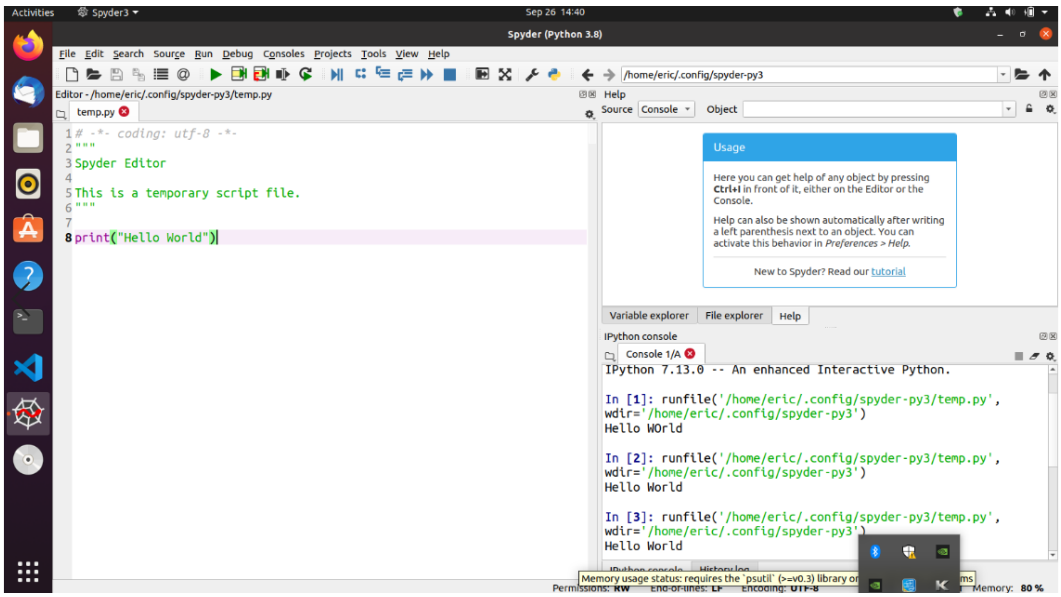
ECE59500 Hands-on Assignment 1

Task 1

1) screenshot of installing pandas package on shell

```
eric@eric-VirtualBox:~$ sudo -i
root@eric-VirtualBox:~# apt-get install python3-pandas
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  fonts-lyx libaec0 libblas3 libblosc1 libgfortran5 libhdf5-103
  libjs-jquery-ui liblapack3 liblbfgsb0 libsnappy1v5 libsz2
  python-matplotlib-data python-tables-data python3-atomicwrites python3-bs4
  python3-cycler python3-et-xmlfile python3-jdcal python3-kiwisolver
  python3-lxml python3-matplotlib python3-numexpr python3-numpy
  python3-openpyxl python3-pandas-lib python3-pluggy python3-py python3-pytest
  python3-scipy python3-soupsieve python3-tables python3-tables-lib
  python3-xlrd python3-xlwt ttf-bitstream-vera
Suggested packages:
  libjs-jquery-ui-docs python-cycler-doc python3-lxml-dbg python-lxml-doc
  dvipng ffmpeg inkscape ipython3 python-matplotlib-doc python3-cairocffi
  python3-gobject python3-nose texlive-extra-utils texlive-latex-extra
  ttf-staypuft gfortran python-numpy-doc python3-numpy-dbg python-pandas-doc
  python3-statsmodels subversion python-scipy-doc python3-netcdf4
  python-tables-doc vitables python-xlrt-doc
The following NEW packages will be installed:
```

2)screenshot of running spyder



Task 2

source code:

```python
wd = int(input("Enter the amount for withdrwal: "))
bills = {
        100 : 0,
        50 : 0,
        20 : 0,
        10 : 0
        }
for i, j in bills.items():
    while wd >= i:
        wd -= i
        bills[i] += 1
        if wd == 0:
            break

if(wd != 0):
    print('The amount cannot be drawn.')
else:
    print('Please collect your bills as follows:')
    for i, j in bills.items():
        if j != 0:
            # print(money)
            # print(f'${i:3}: {j}')
            print('%5s' '%2s' % ('$'+str(i)+':', str(j)))
```

test result:

```
PS D:\PNW Courses\2020 Summer\Big Data\HW1\Hands-on1> & D:/Python3.8.6/python.exe "d:/PNW Courses/2020 Summer/Big Data/HW1/Hands-on1/atm.py"
Enter the amount for withdrwal: 150
Please collect your bills as follows:
$100: 1
 $50: 1
PS D:\PNW Courses\2020 Summer\Big Data\HW1\Hands-on1> & D:/Python3.8.6/python.exe "d:/PNW Courses/2020 Summer/Big Data/HW1/Hands-on1/atm.py"
Enter the amount for withdrwal: 75
The amount cannot be drawn.
PS D:\PNW Courses\2020 Summer\Big Data\HW1\Hands-on1>
```

Task 3:

