Problem Solving and Programming

Date 12 June 2019

Day Objectives

- String Slicing
- Functions in Python
- Basic Problems related to conditional statements using functions
- Iteration in Python
- Python Data Structures - Lists, Tuples and Dictionaries
- Basic Operations on data structures
    - Applying Data Structures to solve problems

In [ ]:
```

```

String Slicing

In [46]:
```python
s1 = 'Python'

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

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

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

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

s1[-2]  # Accessing the penultimate (from last the second one) character of

s1[0:2] # Accessing last two characters in this '0' is inclusive(that is fir

s1[-2:]  #Accessing the last two charcters in a string in any string length

s1[:-2]  #Accessing the whole string excluding the last two characters

s1[4:]  #Accessing the last two character if the above string length  or Acc

# Accessing all character except first and last character

s1[1:-1]

s1[1:] #Accessing all characters except the first one

# Accessing the middle character in odd length a string

s1[len(s1)//2]

# Reverse of a string

s1[-1::-1] # first part starting point second part end point third part is i


s1[-1:-3:-1] # Accessing last two characters in reverse oreder

# Reverse the middle two characters in an even length string

s1[-3:-5:-1]

#s1[len(s1)//2:len(s2)//2:-2:-2]

# Accessing alternate characters in a string
# "Python" --> "Pto"

s1[::2]


# Accessing alternate characters of a string in reverse order
# "Python" --> "nhy"

s1[::-2]

```

Out[46]:
```
'nhy'
```

In [ ]:
```python

```

In [ ]:
```python

```

Functions

In [48]:
```python
# Fuction to reverse a string

def reverseString(s):                    # defining a function
    return s[::-1]

reverseString("Python")
```

Out[48]:
```
'nohtyP'
```

In [ ]:
```python

```

In [59]:
```python
# Function to check if a string is a palindrome
def palindrome(s):
    if s == s[::-1]:
        return True
    else:
        return False

palindrome("madam")     # True
palindrome("123321")   # True
palindrome("racecar") # True
palindrome("r")   # True
palindrome(" ")  # True
palindrome("cc") # True
palindrome("Madam")   #False

```

Out[59]:
```
False
```

In [64]:
```python
# Function to check if a given year is a leap year

def leapYear(n):
    if n%400==0 or (n%100!=0 and n%4==0):
        return True
    return False
print(leapYear(2020))    # TRUE
print(leapYear(1234))     # FALSE

```

```
True
False
```

In [58]:
```
# Function to count the number of digits in a given number

def noOfDigitsInNumber(n):
    return len(str(n))
noOfDigitsInNumber(1233)
noOfDigitsInNumber(123345566)
```

Out[58]:
```
9
```

In [7]:
```
# Function to identify the greatest of 4 numbers

def greatestOfGiven(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
greatestOfGiven(1, 234, 456,34)

```

Out[7]:
```
456
```

In [ ]:
```

```

Iteration

- for
- while

In [59]:
```
# Function to print N natural numbers

def printNNaturalNumbers(n):
    for counter in range(1,n+1):
        print(counter, end=" ")   # if here we are not used end=" " here the
    return
printNNaturalNumbers(30)
print()
printNNaturalNumbers(23)
```

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 3
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
```

In [29]:
```
# Function to print N Natural numbers

def nNaturalNumbers(n):
    counter = 1
    while counter <= n:
        print(counter, end = " ")
        counter = counter + 1
    return


nNaturalNumbers(9)
```

```
1 2 3 4 5 6 7 8 9
```

In [1]:
```
 # Function to print all numbers divisible by 6
 # and not a factor of 100 in a given range(lb, ub) inclusive

def factorOf100andDivisibleBy6(lb, ub):
    for i in range(lb,ub+1):
        if 100%i!=0 and i%6==0:
            print(i,end=" ")

    return
factorOf100andDivisibleBy6(115,180)

```

```
120 126 132 138 144 150 156 162 168 174 180
```

In [19]:
```
# Function to find the average of cubes of all even numbers
# in a given range(lb,ub) inclusive
def avgOfCubesOfAllEvenInaRange(lb,ub):
    sum=0
    i=0
    for lb in range(lb,ub+1):
        if lb%2==0:
            sum=sum+lb**3
            print(sum)   #print step by step sum value for understand
            i = i + 1
    print(sum//i)  #(sum/i) for
    return
avgOfCubesOfAllEvenInaRange(2,10)




```

