# HW1

September 9, 2024

## 1 Homework 1

Please submit the solution to gradescope by 11:59 PM, Sept 12, Thursday.

**Name**: Ivy Nangalia

**PID**: 730670491

```
[ ]: import string
```

### 1.1 Problem 1 (25 points).

**1.1** (20 points) Load a text file `file.txt`, clean its content by

- removing all punctuation
- removing extra white spaces
- transforming the text to lowercase.

Print the cleaned text.

**1.2** (5 points) Count and print the number of vowels present in the cleaned text

Example input: `s = "hello world"`

Example output: `a = 0, e = 1, i = 0, o = 2, u = 0`

```
[ ]: %%file file.txt

     Hello World!
     I'm doing my first STOR320 homework Assignment!
     How exciting.
```

```
Writing file.txt
```

```
[ ]: with open("file.txt") as f:
         f = f.read()
     f = f.translate(str.maketrans('', '', string.punctuation))
     f = f.lower()
     for line in f:
         print(line, end='')
```

```
hello world
im doing my first stor320 homework assignment
how exciting
```

[ ]:
```
def vowel_count(x: str):
    vc: dict = {"a": 0, "e": 0, "i": 0, "o": 0, "u": 0}
    for y in x:
        if y in vc:
            vc[y] += 1
    print(f"a = {vc['a']}, e = {vc['e']}, i = {vc['i']}, o = {vc['o']}, u =␣
 ↪{vc['u']}")


vowel_count(f)
```

```
a = 1, e = 4, i = 6, o = 7, u = 0
```

### 1.2 Problem 2. (25 point).

You are organizing information about several data science courses. Given the nested list of tuples `courses = [(101, 'Intro to Data Science'), (201, 'Machine Learning'), (103, 'Data Visualization'), (102, 'Python Programming')]`, perform the following tasks:

**2.1** (5 points) Convert the list of tuples into a dictionary where the course number is the key and the course name is the value.

**2.2** (5 points) Add a new course `(202, 'Deep Learning')` to the dictionary.

**2.3** (5 points) Sort the dictionary based on the course number.

**2.4** (5 points) Update the dictionary based on the following instruction: `Intro to Data Science` course is combined with `Python Programming` course, called `Intro to Data Science with Python` and the course number is still `101`. Delete the course `Python Programming`.

**2.5** (5 points) Extract the course numbers and course names from the updated dictionary and store them in separate lists, e.g., `[101, 102, 103, 201, 202]`, `['Intro to Data Science with Python', 'Data Visualization',...]`

[ ]:
```
courses = [(101, 'Intro to Data Science'), (201, 'Machine Learning'), (103,␣
 ↪'Data Visualization'), (102, 'Python Programming')]
courses = dict(sorted(courses))
courses[202] = "Deep Learning"
courses[101] = "Intro to Data Science with Python"
courses.pop(102)

print(courses)
print(list(courses.keys()))
print(list(courses.values()))
```

```
{101: 'Intro to Data Science with Python', 103: 'Data Visualization', 201:
'Machine Learning', 202: 'Deep Learning'}
[101, 103, 201, 202]
['Intro to Data Science with Python', 'Data Visualization', 'Machine Learning',
'Deep Learning']
```

### 1.3 Problem 3 (25 points).

You have two lists of student IDs enrolled in two different data science courses: `course1_ids = [101, 102, 102, 103, 104]` and `course2_ids = [103, 104, 104, 105, 106]`. Perform the following tasks:

**3.1** (5 points) Convert the lists to sets to remove duplicate IDs.

**3.2** (10 points) Find the union and intersection of the two sets to determine which students are in either or both courses.

**3.3** (5 points) Convert the union and intersection sets to sorted lists.

**3.4** (5 points) Create a dictionary where the keys are the student IDs from the union list and the values are True if the student is in both courses, otherwise False.

[ ]:
```python
course1_ids = [101, 102, 102, 103, 104]
course2_ids = [103, 104, 104, 105, 106]
course1_ids = set(course1_ids)
course2_ids = set(course2_ids)

union = sorted(course1_ids or course2_ids)
intersection = sorted(course1_ids and course2_ids)

print("Union: " + str(union))
print("Intersection: " + str(intersection))

both_classes = {}
for id in union:
    if id in course1_ids and id in course2_ids:
        both_classes[id] = True
    else:
        both_classes[id] = False

print(both_classes)
```

```
Union: [101, 102, 103, 104]
Intersection: [103, 104, 105, 106]
{101: False, 102: False, 103: True, 104: True}
```

## 2 Problem 4 (25 points)

**4.1** (5 points) Install a package `sympy`. The `sympy` library is a Python library for symbolic mathematics. You do not have to install this library again if it is already installed.

**4.2** (10 points) `isprime` function in `sympy` checks if a number is a prime number. For example, `isprime(7)` returns `True`. Based on this function, define a function to find the closest prime number of an integer. For example, if the input is 8, the function should return 7, as it is the closest prime number.

**4.3** (5 points) Treat your PID number at UNC as an integer, and print its closest prime number.

**4.4** (5 points) Given this sequence `[456,983,482,963,749,12,97]`, create a new list where each element is replaced by the closest prime number to that element, using the function defined in **4.2**.

```
[ ]: def closest_prime(n: int) -> int:
         n_add = n
         n_sub = n
         from sympy import isprime
         while isprime(n) == False:
             while isprime(n_add) == False:
                 n_add += 1
             while isprime(n_sub) == False:
                 n_sub -= 1
             if (n_add - n) > (n - n_sub):
                 n = n_sub
             else:
                 n = n_add
         return n

     print(closest_prime(730670491))

     seq = [456,983,482,963,749,12,97]
     for x in range(0, len(seq)):
         seq[x] = closest_prime(seq[x])

     print(seq)
```

```
730670477
[457, 983, 479, 967, 751, 13, 97]
```

```
[ ]: # discovering prime numbers for fun

     def prime(n: int) -> bool:
         divisable: bool
         if n == 1:
             return False
         n_sub = n - 1
         while n % n_sub != 0:
             n_sub -= 1
         if n_sub == 1:
             divisable = False
         else:
             divisable = True
```

```
    return not divisable

primes = []

for x in range(1, 50):
    if prime(x) == True:
        primes.append(x)

print(primes)

flags = []
for x in primes:
    if not isprime(x):
        flags.append(x)
if len(flags) == 0:
    print("All prime!")
else:
    print("Flags: " + str(flags))
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]
All prime!
```

```
[ ]:
```