda-build/en/latest/user-guide/tutorials/build-pkgs.html

13.4.4 Features

Task	**Conda package and environment manager command**	**Pip package manager command**	**Virtualenv environment manager command**
Install a package Install a package from a channel	conda install $PACKAGE_NAME conda install $PACKAGE_NAME —-channel conda-forge	pip install $PACKAGE_NAME ??	X
Update a package	conda update --name $ENVIRONMENT_NAME $PACKAGE_NAME	pip install --upgrade $PACKAGE_NAME	X
Update package manager	conda update conda	Linux/macOS: pip install -U pip Win: python -m pip install -U pip	X
Uninstall a package	conda remove --name $ENVIRONMENT_NAME $PACKAGE_NAME	pip uninstall $PACKAGE_NAME	X
Create an environment	conda create --name $ENVIRONMENT_NAME python	X	cd $ENV_BASE_DIR; virtualenv $ENVIRONMENT_NAME
Activate an environment	conda activate $ENVIRONMENT_NAME *	X	source $ENV_BASE_DIR/$ENVIRONMENT_NAME/bin/activate
Deactivate an environment	conda deactivate	X	deactivate
Search available packages	conda search $SEARCH_TERM	pip search $SEARCH_TERM	X
Install package from specific source	conda install --channel $URL $PACKAGE_NAME	pip install --index-url $URL $PACKAGE_NAME	X
List installed packages	conda list --name $ENVIRONMENT_NAME	pip list	X
Create requirements file	conda list --export	pip freeze	X
List all environments	conda info --envs	X	Install virtualenv wrapper, then lsvirtualenv

Task	Conda package and environment manager command	Pip package manager command	Virtualenv environment manager command
Install other package manager	conda install pip	pip install conda	X
Install Python	conda install python=x.x	X	X
Update Python	conda update python *	X	X

https://towardsdatascience.com/a-guide-to-conda-environments-bc6180fc533
https://conda.io/projects/conda/en/latest/commands.html#conda-vs-pip-vs-virtualenv-commands
https://www.anaconda.com/understanding-conda-and-pip/

13.5 General Working

13.5.1 Set-up: Create

One time on a machine

13.5.2 Install Dependencies

One time on a machine

13.5.3 Running

Activate
Everytime a program is run, the environment is to be activated

13.5.4 Checking Environment and Dependencies

To list packages in an environment or in base environment:
pip freeze
conda list

https://medium.com/knerd/best-practices-for-python-dependency-management-cc8d1913db82
Python Dependency Management

https://stackoverflow.com/questions/1051254/check-if-python-package-is-installed Check1 for imported modules in Python
https://stackoverflow.com/questions/14050281/how-to-check-if-a-python-module-exists-without-importing-it/14050282#14050282 Check2 for imported modules in Python

A standard function to check that the program that it is in the right environment (else make it critical message)? Work in progress

13.5.5 Transitioning to Miniconda

There are no major differences in commands while using Miniconda and Anaconda

Install miniconda in a new (adjacent) folder and keep it ready for next project.
- It will have the latest conda (which is less error prone to work with)
- Light weight base environment so that it is difficult to break (compared the Anaconda).

If anaconda already exists:
- Environments exist and are working, continue using them as usual.
- If conda version is old and can not be updated. Transition over these old environments when project or package refresh is required.
    - Easier to refresh them in latest Miniconda version than old Anaconda version.
    - Latest Miniconda version will find right packages and install them for you with less hassles.

13.6 General Tips for Smoother Collaborative Python Dev (PyCharm)

(Assuming PyCharm is IDE)
- Start project using an existing, specific virtual environment, or create a new one for the project
- Reference only the required packages/libraries

13.6.1 Building identical conda environments¶ (for developer-to-developer sharing)

From: https://conda.io/docs/user-guide/tasks/manage-environments.html
You can use explicit specification files to build an identical conda environment on the same operating system platform, either on the same machine or on a different machine.

Use the Terminal or an Anaconda Prompt for the following steps.

1. Run `conda list --explicit` to produce a spec list such as:

```
# This file may be used to create an environment using:
# $ conda create --name <env> --file <this file>
# platform: osx-64
@EXPLICIT
https://repo.continuum.io/pkgs/free/osx-64/mkl-11.3.3-0.tar.bz2
https://repo.continuum.io/pkgs/free/osx-64/numpy-1.11.1-
py35_0.tar.bz2
https://repo.continuum.io/pkgs/free/osx-64/openssl-1.0.2h-1.tar.bz2
https://repo.continuum.io/pkgs/free/osx-64/pip-8.1.2-py35_0.tar.bz2
https://repo.continuum.io/pkgs/free/osx-64/python-3.5.2-0.tar.bz2
https://repo.continuum.io/pkgs/free/osx-64/readline-6.2-2.tar.bz2
https://repo.continuum.io/pkgs/free/osx-64/setuptools-25.1.6-
py35_0.tar.bz2
https://repo.continuum.io/pkgs/free/osx-64/sqlite-3.13.0-0.tar.bz2
https://repo.continuum.io/pkgs/free/osx-64/tk-8.5.18-0.tar.bz2
https://repo.continuum.io/pkgs/free/osx-64/wheel-0.29.0-
py35_0.tar.bz2
https://repo.continuum.io/pkgs/free/osx-64/xz-5.2.2-0.tar.bz2
https://repo.continuum.io/pkgs/free/osx-64/zlib-1.2.8-3.tar.bz2
```

2. To create this spec list as a file in the current working directory, run:

```
conda list --explicit > spec-file.txt
```

NOTE: You can use `spec-file.txt` as the filename or replace it with a filename of your choice.

An explicit spec file is not usually cross platform, and therefore has a comment at the top such as
`# platform: osx-64` showing the platform where it was created. This platform is the one where this spec file is known to work. On other platforms, the packages specified might not be available or dependencies might be missing for some of the key packages already in the spec.

To use the spec file to create an identical environment on the same machine or another machine:

```
conda create –name myenv –file spec-file.txt
```

To use the spec file to install its listed packages into an existing environment:
```
conda     install     --name     myenv     --file     spec-file.txt
```

Conda does not check architecture or dependencies when installing from a spec file. To ensure that the packages work correctly, make sure that the file was created from a working environment, and use it on the same architecture, operating system and platform, such as linux-64 or osx-64.

