Engineering Analysis Homework 1

August 24, 2023

Gabe Morris

[1]:
```
# Notebook Preamble
import sympy as sp
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from IPython.display import display, Latex

plt.style.use('maroon_ipynb.mplstyle')
```

Contents

ME 8213 Homework 1 Gabe Morris
gnm54

1 Problem 1

The included data file, Homework1.dat, is a pretend set of grades for an assignment. The first column is the last name, the second column is the first name, and the third column is the grade.

a) Determine the number of students who passed the assignment (grade > 60)
b) Determine the number of students who made a B on the assignment (80<= grade < 90)
c) Print the names of all students who made an A on the assignment. If no students made an A, print a message saying so

[2]:
```
# Getting the information from dat file
data = pd.read_csv('Homework1.dat', delimiter='\t', names=['Last', 'First',␣
 ↪'Grade'])
data_style = data.style.hide(axis='index')
data_style
latex = data_style.to_latex(hrules=True)
display(Latex(fr'\begin{{center}}{latex}\end{{center}}'))
```

Last	First	Grade
Roentgen	Wilhelm	77.385410
Lorentz	Hendrik	82.440226
Curie	Pierre	64.835290
Curie	Marie	97.433619
Thomson	Joseph	92.010022
Michelson	Albert	84.148474
Marconi	Guglielmo	76.428518
Wien	Wilhelm	84.424898
Bragg	William	86.659487
Planck	Max	79.105066
Einstein	Albert	76.980888
Bohr	Niels	72.115877
Millikan	Robert	61.874601
Hertz	Gustav	59.616759
Compton	Arthur	57.725977
Raman	Chandrasekhara	72.897915
Dirac	Paul	63.476188
Fermi	Enrico	76.980724
Yukawa	Hideki	85.933096
Bloch	Felix	95.202058
Bardeen	John	73.423745
Wigner	Eugene	88.479665
Hall	John	80.364778
Higgs	Peter	84.312859
Thorne	Kip	75.144785
Penrose	Roger	84.832464

1.1 Part A

[3]:
```
last, first, grades = np.array(data['Last']), np.array(data['First']), np.
↪array(data['Grade'])

# Number of students that got above 60
passing = grades > 60
list(passing).count(True)
```

[3]: 24

1.2 Part B

[4]:
```
# Number of B's

Bs = np.logical_and(grades >= 80, grades < 90)
list(Bs).count(True)
```

[4]: 9

1.3 Part C

[5]:
```
# Getting all the names of the students that made an A
As = grades >= 90
last_A, first_A = last[As], first[As]

if As.size > 0:
    [print(f'{first_} {last_} made an A') for first_, last_ in zip(first_A,␣
↪last_A)]
else:
    print("'There are no A's")
```

```
Marie Curie made an A
Joseph Thomson made an A
Felix Bloch made an A
```

2 Problem 2

Write a file called primes500.dat which lists all prime numbers less than 500.

[6]:
```
def sieve_eratosthenes(limit_):
    # Create a boolean array "prime[0...limit_]" and initialize all entries as␣
 ↪True
    # A value in prime[x] will finally be False if x is Not a prime, else True.
    prime_ = [True for _ in range(limit_ + 1)]
    p = 2
    while p**2 <= limit_:
        # If prime[i] is not changed, then it is a prime
        if prime_[p]:
            # Update all multiples of p
            for i in range(p**2, limit_ + 1, p):
                prime_[i] = False
        p += 1

    primes_ = [p for p in range(2, limit_) if prime_[p]]
    return primes_

# Generate prime numbers less than 500
limit = 500
primes = sieve_eratosthenes(limit)

# Write the prime numbers to the file primes500.dat
with open('primes500.dat', 'w') as file:
    for prime in primes:
        file.write(str(prime) + '\n')

np.array(primes)
```

[6]:
```
array([  2,   3,   5,   7,  11,  13,  17,  19,  23,  29,  31,  37,  41,
        43,  47,  53,  59,  61,  67,  71,  73,  79,  83,  89,  97, 101,
       103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167,
       173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239,
       241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313,
       317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397,
       401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467,
       479, 487, 491, 499])
```

3 Problem 3

Find the (a) dot product and (b) outer product of the following two vectors (Use numpy):

$$a = (1\ 1\ 0)$$

$$b = (2\ 1\ 3)$$

3.1 Part A

[7]:
```
np.dot((1, 1, 0), (2, 1, 3))
```

[7]:
```
3
```

3.2 Part B

[8]:
```
np.outer((1, 1, 0), (2, 1, 3))
```

[8]:
```
array([[2, 1, 3],
       [2, 1, 3],
       [0, 0, 0]])
```

4 Problem 4

A perfect number is a positive integer that is equal to the sum of its positive divisors, including 1, but excluding itself. As an example, 6 is a perfect number because the divisors of 6 are 1, 2, and 3, and 1+2+3=6. Find all perfect numbers less than 500.

[9]:
```
def find_divisors(n):
    divisors_ = [1]
    for i in range(2, n + 1):
        if n % i == 0:
            divisors_.append(i)
    return divisors_

perfect_numbers = []

for num in range(2, 500):
    divisors = find_divisors(num)
    if num == sum(divisors[:-1]):
        perfect_numbers.append(num)

print("Perfect numbers less than 500:", perfect_numbers)
```

```
Perfect numbers less than 500: [6, 28, 496]
```