Day objectives

Date : 27-Sept-2019

Strings and String slicing

Functions in Python

In [3]:

```
# Strings
# String is a collection of charecters
# String declarations in python
   #  'string'  or "string"
# Strings like an array
# If you want access the string elements here we are two ways to acccess
             # 1. By using positive indexing
             # 2. By using negative indexing
s = "python"
                  #  p  y  t  h  o  n
                  #  0  1  2  3  4  5  -> positive indexing
                  # -6 -5 -4 -3 -2 -1  -> negative indexing
s[0]   # It gives 0 th element of the string
s[1]   # It gives 1 th element of the string
s[-1]  # last value of the string
s[-2]  # second last of the string

# String have upper and lower bound

s[1:3] # Here lowerbound is included and upper bound is excluded

# If you Want to print the alternative elementes of the string we use step method
s[::2]   # string[lower:upper:step]
s[::-1]  # It gives reverse string

```

Out[3]:

```
'nohtyp'
```

In [5]:

```
s = "Python"
len(s)         # It gives the length of the string   #len(string)
```

Out[5]:

```
6
```

In [7]:

```python
s1 = "Python programming"
len(s1)
```

Out[7]:

```
18
```

In [12]:

```python
# check the given string is a palindrome r not
s = input(" enter the string ")
temp=s[::-1]
if s==temp:
    print("palindrome")
else:
    print("not palindrome")
```

```
 enter the string charan
not palindrome
```

In [13]:

```
dir(str) # Predefined functions of the strings
```

Out[13]:

```
['__add__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__getnewargs__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__mod__',
 '__mul__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__rmod__',
 '__rmul__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'capitalize',
 'casefold',
 'center',
 'count',
 'encode',
 'endswith',
 'expandtabs',
 'find',
 'format',
 'format_map',
 'index',
 'isalnum',
 'isalpha',
 'isascii',
 'isdecimal',
 'isdigit',
 'isidentifier',
 'islower',
 'isnumeric',
 'isprintable',
 'isspace',
 'istitle',
 'isupper',
 'join',
 'ljust',
 'lower',
 'lstrip',
 'maketrans',
 'partition',
 'replace',
 'rfind',
 'rindex',
 'rjust',
 'rpartition',
 'rsplit',
 'rstrip',
 'split',
 'splitlines',
 'startswith',
 'strip',
 'swapcase',
 'title',
 'translate',
 'upper',
 'zfill']
```

In [16]:

```
s = 'python'
print(s)
s   # When we call a variable it gives out because as it is stored in the kernal
```

```
python
```

Out[16]:

```
'python'
```

In [17]:

```
s = 'python'
s.capitalize() # It changes first letter to uppercase
```

Out[17]:

```
'Python'
```

In [21]:

```
s = 'python'
s.upper() # It changes lower to upper case letters
```

Out[21]:

```
'PYTHON'
```

In [23]:

```
s.count('t') # It gives number of occurance of letters
             # It is case sensitive
```

Out[23]:

```
1
```

In [26]:

```
# String cheat sheet
s = 'apssdc'
s*2
```

Out[26]:

```
'apssdcapssdc'
```

Functions in Python

In [27]:

```
# Function is block of code
# We have two types of functions
     # 1.Predefined functions
     # 2.use defined functions

# user defined funtions:
    # 1.Without arguments and without return values
    # 2.Without arguments and with  return values
    # 3.With arguments and without return values
    # 4.With arguments and with  return values

#Function declaration in python (syntax)
# def functionname():
#     statements


```

In [30]:

```
# 1.Without arguments and without return values
# Addition of two numbers

def add():              # To declare a function
    a=10
    b=20
    print(a+b)
add()                   # To call a function
```

```
30
```

In [32]:

```
# 2.Without arguments and with  return values

def add():
    a=10
    b=20
    return a+b  # To return a value # It can be called all over the program
add()
```

Out[32]:

```
30
```

In [33]:

```
# 3.With arguments and without return values

def add(a,b):             #  Need Arguments a , b
    print(a+b)
add(2,4)   # Values passed as a argument
```

```
6
```

In [34]:

```
# 4.With arguments and with  return values

def add(a,b):   # Arguments
    return a+b   # Return value
add(4,7)

```

Out[34]:

```
11
```

In [ ]:

```
# Tasks
# 1.create a function to check the given string is a palindrome or not
# 2.create a function to check given number is a prime or not
# 3.create a function to generate all the prime numbers in between the given range
# 4.create a function to generate the fators of the given number
# 5.create a function to find the avarage of 1 to 100 numbers
# 6.create a function to calculate the avarage of given number factors
# 7.create a function to print the prime numbers in a given number factors
# 8.create a function to check the leap year
# 9.Take a string "proble solving using Python"
    #find the length of the string
    #change the string to lower case
    #print the string in reverse order
    #find how many time "s" is repeated in a given string
```

In [38]:

```
# 1.create a function to check the given string is a palindrome or not
s = input()

def palindrome(string):
    temp=string[::-1]
    if temp==string:
        return 'palindrome'
    else:
        return 'not palidrome'

palindrome(s)
```

```
charan
```

Out[38]:

```
'not palidrome'
```

In [2]:

```
# 2.create a function to check given number is a prime or not
num=int(input())
def isprime(number):
    count=0
    for i in range(1,number+1):
        if number%i==0:
            count+=1
    if count==2:
        return 'prime'
    else:
        return 'not prime'
isprime(num)

```

```
7
```

Out[2]:

```
'prime'
```

In [61]:

```
# 3.create a function to generate all the prime numbers in between the given range
ul=int(input('ul'))
ll=int(input('ll'))
def primegen(ll,ul):
    count=0
    for i in range(ll,ul+1):
        for j in range(1,i+1):
            if i%j==0:
                count+=1
    if count==2:
                print(i)
primegen(ll,ul)

```

```
ul15
ll1
```

In [7]:

```
# 4.create a function to generate the fators of the given number
n=int(input())
def factors(num):
    for i in range(1,num+1):
        if num%i==0:
            print(i,end=" ")

factors(n)

```

```
6
1 2 3 6
```

In [15]:

```
# 5.create a function to find the avarage of 1 to 100 numbers
def avg(ll,ul):
    sum=0
    count=0
    for i in range(ll,ul+1):
        sum+=i
        count+=1
    avarage=sum/count
    return avarage
avg(1,100)
```

Out[15]:

```
50.5
```

In [19]:

```
# 6.create a function to calculate the avarage of given number factors
number=int(input())
def avgfact(num):
    sum=0
    count=0
    for i in range(1,num+1):
        if num%i==0:
            sum+=i
            count+=1
    avarage=sum/count
    return avarage
avgfact(number)
```

```
6
```

Out[19]:

```
3.0
```

In [35]:

```
# 7.create a function to print the prime numbers in a given number factors
def prifact(n):
    count=0
    for i in range(1,n+1):
        if n%i==0:
            if isprime(i)=='prime':
                print(i)


prifact(6)
```

```
2
3
```

In [38]:

```
# 8.create a function to check the leap year
year = int(input())
def leapcheck(y):
    if ((y%4==0 and y%100!=0) or (y%400==0)):
        return 'leap year'
    else:
        return 'not leap year'
leapcheck(year)
```

```
2000
```

Out[38]:

```
'leap year'
```

In [56]:

```
# 9.Take a string "problem solving using Python"
    #find the length of the string
    #change the string to lower case
    #print the string in reverse order
    #find how many time "s" is repeated in a given string
s = 'problem solving using PYTHON'
length=len(s)
print('length of the string is : '+ str(length))
lowstr=s.lower()
print('lower case of the string is: '+str(lowstr))
revstr=s[::-1]
print('reversed string is: '+revstr)
occurance=s.count('s')
print('occurance od "s" in the string is :'+ str(occurance))
```

```
length of the string is : 28
lower case of the string is: problem solving using python
reversed string is: NOHTYP gnisu gnivlos melborp
occurance od "s" in the string is :2
```

In [ ]:

```

```