13.6.2 Activating an environment

To activate an environment:

- On Windows, in your Anaconda Prompt, run `activate myenv`

Running codes in both environments:

13.7 Dockers

Studying this further.

https://www.distelli.com/docs/tutorials/build-and-deploy-python-with-docker/ Python and Dockers
https://realpython.com/blog/python/python-virtual-environments-a-primer/
Dockers :
http://mherman.org/docker-workshop/#1
https://www.youtube.com/watch?v=YFl2mCHdv24
https://www.docker.com/what-docker

13.8 Virtual Environments and Notebooks

- Step 1: create a virtual environment
- - Attached is yml file that will create a virtual environment.
- - Place this yml file in a directory

- - Open Anaconda ommand prompt and go to directory with YML file
- - Create the virtual environment is below:
- conda env create -f dca_py35.yaml
- - Ensure all the steps are successful
-
- Step 2: Go to Anaconda Prompt and activate this environment using below command
- activate dca_py35
-
- Use pip to install ipykernel
- pip install --user ipykernel
-
- Step 3: Link the environment to Jupyter using the following command
- python -m ipykernel install --name=dca_py35
-
- The reference is this article: https://janakiev.com/blog/jupyter-virtual-envs/
- In the Jupyter kernels folder, the kernel.json file should have the environment added ot it now.

https://janakiev.com/blog/jupyter-virtual-envs/

13.9 IDEs and Virtual Environments

13.10 References

S.No.	Link	Description
[23]	https://www.datacamp.com/community/tutorials/python-dictionary-tutorial	Running codes in both environments
[24]	https://stackoverflow.com/questions/20383647/pandas-selecting-by-label-sometimes-return-series-sometimes-returns-dataframe	Running codes in both environments
[25]	https://pypi.python.org/pypi/numericalunits	Running codes in both environments

14 COMMON PYTHON PROGRAMS

14.1 Summary

Objective	Input	Output	Section
Project Schedule GANTT chart	.csv file with formatted project schedule data	.png or .svg for use in documents or html.	14.2
Data Heatmap Table	Format data into required format Prepare heatmap from formatted data		14.4
	s		

14.2 Project Schedule

Objective: Prepare a project schedule as a GANTT chart using resources (people and time)

Data Format:

- Data in .csv format
- The data format is the key:
    - Task,Duration,Start,Resource,Dependency
    - Task 1,5,12/31/2016,Joe,
    - Task 2,10,12/31/2016,Mark,Task 1
    - Task 3,1,12/31/2016,Kevin,Task 1
    - …
    - …
    - Task 29,5,12/31/2016,Mark,
    - Task 30,1,12/31/2016,Joe,Task 27

Dependencies:
projectscheduler
svgwrite

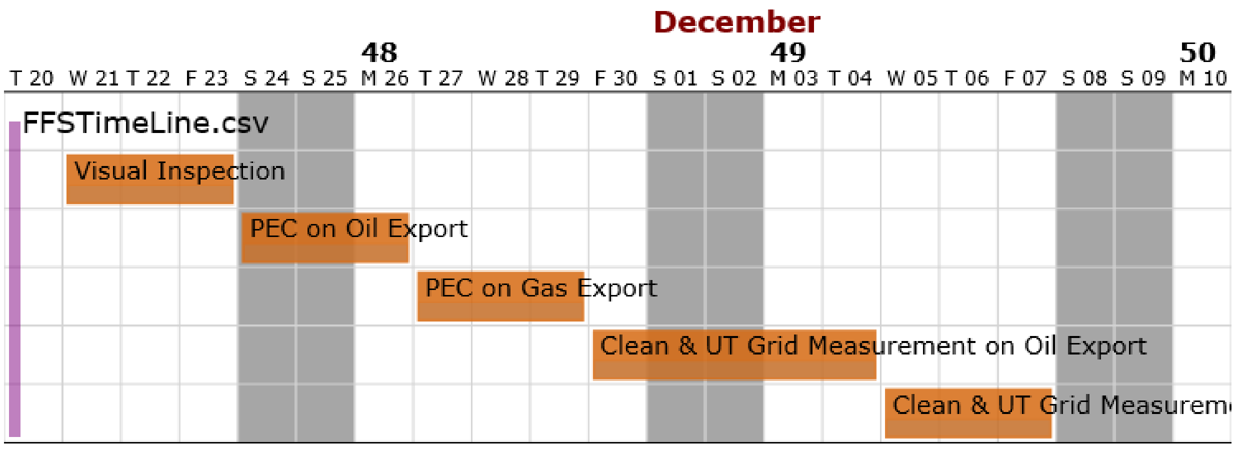
Run in Anaconda Command:
schedule input.csv output.svg

Example Code Location: 0119 Programming\010 ProjectSchedule\

References:
https://pypi.python.org/pypi/projectscheduler/0.5.6
https://raw.githubusercontent.com/traherom/simple-scheduler/master/example/example.csv

Bug: Can not plot historical dates into timeline. See below where the csv dates are for November

.csv
Task,Duration,Start,Resource,Dependency
Visual Inspection,2,10/24/2018,,
PEC on Oil Export,1,10/29/2018,,
PEC on Gas Export,2,10/30/2018,,
Clean & UT Grid Measurement on Oil Export,2,11/4/2018,,
Clean & UT Grid Measurement on Oil Export,2,11/6/2018,,



14.3 Timeline

Need a good timeline chart in Python

TimeLine Charts
https://matplotlib.org/gallery/lines_bars_and_markers/timeline.html
Using Dataframes
https://sukhbinder.wordpress.com/2018/09/11/timeline-in-python-with-matplotlib/
https://github.com/sukhbinder/timeline_in_python

14.4 HeatMap Table- Bokeh

Objective: Prepare a heatmap table to know the status of a machine on a hourly basis for a few days.

Data Format:

- The data format is the key for getting a good heatmap.
- Data needs to be prepared in the following format
-

Dependencies:
bokeh
math

Run in Anaconda Command:
Python heatMap_D10.py
Example: 0119 Programming\011 HeatMap\
Code to be finalized.

References:
http://bokeh.pydata.org/en/0.10.0/docs/gallery/cat_heatmap_chart.html
https://stackoverflow.com/questions/45180810/how-to-create-a-bokeh-heatmap-from-a-table-with-repeating-labels

15 DOCUMENTATION

Jupyter Notebooks can help create in-line documentation of codes for easy understanding.

