Problem Solving And Programming

Date 12June 2019

Day Objectives:-

- String Slicing
- Functions in Python
- Basic Problems related to conditional statements using functions
- Iteration in Python
- Practise more problems

In [ ]:

String Slicing

In [2]:
```
s1= "Python"

s1[0] #Accessing the first character in a string

s1[1] # Accessing the second character in a string

s1[len(s1)-1] # Accessing the last character in a string

# Another way of accessing the last character is
s1[-1]


s1[-2]#  Second character from the last character  or Accessing the penultimate c

s1[0:3] # Accessing the 2 characters

s1[-2:] # Accessing the last two characters in a string

s1[0:] # Accessing the entire word

#s1[4:] # it will help in  string of the letters we know  or it is accessing all

s1[len(s1)-2:] # string of any length


# Accessing all characters except firt and last charcter

s1[1:len(s1)-1] # Accessing the all the characters except first and last characte

s1[1:-1] # another way of accessing the characters except first and last characte

s1[len(s1)//2] # middle character printing

s1[-1::-1]# Reverse of a string

s1[-1:-3:-1]#Access last two characters in reverse order


s1[len(s1)//2:len(s1)//2-2:1]# Reverse the middle two characters in an even lengt

s1[::2]
            #  "python" Accessing alterante characters in a string  in reverse ar

s1[::-2]
```

Out[2]: 'nhy'

In [ ]:

Functions

In [3]:
```
# Function to reverse a string
def reverseString(s):
    return s[::-1]

reverseString("Python")
```

Out[3]: 'nohtyP'

In [4]:
```
#Functions to check if a string is a Palindrome
def Palindrome(s):
    if s == s[::-1]:
        return True
    else:
        return False
s=input("Enter string")
Palindrome(s)
```

```
Enter stringa
```

Out[4]: True

In [1]:
```
# Function check  if given year is a leap year
def isleap(year):
    if(year%400==0 or year%100!=0 and year%4==0):
        return True
    return False
year=int(input("Enter year"))
isleap(year)
```

```
Enter year1200
```

Out[1]: True

In [2]:
```
#Function to count the number of digits in a given number
def count(n):
    return len(str(n))
n=int(input("enter the number"))
count(n)
```

```
enter the number1234
```

Out[2]: 4

In [3]:
```
# Function to identify the greatest of 4 numbers
def greatest4(n1,n2,n3,n4):
    if n1>n2 and n1>n3 and n1>n4:
        return n1
    elif n2>n3 and n2>n4:
        return n2
    elif n3>n4:
        return n3
    return n4
n1=int(input("enter n1"))
n2=int(input("enter n1"))
n3=int(input("enter n1"))
n4=int(input("enter n1"))
greatest4(n1,n2,n3,n4)
```

```
enter n11
enter n12
enter n13
enter n14
```

Out[3]: 4

Iteration

- for
- while

In [4]:
```
# Function to print n numbers
def printNNaturalNumbers(n):
    for i in range (1,n+1):
        print(i,end=" ")
    return
    print()
n=int(input("enter n"))
printNNaturalNumbers(n)
#printNNaturalNumbers(n)
```

```
enter n5
1 2 3 4 5 1 2 3 4 5
```

In [5]:
```
# Function to print N natural numbers

def Natural(n):
    counter=1
    while counter<=n:
        print(counter,end=" ")
        counter=counter+1
    return
n=int(input("enter n"))
Natural(n)
```

```
enter n6
1 2 3 4 5 6
```

In [6]:
```
# Functions to print all numbers divisible by 6 and not a factor of 100

def divby6(lb,ub):
    for i in range(lb,ub):
        if(i%6==0 and lb%100!=0):
            print(i,end=" ")
    return
lb=int(input("Enter lower bound:"))
ub=int(input("Enter upper bound"))
divby6(lb,ub)
```

```
Enter lower bound:1
Enter upper bound10
6
```

