Day Objectives:

- Binary Conversions
- Ord,Chr
- Strings
  - Immutable
  - Indexing
  - Slicing
  - Methods
- List
  - Methods
- Tuples
  - Methods

In [10]:
```
# Task1: srija
num=int(input("Enter a number"))
for i in range(1,11):
    print(num,"x",i,"=",num*i)
```

```
Enter a number10
10 x 1 = 10
10 x 2 = 20
10 x 3 = 30
10 x 4 = 40
10 x 5 = 50
10 x 6 = 60
10 x 7 = 70
10 x 8 = 80
10 x 9 = 90
10 x 10 = 100
```

In [13]:
```
# Task2:
a=int(input("enter a"))
b=int(input("enter b"))
n=int(input("enter operation"))
if n==1 :  print(a+b)
elif n==2 : print(a-b)
elif n==3 :print(a*b)
elif n==4: print(a/b)
elif n==5: print(a//b)
elif n==6: print(a%b)
else : print("not valid")
```

```
enter a12
enter b12
enter operation1
24
```

In [18]:
```
a=int(input())
b=int (input())
print("Enter operator")
c=input()
if(c=="+"):
    print(a+b)
elif(c=="-"):
    print(a-b)
elif(c=="*"):
    print(a*b)
elif(c=="/"):
    print(a/b)
else:
    print(a//b)
```

```
14
2
Enter operator
*
28
```

Number Conversions

- Binary
- Oct
- Hex
- dec

In [19]:
```
bin(12)  # 1100
```

Out[19]: '0b1100'

In [20]:
```
oct(12)
```

Out[20]: '0o14'

In [22]:
```
hex(12) # 10-A....15-F
```

Out[22]: '0xc'

In [25]:
```
# binary to decimal
int("1100",2)
```

Out[25]: 12

In [26]:
```
# binary to oct
int("1100",8)
```

Out[26]: 576

In [27]:
```
# binary to hex
int("1100",16)
```

Out[27]: 4352

In [29]:
```
# char to ASCII
ord("A")
```

Out[29]: 65

In [30]:
```
ord("Z")
```

Out[30]: 90

In [31]:
```
ord("a")
```

Out[31]: 97

In [32]:
```
ord("z")
```

Out[32]: 122

In [34]:
```
ord("0")
```

Out[34]: 48

In [35]:
```
ord("9")
```

Out[35]: 57

In [36]:
```
ord("$")
```

Out[36]: 36

In [37]:
```
ord("+")
```

Out[37]: 43

In [38]:
```
# ASCII to char
chr(100)
```

Out[38]: 'd'

In [41]:
```
chr(125)
```

Out[41]: '}'

In [43]:
```
chr(10)
```

Out[43]: '\n'

Strings

- collection of characters
- it's enclosed with '','""" """,
- Immutable object
- We can't change, modify, update, remove
- no char type data

In [44]:
```
s = "apssdc"
s1 = 'apssdc'
s2 = """apssdc"""
print(type(s),type(s1),type(s2))
```

```
<class 'str'> <class 'str'> <class 'str'>
```

i don't know anything

In [46]:
```
s = 'i don\'t know anything'
print(s)
```

```
i don't know anything
```

In [47]:
```
s = "i don\'t know anything"
print(s)
```

```
i don't know anything
```

In [48]:
```
data = """sdjhfjdhf
dsfhdagf
jkhsdjagfh
jhdgkjhsakjg
kjghajk"""
data
```

Out[48]:
```
'sdjhfjdhf\ndsfhdagf\njkhsdjagfh\njhdgkjhsakjg\nkjghajk'
```

In [51]:
```
s4 = """i am "lavanya"  """
print(s4)
```

```
i am "lavanya"
```

String Indexing and Slicing

- To access the particular element in a string
- checking postion
- It makes slicing easy
-

email 456 lavanya phone branch... 343 jdsfhj
566 dsfbds 100

Positive indexing Negative Indexing

```
apssdc
012345
  a  p   s  s  d   c
  -6 -5 -4 -3 -2 -1
```

[startvalue,endvalue,stepvalue]

- start value always inclusive
- end value always exclusive

In [52]:
```
s  = "apssdc"
s[4]
```

Out[52]: 'd'

In [53]:
```
s[0]
```

Out[53]: 'a'

In [54]:
```
s[-1]
```

Out[54]: 'c'

In [55]:
```
s[2]
```

Out[55]: 's'

In [57]:
```
s[0:2]
```

Out[57]: 'ap'

In [58]:
```
s[2::] # empty means all
```

Out[58]: 'ssdc'

In [59]:
```
s
```

Out[59]: 'apssdc'

In [64]:
```
s[2:6]
```

Out[64]: 'ssdc'