15.1 Latex

https://github.com/andkret/Cookbook/blob/master/Data%20Engineering%20Cookbook.tex

15.2 Markdown

For application documentation

15.3 Jinja2

Key syntax for Jinja2:

- {% ... %} for Statements
- {{ ... }} for Expressions to print to the template output
- {# ... #} for Comments not included in the template output
- # ... ## for Line Statements

https://towardsdatascience.com/creating-pdf-reports-with-python-pdfkit-and-jinja2-templates-64a89158fa2d https://jinja.palletsprojects.com/en/2.11.x/tricks/

https://realpython.com/primer-on-jinja-templating/

https://jinja.palletsprojects.com/en/2.11.x/templates/

https://towardsdatascience.com/creating-pdf-reports-with-python-pdfkit-and-jinja2-templates-64a89158fa2d

https://dev.to/goyder/automatic-reporting-in-python---part-1-from-planning-to-hello-world-32n1

https://jinja.palletsprojects.com/en/2.11.x/faq/

15.4 JINJA2 to PDF

PDFKIT and headless-PDF packages need the set up of filepath for wkhtmltopdf

Simple examples

- pdfkit.from_url
- pdfkit.from_file
- pdfkit.from_string

https://micropyramid.com/blog/how-to-create-pdf-files-in-python-using-pdfkit/

https://pypi.org/project/pdfkit/
https://github.com/JazzCore/python-pdfkit
https://towardsdatascience.com/creating-pdf-reports-with-python-pdfkit-and-jinja2-templates-64a89158fa2d

setting up : wkhtmltopdf
https://stackoverflow.com/questions/27673870/cant-create-pdf-using-python-pdfkit-error-no-wkhtmltopdf-executable-found

```
wkhtmltopdf_path_ : C:\Program Files\wkhtmltopdf\bin\wkhtmltopdf.exe
wkhtmltopdf_path__ : C:\Users\Public\Data\Continuum\Anaconda3\envs\time_series\Lib\site-packages\wkhtmltopdf
wkhtmltopdf_path : common\exe\wkhtmltopdf.exe
```

https://wkhtmltopdf.org/downloads.html

headless-pdfkit (Not tested)
https://pypi.org/project/headless-pdfkit/

https://headless-pdfkit.readthedocs.io/en/latest/#examples

15.5 JINJA2 to Latex

No support or documentation available. Installation is also not working.
https://pypi.org/project/html2latex/

15.6 HTML

To view html code in an easy manner.
https://pypi.org/project/html2text/

16 MEMORY INTENSIVE APPLICATIONS

16.1 Increase CPU Usage

https://superuser.com/questions/679679/how-to-increase-pythons-cpu-usage

16.2 Threading (Parallelization)

https://stackoverflow.com/questions/39350300/run-multiple-servers-in-python-at-same-time-threading

https://www.tutorialspoint.com/python/python_multithreading.htm Python Threading
https://pymotw.com/2/threading/ Threading
https://www.youtube.com/watch?v=NwH0HvMI4EA Threading basics

http://www.instantfundas.com/2012/03/how-to-record-cpu-and-memory-usage-over.html Memory usage recording
https://stackoverflow.com/questions/69332/tracking-cpu-and-memory-usage-per-process Memory usage recording
https://pypi.org/project/memory_profiler/ Memory usage in python
https://pythonfiles.wordpress.com/2017/06/01/hunting-performance-in-python-code-part-3/ CPU usage.

16.3 Data Distribution

Python Dask or Apache Spark, See Big Data Technologies document.

17 FUNCTIONS AND CLASSES

17.1 Simple Function

A simple function

```
def say_hello():
        print('hello')

#Call the function
say_hello()
```

17.2 Lamda Function

```
#Normal function
def square(num):
        return num**2

#The above function can be replaced with
square = lambda num:num**2
print(square(5))


#Another Lamda Function Example
Addnum = lambda x,y:x+y
Addnum(2, 3)

#More examples
reverse = lambda a:a[::-1]
print(reverse('Hello'))
```

17.3 Class and Class Methods

The class is a fundamental building block in Python. A class is a code template for creating objects.
Objects will have attributes:

- Member variables
- Behaviors
- Classes can have functionalities which can be defined by setting attributes

  - Containers for data
  - Functions for attributes or data. These are called class methods
-
- It is a function-based cousin of def ().
- def is used to define a function while class is used to define a class.
- A Class is simply a logical grouping of data and functions (the latter of which are frequently referred to as "methods" when defined within a class).
- What do we mean by "logical grouping"? Well, a class can contain any data we'd like it to, and can have any functions (methods) attached to it.
- Rather than just throwing random things together under the name "class", we try to create classes where there is a logical connection between things. Many times, classes are based on objects in the real world (like Customer or Product). Other times, classes are based on concepts in our system, like HTTPRequest or Owner.
- Classes can be thought of as blueprints for creating objects. When I define a Customer class using the class keyword, I haven't actually created a customer. Instead, what I've created is a sort of instruction manual for constructing "customer" objects.

Example code:

```
class Customer(object):
    """A customer of ABC Bank with a checking account. Customers have the
    following properties:

    Attributes:
        name: A string representing the customer's name.
        balance: A float tracking the current balance of the customer's account.
    """

    def __init__(self, name, balance=0.0):
        """Return a Customer object whose name is *name* and starting
        balance is *balance*."""
        self.name = name
        self.balance = balance

def withdraw(self, amount):
        """Return the balance remaining after withdrawing *amount*
        dollars."""
        if amount > self.balance:
            raise RuntimeError('Amount greater than available balance.')
        self.balance -= amount
        return self.balance
```

- A method like withdraw defines the instructions for withdrawing money from some abstract customer's account. Calling jeff.withdraw(100.0) puts those instructions to use on the Jeff instance.
- When we say def withdraw(self, amount):, we're saying, "here's how you withdraw money from a Customer object (which we'll call self) and a dollar figure (which we'll call amount).
- self is the instance of the Customer that withdraw is being called on.
- When we call __init__, we're in the process of creating an object, Python allows us to extend the self pattern to when objects are constructed as well, even though it doesn't exactly fit. Just imagine that jeff = Customer('Jeff Knupp', 1000.0) is the same as calling jeff = Customer(jeff, 'Jeff Knupp', 1000.0); the jeff that's passed in is also made the result.
- This is why when we call __init__, we initialize objects by saying things like self.name = name. Remember, since self is the instance, this is equivalent to saying jeff.name = name, which is the same as jeff.name = 'Jeff Knupp

17.4 Instance Attributes and Methods

- A function defined in a class is called a "method". Methods have access to all the data contained on the instance of the object; they can access and modify anything previously set on self. Because they use self, they require an instance of the class in order to be used. For this reason, they're often referred to as "instance methods".
- Class attributes are attributes that are set at the class-level, as opposed to the instance-level. Normal attributes are introduced in the __init__ method, but some attributes of a class hold for all instances in all cases.

