QB_SOLUTION_6,7,8

January 17, 2024

1 503 Write a function cust_data() to ask user to enter their names and age to store data in customer.txt file.

[1]:
```
def cust_data():
    name=input("Enter customer name:")
    age=int(input("Enter customer age:"))
    data=str([name,age])
    f=open("customer.txt","a")
    f.write(data)
    f.close()
cust_data()
f=open("customer.txt")
r=f.read()
print(r)
```

```
Enter customer name:Vishal
Enter customer age:31
['Vishal', 31]
```

2 504 Write a python program to create and read the city.txt file in one go and print the contents on the output screen.

[3]:
```
f=open("city.txt","w")
f.write("My city is very clean city.")
f.close()
f=open("city.txt","r")
dt=f.read()
print(dt)
f.close()
```

```
My city is very clean city.
```

3 505 Write a function count_lines() to count and display the total number of lines from the file. Consider the following lines for the file – friends.txt.

Friends are crazy, Friends are naughty !

Friends are honest, Friends are best !

Friends are like keygen, friends are like license key !

We are nothing without friends, Life is not possible without friends !

[9]:
```
%%writefile friends.txt
Friends are crazy, Friends are naughty !
Friends are honest, Friends are best !
Friends are like keygen, friends are like license key !
We are nothing without friends, Life is not possible without friends !
```

```
Overwriting friends.txt
```

[10]:
```
def countline(f):
    #file_read=f.readlines()
    count=0
    for lines in f:
        count+=1
    print(count)

f=open("friends.txt")
countline(f)
```

```
4
```

4 506 Write a function display_oddLines() to display odd number lines from the text file. Consider the following lines for the file – friends.txt.

Friends are crazy, Friends are naughty !

Friends are honest, Friends are best !

Friends are like keygen, friends are like license key !

We are nothing without friends, Life is not possible without friends !

[11]:
```
def oddline(f):
    #file_read=f.readlines()
    count=0
    for lines in f:
        if(count%2!=0):
            print(lines)
        count+=1


f=open("friends.txt")
oddline(f)
```

```
Friends are honest, Friends are best !

We are nothing without friends, Life is not possible without friends !
```

5 507 Write a Python program to read a text file and do following: 1. Print no. of words 2. Print no. statements

[12]:
```
def oddline(f):
    #file_read=f.readlines()
    countLine=0
    words=0
    for lines in f:
        countLine+=1
        for word in lines.split():
            if word.isalpha()==True or word.isalnum()==True:
                words+=1
    print(countLine,words)
f=open("friends.txt")
oddline(f)
```

```
4 28
```

6 508 Write a python program that reads a text file and changes the file by capitalizing each character of file.

[13]:
```
%%writefile sample1.txt
vishal is best in python
B2,B7 and D1 are best in python
```

```
Writing sample1.txt
```

[14]:
```
f=open("sample1.txt")
data=f.read()
data=data.upper()
f.close()
f=open("output.txt","w")
for i in  data:
    f.write(i)
f.close()
```

```
f=open("output.txt")
print(f.read())
```

```
VISHAL IS BEST IN PYTHON
B2,B7 AND D1 ARE BEST IN PYTHON
```

7 509 Write a Python program to copy the contents of a file to another file.

[15]:
```
%%writefile 1.txt
vishal acharya
b2
```

```
Writing 1.txt
```

[16]:
```
%%writefile 2.txt
python
b7,d1
```

```
Writing 2.txt
```

[20]:
```
file1=open("1.txt")
file2=open("1copy.txt","w")
for lines in file1:
    file2.write(lines)
file1.close()
file2.close()

f=open("1copy.txt")
print(f.read())
```

```
vishal acharya
b2
```

8 510 Write a python program to read line by line from a given files file1 & file2 and write into file3.

[21]:
```
file1=open("1.txt")
file2=open("2.txt")
file3=open("3.txt","w")
for lines in file1:
    file3.write(lines)
for lines in file2:
    file3.write(lines)
file1.close()
```

```
file2.close()
file3.close()
f=open("3.txt")
print(f.read())
```

```
vishal acharya
b2
python
b7,d1
```

9 511 Write python program to count the number of lines in a file.

[22]:
```
%%writefile friends.txt
Friends are crazy, Friends are naughty !
Friends are honest, Friends are best !
Friends are like keygen, friends are like license key !
We are nothing without friends, Life is not possible without friends !
```

```
Overwriting friends.txt
```

[23]:
```
def countline(f):
    #file_read=f.readlines()
    count=0
    for lines in f:
        count+=1
    print(count)

f=open("friends.txt")
countline(f)
```

```
4
```

10 512 Write a python program to search for a string in text files.

[29]:
```
%%writefile friends.txt
Friends are crazy, Friends are naughty !
Friends are honest, Friends are best !
Friends are like keygen, friends are like license key !
We are nothing without friends, Life is not possible without friends !
```

```
Overwriting friends.txt
```

[36]:
```
f=open('friends.txt')
lines=f.readlines()
for row in lines:
    word='Lif'
    #print(row.find(word))
    if row.find(word)!=-1:
        print('string exists in file')
        print('line Number:',lines.index(row),row.index(word))
        print(row)
f.close()
```

```
string exists in file
line Number: 3 32
We are nothing without friends, Life is not possible without friends !
```

11 513

12 Using a file which consists of multiple statements, find all the words from the file that can be made from all the characters of given user's string.

- Note: If user enters same characters multiple times in a string, then word from file will only be eligible for output if it contains that character for same or more number of times in it. (if file have apple, greenapple and user's string is 'aepe' then output will only be greenapple. apple is not eligible as it contains e for 1 time only.

For example:

If a given file is:

Apple is a fruit which is healthy

Green apple is tasty and sweet

Oranges and bananas are good for health

Fruits are a good source of vitamins

String entered by user: nas

Output words will be: oranges, bananas, vitamins

String entered by user: isi

Output words will be: vitamins

String entered by user: eee

Output words will be: greenapple

[55]:
```
%%writefile example.txt
Apple is a fruit which is healthy
Greenapple is tasty and sweet
Oranges and bananas are good for health
```

```
Fruits are a good source of vitamins
```

```
Overwriting example.txt
```

[57]:
```
def is_word_eligible(word, user_chars):
    word_count = {}
    for char in word:
        if char in word_count:
            word_count[char] += 1
        else:
            word_count[char] = 1

    user_count = {}
    for char in user_chars:
        if char in user_count:
            user_count[char] += 1
        else:
            user_count[char] = 1
   # print(word_count,user_count)
    for char, count in user_count.items():
        if char not in word_count or word_count[char] < count:
            return False

    return True

def find_words_from_file(file_path, user_chars):
    eligible_words = []

    with open(file_path, 'r') as file:


        for statement in file:
            words = statement.split()
            for word in words:
                #print(word)
                if is_word_eligible(word, user_chars):
                    #print(word)
                    eligible_words.append(word)

    return list(eligible_words)
file_path = 'example.txt'  # Replace 'your_file.txt' with the actual path to␣
 ↪your file
user_input = input("Enter a string of characters: ").lower()

result = find_words_from_file(file_path, user_input)

if result:
        print("Output words will be:", ', '.join(result))
else:
        print("No matching words found.")
```

```
Enter a string of characters: isi
Output words will be: vitamins
```

13 515 Write a Python program to reverse the content of a one file and store it in second file and also convert content of second file into uppercase and store it in third file and also count number of Vowels in third file and also print only 2nd line from the content of third file.

