ENGR 1330-2022 Exam 2-Laboratory Portion

Medrano, Giovanni

R11521018

ENGR 1330 Exam 2 - Laboratory/Programming Skills

Download (right-click, save target as ...) this page as a jupyterlab notebook from: s22-ex1-deploy.ipynb

If you are unable to download the file, create an empty notebook and copy paste the problems into Markdown cells and Code cells (problem-by-problem)

Problem 0 (5 pts) : *Profile your computer*

Execute the code cell below exactly as written. If you get an error just continue to the remaining problems.

In [1]:

```
# Preamble script block to identify host, user, and kernel
import sys
! hostname
! whoami
print(sys.executable)
print(sys.version)
print(sys.version_info)
```

```
DESKTOP-6HAS1BN
desktop-6has1bn\medra
C:\Users\medra\anaconda3\python.exe
3.8.5 (default, Sep  3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)]
sys.version_info(major=3, minor=8, micro=5, releaselevel='final', serial=0)
```

Problem 1 NUMPY (10 points) : *Show me the money!*

Assume that you have access to 900,000 bank accounts belonging to other people. All of these account balances start with \$101.23 USD in them. The interest rate for all the accounts is 5%, compounded daily. [Daily compounding] (https://www.eduworks.com/Documents/Workshops/EdMedia1998/class/compound.htm) generally means that the stated annual rate is applied daily (also: [Daily compounding] (https://www.cuemath.com/daily-compound-interest-formula/)) $\frac{5\%}{365}=0.01369\%$ per day. The interest is added to the balance of each account, with the exception of the fractions of cents (anything past the third decimal place) that are truncated (dropped).

Let's pretend we are online robbers and we want to write some code to skim the excesss interest and place it into our own account. We are going to illegally skim the fractions that get truncated (dropped) out of the other people's accounts evertime their interest is calculated, and add the truncated amounts into our own account. Our account will begin with $0 but grow by the day.

Execute the code block below to create the accounts

In [2]:

```
import numpy as np
# create the accounts
accounts = np.array([101.23 for i in range(900000)])
```

The daily interest is simply the product of the daily compounding rate and the account value

$ P_{n+1} = P_{n} + i*P_{n}$ in this problem the result $i*P_{n}$ is only reported to the penny, any smaller part of the value is skimmed into the illegal bitcoin account.

A function that performs the computations is shown below

In [3]:

```
# execute this code block to prototype the function
def add_interest(current_value,daily_rate):
    compounding_amount = current_value*daily_rate      # compute compounding amount.
    whole_dollar_ammount = current_value*daily_rate//1 # compute compounding amount who
    fractional_dollar_amount = compounding_amount - whole_dollar_ammount # compute comp
    pennies = fractional_dollar_amount*100//1  # compute portion of fraction that are p
    skim = fractional_dollar_amount - pennies/100
    output=(whole_dollar_ammount+pennies/100,skim) # return a tuple of interest to add,
    return(output)
```

Implement the steps necessary to increase each of the 900,000 accounts by (5/365)% per day, truncate the accounts to the nearest penny, and add the truncated amount into a seperate account titled “bitcoin”

Using a repetition structure determine the abount in the bitcoin account after 1 day, 10 days and 100 days of stealing.

In [4]:

```
bitcoin = 0
days = 0
treshold = 1000000
divisor = 0.01369/100




while(1):

    if(bitcoin>=treshold):
        break

    if(days==1):
        print('You have made ', bitcoin,'$ so far in 1 days!')
    if(days==10):
        print('You have made ', bitcoin,'$ so far in 10 days!')
    if(days==100):
        print('You have made ', bitcoin,'$ so far in 100 days!')
    x = add_interest(accounts, divisor)

    accounts = (accounts + x[0])

    bitcoin = (np.sum(x[1]) + bitcoin)

    days =days + 1

print('You have made ', bitcoin, '$ so far in',days,'total days!')
```

```
You have made  3472.548300000006 $ so far in 1 days!
You have made  34780.92750000005 $ so far in 10 days!
You have made  353353.72500000324 $ so far in 100 days!
You have made  1001370.1500000241 $ so far in 275 total days!
```