Classes and Objects:

Class is a blueprint. The below function essentially a key, value pairs are being defined usign class objects

```
Class LotteryPlayer:
        def __init__(self, name, numbers):
                self.name = "Rolf"
                self.numbers = (5,9,12,3,1,21)

        def total(self):
                return sum(self.numbers)

player_one = LotteryPlayer("Rolf", (5,9,12,3,1,21) )   (will get with default properties)
player_two = LotteryPlayer("John", (5,9,12,3,1,21) )   (will get with default properties)
player_two.numbers = (15,19,12,3,1,21)

print(player_one.name)
print(player_one.numbers)
print(player_one.total())    Methods can also be accessed using classes.
```

note that player_one != player_two. They are not the same and are tow different objects

17.5 Difference between Static and Class methods:

```
class Student:
        def __init__(self, name, school)
                self.name = name
                self. school = school
                self.marks = []

        def average(self):
                return sum(self.marks) / len(self.marks)

        def go_to_school(self):
                print("I'm going to {}." .format(self.school))

        def go_to_school_static(self): # Self is needed here but is NOT used. Static method
                print("I'm going to School.")

        @staticmethod
        def go_to_school_static():
                print("I'm going to School.")


        @classmethod
        def go_to_school_static(cls): # A class is passed here.
                print("I'm going to School.")



anna = Student("Anna", "MIT")
anna.marks.append(56)
anna.marks.append(71)
print(anna.average())
```

The reason for defining the Offshore pipe is because, we want to ensure all offshorepipes will have same consistent naming and properties when we want to access in VMStress, OrcaFlex analysis or other programs.

The main purpose of classes and functions is to achieve object oriented programming. Essentially create objects so we can re-use them again and again.

Class Attributes: Data
Class Methods: actions

17.5.1 Inheritance

- Class is the blueprint
- Any instance created from Class will be unique and can be distinguished from each other.
- Default attributes for the class or self can be defined at the top of the class.

  - If class is used, it is called class variables.
  - If self. is used, it is called instance variables
- Inheritance:
  - Utilizes Method resolution
  - Define a new class Developer based on Employee class as follows:
    - Class Developer(Employee):
    - Pass
  - Super().__init__(first, last, pay) (Generic)
  - Self.prog_lang = prog_lang

To understand the attributes of a class and class object:
Print(Student.__dict__)

To get more information on inheritance:
Print(help(Student))

https://stackoverflow.com/questions/9058305/getting-attributes-of

```
class Person:
        def __init__(self, first, last, age):
                self.firstname = first
                self.lastname = last
                self.age = age

        def __str__(self):
                return self.firstname + " " + self.lastname + ", " + str(self.age)

class Employee(Person):
        def __init__(self, first, last, age, staffnum):
                super().__init__(first, last, age)
                self.staffnumber = staffnum

        def __str__(self):
                return super().__str__() + ", " + self.staffnumber

x = Person("Marge", "Simpson", 36)
y = Employee("Homer", "Simpson", 28, "1007")

print(x)
print(y)
```

17.5.2 Decorators

A decorator function is a function called prior to calling another function

Example 1: Simple Decorator Function (2 levels) i.e. my_decorator() calls mainFunction()

```
import functools

def my_decorator(mainFunction):
        @functools.wraps(mainFunction)
        def function_that_runs_mainFunction():
                print("I am the decorator function")
                mainFunction()
                print("End of decorator function")
        return function_that_runs_mainFunction

@my_decorator
def mainFunction():
        print("I am the main function running inside the decorator")

mainFunction()
```

```
I am the decorator function
I am the main function running inside the decorator
End of decorator function
```

Example 2: Advanced Decorator Function (3 levels) with parameters

- 3 levels of functions.
- Very helpful to concise code in certain scenarios such as below:
    - Check if administrator to display certain pages

```
import functools

def decorator_with_arguments(number):
    def my_decorator(mainFunction):
        @functools.wraps(mainFunction)
        def function_that_runs_mainFunction(*args, **kwargs):
            print("I am the decorator function.")
            print("Perform further checks with 'number' argument")
            if number == 56:
                print("Not running the function")
            else:
                print("Running the function")
                mainFunction(*args, **kwargs)
            print("End of decorator function")
        return function_that_runs_mainFunction
    return my_decorator

@decorator_with_arguments(55)
def my_function_too(x,y):
        print(x+y)

my_function_too(57, 67)
```

18 JUPYTER NOTEBOOK

https://plot.ly/python/ipython-notebook-tutorial/
https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html

18.1 Jupyter Notebooks and Sharing:

http://jupyter-notebook.readthedocs.io/en/stable/public_server.html
https://www.reddit.com/r/IPython/comments/7l0bxl/how_can_i_host_a_jupyter_notebook_online_and/
http://amber-md.github.io/pytraj/latest/tutorials/remote_jupyter_notebook
https://hsaghir.github.io/data_science/jupyter-notebook-on-a-remote-machine-linux/
https://ricardodeazambuja.com/jupyter_notebooks/2017/02/10/Jupyter_notebook_remotelly/

https://shiny.rstudio.com/ Shiny

18.2 Add Conda environment to Notebook

Key steps:

https://medium.com/@nrk25693/how-to-add-your-conda-environment-to-your-jupyter-notebook-in-just-4-steps-abeab8b8d084

18.3 Troubleshooting Tips

18.3.1 Kernel died

Try 1: Start anaconda prompt as administrator – This did not work either…
https://stackoverflow.com/questions/49326164/jupyter-notebook-dead-kernel