Examples:

If data file one contains the following data:

Friends are crazy, Friends are naughty !

Friends are honest, Friends are best !

Output 1:

! tseb era sdneirF ,tsenoh era sdneirF

! ythguan era sdneirF ,yzarc era sdneirF

Output 2:

! TSEB ERA SDNEIRF ,TSENOH ERA SDNEIRF

! YTHGUAN ERA SDNEIRF ,YZARC ERA SDNEIRF

Output 3:

Vowels = 22

Output 4:

! YTHGUAN ERA SDNEIRF ,YZARC ERA SDNEIRF

[72]:
```
%%writefile example1.txt
Friends are crazy, Friends are naughty !
Friends are honest, Friends are best !
```

```
Overwriting example1.txt
```

[78]:
```
def reverse_content(input_file, output_file):
    with open(input_file, 'r') as file:
        content = file.read()

    reversed_content = content[::-1]
    reversed_content=reversed_content.lstrip()
    with open(output_file, 'w') as file:
        file.write(reversed_content)
    with open(output_file,"r") as f:
        print("output1")
        print(f.read())

def uppercase_content(input_file, output_file):
    with open(input_file, 'r') as file:
        content = file.read()

    uppercase_content = content.upper()


    with open(output_file, 'w') as file:
        file.write(uppercase_content)
    with open(output_file,"r") as f:
        print("output2")
        print(f.read())

def count_vowels(input_file):
    with open(input_file, 'r') as file:
        content = file.read()

    vowel_count = sum(1 for char in content if char.lower() in 'aeiou')

    return vowel_count

input_file = 'example1.txt'   # Replace 'input.txt' with the actual path to␣
 ↪your first file
output_file1 = 'output1.txt'
output_file2 = 'output2.txt'

# Reverse content from the first file and write to the second file
reverse_content(input_file, output_file1)

# Convert content of the second file to uppercase and write to the third file
uppercase_content(output_file1, output_file2)

# Count the number of vowels in the third file
vowel_count = count_vowels(output_file2)
print(f"Number of vowels in the third file: {vowel_count}")

# Print only the second line from the content of the third file
with open(output_file2, 'r') as file:
        lines = file.readlines()
        if len(lines) >= 2:
            print(f"Second line from the third file: {lines[1].strip()}")
        else:
            print("The third file does not have a second line.")
```

```
output1
! tseb era sdneirF ,tsenoh era sdneirF
! ythguan era sdneirF ,yzarc era sdneirF
output2
! TSEB ERA SDNEIRF ,TSENOH ERA SDNEIRF
! YTHGUAN ERA SDNEIRF ,YZARC ERA SDNEIRF
Number of vowels in the third file: 22
Second line from the third file: ! YTHGUAN ERA SDNEIRF ,YZARC ERA SDNEIRF
```

14 516 Write a python program to extract a list of all four-letter words that start and end with the same letter from a given text file.

[82]:
```
%%writefile friends.txt
hiih
Friends are crazy, Friends are naughty !
Friends are honest, Friends are best !
Friends are like keygen, friends are like license key !
We are nothing without friends, Life is not possible without friends !
1231 visv
```

```
Overwriting friends.txt
```

[85]:
```
with open("friends.txt", 'r') as file:
    text = file.read()
# Split the text into words
words = text.split()

# Extract four-letter words starting and ending with the same letter
four_letter_words = []
for word in words:
    if len(word) == 4 and word[0].lower() == word[-1].lower():
        four_letter_words.append(word)
if four_letter_words:
    print("List of four-letter words starting and ending with the same letter:")
    for word in four_letter_words:
        print(word)
else:
    print("No matching words found.")
```

```
List of four-letter words starting and ending with the same letter:
hiih
1231
visv
```

15 517 Write a python program to read a text file "Story.txt" and print only word starting with 'I' in reverse order.

Example: If value in text file is : 'INDIA IS MY COUNTRY'

Output will be: 'AIDNI SI MY COUNTRY'

[100]:
```
%%writefile Story.txt
INDIA IS MY COUNTRY
is india ?
```

```
Overwriting Story.txt
```

[112]:
```
f=open("Story.txt")
l=f.readlines()
f.close()
result=""
for i in l:
    k=i.split()
    for j in k:
        if j[0].lower()=="i":
            result+= j[::-1]+" "
        else:
            result+=j+" "
    result=result.strip()+"\n"
print(result.strip("\n"))
```

```
AIDNI SI MY COUNTRY
si aidni ?
```

16 518 Write a Python program to count words, characters and spaces from text file.

Input:

Python is a Easy Subject

OOPs is One of the most

interesting Topic

Output:

No of space: 10

No of word: 13

No of character: 64

[116]:
```
%%writefile python.txt
Python is a Easy Subject
OOPs is One of the most
```

```
interesting Topic
```

```
Overwriting python.txt
```

[118]:
```
with open("python.txt", 'r') as file:
    text = file.read()
# Count words
word_count = len(text.split())
# Count characters
char_count = len(text)-text.count("\n")
# Count spaces
space_count = text.count(' ')
print(f"Number of words: {word_count}")
print(f"Number of characters: {char_count}")
print(f"Number of spaces: {space_count}")
```

```
Number of words: 13
Number of characters: 64
Number of spaces: 10
```

17 519 File Filtering. write all lines of a file1, except those that start with a pound sign (#), the comment character for Python to file2. And display data of file2.

Text file1 content:

Friends are crazy, Friends are naughty !

#Friends are honest, Friends are best !

Friends are like keygen, #friends are like license key !

We are nothing without friends, Life is not possible without friends !

Text file2 shoud be:

Friends are crazy, Friends are naughty !

Friends are like keygen,

We are nothing without friends, Life is not possible without friends !