```
8
72
288
800
1800
360
```

In [34]:
```
# Functions to generate the list of factors for a given number
# 12 -> 1 2 3 4 6 12
def factorOfnumber(n):
    for i in range(1,n+1):
        if n%i==0:
            print(i,end=" ")
    return
factorOfnumber(12)

```

```
1 2 3 4 6 12
```

In [54]:
```
# Function to calculate the factorial of a given number
def factorialOfNumber(n):
    f=1
    for i in range(1,n+1):
        f=f*i
    return f
factorialOfNumber(5)
```

Out[54]:
```
120
```

In [98]:
```
# Function to check if a given number is Prime
def isPrime(n):
    i=1
    count=0
    for i in range(i,n+1):       # for i in range(i, n//2): can also we can
        if n%i==0:
            count += 1   # --> count=count+1
    if count==2:
        return True
   # else:
     #   return False
       # print(" n is prime")
   # else:
       # print("n is not a prime")

isPrime(11)
```

Out[98]:
```
True
```

In [103]:
```
# Function to calculate the average first N Prime numbers
def avgNPrimes(n):
    primeCount = 0
    sum = 0
    seqCount = 2
    while(primeCount < n):
        if isPrime(seqCount):
            primeCount += 1
            sum += seqCount
            # print(seqCount)
        seqCount +=1
    return sum/n
avgNPrimes(10)

```

Out[103]:
```
12.9
```

In [112]:
```
# Function to check if a given number is Perfect number
def isPerfect(n):
    sum = 0
    for i in range(1,n):
        if(n%i==0):
            sum+=i
    if(sum == n):
        return True
    #else:
     #   return False
#isPerfect(28)
```

Out[112]:
```
True
```

In [141]:
```
# Function to generate all Perfect numbers in a given range
def isPerfectInRange(lb,ub):
    for i in range(lb,ub+1):
        if(isPerfect(i)):
            print(i)
isPerfectInRange(1,10000)

```

```
6
28
496
8128
```

In [ ]:
```

```

Advanced Problem Set

- Function to calculate average of all factorials in a given range
- Function to generate N odd armstrong numbers
- Function to generate Multiplication table for a number in a given range

```
- 10 in the range(100, 110) inclusive
- 10 x 100 = 1000
- 10 x 101 = 1010
- 10 x 102 = 1020
```

In [75]:

```python
# FUction to calculate average of all factorials in a given range
def avgOfFactorsInRange(lb,ub):
    sum = 0
    count = 0
    for lb in range(lb,ub+1):
       # if factorialOfNumber(lb):
            sum +=factorialOfNumber(lb)
            #print(sum)
            count+=1
    return (sum//count)
avgOfFactorsInRange(1,5)

```

Out[75]:

```
30
```

In [2]:

```python
# Function to generate N odd armstrong numbers
def generateNumbers(n):
    for i in range(1,n+1):
        isArmstrong(i)
    return
def isArmstrong(number):
    sum = 0
    temp = number
    while(number>0):
        a =(number%10)**len(str(number))
        sum = sum + a
        number = number//10
    if(temp == sum and temp%2!=0):
        print(sum)
    return

generateNumbers(100)

```

```
1
3
5
7
9
89
```

In [5]:
```
# Function to generate Multiplication table for a number in a given range
def multiplicationTable(table,start,end):
    for i in range(start,end+1):
        print(table,"x",i,"=",table*i)
table=int(input("enter a number"))
start=int(input("enter start number"))
end=int(input("enter end number"))
multiplicationTable(table,start,end)
```

```
enter a number10
enter start number100
enter end number110
10 x 100 = 1000
10 x 101 = 1010
10 x 102 = 1020
10 x 103 = 1030
10 x 104 = 1040
10 x 105 = 1050
10 x 106 = 1060
10 x 107 = 1070
10 x 108 = 1080
10 x 109 = 1090
10 x 110 = 1100
```

In [51]:
```
# Function to print the alternate values in a range
# [500,550]  --> in Mathematics square bracket means inclusive range i.e 500
# (500,550)  --> in Mathematics open bracket means exclusive range  i.e 500
# range(500 ,550)  -> 500 501 502 503 .....509
# All set based functions in Python have start inclusive  end range exclusiv

def alternateValues(start, end):
    for value in range(start, end+1, 4):  # 4 represents every 4th number ha
        print(value, end=" ")
    return
alternateValues(500,525)

```