Try 2: Install (or reinstall) following packages in a dedicated environment. (Successful)
ipykernel
ipython
jupyter_client
jupyter_core
traitlets
ipython_genutils

https://github.com/jupyter/notebook/issues/1892

19 REPORTING

https://stackabuse.com/how-to-send-emails-with-gmail-using-python/

https://realpython.com/python-send-email/

Sample Email:

```
import urllib3
from exchangelib import Credentials, Account, FileAttachment, Message
from exchangelib.protocol import BaseProtocol, NoVerifyHTTPAdapter


def disableWarnings():
    urllib3.disable_warnings()  # disable warnings about SSL errors
    urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)
    BaseProtocol.HTTP_ADAPTER_CLS = NoVerifyHTTPAdapter  # ignore SSL errors caused by company firewall


def sendEmail(subject, body, recipients, recipientsCC):
    disableWarnings()
    credentials = Credentials('server_name', 'mypassword')
    account = Account(primary_smtp_address='server@company.com', credentials=credentials, autodiscover=True)

    m = Message(account=account,
                subject=subject,
                body=body,
                to_recipients=recipients,
                cc_recipients=recipientsCC)

    m.send()

# This is the entry point into the code
if __name__ == "__main__":
    sendEmail("Apollo 13", "Houston, we have a problem.", ["achantav@gmail.com"])
```

20 API

SOAP vs REST vs GraphQL

https://realpython.com/python-api/

21 REFERENCES

http://matplotlib.org/1.3.0/faq/howto_faq.html#align-my-ylabels-across-multiple-subplots

http://stackoverflow.com/questions/6963035/pyplot-axes-labels-for-subplots

http://matplotlib.org/users/gridspec.html

http://pandas.pydata.org/pandas-docs/stable/text.html

http://pandas.pydata.org/pandas-docs/version/0.17.0/pandas.pdf

http://www.labri.fr/perso/nrougier/teaching/matplotlib/

https://www.youtube.com/watch?v=p8hle-ni-DM

Various packages can be downloaded from below link:
http://www.lfd.uci.edu/~gohlke/pythonlibs/#numpy

https://nbviewer.jupyter.org/github/obspy/docs/blob/master/workshops/2015-08-03_iris/01_Python_Crash_Course_with_output_and_solutions.ipynb Python Crash Course

Appendix 1.0 – BEGGINER AUXILIARY SECTIONS

1.1 IDE and Consoles

IDEs and consoles are an easy way to work with Python.

https://realpython.com/python-ides-code-editors-guide/

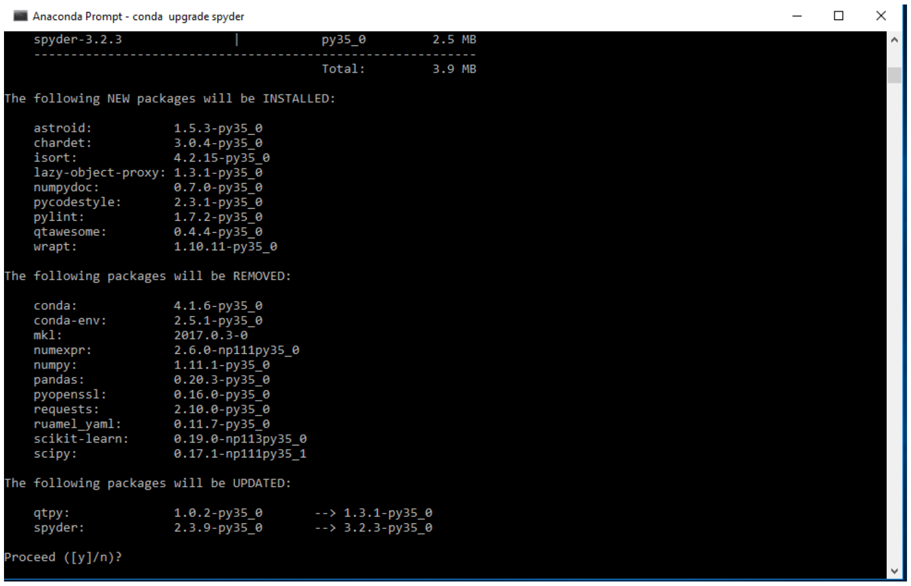
1.1.1 Python Anaconda

- Python Anaconda is an enterprise version.
- Very easy to use, versatile, can execute stand alone functions after the entire functions are executed.
- The IDE (or front-end) of Anaconda is Spyder

https://ipython.org/ipython-doc/3/interactive/qtconsole.html

Problem: Anaconda SPYDER GUI **does not** open.
Resolution unknown.

When passing command conda upgrade spyder, it tries to uninstall conda and other packages. Bad sign.



```
    spyder-3.2.3               |         py35_0         2.5 MB
    ------------------------------------------------------------
                                           Total:         3.9 MB

The following NEW packages will be INSTALLED:

    astroid:           1.5.3-py35_0
    chardet:           3.0.4-py35_0
    isort:             4.2.15-py35_0
    lazy-object-proxy: 1.3.1-py35_0
    numpydoc:          0.7.0-py35_0
    pycodestyle:       2.3.1-py35_0
    pylint:            1.7.2-py35_0
    qtawesome:         0.4.4-py35_0
    wrapt:             1.10.11-py35_0

The following packages will be REMOVED:

    conda:             4.1.6-py35_0
    conda-env:         2.5.1-py35_0
    mkl:               2017.0.3-0
    numexpr:           2.6.0-np111py35_0
    numpy:             1.11.1-py35_0
    pandas:            0.20.3-py35_0
    pyopenssl:         0.16.0-py35_0
    requests:          2.10.0-py35_0
    ruamel_yaml:       0.11.7-py35_0
    scikit-learn:      0.19.0-np113py35_0
    scipy:             0.17.1-np111py35_1

The following packages will be UPDATED:

    qtpy:              1.0.2-py35_0       --> 1.3.1-py35_0
    spyder:            2.3.9-py35_0       --> 3.2.3-py35_0

Proceed ([y]/n)?
```