In [68]:
```
# pa
s[-5:-7:-1]
```

Out[68]: 'pa'

In [69]:
```
s[::-1] # reverse order
```

Out[69]: 'cdsspa'

In [73]:
```
# Task : check the given data is penlindrome or not
a=input()
b=a[::-1]
if(a==b):
    print("Palindrome")
else:
    print("Not a Palindrome")
# code length
# memory allocation
```

```
1223345456
Not a Palindrome
```

In [72]:
```
string=input(("Enter a string:"))
if string==string[::-1]:
    print("The string is a palindrome")
else:
    print("Not a palindrome")
```

```
Enter a string:mom
The string is a palindrome
```

String Methods

In [74]:
```
dir(str)
```

Out[74]:
```
['__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 [77]:
```
s = "god is great all the time"
s.capitalize()
```

Out[77]: 'God is great all the time'

In [78]:
```
s.title()
```

Out[78]: 'God Is Great All The Time'

In [79]:
```
s.split() # split returns  list, default it consider whitespace
```

Out[79]: ['god', 'is', 'great', 'all', 'the', 'time']

In [81]:
```
s.split("all")
```

Out[81]: ['god is great ', ' the time']

In [83]:
```
email = "lavanya_p@apssdc.in"
email.split("@")
```

Out[83]: ['lavanya_p', 'apssdc.in']

In [86]:
```
li = ["lavanya_p@apssdc.in",
      "geethaandraju88@gmail.com",
     "siddineni2000@gmail.com",
     "tara123",
     "eswargandhi99@gmail.com"]
for email in li:
    print(email.split("gmail.com"))
```

```
['lavanya_p@apssdc.in']
['geethaandraju88@', '']
['siddineni2000@', '']
['tara123']
['eswargandhi99@', '']
```

In [88]:
```
s.startswith("god")
```

Out[88]: True

In [89]:
```
s
```

Out[89]: 'god is great all the time'

In [90]:
```
s.endswith("poo")
```

Out[90]: False

In [93]:
```
li = ["lavanya_p@apssdc.in",
      "geethaandraju88@gmail.com",
     "siddineni2000@gmail.com",
     "tara123",
     "eswargandhi99@gmail.com"]
for email in li:
    if email.endswith("gmail.com"):
        print(email)
```

```
geethaandraju88@gmail.com
siddineni2000@gmail.com
eswargandhi99@gmail.com
```

In [95]:
```
s.count("t")
```

Out[95]: 3

In [96]:
```
s.count(" ")
```

Out[96]: 5

In [102]:
```
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]:
```
s.center(50)
```

Out[101]:
```
'            god is great all the time             '
```

In [104]:
```
s = "sadSDFG"
s.swapcase()
```

Out[104]:
```
'SADsdfg'
```

In [107]:
```
s = "lavanya.123@gmail.com"
s.isalpha()
```

Out[107]:
```
False
```

In [108]:
```
s.isdigit()
```

Out[108]:
```
False
```

In [112]:
```
s = "lavanya123gmailcom"
s.isalnum()
```

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

In [113]:
```
# Task : Count number of digits in a given string
# or
# Task : Print the alphabets in a given string

s = input("enter your data")
for char in s:
    if char.isalpha():
        print(char)
```

```
enter your datagod is good 12234
g
o
d
i
s
g
o
o
d
```

In [114]:
```
s = input("enter your data")
for char in s:
    if not char.isalnum():
        print(char)
```

```
enter your datahdshfgdhj%426@35
%
@
```

In [120]:
```
s = "****styl))))))"
s.strip() # default it removes space
```

Out[120]:
```
'****styl))))))'
```

In [118]:
```
s.lstrip()
```

Out[118]:
```
'style   '
```

In [119]:
```
s.rstrip()
```

Out[119]:
```
'     style'
```

In [121]:
```
s.lstrip("*")
```

Out[121]:
```
'styl))))))'
```

In [122]:
```
s.rstrip(")")
```

Out[122]:
```
'****styl'
```

List :

- collection of data
- Arranged in [] , and seperated by ,
- Mutable object
- Heterogeneous type - int,str,float,list..etc
- data structure object

In [124]:
```
# intialize
l = []
l = list()
type(l)
```

Out[124]:
```
list
```

In [125]:
```
l1 = [1,2,3,4,4]
id(l1)
```

Out[125]:
```
2053733094208
```

In [126]:
```
l1 = ["Tulasi","geetha","Prasanna","gurucharan","lavanya"]
l1[-1]
```

Out[126]: 'lavanya'

In [127]:
```
l1[1]
```

Out[127]: 'geetha'

In [129]:
```
l1[1::2] # step by 2
```

Out[129]: ['geetha', 'gurucharan']