[119]:
```
%%writefile Friends.txt
Friends are crazy, Friends are naughty !
#Friends are honest, Friends are best !
Friends are like keygen, #friends are like license key !
We are nothing without friends, Life is not possible without friends !
```

```
Overwriting Friends.txt
```

[126]:
```python
f=open("Friends.txt")
d=open("vishal.txt","w")

while(True):
    buff=f.readline()
    if len(buff)!=0:
        if buff[0]=="#":
            continue
        else:

            if "#" in buff:
                for index in range (1,len(buff)):
                    if buff[index]=="#":
                        d.write(buff[0:index]+"\n")
            else:
                d.write(buff)
    else:
        break

f.close()
d.close()
c=open("vishal.txt")
print(c.read())
```

```
Friends are crazy, Friends are naughty !
Friends are like keygen,
We are nothing without friends, Life is not possible without friends !
```

18 520 Write a python program to accept string/sentence from user till the user enters "END". Each string/sentence entered by user should be a newline in file. Save all the lines in file and display only those lines which begin with capital letter.

Example:

Enter Something (for quit enter END):Hi Friends

Enter Something (for quit enter END):how are you all

Enter Something (for quit enter END):I am fine

Enter Something (for quit enter END):hope you all are fine

Enter Something (for quit enter END):END

The Line started with Capital Letters:

Hi Friends

I am fine

[127]:
```
with open("B7.txt", 'w') as file:
    while True:
        user_input = input("Enter a string/sentence (type 'END' to stop): ")

        if user_input.upper() == 'END':
            break

        file.write(user_input + '\n')

# Display lines that begin with a capital letter
with open("B7.txt", 'r') as file:
    lines = file.readlines()

capital_lines = [line.strip() for line in lines if line[0].isupper()]

print("Lines that begin with a capital letter:")
for line in capital_lines:
    print(line)
```

```
Enter a string/sentence (type 'END' to stop): Hi Friends
Enter a string/sentence (type 'END' to stop): how are you all
Enter a string/sentence (type 'END' to stop): I am fine
Enter a string/sentence (type 'END' to stop): hope you all are fine
Enter a string/sentence (type 'END' to stop): END
Lines that begin with a capital letter:
Hi Friends
I am fine
```

[128]:
```
with open("B7.txt", 'w') as file:
    while True:
        user_input = input("Enter a string/sentence (type 'END' to stop): ")

        if user_input.upper() == 'END':
            break

        file.write(user_input + '\n')

# Display lines that begin with a capital letter
with open("B7.txt", 'r') as file:
    lines = file.readlines()

capital_lines=[]
for line in lines:
    if line[0].isupper():
        capital_lines.append(line.strip())
```

```
print("Lines that begin with a capital letter:")
for line in capital_lines:
    print(line)
```

```
Enter a string/sentence (type 'END' to stop): Hi Friends
Enter a string/sentence (type 'END' to stop): how are you all
Enter a string/sentence (type 'END' to stop): I am fine
Enter a string/sentence (type 'END' to stop): hope you all are fine
Enter a string/sentence (type 'END' to stop): END
Lines that begin with a capital letter:
Hi Friends
I am fine
```

19 521 Write a program to compare two text files. If they are different, give the line and column numbers in the files where the first difference occurs.

Example:

File 1: python1.txt

Friends are crazy, Friends are naughty !

Friends are honest, Friends are best !

Friends are like keygen, friends are like license key !

new We are nothing without friends, Life is not possible without friends !

File 2: python2.txt

Friends are crazy, Friends are naughty !

Friends 6re honest, Friends are best !

Friends are like keygen, friends are like license key !

new We are nothing without friends, Life is not possible without friends !

Output:

line number 2 colNo. 9

[130]:
```
%%writefile python1.txt
Friends are crazy, Friends are naughty !
Friends are honest, Friends are best !
Friends are like keygen, friends are like license key !
new We are nothing without friends, Life is not possible without friends !
```

```
Writing python1.txt
```

[131]:
```
%%writefile python2.txt
Friends are crazy, Friends are naughty !
Friends 6re honest, Friends are best !
Friends are like keygen, friends are like license key !
new We are nothing without friends, Life is not possible without friends !
```

```
Writing python2.txt
```

[132]:
```
f1 = open("python1.txt", "r")
f2 = open("python2.txt", "r")

fileOne = f1.readlines()
fileTwo = f2.readlines()

f1.close()
f2.close()



for index,line in enumerate(fileOne):
   if line!= fileTwo[index]:
    for ind,char in enumerate(line):
        if char!=fileTwo[index][ind]:
            print("line number",index+1,"colum",ind+1)
            break
    break
```

```
line number 2 colum 9
```

20 522 Write a Python program to count the frequency of words in a file

Example:

File 1: python1.txt

Friends are crazy, Friends are naughty !

Friends are honest, Friends are best !

Friends are like keygen, friends are like license key !

new We are nothing without friends, Life is not possible without friends !

Output:

Friends - 5,are - 7,crazy, - 1,naughty - 1,! - 4,honest, - 1,best - 1,like - 2,keygen, - 1,friends - 2,license - 1,key - 1,new - 1,We - 1,nothing - 1,without - 2,friends, - 1,Life - 1,is - 1,not - 1,possible - 1,

[138]:
```
f=open("python1.txt")
d={}
for line in f:
    for words in line.split():
        d[words]=d.get(words,0)+1
```

```
#print(d)
for i,j in d.items():
    print(i,"-",j,end=",")
```

```
Friends - 5,are - 7,crazy, - 1,naughty - 1,! - 4,honest, - 1,best - 1,like -
2,keygen, - 1,friends - 2,license - 1,key - 1,new - 1,We - 1,nothing - 1,without
- 2,friends, - 1,Life - 1,is - 1,not - 1,possible - 1,
```

21 540 Write a python program to make a module which contain all the basic functions related to string and import that module in another file and use that fuctions with string given by user.

[140]:
```
%%writefile string_functions.py

def count_vowels(s):
    """Count the number of vowels in a string."""
    vowels = "aeiouAEIOU"
    return sum(1 for char in s if char in vowels)

def reverse_string(s):
    """Reverse a string."""
    return s[::-1]

def capitalize_words(s):
    """Capitalize the first letter of each word in a string."""
    return ' '.join(word.capitalize() for word in s.split())
```

```
Writing string_functions.py
```

[141]:
```
# main.py
import string_functions

# Get a string from the user
user_input = input("Enter a string: ")

# Use functions from the module
vowel_count = string_functions.count_vowels(user_input)
reversed_string = string_functions.reverse_string(user_input)
capitalized_words = string_functions.capitalize_words(user_input)

# Display the results
print(f"Number of vowels: {vowel_count}")
print(f"Reversed string: {reversed_string}")
print(f"Capitalized words: {capitalized_words}")
```

```
Enter a string: vishal is best in python
Number of vowels: 6
```

```
Reversed string: nohtyp ni tseb si lahsiv
Capitalized words: Vishal Is Best In Python
```

22 541 Write a python program to create a directory and sub-directory. It should print the current working directory path and list of names of files present in the given directory.