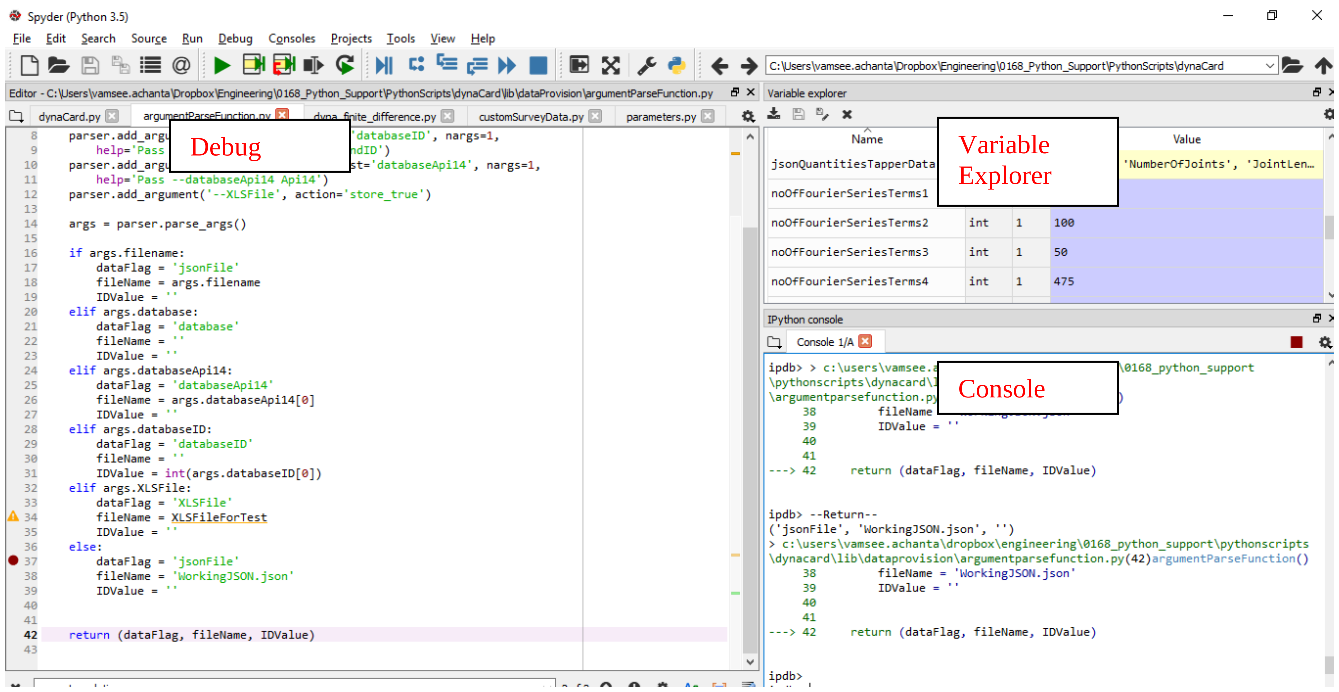
https://stackoverflow.com/questions/37750357/why-is-my-spyder-ide-no-longer-opening
https://pypi.python.org/pypi/setuptools

1.1.2 Spyder IDE

The general IDE layout is given below:

Debug is available:

- Line by line with following icons. Use 2nd icon (or Control F12) to run line by line
- This also steps into a separate function
-
- Can troubleshoot variables after stepping through the functions.
- Running Similar to command window in Spyder:


Spyder (Python 3.5)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\vamsee.achanta\Dropbox\Engineering\0168_Python_Support\PythonScripts\dynaCard
Editor - C:\Users\vamsee.achanta\Dropbox\Engineering\0168_Python_Support\PythonScripts\dynaCard\lib\dataProvision\argumentParseFunction.py
dynaCard.py
argumentParseFunction.py
dyna_finite_difference.py
customSurveyData.py
parameters.py
Debug
Variable explorer
Variable Explorer
IPython console
Console 1/A
Console

```
    parser.add_argu                          'databaseID', nargs=1,
        help='Pass                           ndID')
    parser.add_argu                          st='databaseApi14', nargs=1,
        help='Pass --databaseApi14 Api14')
    parser.add_argument('--XLSFile', action='store_true')

    args = parser.parse_args()

    if args.filename:
        dataFlag = 'jsonFile'
        fileName = args.filename
        IDValue = ''
    elif args.database:
        dataFlag = 'database'
        fileName = ''
        IDValue = ''
    elif args.databaseApi14:
        dataFlag = 'databaseApi14'
        fileName = args.databaseApi14[0]
        IDValue = ''
    elif args.databaseID:
        dataFlag = 'databaseID'
        fileName = ''
        IDValue = int(args.databaseID[0])
    elif args.XLSFile:
        dataFlag = 'XLSFile'
        fileName = XLSFileForTest
        IDValue = ''
    else:
        dataFlag = 'jsonFile'
        fileName = 'WorkingJSON.json'
        IDValue = ''


    return (dataFlag, fileName, IDValue)
```

Name			Value
jsonQuantitiesTapperData			'NumberOfJoints', 'JointLen…
noOfFourierSeriesTerms1			
noOfFourierSeriesTerms2	int	1	100
noOfFourierSeriesTerms3	int	1	50
noOfFourierSeriesTerms4	int	1	475

```
ipdb> > c:\users\vamsee.a                    \0168_python_support
\pythonscripts\dynacard\l
\argumentparsefunction.py                    )
     38         fileName
     39         IDValue = ''
     40
     41
---> 42     return (dataFlag, fileName, IDValue)


ipdb> --Return--
('jsonFile', 'WorkingJSON.json', '')
> c:\users\vamsee.achanta\dropbox\engineering\0168_python_support\pythonscripts
\dynacard\lib\dataprovision\argumentparsefunction.py(42)argumentParseFunction()
     38         fileName = 'WorkingJSON.json'
     39         IDValue = ''
     40
     41
---> 42     return (dataFlag, fileName, IDValue)


ipdb>
```

1.1.3 Passing argument in Spyder similar to that of running in a command window

Method 1 : GUI

- TO run command line options in Spyder, click on "Run",
- select "Run Configuration per file"
- Go to "General Settings"
- Check the "Command Line" options to provide the command arguments.
- Reference: http://www.agcross.com/2014/08/debug-python-spyder-command-line-flags/

Method 2 : Passing/opening Command Prompt to Spyder in-line

# -*- coding: utf-8 -*-
"""
Created on Mon Feb 26 11:58:29 2018

```
@author: martinj15
"""

def pullDynoData(databaseID):
    '''
    Wrapper to load data the way Vamsee provided. I call it Vamsee-style :)
    '''
#    import os
    go_to_dir       = "cd P:\Projects\PythonScripts\dynaCardData"
    set_environment = "activate dynaCard_env"
    create_excel    = 'python dynaCardData.py --databaseID '+str(databaseID)
    deactivate_environment = "deactivate"

    return go_to_dir+'&&'+set_environment+' && '+ create_excel+' && '+deactivate_environment

python_command = pullDynoData(357697)

import os
os.system(python_command)
```

