**DAY7-**PROGRAMMING FUNDAMENTALS USING PYTHON

**Assignment 46:**

Write a python function, **nearest\_palindrome()** which accepts a number and returns the nearest palindrome greater than the given number.  
Also write the pytest test cases to test the program.

|  |  |
| --- | --- |
| **Sample Input** | **Expected Output** |
| 12300 | 12321 |
| 12331 | 12421 |

#PF-Assgn-46

def nearest\_palindrome(number):

#start writitng your code here

while True :

revnum= str(number)

revnum=revnum[::-1]

revnum=int(revnum)

if number == revnum:

return number

else:

number+=1

number=12300

print(nearest\_palindrome(number))

**Assignment 47:**

Write a python function, **encrypt\_sentence()** which accepts a message and encrypts it based on rules given below and returns the encrypted message.  
Words at odd position -> Reverse It  
Words at even position -> Rearrange the characters so that all consonants appear before the vowels and their order should not change  
**Note**:

1. Assume that the sentence would begin with a word and there will be only a single space between the words.
2. Perform case sensitive string operations wherever necessary.

Also write the pytest test cases to test the program.

|  |  |
| --- | --- |
| **Sample Input** | **Expected Output** |
| the sun rises in the east | eht snu sesir ni eht stea |

#PF-Assgn-47

def encrypt\_sentence(sentence):

#start writing your code here

vowel=set("AEIOUaeiou")

s1=sentence

var=""

li = list(sentence.split(" "))

for i in range(len(li)):

var=li[i]

if(i%2==0):

var=var[::-1]

li[i]=var

else:

t=""

t2=""

for j in var:

if(j in vowel):

t2=t2+j

else:

t=t+j

t=t+t2

li[i]=t

var2=""

for i in range(len(li)):

var2=var2+li[i]

if(i != len(li)-1):

var2=var2+" "

return var2

sentence="The sun rises in the east"

encrypted\_sentence=encrypt\_sentence(sentence)

print(encrypted\_sentence)

Assignment 48:

Write a python function,**find\_correct()** which accepts a dictionary and returns a list as per the rules mentioned below.  
The input dictionary will contain correct spelling of a word as key and the spelling provided by a contestant as the value.  
The function should identify the degree of correctness as mentioned below:  
CORRECT, if it is an exact match  
ALMOST CORRECT, if no more than 2 letters are wrong  
WRONG, if more than 2 letters are wrong or if length (correct spelling versus spelling given by contestant) mismatches.  
and return a list containing the number of CORRECT answers, number of ALMOST CORRECT answers and number of WRONG answers.  
Assume that the words contain only uppercase letters and the maximum word length is 10.  
Also write the pytest test cases to test the program.

|  |  |
| --- | --- |
| **Sample Input** | **Expected Output** |
| {"THEIR": "THEIR", "BUSINESS": "BISINESS","WINDOWS":"WINDMILL","WERE":"WEAR","SAMPLE":"SAMPLE"} | [2, 2, 1] |

#PF-Assgn-48

def find\_correct(word\_dict):

#start writing your code here

c1,c2,c3,cou=0,0,0,0

for i in word\_dict:

if(word\_dict[i]==i):

c1=c1+1

elif(len(word\_dict[i])!=len(i)):

c3=c3+1

else:

x=word\_dict[i]

for j in range(len(x)):

if(x[j]!=i[j]):

cou=cou+1

if(cou<=2):

c2+=1

else:

c3+=1

cou=0

list=[c1,c2,c3]

return list

word\_dict={"THEIR": "THEIR","BUSINESS":"BISINESS","WINDOWS":"WINDMILL","WERE":"WEAR","SAMPLE":"SAMPLE"}

print(find\_correct(word\_dict))

Assignment 49:

In a fair coin we have an equal chance (50%) of either getting a ‘head’ or ‘tail’.  That is if we toss the coin a large number of times we would observe head approximately 50% of the time. Write a program to implement a biased coin toss where the chance of getting a head is 70% (and tail 30%). That is if we invoke the program 1000 times we should see the head randomly approximately 700 times.

#PF-Tryout

#Start writing your code here

import random

head,tail=0,0

for i in range(1000):

if random.random()<0.7:

head=head+1

else:

tail=tail+1

print("Heads: "+str(head))

print("Tails: "+str(tail))

Assignment 50:

Write python function, **sms\_encoding()** which accepts a sentence and converts it into an abbreviated sentence to be sent as SMS and returns the abbreviated sentence.  
Rules are as follows:  
a. Spaces are to be retained as is  
b. Each word should be encoded separately

* If a word has only vowels then retain the word as is
* If a word has a consonant (at least 1) then retain only those consonants

**Note:**Assume that the sentence would begin with a word and there will be only a single space between the words.

|  |  |
| --- | --- |
| **Sample Input** | **Expected Output** |
| I love Python | I lv Pythn |
| MSD says I love cricket and tennis too | MSD sys I lv crckt nd tnns t |
| I will not repeat mistakes | I wll nt rpt mstks |

#PF-Assgn-50

def sms\_encoding(data):

#start writing your code here

vowels=set("AEIOUaeiou")

st=""

word=data.split()

for i in word:

if(len(i)==1):

st=st+i

else:

for j in i:

if j not in vowels:

st=st+j

st=st+" "

return st[0:-1]

data="I love Python"

print(sms\_encoding(data))

Assignment 51 :

