Factorial of a number

In [7]:
```
def fact(x):
    f=1
    for i in range(x,0,-1):
        f=f*i
    return f
n=int(input())
result=fact(n)
print("result is",result)
```

```
5
result is 120
```

Recursion

when a function calls itself again & again

In [10]:
```
def abc():
    print("Raushan")

abc()
```

```
Raushan
```

In [12]:
```
def fact(x):
    if x ==1:
        return 1
    else:
        return (x * fact(x-1))

n=int(input())
result=fact(n)
print("result is",result)
```

```
5
result is 120
```

Anonymous Function or Lamba Function

In [13]:
```
#  A function without having a name
#  Single line function
# Not having return or def
```

Syntax: Lammbda multiple_arguments: expression

where expression returns an object and it is only the whole function multiple_arguments used a comma to separate multiple arguments

In [14]:
```
def  add_10(x):  # Normal function definition
    return x+10

print(add_10(47))
```

```
57
```

In [15]:
```
lambda_10 = lambda x:x+10  # Lambda function definition

print(add_10(47))
```

```
57
```

In [21]:
```
def add_10(x):   # Normal function definition
```

```
    return x+10

z=add_10(56)
print(z)
```

```
66
```

In [17]:

```
def add_10(x):      # Normal function definition
    return x+10

result=add_10(47)
print(result)
```

```
57
```

In [20]:

```
lambda_10=lambda x:x+10    # Lambda function definition

print(lambda_10(49))
```

```
59
```

In [2]:

```
def sum(x,y,z):
    return x+y+z

r=sum(10,20,30)
print(r)
```

```
60
```

In [1]:

```
lambda_sum = lambda x,y,z:x+y+z

print(lambda_sum(10,20,30))
```

```
60
```

LISTS

Lists are used to store multiple items in a single variable

Lists are created by just placing the sequence inside the square brackets[].

Lists are mutable

Lists are ordered and have a definite count

Indexing in list start from 0

In [4]:

```
# Blank lists
list1 = []
print(list1)

# List having numbers
list1 = [2,4,6,8,2,4]
print(list1)
print(list1[4])
print(list1[-4])

# To know length of list
print(len(list1))

list2 = ["Chitkara","University", "Punjab"]
print(list2[1])

# Nested List
list3=[["Chitkara","University"],["Punjab"]]
print(list3)
print(list3[1])
print(list3[0][1])
```

```
[]
[2, 4, 6, 8, 2, 4]
```

```
2
6
6
University
[['Chitkara', 'University'], ['Punjab']]
['Punjab']
University
```

In [ ]:

```
# list1= [2,4,6,8,2,4]
list1.append(7)
print(list1)

list2=[9,10]
list1.append(list2)
print(list1)
print(list1[7])


list1.insert(0,"Raushan")
print(list1)
list1.insert(7,18)
print(list1)

# Add multiple elements at one time at end at end of list
list1.extend([12,13,14,"great"])
print(list1)
```

Removing elements

In [6]:

```
list1 = [2,4,6,8,2,4]
list1.remove(6)
print(list1)


list1.remove(4)
print(list1)

list1.pop()   #remove last element from list
print(list1)
```

```
[2, 4, 8, 2, 4]
[2, 8, 2, 4]
[2, 8, 2]
```

Slicing

In [10]:

```
list1 = [2,4,6,8,10,14]
sliced_list1=list1[1:5]
print(sliced_list1)

sliced_list1=list1[::]
print(sliced_list1)

sliced_list1=list1[2:]
print(sliced_list1)

sliced_list1=list1[-2:-5:-1]
print(sliced_list1)
```

```
[4, 6, 8, 10]
[2, 4, 6, 8, 10, 14]
[6, 8, 10, 14]
[10, 8, 6]
```

List comprehension

To create a new lists from other iterables like tuples, srtings, arrays, lists

syntax

newList= [expression(element) for element in oldList if condition]

Even square

In [11]:

```
even_square = [x ** 2 for x in range(1,21) if x % 2 == 0]
print (even_square)
```

```
[4, 16, 36, 64, 100, 144, 196, 256, 324, 400]
```

In [13]:

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]
newlist = [x for x in fruits if "e" in x]

print(newlist)
```

```
['apple', 'cherry']
```

>>>Raushan Raj

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js