1.1.4 IDE

Spyder is an easy to use IDE

- Comes with Anaconda

1.1.5 Canopy

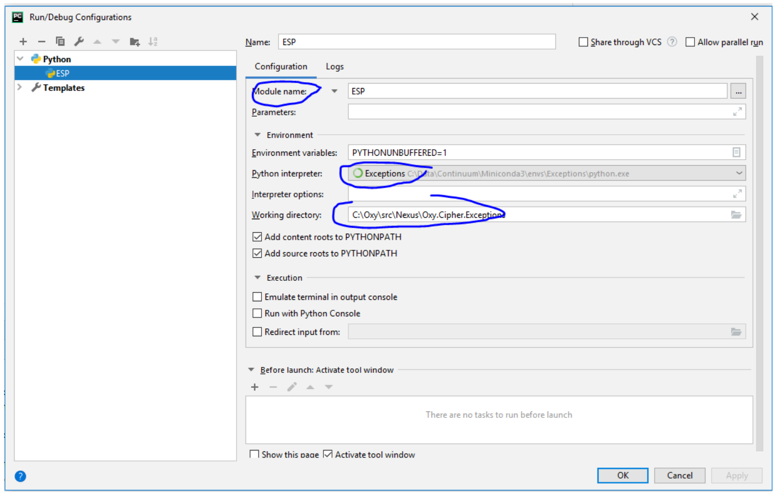
Canopy is an easy to use Python IDE

- Canopy is capable of interactive Python debugger and variable browser. However, it is only available to commercial subscribers

1.1.6 PyCharm

- Paid professional IDE. Community version is free.
- Very good for development

Running Modules in PyCharm:



Run/Debug Configurations

- Python
  - ESP
- Templates

Name: ESP ☐ Share through VCS ☐ Allow parallel run

Configuration | Logs

Module name: ESP
Parameters:

Environment
Environment variables: PYTHONUNBUFFERED=1
Python interpreter: Exceptions C:\Data\Continuum\Miniconda3\envs\Exceptions\python.exe
Interpreter options:
Working directory: C:\Oxy\src\Nexus\Oxy.Cipher.Exceptions
☑ Add content roots to PYTHONPATH
☑ Add source roots to PYTHONPATH

Execution
☐ Emulate terminal in output console
☐ Run with Python Console
☐ Redirect input from:

Before launch: Activate tool window
There are no tasks to run before launch
☐ Show this page ☑ Activate tool window

1.2 Running PY Files

See Section 2.2 for the simples way of running python files. Alternative methods are further given in this section.

1.2.1 Method 1

- In command prompt, go the directory where the module set up files are located. Example shown blow:

CD "C:\Users\vamsee.achanta\Downloads\PyPDF2-1.25.1\"

- Run Python on the setup file as follows:

C:\Python\Python35-32\python.exe setup.py install

- Restart Python (if necessary).
- The files should be installed and ready to use.

References:
http://dubroy.com/blog/so-you-want-to-install-a-python-package/
https://docs.python.org/3.6/installing/index.html

1.2.2 Calling .PY files

In Shell IDLE
subprocess.call(['python','HelloWorld.py'])
subprocess.check_output (['python','HelloWorld.py'])

In Command Prompt:
import subprocess

subprocess.call(['python','HelloWorld.py'])

1.2.3 Getting Started

Finding program Path

In windows machine, typical path for first time installation (versions older than 3.5) is:
C:\Python

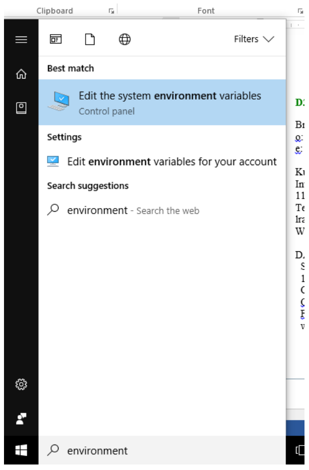
A typical 3.6 version path is located in folder below. A search may be required.
C:\Users\vamsee.achanta\AppData\Local\Programs\Python\Python36-32

Adding Path to Run Python from DOS

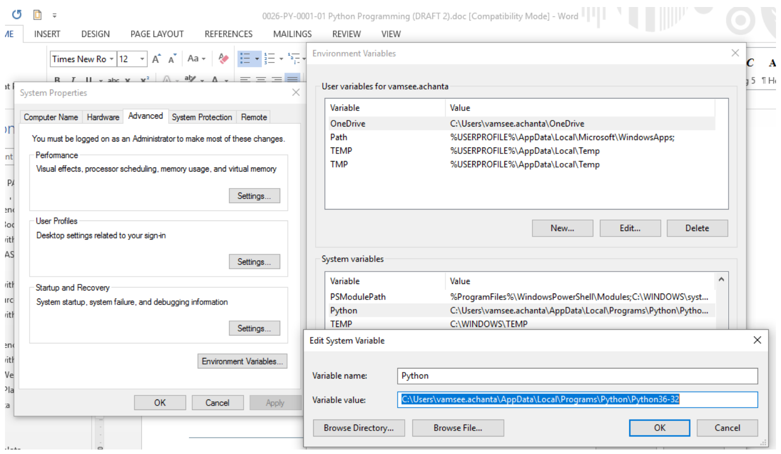
Method 1:
In windows, set path to the Python executable program folder:
set path=%path%;C:\Users\vamsee.achanta\AppData\Local\Programs\Python\Python36-32

Method 2:

- Search for Environment variables in Windows and Edit System Environment variables


Clipboard
Font
Filters
Best match
Edit the system environment variables
Control panel
Settings
Edit environment variables for your account
Search suggestions
environment - Search the web
environment

- Click on Environment variables
- Add Python Variable
- Good practice: Ensure this path is added by typing in “path” in the DOS Window and verifying the path you just added.
    - Sometimes a system administrator may be required to do this



You must be logged on as an Administrator to make most of these changes.

Performance
Visual effects, processor scheduling, memory usage, and virtual memory

User Profiles
Desktop settings related to your sign-in

Startup and Recovery
System startup, system failure, and debugging information

User variables for vamsee.achanta

Variable	Value
OneDrive	C:\Users\vamsee.achanta\OneDrive
Path	%USERPROFILE%\AppData\Local\Microsoft\WindowsApps;
TEMP	%USERPROFILE%\AppData\Local\Temp
TMP	%USERPROFILE%\AppData\Local\Temp

System variables

Variable	Value
PSModulePath	%ProgramFiles%\WindowsPowerShell\Modules;C:\WINDOWS\syst...
Python	C:\Users\vamsee.achanta\AppData\Local\Programs\Python\Pytho...
TEMP	C:\WINDOWS\TEMP

Variable name: Python
Variable value: C:\Users\vamsee.achanta\AppData\Local\Programs\Python\Python36-32

1.3 Python Installation Errors

Error1: Cannot uninstall 'certifi'. It is a distutils installed project and thus we cannot accurately determine which files belong to it which would lead to only a partial uninstall.

