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
- Problem set for Practice

In [ ]:

String Slicing

In [26]:
```
s1 = 'Python'
s1[0] # Accessing the first character in astring
s1[1] # Accessing the second character in staring
s1[len(s1)-1] # Accessing the last character in a String
s1[-1] # Accessing the last character in a string
s1[-2] # Accessing the penultimate character of a string
s1[0:2] # Accessing the first two characters in
s1[-2:]
s1[len(s1)-2:]
s1[1:-1]# Accessing the middle character except first character
s1[len(s1)//2]# Accessing the middle character
s1[-1::-1]
# Access las two characters in reverse order

s1[-1:-3:-1]

# Reverse the middle two characters in an even lenth string
s2="abcdef"
s2[len(s2)//2:len(s2)//2-2:-1]

# Accessing alternative characters in a string
#"python" -> "Pto"
s2[::2]

#Accessing  alternative characters of String in reverse order
s2[::-2]
```

Out[26]: 'fdb'

In [ ]:

Functions

In [28]:
```
#Function to reverse a string
def reverseString(s):
    return s[::-1]
reverseString("Python")
```

Out[28]: 'nohtyP'

In [37]:
```
# Function to check if string is a plaindrome
def palindrome(s):
    if s == s[::-1]:
        return True
    else:
        return False
palindrome("mastan")
palindrome("cc")
```

Out[37]: True

In [45]:
```
# Function to check if given year is a leap year
def isLeapYear(year):
    if year%400==0 or (year%100!=0 and year%4==0):
        return True
    return False
print(isLeapYear(2004))
print(isLeapYear(2005))
print(isLeapYear(100))
```

```
True
False
False
```

In [47]:
```
#Function to count the number of digits in a given number
def numberofDigits(s):
    return len(str(s))
numberofDigits(12345)
```

Out[47]: 5

In [49]:
```
# 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
greatest4(1,123,143,148)
```

Out[49]: 148

In [ ]:

Iteration

- for
- while

In [50]:
```
# Function to print n natural numbers
def printnNaturalNumbers(n):
    for counter in range(1,n+1):
        print(counter,end=" ")
    return
print(printnNaturalNumbers(30))
```

```
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 None
```

In [15]:
```
# Function to print the alternate values in a range
#[500, 550]--> 500 502 504.........550(Inclusive range)
#(500,550)--->501 503 505.......549(Exclusive)
#range(500,550)-->500 501 502..........549
#All set based functions in python have start value inclusive and end value is ex
def alternateValues(lb,ub):
    for i in range(lb,ub+1,4):
        print(i,end=" ")
    return
alternateValues(10,20)
```

```
10 14 18
```

In [27]:
```
# Function to print reverse of given range in the same
def reverseOrder(lb,ub):
    for i in range(ub,lb-1,-1):
        print(i,end=" ")
    return
reverseOrder(10,20)
```

```
20 19 18 17 16 15 14 13 12 11 10
```

In [28]:
```
# Function to print the odd numbers in reverse order in a given range
def reverseOrder(lb,ub):
    for i in range(ub,lb-1,-1):
        if i%2!=0:
            print(i,end=" ")
    return
reverseOrder(10,20)
```

```
19 17 15 13 11
```

In [37]:
```
# Function to calculate the sum of numbers in a range
def sumofinaRange(lb,ub):
    sum=0
    for i in range(lb,ub+1):
        sum+=i
    return sum

sumofinaRange(100,200)
```

Out[37]: 15150

In [41]:
```
# Function to calculate the average of a given range
def avgofGivenRange(lb,ub):
    sum=0
    count=0
    for i in range(lb,ub+1):
        count+=1
        sum+=i
    avg=sum//count
    return avg
avgofGivenRange(1000,5000)
```

Out[41]: 3000

In [67]:
```
# Function to generate all leap years in agiven time period
# 2000 - 2020 -->2000 2004 2008 2012 2016 2020
def isLeapYear(i):
    if i%400==0 or (i%100!=0 and i%4==0):
        return True
    return False
def generateLeapYear(startYear,endYear):
    for i in range(startYear,endYear+1):
            if(isLeapYear(i)):
                print(i,end=" ")

generateLeapYear(1919,2019)
```

```
1920 1924 1928 1932 1936 1940 1944 1948 1952 1956 1960 1964 1968 1972 1976 1980
1984 1988 1992 1996 2000 2004 2008 2012 2016
```

In [90]:
```
# Caculate number of days in a given time Period using leap Year logic
# for every year in the given time period.
#if the year is not a leap year -->add 365 to sum
# if leap year--> add 366
def numberofDaysinBetweenYears(n1,n2):
    sum=0
    for i in range(n1,n2+1):
        if isLeapYear(i):
            sum=sum+366
        else:
            sum=sum+365
    return sum
numberofDaysinBetweenYears(2017,2018)
```