Q1)

```python
import jinja2
import pandas as pd
from tabulate import tabulate
from pandas import DataFrame

# open the file and read the contens
df = pd.read_csv("records.txt")

# Q1

def scoreEval(score):
        if score >= 90:
                return 'A'
        elif score >= 80:
                return 'B'
        elif score >= 70:
                return 'C'
        elif sscore >= 60:
                return 'D'
        else:
                return 'F'

df['ece595'] = df['ece595'].apply(scoreEval)
df['ece547'] = df['ece547'].apply(scoreEval)
df['ece354'] = df['ece354'].apply(scoreEval)
print(tabulate(df, showindex=False, headers = df.columns, tablefmt = "grid"))
```

```
PS D:\PNW Courses\2020 Summer\Big Data\HW1\Hands-on1> & D:/Python3.8.6/python.exe "d:/PNW Courses/2020 Summer/Big Data/HW1/Hands-on1/registrar.py"
+-----------+----------+----------+----------+
| student   | ece595   | ece547   | ece354   |
+===========+==========+==========+==========+
| ahmad     | A        | B        | A        |
+-----------+----------+----------+----------+
| bob       | A        | B        | B        |
+-----------+----------+----------+----------+
| chen      | C        | B        | C        |
+-----------+----------+----------+----------+
| dev       | B        | A        | A        |
+-----------+----------+----------+----------+
| eva       | B        | A        | B        |
+-----------+----------+----------+----------+
| fatima    | B        | B        | B        |
+-----------+----------+----------+----------+
| geeta     | A        | B        | C        |
+-----------+----------+----------+----------+
| hailu     | A        | B        | B        |
+-----------+----------+----------+----------+
| ian       | B        | B        | A        |
+-----------+----------+----------+----------+
| john      | B        | C        | C        |
+-----------+----------+----------+----------+
```

Q2)

```
df_score595 = df[df.ece595.eq(df['ece595'].max())].drop(columns = ['ece547', 'ece354'])
df_score595.columns = ['scorer', 'score']
df_score595.insert(0, 'subject', 'ece595')

df_score547 = df[df.ece547.eq(df['ece547'].max())].drop(columns = ['ece595', 'ece354'])
df_score547.columns = ['scorer', 'score']
df_score547.insert(0, 'subject', 'ece547')

df_score354 = df[df.ece354.eq(df['ece354'].max())].drop(columns = ['ece595', 'ece547'])
df_score354.columns = ['scorer', 'score']
df_score354.insert(0, 'subject', 'ece354')

df = pd.DataFrame(data = df_score595)
df = df.append(df_score547)
df = df.append(df_score354)
print(tabulate(df, showindex=False, headers = df.columns, tablefmt = "grid"))
```

```
+-----------+----------+---------+
| subject   | scorer   |   score |
+===========+==========+=========+
| ece595    | hailu    |      93 |
+-----------+----------+---------+
| ece547    | eva      |      92 |
+-----------+----------+---------+
| ece354    | ian      |      95 |
+-----------+----------+---------+
```

Q3)

```
def scoreEval(score):
        if score >= 90:
                return 4
        elif score >= 80:
                return 3
        elif score >= 70:
                return 2
        elif sscore >= 60:
                return 1
        else:
                return 0

df['ece595'] = df['ece595'].apply(scoreEval)
df['ece547'] = df['ece547'].apply(scoreEval)
df['ece354'] = df['ece354'].apply(scoreEval)
df['Gpa'] = df.mean(axis = 1).round(decimals = 2)
df = df.drop(['ece595', 'ece547', 'ece354'], axis = 1)
print(tabulate(df, showindex=False, headers = df.columns, tablefmt = "grid"))
```

student	Gpa
ahmad	3.67
bob	3.33
chen	2.33
dev	3.67
eva	3.33
fatima	3
geeta	3
hailu	3.33
ian	3.33
john	2.33

Task4

The author tired using a file consisting of 6.75 million 16-byte records containing uniformly distributed random data to extract the median age by sex for each country. With a 128-GB RAM on a $15000 Dell server, data in-memory can be ran off in less than 1 min, such that he hesitated to call it "**big data**". He then loaded subsets consisting of up to 1 billion rows of just three columns, and the execution of time increased rapidly as the number of rows grew past 1 million, and the query ended up taking more than 24 hours to finish, presumably because of the poor algorithm for the given data and query, which indicated that as dataset sizes grow, it becomes increasingly important to flexibly choose diverse **algorithms** for exploiting the efficiency of dealing with datasets. The pathologies of big data are primarily those of analysis, including transaction processing and data storage, the trouble usually comes when we want to take that accumulated data and make some analysis on it. **Data warehousing** is usually the solution for it, as in a copy of transaction data followed by reconstitution in a different database, which is what we so-called "**extract, load, transform**" process. Then the definition of Big Data became more and more clearly that "data whose size forces us the look beyond the tried-and-true methods that are prevalent at that time". What makes big data getting bigger is repeated observations over time and space. For example, the cellphone database will record time and location every 15s. That kind of database storing time-series data will likely to cause some problems as the data grew large enough that it no longer fits in memory. In modern systems, random access to memory is typically more than 150000 times slower than sequential access to disk, and aside from that, hard limits is also another challenge for data analysis. Both them drove us to use another tactic which is called **Distributed System** for handling with increasingly growing datasets. The cheap price and flexible replications brought up the advantage of Distributed System for dealing with huge amount of data, although so, it also has significant performance costs: the **bandwidth** and **latency** are several orders of magnitude worse than RAM. Meanwhile, once a large dataset has been distributed

to multiple nodes in this way, a huge advantage can be obtained by distributing the processing as well. In distributed analysis, processing must contain a significant component that is local in the data- that is, does not require simultaneous processing of many disparate parts of the dataset. Ideally, each node will have the same amount of workload, if this is not the case, then some code with the most work may take a lot time to give us the result, then there's no benefit of using Distributed System at all. On top of that, there's another issue might become the bottle-neck of the overall distributed system which is **reliability**. Since most of the hardware components tend to be unreliable when the dataset gets huge, and it'll go hard to keep **multiple copies** at the same time.