[142]:
```python
import os

def create_directory_structure():
    # Specify directory names
    main_directory = 'my_directory'
    sub_directory = 'my_subdirectory'

    # Create main directory
    os.makedirs(main_directory)

    # Create subdirectory inside the main directory
    os.makedirs(os.path.join(main_directory, sub_directory))

def print_directory_contents():
    # Print current working directory path
    current_path = os.getcwd()
    print(f"Current working directory: {current_path}")

    # List names of files in the given directory
    files = os.listdir()
    print("List of files in the directory:")
    for file in files:
        print(file)

if __name__ == "__main__":
    create_directory_structure()
    print_directory_contents()
```

```
Current working directory: D:\202324python\T3-PYTHON1\QB
List of files in the directory:
.ipynb_checkpoints
1.txt
1copy.txt
2.txt
3.txt
B7.txt
city.txt
customer.txt
example.txt
example1.txt
```

```
friends.txt
my_directory
output.txt
output1.txt
output2.txt
python.txt
python1.txt
python2.txt
QB_SOLUTION_6,7,8.ipynb
sample1.txt
Story.txt
string_functions.py
vishal.txt
__pycache__
```

23 542 Write a python program to make a module named cal.py which contain all the basic functions related to calculator like addition, subtraction, multiplication, and division import that module in another file and use that functions with number inputs given by user.

[143]:
```
%%writefile cal.py

def add(x, y):
    """Addition function."""
    return x + y

def subtract(x, y):
    """Subtraction function."""
    return x - y

def multiply(x, y):
    """Multiplication function."""
    return x * y

def divide(x, y):
    """Division function."""
    if y != 0:
        return x / y
    else:
        return "Cannot divide by zero"
```

```
Writing cal.py
```

[144]:
```
# main.py
import cal
```

```
# Get number inputs from the user
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
# Use functions from the module
result_add = cal.add(num1, num2)
result_subtract = cal.subtract(num1, num2)
result_multiply = cal.multiply(num1, num2)
result_divide = cal.divide(num1, num2)
# Display the results
print(f"Addition: {result_add}")
print(f"Subtraction: {result_subtract}")
print(f"Multiplication: {result_multiply}")
print(f"Division: {result_divide}")
```

```
Enter the first number: 5
Enter the second number: 4
Addition: 9.0
Subtraction: 1.0
Multiplication: 20.0
Division: 1.25
```

24 543 Write a program to create a module 'first_word.py', which returns the first word of any string passed. Show the working of the module, by calling the module with any suitable example.

Input: 'This is Python Programming'

Output: 'This'

[145]:
```
%%writefile first_word.py

def get_first_word(input_string):
    """Return the first word of a string."""
    words = input_string.split()
    if words:
        return words[0]
    else:
        return "No words in the input string"
```

```
Writing first_word.py
```

[146]:
```
# main.py
import first_word
# Example usage
input_string = "Hello, this is an example string."
result = first_word.get_first_word(input_string)
print(f"The first word of the string is: {result}")
```

```
The first word of the string is: Hello,
```

25 574 Create a class called NumberSet that accepts 2 integers as input, and defines two instance variables: num1 and num2, which hold each of the input integers. Then, create an instance of NumberSet where its num1 is 6 and its num2 is 10. Save this instance to a variable t

[1]:
```python
class NumberSet:
   def __init__(self, num1, num2):
       self.num1 = num1
       self.num2 = num2

t = NumberSet(6, 10)

print(f"t.num1: {t.num1}")
print(f"t.num2: {t.num2}")
```

```
t.num1: 6
t.num2: 10
```

26 575 Create a class called Animal that accepts two numbers as inputs and assigns them respectively to two instance variables: arms and legs. Create an instance method called limbs that, when called, returns the total number of limbs the animal has. To the variable name spider, assign an instance of Animal that has 4 arms and 4 legs. Call the limbs method on the spider instance and save the result to the variable name spidlimbs

[4]:
```python
class Animal:
    def __init__(self, arms, legs):
        self.arms = arms
        self.legs = legs

    def limbs(self):
        return self.arms + self.legs

spider = Animal(4, 4)
spidlimbs = spider.limbs()

print(f"The spider has {spidlimbs} limbs.")
```

```
The spider has 8 limbs.
```

27 576 Write a Python program to create a Vehicle class with max_speed and mileage instance attributes.

[5]:
```python
class Vehicle:
    def __init__(self, max_speed, mileage):
        """Initializes a Vehicle object with maximum speed and mileage."""
        self.max_speed = max_speed
        self.mileage = mileage

# Example usage:
# Create a car with a max speed of 200 km/h and mileage of 15 km/l
car = Vehicle(200, 15)
# Create a bike with a max speed of 100 km/h and mileage of 50 km/l
bike = Vehicle(100, 50)
print("Car's max speed:", car.max_speed)
print("Car's mileage:", car.mileage)
print("Bike's max speed:", bike.max_speed)
print("Bike's mileage:", bike.mileage)
```

```
Car's max speed: 200
Car's mileage: 15
Bike's max speed: 100
Bike's mileage: 50
```

28 577 Write a Python class named Student with two attributes student_name, marks. Modify the attribute values of the said class and print the original and modified values of the said attributes.

[6]:
```python
class Student:
    def __init__(self, student_name, marks):
        self.student_name = student_name
        self.marks = marks

# Create a student object
student1 = Student("vishal", 9)
# Print original attribute values
print(f"Original student name: {student1.student_name}")
print(f"Original marks: {student1.marks}")
# Modify attribute values
student1.student_name = "vishal"
student1.marks = 10
# Print modified attribute values
print(f"Modified student name: {student1.student_name}")
print(f"Modified marks: {student1.marks}")
```

```
Original student name: vishal
Original marks: 9
Modified student name: vishal
Modified marks: 10
```

29 578 Write a Python class named Student with two attributes student_id, student_name. Add a new attribute student_class. Create a function to display the entire attribute and their values in Student class.

[8]:
```
class Student:
   def __init__(self, student_id, student_name):
       self.student_id = student_id
       self.student_name = student_name
       self.student_class = None  # Initially set to None

   def display_details(self):
       """
       Displays all attributes and their values for the Student object.
       """
       print(f"Student ID: {self.student_id}")
       print(f"Student Name: {self.student_name}")
       print(f"Student Class: {self.student_class}")  # Shows None if not set

# Example usage
student1 = Student(10, "vishal")
student1.student_class = "B2"  # Set the student class

student1.display_details()  # Print all details

# Another student
student2 = Student(1, "Vishal10")
student2.display_details()  # Shows None for student class

student2.student_class = "B7"
student2.display_details()  # Now shows updated student class
```

```
Student ID: 10
Student Name: vishal
Student Class: B2
Student ID: 1
Student Name: Vishal10
Student Class: None
Student ID: 1
Student Name: Vishal10
Student Class: B7
```

30 579 Write a Python class named Rectangle constructed by a length and width and a method which will compute the area of a rectangle.