List Method

In [130]:
```
dir(list)
```

Out[130]:
```
['__add__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__delitem__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__gt__',
 '__hash__',
 '__iadd__',
 '__imul__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__mul__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__reversed__',
 '__rmul__',
 '__setattr__',
 '__setitem__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'append',
 'clear',
 'copy',
 'count',
 'extend',
 'index',
 'insert',
 'pop',
 'remove',
 'reverse',
 'sort']
```

In [132]:
```
l1 = ["asad","saf",90,9,90.78,78.56]
l1.append("Tulasi")
l1
```

Out[132]:
```
['asad', 'saf', 90, 9, 90.78, 78.56, 'Tulasi']
```

In [135]:
```
li.clear()
```

In [136]:
```
li
```

Out[136]:
```
[]
```

In [138]:
```
l2 = l1.copy()
l2
```

Out[138]:
```
['asad', 'saf', 90, 9, 90.78, 78.56, 'Tulasi']
```

In [139]:
```
print(id(l1))
id(l2)
```

```
2053732070528
```

Out[139]:
```
2053732137024
```

In [141]:
```
l1.count(90)
```

Out[141]:
```
1
```

In [142]:
```
l1.append("lavanya")
l1
```

Out[142]:
```
['asad', 'saf', 90, 9, 90.78, 78.56, 'Tulasi', 'lavanya']
```

In [143]:
```
l1.extend("lavanya")
```

In [144]:
```
l1
```

Out[144]:
```
['asad',
 'saf',
 90,
 9,
 90.78,
 78.56,
 'Tulasi',
 'lavanya',
 'l',
 'a',
 'v',
 'a',
 'n',
 'y',
 'a']
```

In [147]:
```
l1.append([1,2,3])
l1
```

Out[147]:
```
['asad',
 'saf',
 90,
 9,
 90.78,
 78.56,
 'Tulasi',
 'lavanya',
 'l',
 'a',
 'v',
 'a',
 'n',
 'y',
 'a',
 [1, 2, 3],
 [1, 2, 3],
 [1, 2, 3]]
```

In [148]:
```
l1.extend([1,2,3])
```

In [149]:
```
l1
```

Out[149]:
```
['asad',
 'saf',
 90,
 9,
 90.78,
 78.56,
 'Tulasi',
 'lavanya',
 'l',
 'a',
 'v',
 'a',
 'n',
 'y',
 'a',
 [1, 2, 3],
 [1, 2, 3],
 [1, 2, 3],
 1,
 2,
 3]
```

In [150]:
```
l = [1,2,3,4]
l.append([5,6,7])
l
```

Out[150]:
```
[1, 2, 3, 4, [5, 6, 7]]
```

In [151]:
```
l = [1,2,3,4]
l.extend([5,6,7])
l
```

Out[151]:
```
[1, 2, 3, 4, 5, 6, 7]
```

In [154]:
```
l.insert(2,100)
l
```

Out[154]:
```
[1, 2, 100, 3, 4, 5, 6, 7, 2]
```

In [155]:
```
l.index(100)
```

Out[155]:
```
2
```

In [156]:
```
l.remove(100)
```

In [157]:
```
l
```

Out[157]:
```
[1, 2, 3, 4, 5, 6, 7, 2]
```

In [159]:
```
sorted(l)
```

Out[159]:
```
[1, 2, 2, 3, 4, 5, 6, 7]
```

In [161]:
```
sorted(l,reverse=False)
```

Out[161]:
```
[1, 2, 2, 3, 4, 5, 6, 7]
```

Task:

- count number of alpha, digits, spl chars
  - alpha : 12
  - spl char : 3
  - digits : 10
  - space : 5

Task:

- Take one empty list
- ask your friends to enter their mail id and add to the list
- print only the user name/id

Tuples

- Read only list
- Which are used to store multiple items in a single variable
- Collection of elements which are enclosed in ()
- immutable data type
- Duplicates are allowed

In [162]:
```
t = ()
type(t)
```

Out[162]:
```
tuple
```

In [163]:
```
t = tuple()
type(t)
```

Out[163]:
```
tuple
```

In [164]:
```
dir(t)
```

Out[164]:
```
['__add__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__getnewargs__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__mul__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__rmul__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'count',
 'index']
```

In [165]:
```
# applications / used of tuple

t = (10,24,56.56,465,56)
t.count(10)
```

Out[165]:
```
1
```

In [166]:
```
t.index(465)
```

Out[166]:
```
3
```

In [167]:
```
t.index(0)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-167-586cccc47b32> in <module>
----> 1 t.index(0)

ValueError: tuple.index(x): x not in tuple
```

In [169]:
```
sorted(t)  #
```

Out[169]:
```
[10, 24, 56, 56.56, 465]
```

dict, sets