In [7]:
```
# Function to find the average of all even numbers in a given range (lb,ub)
def avgeven(lb,ub):
    count=0
    sum=0
    for i in range(lb,ub):
        if(i%2==0):
            sum=sum+i
            count=count+1
    avg=sum/count
    print(avg)
    return
lb=int(input("Enter lower bound:"))
ub=int(input("Enter upper bound"))
avgeven(lb,ub)
```

```
Enter lower bound:1
Enter upper bound20
10.0
```

In [8]:
```
# Function to find the average of cubes of all even numbers  in a given range(lb,
def avgeven(lb,ub):
    count=0
    sum=0
    for i in range(lb,ub):
        if(i%2==0):
            k=i*i*i
            sum=sum+k
            count=count+1
    avg=sum//count
    print(avg)
    return
lb=int(input("Enter lower bound:"))
ub=int(input("Enter upper bound"))
avgeven(lb,ub)
```

```
Enter lower bound:23
Enter upper bound34
22624
```

In [9]:
```
# Functions to generate the list of factors for a given number 12---->1 2 3 4 6 1
def factors(n):
    sum=0
    for i in range (1,n+1):
        if(n%i==0):
            print(i,end=" ")
            sum=sum+i
    return sum
n=int(input("Enter n"))
factors(n)
```

```
Enter n12
1 2 3 4 6 12
```

Out[9]:
```
28
```

In [1]:
```
# Functions to calculate the factorial of a given number
def factorial(n):
    fact=1
    for i in range(1,n+1):
        fact=fact*n
        n=n-1

    print(fact)
    return
n=int(input("Enter n"))
factorial(n)
```

```
Enter n5
120
```

In [11]:
```
# Function to check if a given number is Prime
def primecheck(n):
    c=0
    for i in range(1,n+1):
        if(n%i==0):
            c=c+1
    if(c==2):
        print("Prime")
    else:
        print("not Prime")
    return
#n=int(input("Enter n"))
primecheck(3)
```

```
Prime
```

In [ ]:
```
# Function to calculate the avearage first N Prime numbers
def primeavg(n):
    primeCount=0
    sum=0
    seqCount=2
    while(primeCount<n):
        if primecheck(seqCount):
            primeCount +=1
            sum+=seqCount
        seqCount+=1
    return sum/n
n=int(input("Enter n"))
primeavg(n)
```

In [ ]:
```
##### Function to generate all perfect numbers in a given range

def isperfect(n):
    if factors(n)==n:
        return True
    return False
def generatePerfect(lb,ub):
    for i in range(lb,ub):
        if isperfect(i):
            print(i,end=" ")
    return
generatePerfect(1,100)
```

In [4]:
```
# Function to generate all perfect numbers in a given range

def prefectrange(lb,ub):
    count=0
    sum=0
    for i  in range (lb,ub):
        sum=0
        for j in range (1,i+1):
            if(i%j==0):
                sum=sum+j
        if(sum==i):
            print(i,end=" ")
    return
lb=int(input("Enter lower bound:"))
ub=int(input("Enter upper bound"))
prefectrange(lb,ub)
```

```
Enter lower bound:1
Enter upper bound10
1
```

In [5]:
```
#  Function to calculate the avearage first N Prime numbers
def avgprime(lb,ub,n):
    count=0
    sum=0
    fact=0
    while(fact==n):
        for i in range (lb,ub):
            for j in range (1,i+1):
                if(i%j==0):
                    count=count+1
            if(count==2):
                sum=sum+i
                fact=fact+1
        print(sum)
    #return sum/n
n=int(input("Enter n"))

lb=int(input("Enter lower bound:"))
ub=int(input("Enter upper bound"))
avgprime(lb,ub,n)
```

```
Enter n3
Enter lower bound:1
Enter upper bound10
```

In [ ]:
```
# def prime(u):
    for j in range(u):
        count=0
        n=j
        for i in range(1,n+1):
            if(n%i==0):
                count+=1
        if(count==2):
               print(n)

prime(20)
```

In [11]:
```
# Advanced Problem Set (Optional)
# Function to calculate average of all factorials in a given range
# Function to generate N odd arm strong numbers
# Function to generate Multiplication table for a number in a given range
      #10 in the range (100,102)  inclusive
      #10*100=1000
      #10*101=1010
      #10*102=1020


def mul_table(lb,ub,n):
    for i in range(lb,ub,n):
        print(n,'X',i, "=" ,n*i)
    return
lb=int(input("Enter lower bound:"))
ub=int(input("Enter upper bound"))
n=int(input("enter n"))
mul_table(lb,ub,n)
```

```
Enter lower bound:100
Enter upper bound102
enter n10
10 X 100 = 1000
```

