In [22]:
```python
# Login Credentials
uname="gistcse"
pwd="Gist@cse2"
user=input("User Name:")
password=input("Password:")
if user==uname and password==pwd:
    print("Login Successful...!")
else:
    if user!=uname and password==pwd:
        print("Invalid user name..!")
    elif user==uname and password!=pwd:
        print("Wrong Password..!")
    else:
        print("xx.. Login Failed.. xx")

```
...

Loops in python

- iteration of statements
- for,while loops
    - **For Loop**
        - **syntax**
            - for iterator in iterable:
                - statements
            - for iterator in range(len(iterable)):
                - statements
        - default incrementation is 1

In [23]:
```python
# why loops?
# display your name for 5times
print("A")
print("A")
print("A")
print("A")
print("A")
```
...

In [24]:
```python
for val in range(1,10): # 1,2,3,4,..9
    print("A")
```
...

In [25]:
```python
for val in range(1,11):
    print(val,end=" ")
```

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

In [26]:
```python
for val in range(10,0,-1): # (start,stop,step_count)
    print(val,end=" ")

```

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

In [27]:
```python
for val in range(1,20,2):
    print(val,end=" ")
```

```
1 3 5 7 9 11 13 15 17 19
```

In [28]:
```python
# write a python program to print nth multiplication table
# multiplication table for 9
'''
9x1=9
9x2=18
9x3=27
.....
9x10=90
'''
n=int(input("enter the number:"))
for v in range(1,11):
    print(n,'x',v,'=',n*v)

```

...

In [29]:
```python
for ch in "GIST":
    print(ch)
```

...

while Loop

- condition based loop
- user incrementation
- **syntax**
    - while condition:
        - statements

In [1]:
```
# write a python program to print the even no.s present in a range
#using while
lw,up=int(input()),int(input()) # 1,10
while lw<=up: # universal truth-->infinite loop
    if lw%2==0:
        print(lw,end=" ")
    lw+=1 #==up


```

...

In [6]:
```
# while loop in reverse order
x,y=int(input()),int(input()) # 20,1
while x>=y:
    if x%2==1:
        print(x,end=" ")
    x-=1
```

...

Nested Loops

- loops within loop
  - i loop
    - j loop

2 iterables at a time

```
....
....
....
```

In [15]:
```
for r in range(1,6):
    for cl in range(1,5):
        print('*',end=" ")
    print()
```

...

In [16]:
```
#iterate col w r t row
for r in range(1,6):
    for c in range(1,r+1):
        print("*",end=" ")
    print()
```

...

In [17]:
```
# write a python program to print the following pattern
* * * *
  * * *
    * *
      *
```

...

In [18]:
```
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *

```

```
  File "C:\Users\ruthu\AppData\Local\Temp\ipykernel_20672\965164629.py", line 1
    * * * * *
      ^
SyntaxError: invalid syntax
```

In [21]:
```
for row in range(1,6):
    for col in range(1,6):
        if row==col:
            print("@",end=" ")
        elif row>col:
            print("#",end=" ")
        else:
            print("*",end=" ")
    print()
```

...

Functions

- Group/collection of statements executed to perform a specific task
- There 2types of functions
  1. Pre-defined/built in functions
     - These functions were defined when the language is developed
     - these functions are highlighted in green colour
     - Ex:print(),input(),int(),float(),type(),bool(),bin(),dir(),str(),set(),dict(),enumerate(),sum(),s etc
     - a user can call and use those functions
  2. User Defined

In [22]:
```
print()
input()
sum()
int()
max()
min()
list()
tuple()
dir()
help()
type()
bin()
ord()
chr()
enumerate()
open()
sorted()
str(),tuple(),list(),set(),dict()
```

...

In [23]:
```
help(print)
```

...

In [24]:
```
help(input)
```

...

User defined Functions

- These functions are defined by user
- def is the keyword that represents function
- **syntax**
  - def function_name(arguments)
    - statements
- recursive function
- non-recursive(mostly used)
- **Features of function**
  - reusability of code
  - follows the modularity
- function definition
  - variables used here called as arguments
  - reference var
- function call
  - parameters ,actual data points

In [26]:
```
# def function_name(arguments):
def example():
    print("working with functions")

example()
```

...

In [33]:
```
# function with argument and with return
def add(x,y):
    return x+y

def display(a,b):
    return f"I am {a} and my friend is {b}"

print("sum=",add(9,10)) # function call
display('Kavitha','Bhavya')
```

...

In [30]:
```
display("Ruthu",'vanitha') # fun call
```

Out[30]:
```
'I am Ruthu and my friend is vanitha'
```

In [34]:
```
# function with argument and without return
def table(n):
    for i in range(1,11):
        print(n,'x',i,'=',n*i)

table(8)
```

...

In [35]:
```
# function without argument and with return
def empty():
    #local variables
    a,b=int(input()),int(input())
    return a**b

empty()
```

...

In [36]:
```python
# function without args and without return

def final():
    print("no.of occurences=",st.count(ch))

st="college"
ch='e'
final()
```

```
no.of occurences= 2
```

In [40]:
```python
# define a function that prints the prime no.s in a range
#prime no.s has only 2 factors,i.e., 1 & n itself
num=int(input("enter the number:"))
nfs=0
for v in range(1,num+1): # 10:1,2,5,10
    if num%v==0:
        print(v,end=" ")
        nfs+=1
if nfs==2:
    print("given number is prime")
else:
    print("non prime")
```

```
enter the number:11
1 11 given number is prime
```

In [43]:
```python
def is_prime(n):
    count=0
    for v in range(1,n+1):
        if n%v==0:
            count+=1
    if count==2:
        return True
    else:return False

is_prime(11)
```

Out[43]:
```
True
```

In [42]:
```python
is_prime(10)
```

Out[42]:
```
False
```

In [45]:
```python
def is_prime(n):
    count=0
    for v in range(1,n+1):
        if n%v==0:
            count+=1
    if count==2:
        return True
    else:return False

#is_prime(11)
lw,up=int(input()),int(input())
for v in range(lw,up+1):
    if is_prime(v):
        print(v,end=" ")

```

...

In [46]:
```python
#Recursive functions
# function calling itself
# factorial of number # n-->n*n-1*n-2*n-3..n*(n-n-1)..1
#5--5*4*3*2*1
val=int(input())
fact=1
for dig in range(val,0,-1):
    fact*=dig
print("factorial value=",fact)
```

```
5
factorial value= 120
```

In [47]:
```python
#finding factorial value using function
def factorial(n):
    if n==0 or n==1:
        return 1
    else:
        return n*factorial(n-1)
factorial(4)
```

Out[47]:
```
24
```

Strings

- Group of characters
- anything that is enclosed in the quotations called as string
- a single character is also a string
- str() that represents the string data
- str="/""/''' '''

In [51]:
```
#String declaration
name='Afifa'
frnd="Bhavana"
other='''Susmitha'''
clsmt=input("enter the name:") # Karishma
print(name,frnd,other,'and',clsmt,"are friends",end=".")
for ch in name:
    print(ch)
```

...

In [52]:
```
for ch in clsmt: # iteration of string
    print(ch,end=" ")
```

```
K a r i s h m a
```

In [58]:
```
# iteration of string using index
for ix in range(len(name)): # affia:5 characters:0 to 4
    print(ix,name[ix],sep=":")
```

...

String Methods

- the functions can be applied on only strings
- dir(str)

In [59]:
```
print(dir(str))
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__',
'__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewa
rgs__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__l
e__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new__', '__reduce
__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__', '__siz
eof__', '__str__', '__subclasshook__', 'capitalize', 'casefold', 'center', 'cou
nt', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'format_map', 'inde
x', 'isalnum', 'isalpha', 'isascii', 'isdecimal', 'isdigit', 'isidentifier', 'i
slower', 'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper', 'join',
'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'removeprefix', 'removesu
ffix', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip',
'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate',
'upper', 'zfill']
```

In [60]:
```
#format_map,isnumeric,lower,partition
help(str.format_map)
```

```
Help on method_descriptor:

format_map(...)
    S.format_map(mapping) -> str

    Return a formatted version of S, using substitutions from mapping.
    The substitutions are identified by braces ('{' and '}').
```

In [61]:
```
help(str.isnumeric)
```

```
Help on method_descriptor:

isnumeric(self, /)
    Return True if the string is a numeric string, False otherwise.

    A string is numeric if all characters in the string are numeric and there i
s at
    least one character in the string.
```

In [67]:
```
num="t123"
num.isnumeric()
```

...

In [68]:
```
clg="Geethanjali Inst of Science & Technology"
clg.isalpha()
```

Out[68]:
```
False
```

In [69]:
```
"ramya".isalpha()
```

Out[69]:
```
True
```

In [70]:
```
"CSE2".isalnum()
```

Out[70]:
```
True
```

In [72]:
```
pwd="CSE123"
pwd[3].isdigit()
```

Out[72]:
```
True
```

In [73]:
```
pwd[3].isnumeric()
```
Out[73]:
```
True
```

In [74]:
```
clg
```
Out[74]:
```
'Geethanjali Inst of Science & Technology'
```

In [75]:
```
clg.lower()
```
Out[75]:
```
'geethanjali inst of science & technology'
```

In [76]:
```
clg.upper()
```
Out[76]:
```
'GEETHANJALI INST OF SCIENCE & TECHNOLOGY'
```

In [77]:
```
clg.capitalize()
```
Out[77]:
```
'Geethanjali inst of science & technology'
```

In [78]:
```
clg.title()
```
Out[78]:
```
'Geethanjali Inst Of Science & Technology'
```

In [80]:
```
"I am {0} and working at {1}".format('Ruthu','Aylin Technologies')
```
...

In [87]:
```
# split
#strip,lstrip,rstring
#center
#zfill
```

In [88]:
```
#str.split
clg
```
Out[88]:
```
'Geethanjali Inst of Science & Technology'
```

In [89]:
```
clg.split()
```
Out[89]:
```
['Geethanjali', 'Inst', 'of', 'Science', '&', 'Technology']
```

In [91]:
```
#strip will remove the spaces from either side of the string
a="    hello"
b="hey       "
c="    hi hello      "
a.lstrip()
```

...

In [92]:
```
b.rstrip()
```

...

In [93]:
```
c.strip()
```

...

In [94]:
```
a.zfill(40)
```

Out[94]:
```
'0000000000000000000000000000000    hello'
```

In [95]:
```
dir(str)
```

...

In [96]:
```
clg.casefold()
```

Out[96]:
```
'geethanjali inst of science & technology'
```

In [97]:
```
help(str.center)
```

```
Help on method_descriptor:

center(self, width, fillchar=' ', /)
    Return a centered string of length width.

    Padding is done using the specified fill character (default is a space).
```

In [101]:
```
clg.center(5)
```

Out[101]:
```
'Geethanjali Inst of Science & Technology'
```

In [102]:
```
#str.join
help(str.join)
```

...

In [105]:
```
'@'.join([name,clsmt,frnd,other]) # multiple strings together
```
Out[105]:
```
'kavya@Karishma@Bhavana@Susmitha'
```

In [104]:
```
name="kavya"
name.center(40)
```
Out[104]:
```
'                 kavya                  '
```

In [ ]:
```

```