Exam1

September 28, 2024

1 CPE 4040: Exam1

1.1 Follow the instructions below:

- Write your answer in the cell provided under each question.
- You must write comments to explain your thoughts to earn full credit.
- Show your execution result.
- Do your own work. Discussion with peers is prohibited.
- Do not copy-and-paste other people's (or from Generative AI) codes.
- Please do not use pandas since it is not covered yet.

1.1.1 Student's Name: Ryan Brinson

1.2 Q1: Happy Twosday! (15 Points)

February 22, 2022 was a rare day that all five digits of the date, 22222, are twos. The string is both palindrome and symmetric. Recall that a sequence is said to be palindrome if one half of the string is the reverse of the other half, e.g., "madam". A sequence is said to be symmetrical if both halves of the string are the same, e.g., "abcabc". If the length of a sequence is an odd number, the middle element in the sequence is not considered.

1.2.1 Q1.1 Write a code to confirm that February 22 (22222) is palindrome and symmetrical.

[232]:
```
from datetime import date

## IsPalindrome iterates forwards and backwards
## at the same time checking if the elements are
## the same. If they are different, returns false
## and if they are same it returns true. It
## also ignores the middle value
def IsPalindrome(date):
    date = str(date)
    mid = len(date)//2
    for i in range(mid):
        if date[i] != date[len(date)-i-1]:
            return False
    return True
```

```
## IsSymmetric iterates the elements of the first
## and second half of the string at the same time
## in the forward direction, checking if the
## elements are the same. If they are, returns
## True, else it returns False
def IsSymmetric(date):
    date = str(date)
    mid = len(date)//2
    for i in range(mid):
        if date[i] != date[mid + 1 + i]:
            return False
    return True

print(IsPalindrome(22222) and IsSymmetric(22222))
```

```
True
```

1.2.2 Q1.2 There are more dates in February 2022 that are also both palindrome and symmetric.

Write a Python loop to find out all the other dates.

[233]:
```
dates = []

## Iterates through the days of the month
## and if the number is a palindrome and
## symmetric, appends it to the dates array
for i in range(20122, 23122, 100):
    if IsSymmetric(i):
        dates.append(i)
print(dates)
```

```
[22022, 22122, 22222, 22322, 22422, 22522, 22622, 22722, 22822, 22922]
```

1.2.3 Q1.3 Super Palindrome

A number is called a Super Palindrome if it is a palindrome and also a square of another palindrome. For Example, 10201 is a Super Palindrome since 101 is a palindrome and 10201 = 101^2. Write a Python code to find all five-digit super palindromes, that is, numbers from 10000 to 99999.

[234]:
```
## FindSuperPalindrome iterates through the number
## range and tests if each number is a palindrome
## using the previous function, and if it a square.
## If those to criteria are met it appends the number
## to an array
def FindSuperPalindrome():
    sp = []
    for i in range(10000, 99999):
        x = int(i**0.5)
        if IsPalindrome(i) and i == x**2:
            sp.append(i)
    return sp


print(FindSuperPalindrome())
```

```
[10201, 12321, 14641, 40804, 44944, 69696, 94249]
```

1.3 Q2: Python Code Debugging (10 points)

1.3.1 Q2.1 The code below is supposed to count all of the even numbers between 1 and 100 (inclusive) that are also a multiple of 5, but it is incomplete and contains some errors. Please fix the errors and show your execution results

[235]:
```
# while i < 100:
#     if i % 2 or i % 5:
#         total + 1
# print (total)
```

Explain the bugs here - The code never initializes total - I is never initialized or iterated - Uses an **or** in the if statement so only one of the conditions have to be met - total is not properly updated

Revise the code and execute it

[236]:
```
total = 0
i = 1
while i < 100:
    if i % 2 and i % 5:
        total = total + 1
    i += 1
print (total)
```

```
40
```

1.3.2 Q2.2 The code below is supposed to determine if the given list of numbers is sorted. That is, it should return True if each item in the list is less than the next item. Unfortunately, there are a few errors in the code below. Please fix the code and show your execution.