Out[90]: 730

In [91]:
```
# Function to caculate number of hours for given period of time
#(11,1975,3,1999
def numberofDaysMonth(month,year):
    if month==2:
        if isLeapYear(year):
            return 29
        else:
            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
    days+=daysInStartYear(startmonth,startyear)
    days+=daysInEndYear(endmonth,endyear)
    if endyear-startyear == 1:
        days+=numberofDaysinBetweenYears(startyear+1,startyear+1)
    elif endyear - startyear > 2:
        days+=numberofDaysinBetweenYears(startyear+1,endyear-1)
    return 24*days

numberofHours(11,1975,3,1999)
```

Out[91]: 203088

In [ ]:
```
```

In [78]:
```
def no_of_daysinmonth(n):
    if(n<=7 and n%2!=0 ):
        return 31
    elif(n==2):
        return 28
    else:
        return 30
no_of_daysinmonth(2)
```

Out[78]: 28

In [79]:
```
def hours(m1,y1,m2,y2):
    h1=0
    for i in range(m1,12):
        h1=h1+(no_of_daysinmonth(i)*24)
    h2=0
    for j in range(m2,12):
        h2=h2+(no_of_daysinmonth(j)*24)
    h=abs((y2-y1)*365)*24
    return h1+h2+h
hours(11,1975,3,1999)
```

Out[79]: 217512

In [62]:
```
#Function to print all numbers divisible by 6 and not a factor of 100 in a given
def divisiblyBy6(n1,n2):
    for i in range(n1,n2+1):
        if i%6==0 and i%100!=0:
                print(i,end=" ")
divisiblyBy6(10,20)
```

```
12 18
```

In [76]:
```
# Function to find the average of cubes of all even numbers in a given range(lb,
def cubesofAllevenAverage(n1,n2):
    sum=0
    count=0
    for i in range(n1,n2+1):
        if i%2==0:
            sum=sum+i**3
            count=count+1
    print(sum//count)
cubesofAllevenAverage(1,3)
```

```
8
```

In [75]:
```
# Function to generate the sum  of factors for a given number
#12-->1,2,3,4,5,6,12
def factors(n):
    sum=0
    for i in range(1,n//2+1):
        if n%i==0:
            sum+=i
            print(i)
            return sum
factors(12)
```

```
1
2
3
4
6
```

In [71]:
```
# Function to Caculate the factorial of a given number
def printFactorial(n):
    fact=1
    if n==0:
        return 1
    for i in range(1,n+1):
        fact=fact*i
    print(fact)
printFactorial(4)
```

```
24
```

In [84]:
```
# Function to check if a given number is prime
def isPrime(n):
    flag = True
    for i in range(2,n//2+1):
        if n%i==0:
            flag=False
            return flag
    return flag
isPrime(99)
```

Out[84]:
```
False
```

In [85]:
```
# Function to calculate the average  first N Prime Numbers
def avgNPrimes(n):
    primeCount=0
    sequenceCount = 2
    sum=0
    while(primeCount<n):
        if isPrime(sequenceCount):
            primeCount+=1
            sum+=sequenceCount
        sequenceCount+=1
    return sum/n

avgNPrimes(10)
```

Out[85]: 12.9

In [1]:
```
# Function to generate all perfect numbers in a given numbers
def isPerfect(n):
    if factors(n)==n:
        return True
    return False
def generatePerfect(lb,ub):
    for i in range(lb,ub+1):
        if isPerfect(i):
            print(i,end=" ")
    return
generatePerfect(1,10000)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-6f38a960613c> in <module>
      9             print(i,end=" ")
     10     return
---> 11 generatePerfect(1,10000)

<ipython-input-1-6f38a960613c> in generatePerfect(lb, ub)
      6 def generatePerfect(lb,ub):
      7     for i in range(lb,ub+1):
----> 8         if isPerfect(i):
      9             print(i,end=" ")
     10     return

<ipython-input-1-6f38a960613c> in isPerfect(n)
      1 # Function to generate all perfect numbers in a given numbers
      2 def isPerfect(n):
----> 3     if factors(n)==n:
      4         return True
      5     return False

NameError: name 'factors' is not defined
```

Advanced Problem Set

In [51]:
```
# Function to calculate average of all factorials in a given range
def avgofFactorial(lb,ub):
    fact=1
    sum=0
    count=0
    for i in range(lb,ub+1):
            fact=fact*i
            sum+=fact
            count+=1
    print(fact)
    print(sum)
    print(count)
    print(sum//count)
avgofFactorial(1,5)
```

```
120
153
5
30
```

In [ ]:
```

```