Problem 2 (5 pts) : *Get the data files*

In [5]:

```
######### CODE TO AUTOMATICALLY DOWNLOAD THE DATABASE ################
import requests # import needed modules to interact with the internet
# make the connection to the remote file (actually its implementing "bash curl -O http:
remote_url = 'http://54.243.252.9/engr-1330-webroot/8-Labs/Lab15/galton_subset.csv' # a
response = requests.get(remote_url) # Gets the file contents puts into an object
output = open('galton_subset.csv', 'wb') # Prepare a destination, local
output.write(response.content) # write contents of object to named local file
output.close() # close the connection
```

Problem 3: Dataframe read/organize (10 pts): *read file into a dataframe, rename columns, show a few rows of the dataframe*

Read the `galton_subset.csv` into a dataframe. Name/rename the columns to **"Father","Mother","Child"**. Display the first 9 rows of the dataframe.

In [6]:

```
# import necessary packages
import pandas as pd
# read the csv file into a dataframe
df1 = pd.read_csv('galton_subset.csv')
#print(df)
# rename son to child
df1.columns = ['Father', 'Mother', 'Child']
# show dataframe head
df1.head(10)
```

Out[6]:

	Father	Mother	Child
0	78.5	67.0	73.2
1	75.5	66.5	73.5
2	75.0	64.0	71.0
3	75.0	64.0	70.5
4	75.0	58.5	72.0

	Father	Mother	Child
5	74.0	68.0	76.5
6	74.0	62.0	74.0
7	73.0	67.0	71.0
8	73.0	67.0	68.0
9	73.0	66.5	71.0

Problem 4 Dataframe analyze/summarize (10 pts): *find mean values, father, mother, child*

Using the dataframe from the previous problem, find the average height of **"Father","Mother","Child"** columns (series) and the standard deviation of these columns.

On average which category is taller (father, mother, child)?

In [8]:

```
# query dataframe by each series
df2 = df1.mean()
print(df2)
# print average height and standard deviation
df1[['Father', 'Mother','Child']].std()
```

```
Father    69.098883
Mother    63.993855
Child     70.454749
dtype: float64
```

Out[8]:

```
Father    2.546555
Mother    2.366816
Child     2.557061
dtype: float64
```

who is taller?

On average the Child is the tallest then then the mother then the father

---

Problem 5 Dataframe Histograms (10 points) : *histograms; df['...'].hist(alpha=0.5,color='...') OR using subplots*

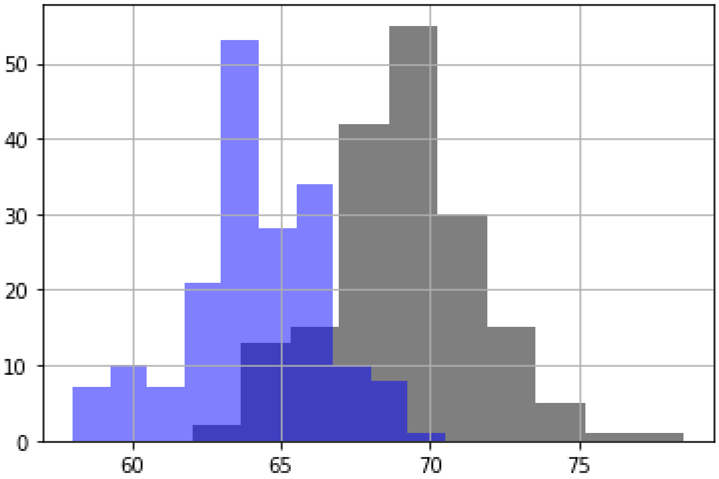
Using the dataframe from the Problem 4, produce a histogram of fathers and a histogram of mothers on the same plot. Do the histograms suggest that one group is shorter than the other?

In [9]:

```
# build histograms df['...'].hist(alpha=0.5,color='...') of father and mother
df1['Father'].hist(alpha=0.5,color='black')
# show the plot same figure
df1['Mother'].hist(alpha=0.5,color='blue')
```

Out[9]:

```
<AxesSubplot:>
```



who is shorter?

The Mother is shorter according to the histogram and data.

In [ ]: