Problem Solving And Programming

Date 12 June 19

Day Objectives

- String slicing
- Functions in Python
- Basic problems related to conditional statements using functions
- Iteration in Python
- Problem set for practice

In [ ]:
```
1
```

String Slicing

In [1]:
```
1 s1 = "Python"
2
3 s1
```

Out[1]: 'Python'

In [2]:
```
1 # To Access paticular character in the string
2
3 # To access the first character of the string
4
5 s1[0]
```

Out[2]: 'P'

In [14]:
```
# To access the last character in the string

s1[-1]

# Another way

s1[len(s1)-1]

s1[-2] # Accessing the penultimate character of a string

s1[0:2]  # second part of the slice is exclusive so have to give +1 extra (A

# Work for any string
s1[-2:]   # Access the last two characters of a string

# another way

# It won't work for all strings since it is based on length of string
s1[4:]  # it is accessing the 5th character to end of the string

```

Out[14]: 'on'

In [15]:
```
# Accessing all character except first and last character

s1[1:-1]
```

Out[15]: 'ytho'

In [24]:
```
# Accessing the middle character in a string
# for odd length string
s1[len(s1)//2]  # // means integer division


# for even length string



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

#another way
s1[::-1]



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


#Reverse the middle two characters in an even length string

s1[len(s1)//2:len(s1)//2-2:-1]


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

s1[::2]

#Accessing alternate characters in reverse order
## "Python" --> "nhy"
s1[::-2]
```

Out[24]: 'nhy'

In [ ]:
```

```

Functions

In [27]:
```
#Function to reverse a string
def reverseString(s):
    return s[::-1]

reverseString("Python")
```

Out[27]: 'nohtyP'

In [30]:
```
#Function to check if a string is a palindrome

def palindrome(s):
    if s == s[::-1]:
        return True
    else:
        return False

palindrome("racecar")
```

Out[30]: True

In [31]:
```
# C
```

In [34]:
```
#Function to Check if  a 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

isLeapYear(int(input("Enter the year ")))
```

```
Enter the year 2016
```

Out[34]: True

In [50]:
```
# Function to Count the number of digits in a given number

def countDigits(n):
    return len(n)

countDigits("12345678")




#Another way

def countDigits(n):
    return len(str(n))

countDigits(12345678)
```

Out[50]: 8

In [52]:
```
#Funcion to identify the greatest of 4 numbers

def greatestNum(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
    else:
        return n4


greatestNum(12,524,36,234)

```

Out[52]: 524

In [ ]:
```

```

Iteration

- for
- while

In [54]:
```
#Function to print n natural numbers
def nNaturalnums(n):
    for i in range(1,n+1):                #second parameter is exclusive
        print(i, end=" ")
    return

nNaturalnums(int(input("Enter the number range : ")))

```

```
Enter the number range : 20
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
```

In [57]:
```
def nNaturalnumbers(n):
    counter = 1
    while counter <= n:
        print(counter,end= " ")
        counter = counter + 1
    return

nNaturalnumbers(int(input("Enter the number range: ")))

```

```
Enter the number range: 20
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
```

In [ ]:
```

```

In [6]:
```
# Function to print the alternate values in a  range in the same line
#[500,550] --> 500 502 504 506 ..... 550   here [] means inclusive of
# (500,550) --> 501 503 505 507 ..... 549 here () means both are exclus
# range(500,550) --> 500 501 502 ..... 549 for all set based functions


def alternateValues(lb,ub):
    for i in range(lb,ub + 1,2):
        print(i,end= " ")
    return

alternateValues(500,525)
```

```
500 502 504 506 508 510 512 514 516 518 520 522 524
```

In [11]:
```
# Function to print the reverse of a number in the given range


def alternaterevValues(lb,ub):
    for value in range(ub, lb -1  ,-1):
        print(value,end= " ")
    return

alternaterevValues(1,10)