In [12]:
```
# function to print the alternate values in a range
# [500,550]--> 500 502 ......550
# (500,550)--> 501 503 503.....549
#range(500,550)---> 500 501 502....549
# All set based functions in Python have start value



def alternateValues(lb,ub):
    for i in range(lb,ub+1,2):
        print(i,end=" ")
    return
lb=int(input("Enter lower bound:"))
ub=int(input("Enter upper bound"))
alternateValues(lb,ub)
```

```
Enter lower bound:100
Enter upper bound120
100 102 104 106 108 110 112 114 116 118 120
```

In [27]:
```
 def avg_factors(lb,ub):
    count=0
    sum=0
    avg=0
    for i in range (lb,ub):
        sum=0

        for j in range (1,i+1):
            if(i%j==0):
                sum=sum+j
                count=count+1
    avg=sum/count
    print(avg)
    return
lb=int(input("Enter lower bound:"))
ub=int(input("Enter upper bound"))
avg_factorials(lb,ub)
```

```
Enter lower bound:1
Enter upper bound10
3.0
```

In [16]:
```
def Sfactorial(n):
    fact=1
    for i in range (1,n+1):
        fact=fact*i
    return fact
#n=int(input("enter n"))
#Sfactorial(n)
#def factorials(lb,ub):
```

```
enter n6
```

Out[16]:
```
720
```

In [39]:
```
def factorialavg(lb,ub):
    count=0
    sum=0
    f=1
    for i in range(lb,ub+1):
        f=f*i
        print(f,end=" ")
        sum=sum+f
        count+=1
    avg=sum/count
    return avg
lb=int(input("Enter start range"))
ub=int(input("Enter end range"))
factorialavg(lb,ub)
```

```
Enter start range1
Enter end range5
1 2 6 24 120
```

Out[39]:
```
30.6
```

In [17]:
```
# Function to print reverse of given range in the same  line
def reverse(lb,ub):
    for i in range(ub,lb-1,-1):
        print(i,end=" ")
        #i=i-1
    return
lb=int(input("Enter lower bound:"))
ub=int(input("Enter upper bound"))
reverse(lb,ub)
```

```
Enter lower bound:1
Enter upper bound10
10 9 8 7 6 5 4 3 2 1
```

In [30]:
```
# Function to print odd numbers in reverse order in a range
def reverse_odd(lb,ub):
    for i in range(ub,lb,-1):
        if(i%2!=0):
            print(i,end=" ")
    return
lb=int(input("Enter lower bound:"))
ub=int(input("Enter upper bound:"))
reverse_odd(lb,ub)
```

```
Enter lower bound:1
Enter upper bound:10
9 7 5 3
```

In [ ]:
```

```

In [65]:
```
# Function to calculate the sum of numbers in a range
def suminarange(lb,ub):
    sum=0
    count=0
    for i in range(lb,ub+1):
        sum=sum+i
        count+=1
    return sum
```

In [18]:
```
def sumcount(lb,ub):
    count=0
    for i in range(lb,ub+1):
        count+=1
    return count
```

In [67]:
```
# Function to calculate the average of a given range
suminarange(1,5)
```

Out[67]: 15

In [68]:
```
c=sumcount(1,5)
```

In [69]:
```
s=suminarange(1,5)
```

In [70]:
```
s//c
```

Out[70]: 3

In [20]:
```
# Function to calculate the average of a given range
def avgrange(lb,ub):
    sum=0
    count=0
    for i in range(lb,ub+1):
        sum=sum+i
        count+=1
    avg=sum/count
    return avg
lb=int(input("enter start range"))
ub=int(input("enter end range"))
avgrange(lb,ub)
```

```
enter start range1
enter end range5
```