[9]:
```python
class Rectangle:
    # define constructor with attributes: length and width
    def __init__(self, length , width):
        self.length = length
        self.width = width

    # Create Perimeter method
    def Perimeter(self):
        return 2*(self.length + self.width)

    # Create area method
    def Area(self):
        return self.length*self.width

    # create display method
    def display(self):
        print("The length of rectangle is: ", self.length)
        print("The width of rectangle is: ", self.width)
        print("The perimeter of rectangle is: ", self.Perimeter())
        print("The area of rectangle is: ", self.Area())
myRectangle = Rectangle(7 , 5)
print("-----------------------------------")
myRectangle.display()
```

```
-----------------------------------
The length of rectangle is:  7
The width of rectangle is:  5
The perimeter of rectangle is:  24
The area of rectangle is:  35
```

31 580 Write a Python class named Circle constructed by a radius and two methods which will compute the area and the perimeter of a circle.

[10]:
```python
import math

class Circle:
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        """Calculates and returns the area of the circle."""
        return math.pi * self.radius**2

    def perimeter(self):
        """Calculates and returns the perimeter (circumference) of the circle.
 ↪"""
        return 2 * math.pi * self.radius

# Example usage:
circle1 = Circle(5)
print("Area of circle1:", circle1.area())
print("Perimeter of circle1:", circle1.perimeter())

circle2 = Circle(10)
print("Area of circle2:", circle2.area())
print("Perimeter of circle2:", circle2.perimeter())
```

```
Area of circle1: 78.53981633974483
Perimeter of circle1: 31.41592653589793
Area of circle2: 314.1592653589793
Perimeter of circle2: 62.83185307179586
```

32 581 Write a python program to demonstrate the use of try-except-else in Exception handling

[11]:
```
def divide_numbers(x, y):
    """Divides two numbers and handles potential exceptions."""

    try:
        result = x / y
    except ZeroDivisionError:
        print("Error: Cannot divide by zero.")
    else:
        print("The result of", x, "divided by", y, "is", result)

# Example usage
# Output: The result of 10 divided by 2 is 5.0
divide_numbers(10, 2)
# Output: Error: Cannot divide by zero.
divide_numbers(10, 0)
```

```
The result of 10 divided by 2 is 5.0
Error: Cannot divide by zero.
```

33 582 Write a python program to demonstrate the use of raise in Exception handling

[12]:
```
def validate_age(age):
    """Raises a ValueError if the age is not within a valid range (18-120)."""

    if not 18 <= age <= 120:
        raise ValueError("Age must be between 18 and 120.")

def get_user_info():
    """Prompts the user for their age and validates it using the validate_age␣
 ↪function."""

    while True:
        try:
            age = int(input("Enter your age: "))
            validate_age(age)
            return age
        except ValueError as e:
            print(e)

# Example usage
try:
    user_age = get_user_info()
    print("Your age is valid:", user_age)
except ValueError:
    print("An error occurred while validating the age. Please try again.")
```

```
Enter your age: 4
Age must be between 18 and 120.
Enter your age: 12
Age must be between 18 and 120.
Enter your age: 19
Your age is valid: 19
```

34 583 Write a python program to demonstrate the use of custom exceptions in Exception handling.

[13]:
```
class InvalidAgeException(Exception):
   """Raised when an invalid age is encountered."""

class Person:
   def __init__(self, name, age):
       self.name = name
       self.validate_age(age)
```

```
    def validate_age(self, age):
        if not 18 <= age <= 120:
            raise InvalidAgeException("Age must be between 18 and 120.")
        self.age = age

def create_person():
    name = input("Enter name: ")
    age = int(input("Enter age: "))

    try:
        person = Person(name, age)
        print("Person created successfully:", person.name, person.age)
    except InvalidAgeException as e:
        print("Error:", e)

# Example usage
create_person()
```

```
Enter name: vishal
Enter age: 14
Error: Age must be between 18 and 120.
```

35 584 Write a program to build a simple Student Management System using Object Oriented Programming in Python which can perform the following operations:

- accept-This method takes details from the user like name, roll number, and marks for two different subjects.
- display-This method displays the details of every student.
- search-This method searches for a particular student from the list of students. This method will ask the user for roll number and then search according to the roll number
- delete-This method deletes the record of a particular student with a matching roll number.
- update-This method updates the roll number of the student. This method will ask for the old roll number and new roll number. It will replace the old roll number with a new roll number.

The following instructions need to be considered while making a program.

1. Give class name as Student
2. Include methods name as accept, display, search, delete and update. (1 mark for each correct method to be formed).
3. Also form constructor with **init** () method (2 marks for forming constructor).
4. 2 marks for correct object prepared like after deletion of one roll no of student it should update the list with new roll no. and should display it.

The example is just for understanding but logic should be for any n number of students.

For Example:

```
List of Students
Name : A
RollNo : 1
Marks1 : 100
Marks2 : 100
Name : B
RollNo : 2
Marks1 : 90
Marks2 : 90
Name : C
RollNo : 3
Marks1 : 80
Marks2 : 80
Student Found,
Name : B
RollNo : 2
Marks1 : 90
Marks2 : 90
List after deletion
Name : A
RollNo : 1
Marks1 : 100
Marks2 : 100
Name : C
RollNo : 3
Marks1 : 80
Marks2 : 80
List after updation
Name : A
RollNo : 1
Marks1 : 100
```

```
Marks2 : 100

Name : C

RollNo : 2

Marks1 : 80

Marks2 : 80
```