[237]:
```
# def is_sorted(numbers):
#     ''' Return whether or not the list of numbers is sorted '''
#     for i in numbers:
#         if numbers[i] < numbers[i + 1]:
#             print ('True')
#         else:
#             print ('False')
```

```
#
#
# num_list = [1, 2, -3, 6, 7]
# is_sorted(num_list)
```

Explain the bugs here - The i goes out of bounds of the array - The function doesn't return anything, just prints it - Needs to be checking if the list is not in order and return False - Does not need an else statement, just need to return True if it makes it all the way through

Revise the code and execute it

[238]:
```
def is_sorted(numbers):
    ''' Return whether or not the list of numbers is sorted '''
    for i in range(len(numbers)-1):
        if numbers[i] > numbers[i + 1]:
            return False
    return True



num_list = [1, 2, -3, 6, 7]
is_sorted(num_list)
```

[238]:
```
False
```

1.4 Q3: Create a custom module math_operations.py that defines two functions: divide(a, b) and power(a, b). Modify the divide() function to raise a custom exception DivisionByZeroError when the divisor is zero. Write the main program to handle this exception gracefully.

[239]:
```
import math_operations as mo

print(mo.divide(1, 2))
print(mo.divide(1, 0))
print(mo.power(1, 2))
```

```
0.5
Can't divide by zero
None
1
```

1.5 Q4: Write a Python program that takes a list of tuples, where each tuple contains a product name and its price. Write a lambda function to filter out products that cost more than a given threshold, and then sort the remaining products by their price in ascending order.

[240]:
```
grocery = [("eggs", 2.99), ("oranges", 5.99), ("cheese", 19.99), ("bread", 3.
 ↪25)]

f = lambda a : a[1] < 10
new_grocery = [item for item in grocery if f(item)]
new_grocery.sort(key = lambda x : x[1])

print(new_grocery)
```

```
[('eggs', 2.99), ('bread', 3.25), ('oranges', 5.99)]
```

1.6 Q5: Write a Python program that imports the datetime module. Create a function work_week that accepts a start date and calculates the dates of all the weekdays (Monday to Friday) for the next 4 weeks (excluding weekends). Return a list of these dates in YYYY-MM-DD format.

[241]:
```
import datetime as dt

def work_week(day: dt.date):
    Saturday = 5
    Sunday = 6
    weekdays = []
    fourweeks = day + dt.timedelta(weeks=4)

    while day < fourweeks:
        if day.weekday() != Saturday and day.weekday() != Sunday:
            weekdays.append(day.isoformat())
        day = day + dt.timedelta(days=1)
    return weekdays

startday = dt.date(2024, 9, 28)
daylist = work_week(startday)
daylist
```

[241]:
```
['2024-09-30',
 '2024-10-01',
 '2024-10-02',
 '2024-10-03',
 '2024-10-04',
 '2024-10-07',
 '2024-10-08',
 '2024-10-09',
 '2024-10-10',
 '2024-10-11',
 '2024-10-14',
 '2024-10-15',
 '2024-10-16',
 '2024-10-17',
 '2024-10-18',
 '2024-10-21',
 '2024-10-22',
 '2024-10-23',
 '2024-10-24',
 '2024-10-25']
```

1.7 Q6: Write a Python function group_anagrams that takes a list of strings and groups them into lists of anagrams. Two words are anagrams if they contain the same characters in the same frequencies, regardless of order. The function should return a list of lists, where each sublist contains words that are anagrams of each other.

[242]:
```
def group_anagrams(words):
    anagrams = {}

    for word in words:
        sorted_word = ''.join(sorted(word))

        if sorted_word in anagrams:
            anagrams[sorted_word].append(word)
        else:
            anagrams[sorted_word] = [word]

    return list(anagrams.values())


word_list = ["time", "group", "hero", "blast", "cheater", "teacher", "bust",␣
 ↪"stub", "rail", "liar", "tom", "car"]
group_anagrams(word_list)
```

[242]:
```
[['time'],
 ['group'],
 ['hero'],
 ['blast'],
 ['cheater', 'teacher'],
 ['bust', 'stub'],
 ['rail', 'liar'],
 ['tom'],
 ['car']]
```