```

```
10 9 8 7 6 5 4 3 2 1
```

In [20]:
```
# Funtion to print odd numbers in reverse order in a given range

def oddReverse(lb,ub):
    for values in range(ub,lb-1,-1):
        if values % 2 != 0:
            print(values, end= " ")
    return

oddReverse(10,20)
```

```
19 17 15 13 11
```

In [21]:
```
#Function to calculate the sum of all the numbers in given range

def sumRange(lb,ub):
    sum = 0
    for i in range(lb,ub+1):
        sum =  sum + i
    return sum

sumRange(10,20)


## 200 * 201/2 - (100 * 101/2)

```

Out[21]: 165

In [35]:
```
#Function to calculate the average of a given range

## (1,5) --> average is 3

def avgRange(lb,ub):
    sum =0
    count = 0
    for i in range(lb,ub+1):
        sum = sum + i
        #count = count +1
    return sum//(ub-lb)

avgRange(10,50)
```

Out[35]: 30.75

In [27]:
```
## Another way


def avgRange(lb,ub):
    sum =0
    count = 0
    for i in range(lb,ub+1):
        sum = sum + i
        count = count + 1
    return sum//count

avgRange(10,50)

```

Out[27]: 30

In [69]:
```
# Function to print all numbers divisible by 6 and not a factor of  100 in a


def divisibleby6(lb,ub):
    for i in range(lb,ub+1):
        if i % 6 == 0 and 100 % i != 0:
            print(i, end= " ")
    return


divisibleby6(100,200)
```

```
102 108 114 120 126 132 138 144 150 156 162 168 174 180 186 192 198
```

In [78]:
```
#Function to find the average of cubes of all even numbers in a given range(

# 1,10 -> 2,4,6,8,10 -> avg(8,64,216,64*8,10000) --> result

def avgCubesEven(lb,ub):
    sum = 0
    count = 0
    for i in range(lb,ub + 1):
        if i % 2 == 0:
            sum += i ** 3
            count += 1
    return sum /count

avgCubesEven(1,3)
```

Out[78]:
```
8.0
```

In [97]:
```

#Function to generate the sum of factors for a given number
# 12 --> 1,2,3,4,6,12

def factorsofaNum(n):
    sum =0
    for i in range(1,n//2+1):
        if n % i ==0:
            sum += i
            print(i, end= " ")
    return sum


factorsofaNum(120)
```

```
1 2 3 4 5 6 8 10 12 15 20 24 30 40 60
```

Out[97]:
```
240
```

In [75]:
```
#Function to calculate the factorial of a given number

def factorial(n):
    fact=1
    for i in range(2,n+1):
        fact *= i
    return fact

factorial(10)
```

Out[75]: 3628800

In [91]:
```
#Function to Check if a given number is prime

def isPrime(n):
    flag = True
    for i in range(2,n//2):
        if n% i == 0:
            flag == False
            return flag
    return flag

isPrime(19)
```

Out[91]: True

In [90]:
```
#Function to calculate the average of 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[90]: 6.5

In [96]:
```
#Function to generate all Perfect numbers in a given range

def isPerfect(n):
    if factorsofaNum(n) == n:
        return True
    return False

def genPerfect(lb,ub):
    for i in range(lb, ub + 1):
        if isPerfect(i):
            print(i, end= " ")
    return

genPerfect(1,10)
```

```
1 1 1 2 1 1 2 3 6 1 1 2 4 1 3 1 2 5
```

In [ ]:
```

```

In [1]:
```
# Function to generate all leap years in a given time period
# [2000, 2020] -> 2000 2004 2008 2012 2016 2020

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 generateLeapYears(startyear, endyear): # uses the isLeapYear() to select
    for year in range(startyear, endyear+1):
        if isLeapYear(year):
            print(year, end=" ")
    return

generateLeapYears(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 [2]:
```
# Calculate number of days in a given time period using leapYearlogic
# For every year in the given time period, if the year is not a leap year ->

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)
```

Out[2]: 730

In [3]:

```
# 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[3]: 1464

In [ ]:
```
1
```