[15]:
```python
class Student:
    students_list = []

    def __init__(self, name, roll_no, marks1, marks2):
        self.name = name
        self.roll_no = roll_no
        self.marks1 = marks1
        self.marks2 = marks2
        Student.students_list.append(self)

    def accept(self):
        name = input("Enter student name: ")
        roll_no = int(input("Enter roll number: "))
        marks1 = int(input("Enter marks for subject 1: "))
        marks2 = int(input("Enter marks for subject 2: "))
        student = Student(name, roll_no, marks1, marks2)
        print(f"Student {name} with Roll No {roll_no} added successfully.")

    def display(self):
        print("\nList of Students")
        for student in Student.students_list:
            print(f"\nName : {student.name}\nRollNo : {student.roll_no}\nMarks1␣
↪: {student.marks1}\nMarks2 : {student.marks2}")

    def search(self):
        roll_no = int(input("Enter the roll number to search: "))
        for student in Student.students_list:
            if student.roll_no == roll_no:
                print("\nStudent Found,\n")
                print(f"Name : {student.name}\nRollNo : {student.
↪roll_no}\nMarks1 : {student.marks1}\nMarks2 : {student.marks2}")
                return
        print(f"\nStudent with Roll No {roll_no} not found.")

    def delete(self):
        roll_no = int(input("Enter the roll number to delete: "))
        for student in Student.students_list:
            if student.roll_no == roll_no:
                Student.students_list.remove(student)
                print("\nList after deletion")
                self.display()
                return
        print(f"\nStudent with Roll No {roll_no} not found.")

    def update(self):
        old_roll_no = int(input("Enter the old roll number: "))
        new_roll_no = int(input("Enter the new roll number: "))
        for student in Student.students_list:
            if student.roll_no == old_roll_no:
                student.roll_no = new_roll_no
                print("\nList after updation")
                self.display()
                return
        print(f"\nStudent with Roll No {old_roll_no} not found.")

# Example Usage
student_system = Student("vishal",1,10,9)

student_system.accept()
student_system.accept()
student_system.accept()

student_system.display()

student_system.search()

student_system.delete()

student_system.update()
```

```
Enter student name: vishal1
Enter roll number: 2
Enter marks for subject 1: 10
Enter marks for subject 2: 9
Student vishal1 with Roll No 2 added successfully.
Enter student name: vishal2
Enter roll number: 3
Enter marks for subject 1: 10
Enter marks for subject 2: 9
Student vishal2 with Roll No 3 added successfully.
Enter student name: vishal2
Enter roll number: 5
Enter marks for subject 1: 10
Enter marks for subject 2: 9
Student vishal2 with Roll No 5 added successfully.

List of Students
```

```
Name : vishal
RollNo : 1
Marks1 : 10
Marks2 : 9

Name : vishal1
RollNo : 2
Marks1 : 10
Marks2 : 9

Name : vishal2
RollNo : 3
Marks1 : 10
Marks2 : 9

Name : vishal2
RollNo : 5
Marks1 : 10
Marks2 : 9
Enter the roll number to search: 5

Student Found,

Name : vishal2
RollNo : 5
Marks1 : 10
Marks2 : 9
Enter the roll number to delete: 3

List after deletion

List of Students

Name : vishal
RollNo : 1
Marks1 : 10
Marks2 : 9

Name : vishal1
RollNo : 2
Marks1 : 10
Marks2 : 9

Name : vishal2
RollNo : 5
Marks1 : 10
Marks2 : 9
```

```
Enter the old roll number: 1
Enter the new roll number: 11

List after updation

List of Students

Name : vishal
RollNo : 11
Marks1 : 10
Marks2 : 9

Name : vishal1
RollNo : 2
Marks1 : 10
Marks2 : 9

Name : vishal2
RollNo : 5
Marks1 : 10
Marks2 : 9
```

36 585 You own a pizzeria named Olly's Pizzas and want to create a Python program to handle the customers and revenue. Create the following classes with the following methods:

Class Pizza containing 1. init method: to initialize the size (small, medium, large), toppings (corn, tomato, onion, capsicum, mushroom, olives, broccoli), cheese (mozzarella, feta, cheddar). Note: One pizza can have only one size but many toppings and cheese. (1.5 marks) Throw custom exceptions if the selects toppings or cheese not available in lists given above. (1 mark) 2. price method: to calculate the prize of the pizza in the following way: - small = 50, medium = 100, large = 200

Each topping costs 20 rupees extra, except broccoli, olives and mushroom, which are exotic and so cost 50 rupees each.

Each type of cheese costs an extra 50 rupees. (1.5 marks)

Class Order containing

1. init method: to initialize the name, customerid of the customer who placed the order (0.5 marks)
2. order method: to allow the customer to select pizzas with choice of toppings and cheese (1 mark)
3. bill method: to generate details about each pizza ordered by the customer and the total cost of the order. (2 marks)

*Note: A customer can get multiple pizzas in one order.

1.5 marks for creating appropriate objects of these classes and writing correct output.

[19]:
```
class InvalidToppingError(Exception):
    pass

class InvalidCheeseError(Exception):
    pass
class Pizza:
    def __init__(self, size, toppings, cheese):
        self.size = size
        self.toppings = toppings
        self.cheese = cheese

    def price(self):
        self.cost = 0
        if self.size == 'small':
            self.cost += 50
        elif self.size == 'medium':
            self.cost += 100
        else:
            self.cost += 200
        topping_prices_20 = ['corn', 'tomato', 'onion', 'capsicum']
        topping_prices_50 = ['mushroom', 'olives', 'broccoli']
        for topping in self.toppings:
            if topping in topping_prices_20:
                self.cost += 20
            else:
                self.cost += 50
        #for cheese
        self.cost += 50 * len(self.cheese)
        return self.cost

class Order:
    def __init__(self, name, customerid):
        self.name = name
        self.customerid = customerid

    def order(self, n):
        self.pizzas = []
        for i in range(n):
            toppings = []
            cheese = []
            print('Customize Pizza',i+1)
            size = input('Select size: ')
            t = int(input('How many toppings: '))
            for i in range(t):
                try:
                    a=input('Enter toppings: ')
                    if a not in ['corn', 'tomato', 'onion',␣
↪'capsicum','mushroom', 'olives', 'broccoli']:
                        raise InvalidToppingError("enter valid toppings")
                    toppings.append(a)
                except InvalidToppingError as e:
                    print(e)
                    a=input('Enter toppings: ')
                    toppings.append(a)
            t = int(input('How many cheese: '))
            for i in range(t):
                try:
                    b=input('Enter cheese: ')
                    if b not in ("mozzarella", "feta", "cheddar"):
                        raise InvalidCheeseError("enter valid cheese")

                except InvalidCheeseError as e:
                    print(e)
                    b=input('Enter cheese: ')
                    cheese.append(b)
                else:
                    cheese.append(b)



            self.pizzas.append(Pizza(size, toppings, cheese))

    def bill(self):
        self.total = 0
        count = 1
        for p in self.pizzas:
            print('Pizza', count)
            print(p.size, p.toppings, p.cheese)
            self.total += p.price()
            count += 1
        print('Total bill amount:', self.total)

number=int(input("How many pizzas you want to order: "))
order1 = Order('VISHAL', number)
order1.order(number)
order1.bill()
```

```
How many pizzas you want to order: 1
Customize Pizza 1
Select size: large
How many toppings: 1
Enter toppings: tomato
How many cheese: 1
```

```
Enter cheese: gh
enter valid cheese
Enter cheese: feta
Pizza 1
large ['tomato'] ['feta']
Total bill amount: 270
```

37 586 Write a class called WordPlay. It should have a constructor that holds a list of words. The user of the class should pass the list of words through constructor, which user wants to use for the class. The class should have following methods:

words_with_length(length) — returns a list of all the words of length length

starts_with(char1) — returns a list of all the words that start with char1

ends_with(char2) — returns a list of all the words that end with char2

palindromes() — returns a list of all the palindromes in the list

only(str1) — returns a list of the words that contain only those letters in str1

avoids(str2) — returns a list of the words that contain none of the letters in str2

Make Required object for WordPlay class and test all the methods.