Out[20]:
```
3.0
```

In [41]:
```
# Function to generate all leap years in a given time period
# 2000 -2020---> 2000 2004.......2020

def isLeap(n):
    if(n%400==0 or n%100!=0 and n%4==0):
        return True
    else:
        return False
#n=int(input("enter year"))
#isLeap(n)

def generateLeapYears(lb,ub):
    for i in range(lb, ub+1):
        if isLeap(i):
            print(i,end=" ")
lb=int(input("enter start range"))
ub=int(input("enter end range"))
generateLeapYears(lb,ub)
```

```
enter start range2000
enter end range2005
2000 2004
```

In [26]:
```
# Function to calculate number of days in a given time period using Leapyear
def no_of_days(lb,ub):
    sum=0
    for i in range (lb,ub+1):
        if isLeap(i):
            sum=sum+366
        else:
            sum=sum+365
    return sum
lb=int(input("enter start range"))
ub=int(input("enter end range"))
no_of_days(lb,ub)
```

```
enter start range2000
enter end range2005
```

Out[26]: 2192

In [28]:
```
# Function to caluculate number of hours for a given period
def no_of_hours(sm,sy,em,ey):
    k=(31+30)*24*60
    return k
no_of_hours(5,2019,6,2019)
```

Out[28]: 87840

In [16]:
```
# Function to generate the armstrong numbers in a given range
def  armstrong(n):
    rem=0
    t=n
    while(n>0):
        n=n%10
        rem=rem+n**3
        n=n/10
        if(rem==n):
            print(rem)
armstrong(153)
```

In [ ]:
```
# Function to calculate the avg prime in range for given n
```

In [8]:
```
# Function to find the no of hours in a given time period

def isLeapYear(year): # To check if a given year is a Leap Year
    if year % 400 == 0 or (year % 100 != 0 and year % 4 == 0):
        return True
    return False




def numberOfDays(startyear, endyear):
    sum = 0
    for year in range(startyear, endyear+1):
        if isLeapYear(year):
            sum = sum + 366
        else:
            sum = sum + 365
    return sum
#number of days in middle years of 2016 2019
numberOfDays(2017, 2018)

def numberOfDaysMonth(month, year):
    if month == 2:
        if isLeapYear(year):
            return 29
        return 28
    elif (month <= 7 and month % 2!= 0) or (month >= 8 and month % 2 == 0):
        return 31
    else:
        return 30

def daysInStartYear(startmonth, startyear):
    days = 0
    for month in range(startmonth, 13):
        days += numberOfDaysMonth(month, startyear)
    return days

def daysInEndYear(endmonth, endyear):
    days = 0
    for month in range(1, endmonth+1):
        days += numberOfDaysMonth(month, endyear)
    return days

def numberOfHours(startmonth, startyear, endmonth, endyear):
    days = 0
    if startyear != endyear:
        days += daysInStartYear(startmonth, startyear)
        days += daysInEndYear(endmonth, endyear)
        if endyear - startyear == 2: # 2019 - 2017
            days += numberOfDays(startyear+1, startyear+1)
        elif endyear - startyear > 2:
            days += numberOfDays(startyear+1, endyear-1)
    else:
        for month in range(startmonth, endmonth+1):
            days += numberOfDaysMonth(month, startyear)
    return 24 * days
```

```
numberOfHours(11, 1975,3, 2018)
```

Out[8]:
```
371808
```

In [2]:
```
def isPrime(n):
    flag = True
    for i in range(2, n//2+1):
        if n % i == 0:
            flag = False
            return flag
    return flag

#isPrime(4)


def avgNPrimes(n):
    primeCount = 0
    sum = 0
    seqCount = 2
    while(primeCount < n):
        if isPrime(seqCount):
            primeCount += 1
            sum += seqCount
        seqCount += 1
    return sum/n

n=int(input("enter n"))
avgNPrimes(n)
```

```
enter n3
```

Out[2]:
```
3.3333333333333335
```

In [93]:
```
def isPrime(n):
    c=0
    for i in range(1, n+1):
        if n % i== 0:
            c=c+1
    if(c==2):
        print(n)
n=int(input("enter n"))
isPrime(n)
```

```
enter n7
7
```

In [ ]:
```

```