```
500 504 508 512 516 520 524
```

In [15]:
```
# Fuction to print reverse of given range in a same line
def reverseOfaRange(start,end):
    for count in range(end,start+1,-2):
        print(count,end=" ")
    return
reverseOfaRange(1,35)
```

```
35 33 31 29 27 25 23 21 19 17 15 13 11 9 7 5 3
```

In [24]:
```
# Function to print odd numbers in reverse order in a range
def reverseOfaRangeOfOdd(start,end):
    for value in range(end,start-1,-1):
        if(value%2!=0):
            print(value,end=" ")
    return
reverseOfaRangeOfOdd(1,10)
```

```
9 7 5 3 1
```

In [28]:
```
# Function to calculate the sum of numbers in a range
def sumOfRange(start,end):
    sum = 0
    for i in range(start,end+1):
        sum+=i  # sum = sum + i
    return sum   # Here in the question "calculate" is there  so we have to
sumOfRange(100,200)

# 200*201/2 - (100*101/2)  #  Formula tosum of numbers btw 100, 200
```

Out[28]:
```
15050.0
```

In [39]:
```
# Function to calculate the average of a given range
def avgOfRange(start,end):
    sum = 0
   # count = 0
    for i in range(start,end+1):
        sum = sum + i  # Sum Calculation
        #count+=1   # Counting
    #return (sum
    return (sum//end+1 - start)
avgOfRange(1,5)
```

Out[39]:
```
3
```

In [81]:
```
# Function to generate all leap years in a given time period
# [2000 - 2020] -> 2000 2004 2008 2012 2016 2020
# isLeapYear(year)
# generateLeapYears(startyear,endyear)
def generateLeapYears(startyear, endyear):
    for i in range(startyear, endyear+1):
        if(isLeapYear(i)):
            print(i,end=" ")


def isLeapYear(year):
    if(year%400==0 or (year%100!=0 and year%4==0)):
        return True
    else:
        return False

generateLeapYears(2000,2020)
#isLeapYear()
```

```
2000 2004 2008 2012 2016 2020
```

In [ ]:

```

```

In [100]:

```
# Calculate number of days in a given time period using leapYear
# For every year in the given time period ,if the year is not a leap year  -
def daysOfGivenTimePeriodIncludeLeapYears(startyear,endyear):
    sum=0
    for year in range(startyear,endyear+1):
        if isLeapYear(year):
            sum+=366
        else:
            sum+=365
    return sum
daysOfGivenTimePeriodIncludeLeapYears(2012,2020)
```

Out[100]:

```
3288
```

```

```

In [ ]:

```



```

In [1]:

```python

# Function to calculate number of hours for a given period in the format(mon
# numberOfHours(11, 1975, 3, 1999) -> 204504 or 205248
# numberOfHours(5, 2019, 6, 2019) -> 1464
# 2, 2016 , 6, 2019
#
#  [all days from feb 2016 to dec 2016,
# .  all days for years between 2016+1 and 2019-1,
#     all days from Jan to June 2019]
#No of hours = 24 * No of days
# 3 steps
    #1. start month year to end of year - calculate no of days
    #2. Calculate days for all years between start year and end year exclusi
            # 2017, 2018 - 365 * no of years
    #3. calculate days from Jan to end month year

# Excluding Feb
# First Six months - 1, 3, 4, 5, 6, 7
                    # All odd months have 31 days
                    # All even months have 30 days
# Last Six months - 8, 9, 10, 11, 12
                    # All even months have 31 days
                    # All odd months have 30 days

# 31 days - (month <= 7 and month % 2 != 0 and month != 2) || (month >= 8 an
#                return 31
#
#            else
#                return 30



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(6, 2018, 7, 2018)

```

Out[1]:
```
1464
```

In [ ]:
```

```