Solution 1: Try removing from site-packages of the environment or package
https://github.com/pypa/pip/issues/5247

Solution2: Try re-installing the entire python package/engine

Error2:

```
==== Python Module Installation Errors ====
Collecting pandas
  Retrying (Retry(total=4, connect=None, read=None, redirect=None, status=None)) after connection broken by 'SSLError(SSLError(1, '[SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed (_ssl.c:719)'),)': /simple/pandas/
  Retrying (Retry(total=3, connect=None, read=None, redirect=None, status=None)) after connection broken by 'SSLError(SSLError(1, '[SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed (_ssl.c:719)'),)': /simple/pandas/
  Retrying (Retry(total=2, connect=None, read=None, redirect=None, status=None)) after connection broken by 'SSLError(SSLError(1, '[SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed (_ssl.c:719)'),)': /simple/pandas/
  Retrying (Retry(total=1, connect=None, read=None, redirect=None, status=None)) after connection broken by 'SSLError(SSLError(1, '[SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed (_ssl.c:719)'),)': /simple/pandas/
  Retrying (Retry(total=0, connect=None, read=None, redirect=None, status=None)) after connection broken by 'SSLError(SSLError(1, '[SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed (_ssl.c:719)'),)': /simple/pandas/
  Could not fetch URL https://pypi.org/simple/pandas/: There was a problem confirming the ssl certificate: HTTPSConnectionPool(host='pypi.org', port=443): Max retries exceeded with url:
```

/simple/pandas/ (Caused by SSLError(SSLError(1, '[SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed (_ssl.c:719)'),)) - skipping
  Could not find a version that satisfies the requirement pandas (from versions:)
No matching distribution found for pandas
==================================
Solution: If http website is accessed , the modules will be automatically installed.

1.4 VS Code

Working in Visual Studio Code:
 How to set up environments in visual studio code

Appendix 2.0– RESOURCES

https://www.youtube.com/watch?v=B9Yi9YE1wnw   Python for Engineers
http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-189-a-gentle-introduction-to-programming-using-python-january-iap-2011/assignments/
https://www.enthought.com/services/training/scientific/
https://automatetheboringstuff.com/chapter12/   Python and Excel
https://automatetheboringstuff.com/chapter13/   Python, Excel and PDFs
https://automatetheboringstuff.com/#toc   Automate the boring stuff
http://stackoverflow.com/questions/28192977/how-to-unlock-a-secured-read-protected-pdf-in-python   How to unlock pdfs

https://en.wikipedia.org/wiki/Jenkins_(software)

https://www.udemy.com/learning-python-not-the-snake/

Python Interview Questions

https://www.codementor.io/sheena/essential-python-interview-questions-du107ozr6
https://intellipaat.com/interview-question/python-interview-questions/

Appendix 3.0 – TESTING

3.1 Performance Profiling

- cprofile and profile provide deterministic profiling of python programs. Statistic will be generated for:
    - How often the program executed

    - How long various parts of the program executed.
- Cprofile is recommended for long running programs
- Profile
- pstats module can convert statistics into reports
-

https://docs.python.org/2/library/profile.html
https://stackoverflow.com/questions/582336/how-can-you-profile-a-script

http://docs.python-guide.org/en/latest/writing/tests/
http://tjelvarolsson.com/blog/four-tools-for-testing-your-python-code/

Programs to be Installed:
- Microsoft Visual Studio .NET 2012?
- SQL Client Server
- Microsoft SQL server Management (X64)

microsoft sql server management studio 2017
SQL Client
(SQL Server 2008 R2 Client tools?)

Appendix 4.0 - **DS STAN**

Stan is a modelling languge is a programming guide with an extensive collection of modelling techniques and accompanying example models. Stan's modelling language is portable across all interaces.

Stan is a state of the art platform for statistical modelling and hig performance statistical computation. Users can specifiy log density functions in Stan's probabilistic programming language and get:
- Full Bayesian statistical inference with MCMC sampling (NUTS, HMC)
- Approximate Bayesian inference with variational inference (ADVI)
- Penalized maximum likelihood estimation with optimization (L-BFGS)

http://mc-stan.org/users/

Welltest Validation

Set up python environment for running Stan libraries:
https://pystan.readthedocs.io/en/latest/windows.html

Define a model. Optimize a model. Perform statistical methods:
https://pystan.readthedocs.io/en/latest/optimizing.html

Avoid recompilation of models

-a-class

Appendix 5.0- CREATE AND UPDATE PYTHON PACKAGE

5.1 Introduction

5.2 Summary

The good practices are as follows:

- Write tests. Preferably utilize pytest.
- Example test and file structure
    - https://github.com/jumptrading/luddite
    - Utilized test_package.py for all tests
    - pytest.ini file for pytest configurations
    - Utilize github test workflows
        - https://github.com/jumptrading/luddite/blob/master/.github/workflows/tests.yml
    - Others?
- Pypi supports Readme.rst (restructured Text) and Readme.md files. If only 1 format is used in a package, prefer to utilize a package to alternate if required.
    - https://stackoverflow.com/questions/10718767/have-the-same-readme-both-in-markdown-and-restructuredtext
-

5.3 Steps – Overview

Step	Description	Commands/Detailed Description

1	Create python project with directory structure	Follow pep8 guidelines
2	Package compliance	Ensure all directories are package modules using __init__.py
3	Add setup.py and build wheels	python setup.py sdist bdist_wheel
4	Create account on pypi and upload using twine package	These commands will push the .whl and .tar.gz file into the pypi repository conda install twine twine upload dist/*

5.4 References

Good References:

https://www.freecodecamp.org/news/build-your-first-python-package/
https://python-packaging-tutorial.readthedocs.io/en/latest/setup_py.html
https://packaging.python.org/

Libraries and Guidelines to contribute:

https://pandas.pydata.org/docs/development/contributing.html#contributing