For Example:

If input list entered by user is: [‘apple’, ‘banana’, ‘find’, ‘dictionary’, ‘set’, ‘tuple’, ‘list’, ‘malayalam’, ‘nayan’, ‘grind’, ‘apricot’]

words_with_length (5) should return [‘apple’, ‘tuple’, ‘nayan’, ‘grind’]

starts_with (‘a’) should return [‘apple’, ‘apricot’]

ends_with (‘d’) should return [‘find’, ‘grind’]

palindromes () should return [‘malayalam’, ‘nayan’]

only (‘bna’) should return [‘banana’]

avoids (‘amkd’) should return [‘set’, ‘tuple’, ‘list’]

[150]:
```
class Wordplay:
    def __init__(self):
        self.words_list = []


    def words_with_length(self,length):
        wordlist = []
        for w in self.words_list:
            if len(w) == length:
                wordlist.append(w)
        return  "Words with length {} {}".format(length,wordlist)

    def start_with(self, s):
        wordlist = []
        for w in self.words_list:
            if (w[0] == s):
                wordlist.append(w)
        return "Words Start with letter {} {}".format(s,wordlist)

    def ends_with(self,s):
        wordlist =[]
        for w in self.words_list:
            if w[-1] == s:
                wordlist.append(w)
        return "Words Ending with letter {} {}".format(s,wordlist)

    def palindrome(self):
        wordlist =[]
        for w in self.words_list:
            if w[::1] == w[::-1]:
                wordlist.append(w)
        return "Palindromes {}".format(wordlist)

    def only_s(self,s):
        s=set(s)
        wordlist =[]
        for w in self.words_list:
            word=set(w)
            if s==word:
                wordlist.append(w)
        return "Words containing L {}".format(wordlist)


    def avoid(self,s):
        s=set(s)
        wordlist = []
        for w in self.words_list:
            word=set(w)
            if word.intersection(s)==set():
                wordlist.append(w)
        return "Words Not containing L {}".format(wordlist)


a = Wordplay()

a.words_list=['apple', 'banana', 'find', 'dictionary', 'set', 'tuple', 'list',␣
 ↪'malayalam', 'nayan', 'grind', 'apricot']
```

```python
print(a.words_with_length(5))
print(a.start_with("a"))
print(a.ends_with("d"))
print(a.palindrome())
print(a.only_s("ban"))
print(a.avoid("amkd"))
```

```
Words with length 5 ['apple', 'tuple', 'nayan', 'grind']
Words Start with letter a ['apple', 'apricot']
Words Ending with letter d ['find', 'grind']
Palindromes ['malayalam', 'nayan']
Words containing L ['banana']
Words Not containing L ['set', 'tuple', 'list']
```

38 587 Write a python program that has class store which keeps record of code and price of each product. Display a menu of all products to the user and prompt him to enter the quantity of each item required. generate a bill and display total amount.

Sample Output:

enter no of items: 3

enter code of item: milk

enter cost of item: 30

enter code of item: apple

enter cost of item: 35

enter code of item: gems

enter cost of item: 40

Item Code Price

milk 30

apple 35

gems 40

Enter quantity of each item:

Enter quantity of milk : 2

Enter quantity of apple : 3

Enter quantity of gems : 4

***********************Bill*********************

ITEM PRICE QUANTITY SUBTOTAL

milk 30 2 60

apple 35 3 105

gems 40 4 160 *************************************** Total= 325

[20]:
```python
class Store:
    def __init__(self):
        self.products = {}

    def add_product(self, code, price):
        self.products[code] = price

    def display_menu(self):
        print("\nItem Code Price")
        for code, price in self.products.items():
            print(f"{code} {price}")

    def generate_bill(self, quantities):
        print("\n****Bill****\n")
        print("ITEM PRICE QUANTITY SUBTOTAL")
        total_amount = 0

        for code, quantity in quantities.items():
            price = self.products[code]
            subtotal = price * quantity
            total_amount += subtotal
            print(f"{code} {price} {quantity} {subtotal}")

        print(f"\nTotal= {total_amount}")


# Example Usage
store = Store()

num_items = int(input("Enter the number of items: "))

for _ in range(num_items):
    code = input("Enter code of item: ")
    cost = float(input("Enter cost of item: "))
    store.add_product(code, cost)

store.display_menu()

quantities = {}
for code in store.products.keys():
    quantity = int(input(f"Enter quantity of {code}: "))
    quantities[code] = quantity
```

```
store.generate_bill(quantities)
```

```
Enter the number of items: 3
Enter code of item: milk
Enter cost of item: 30
Enter code of item: apple
Enter cost of item: 35
Enter code of item: gems
Enter cost of item: 40

Item Code Price
milk 30.0
apple 35.0
gems 40.0
Enter quantity of milk: 2
Enter quantity of apple: 3
Enter quantity of gems: 4

****Bill****

ITEM PRICE QUANTITY SUBTOTAL
milk 30.0 2 60.0
apple 35.0 3 105.0
gems 40.0 4 160.0

Total= 325.0
```

39 588 Write a python program that has a class Point with attributes as the x and y co-ordinates.

1. Add a method 'distance from origin' to class Point which returns the distance of the given point from origin. The equation is
2. Add a method 'translate' to class Point, which returns a new position of point after translation
3. Add a method 'reflect_x' to class Point, which returns a new point which is the reflection of the point about the x-axis.
4. Add a method 'distance' to return distance of the given point with respect to the other point. The formula for calculating distance between A(x1,y1) and B(x2,y2) is

After creating class blueprint run the following test case -

Test Case – Point (1,2)

Distance from origin - 2.23

Translate method - point (1,2) translated by (1,1) increment will be at (2,3) now

Reflect_x Method - Point (2,3) after given reflection will be at (2,-3)

```
Distance Method - distance between point (2,-3) and (3,4) is 1.41
```

[21]:
```
import math

class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def distance_from_origin(self):
        distance = math.sqrt(self.x**2 + self.y**2)
        return round(distance, 2)

    def translate(self, dx, dy):
        new_x = self.x + dx
        new_y = self.y + dy
        return Point(new_x, new_y)

    def reflect_x(self):
        return Point(self.x, -self.y)

    def distance(self, other_point):
        distance = math.sqrt((self.x - other_point.x)**2 + (self.y -␣
 ↪other_point.y)**2)
        return round(distance, 2)

# Test Case
point1 = Point(1, 2)

# Distance from origin
print("Distance from origin:", point1.distance_from_origin())

# Translate method
translation_point = point1.translate(1, 1)
print("Translate method - point (1,2) translated by (1,1):", translation_point.
 ↪x, translation_point.y)

# Reflect_x Method
reflection_point = translation_point.reflect_x()
print("Reflect_x Method - Point (2,3) after given reflection:",␣
 ↪reflection_point.x, reflection_point.y)

# Distance Method
point2 = Point(3, 4)
distance_between_points = reflection_point.distance(point2)
print("Distance Method - distance between point (2,-3) and (3,4):",␣
 ↪distance_between_points)
```

```
Distance from origin: 2.24
Translate method - point (1,2) translated by (1,1): 2 3
Reflect_x Method - Point (2,3) after given reflection: 2 -3
Distance Method - distance between point (2,-3) and (3,4): 7.07
```

40 589 A possible collection of classes which can be used to represent a music collection (for example, inside a music player), focusing on how they would be related by composition. You should include classes for songs, artists, albums and playlists.

- For simplicity you can assume that any song or album has a single "artist" value (which could represent more than one person), but you should include compilation albums (which contain songs by a selection of different artists). The "artist" of a compilation album can be a special value like "Various Artists". You can also assume that each song is associated with a single album, but that multiple copies of the same song (which are included in different albums) can exist.
- Write a simple implementation of this model which clearly shows how the different classes are composed. Write some example code to show how you would use your classes to create an album and add all its songs to a playlist. Class Album should have a method to add track, class Artist should have methods to add album and add song, class Playlist should also have a method to add song.

[22]:
```python
class Song:
    def __init__(self, title, duration):
        self.title = title
        self.duration = duration


class Artist:
    def __init__(self, name):
        self.name = name
        self.albums = []
        self.songs = []

    def add_album(self, album):
        self.albums.append(album)

    def add_song(self, song):
        self.songs.append(song)


class Album:
    def __init__(self, title, artist):
        self.title = title
        self.artist = artist
        self.tracks = []
    def add_track(self, track):
        self.tracks.append(track)


class Playlist:
    def __init__(self, name):
        self.name = name
        self.songs = []

    def add_song(self, song):
        self.songs.append(song)


# Example Usage
# Create artists
artist1 = Artist("Artist1")
artist2 = Artist("Various Artists")

# Create songs
song1 = Song("Song1", 180)
song2 = Song("Song2", 200)
song3 = Song("Song3", 150)

# Create albums
album1 = Album("Album1", artist1)
album1.add_track(song1)
album1.add_track(song2)

album2 = Album("Compilation Album", artist2)
album2.add_track(song2)
album2.add_track(song3)

# Add albums to artists
artist1.add_album(album1)
artist2.add_album(album2)

# Create a playlist and add songs
playlist1 = Playlist("My Playlist")
playlist1.add_song(song1)
playlist1.add_song(song3)

# Display information
print("Artist1's Albums:")
for album in artist1.albums:
    print(f"- {album.title}")
print("\nVarious Artists' Compilation Album:")
for track in album2.tracks:
    print(f"- {track.title}")

print("\nPlaylist 'My Playlist':")
for song in playlist1.songs:
    print(f"- {song.title}")

# Note: This is a simplified example. In a real-world scenario, you may have␣
↪more features and complexity.
```

```
Artist1's Albums:
- Album1

Various Artists' Compilation Album:
- Song2
- Song3

Playlist 'My Playlist':
- Song1
- Song3
```

[23]:
```
class Song:

    def __init__(self, title, artist, album, track_number):
        self.title = title
        self.artist = artist
        self.album = album
        self.track_number = track_number

        artist.add_song(self)


class Album:

    def __init__(self, title, artist, year):
        self.title = title
        self.artist = artist
        self.year = year

        self.tracks = []

        artist.add_album(self)

    def add_track(self, title, artist=None):
        if artist is None:
            artist = self.artist
        track_number = len(self.tracks)

        song = Song(title, artist, self, track_number)

        self.tracks.append(song)


class Artist:
    def __init__(self, name):
        self.name = name

        self.albums = []
        self.songs = []

    def add_album(self, album):
        self.albums.append(album)

    def add_song(self, song):
        self.songs.append(song)


class Playlist:
    def __init__(self, name):
        self.name = name
        self.songs = []

    def add_song(self, song):
        self.songs.append(song)

band = Artist("Bob's Awesome Band")
album = Album("Bob's First Single", band, 2013)
album.add_track("A Ballad about Cheese")
album.add_track("A Ballad about Cheese (dance remix)")
album.add_track("A Third Song to Use Up the Rest of the Space")

playlist = Playlist("My Favourite Songs")

for song in album.tracks:
    playlist.add_song(song)
```

41 590 Stacks and Queues. Write a class SQ that defines a data structure that can behave as both a queue (FIFO) or a stack (LIFO), There are five methods that should be implemented:

1. make a constructor with a valid parameter
2. shift() returns the first element and removes it from the list. Also, use the custom(raise) exception in this method.
3. unshift() "pushes" a new element to the front or head of the list
4. push() adds a new element to the end of a list
5. pop() returns the last element and removes it from the list
6. remove() returns the maximum element of the list and removes it from the list.
7. Create the object and call all methods of the SQ class.

[28]:
```
class StackQueue:
    def __init__(self, L):
        self.L = L
    def shift(self):
        if len(self.L) == 0:

            raise Exception('List is empty')
        try:
            x = self.L.pop(0)
            return x
        except Exception:
            print(Exception)
    def unshift(self, n):
        self.L.insert(0, n)
    def push(self, n):
        self.L.append(n)
    def pop(self):
        try:
            if len(self.L) == 0:
                raise Exception('List is empty')
            x = self.L.pop()
            return x
        except Exception as e:
            print(e)
    def remove(self):
        a = max(self.L)
        print(a)
        self.L.remove(a)
    def display(self):
        return self.L

sq = StackQueue([1,4,6,8,9])
print(sq.shift())
```

```python
sq.unshift(7)
sq.push(10)
print("max")
sq.remove()
print(sq.pop())
print(sq.display())
print(sq.pop())
print(sq.display())
print(sq.pop())
print(sq.pop())
print(sq.pop())
print(sq.pop())
print(sq.display())
```

```
1
max
10
9
[7, 4, 6, 8]
8
[7, 4, 6]
6
4
7
List is empty
None
[]
```