In [1]:
```
# print fibonnaci sequence upto 10 numbers
```

In [10]:
```
def fib(n):
    a = 0
    b = 1
    if n == 1:
        print(a)
    else:

        print(0)
        print(1)
        for i in range(0,n):
            c = a + b
            a = b
            b = c
            print(c)

fib(10)
```

```
0
1
1
2
3
5
8
13
21
34
55
89
```

In [11]:
```
# FACTORIAL OF A NUMBER IN PYTHON
```

In [20]:
```
def fact(n):
    f = 1
    for i in range(1,n+1):
        f = f*i
    return f


fact(5)
```

Out[20]: 120

In [21]:
```
''' recurrsion- function calling itself'''
```

Out[21]: ' recurrsion- function calling itself'

In [22]:
```
''' tell me what do you know about recurrsion?'''
''' recurrsion means the function calling the function itself inside the scope.
recurrsion has 2 rules
```

```
1)function must call itself
2)it must have the base case.
  where the function stop calling itself
  (otherwise it would call itself infinite times)
  and this is simple case which should provide the
  answer without calling the function itself

 1) example --> countdown(print)
 suppose you want to countdown in the form  5 4 3 2 1- normal way

  using reccursion we can use countdown

  def countdown(n):
      if n == 1:
          return (1) # not print it should be return
       else:
          print(n)
          countdown(n-1)'''
```

Out[22]:
```
' recurrsion means the function calling the function itself inside the scope.\nrec
urrsion has 2 rules\n1)function must call itself\n2)it must have the base case.\n
where the function stop calling itself\n  (otherwise it would call itself infinite
times) \n  and this is simple case which should provide the\n  answer without call
ing the function itself \n  \n 1) example --> countdown(print)\n suppose you want
to countdown in the form  5 4 3 2 1- normal way\n  \n  using reccursion we can use
countdown\n  \n  def countdown(n):\n      if n == 1:\n          print(1)\n       e
lse:\n          print(n)\n          countdown(n-1)'
```

In [35]:
```
def wish(n):
    for i in range(n):
        i+=1
        print('hello',i)
        wish(n)

wish(50)
```

```
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
```

```
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
```

```
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
hello 1
```

```
---------------------------------------------------------------------------
RecursionError                            Traceback (most recent call last)
Cell In[35], line 7
      4         print('hello',i)
      5         wish(n)
----> 7 wish(50)

Cell In[35], line 5, in wish(n)
      3 i+=1
      4 print('hello',i)
----> 5 wish(n)

Cell In[35], line 5, in wish(n)
      3 i+=1
      4 print('hello',i)
----> 5 wish(n)

    [... skipping similar frames: wish at line 5 (122 times)]

Cell In[35], line 5, in wish(n)
      3 i+=1
      4 print('hello',i)
----> 5 wish(n)

Cell In[35], line 4, in wish(n)
      2 for i in range(n):
      3     i+=1
----> 4     print('hello',i)
      5     wish(n)

File ~\anaconda3\Lib\site-packages\ipykernel\iostream.py:664, in OutStream.write(sel
f, string)
    655 def write(self, string: str) -> Optional[int]:  # type:ignore[override]
    656     """Write to current stream after encoding if necessary
    657
    658     Returns
   (...)
    662
    663     """
--> 664     parent = self.parent_header
    666     if not isinstance(string, str):
    667         msg = f"write() argument must be str, not {type(string)}"  # type:ig
nore[unreachable]

RecursionError: maximum recursion depth exceeded
```

In [30]:
```
import sys
print(sys.getrecursionlimit())
```

```
3000
```

In [36]:
```
import sys
sys.setrecursionlimit(150)
print(sys.getrecursionlimit())
i = 0
def wish():
```

```
    global i
    print('hello',i)
    i += 1
    wish()
wish()
```

```
150
hello 0
hello 1
hello 2
hello 3
hello 4
hello 5
hello 6
hello 7
hello 8
hello 9
hello 10
hello 11
hello 12
hello 13
hello 14
hello 15
hello 16
hello 17
hello 18
hello 19
hello 20
hello 21
hello 22
hello 23
hello 24
hello 25
hello 26
hello 27
hello 28
hello 29
hello 30
hello 31
hello 32
hello 33
hello 34
hello 35
hello 36
hello 37
hello 38
hello 39
hello 40
hello 41
hello 42
hello 43
hello 44
hello 45
hello 46
hello 47
hello 48
hello 49
hello 50
hello 51
hello 52
hello 53
hello 54
```

```
hello 55
hello 56
hello 57
hello 58
hello 59
hello 60
hello 61
hello 62
hello 63
hello 64
hello 65
hello 66
hello 67
hello 68
hello 69
hello 70
hello 71
hello 72
hello 73
hello 74
hello 75
hello 76
hello 77
hello 78
hello 79
hello 80
hello 81
hello 82
hello 83
hello 84
hello 85
hello 86
hello 87
hello 88
hello 89
hello 90
hello 91
hello 92
hello 93
hello 94
hello 95
hello 96
hello 97
hello 98
hello 99
hello 100
hello 101
hello 102
hello 103
hello 104
hello 105
hello 106
hello 107
hello 108
hello 109
hello 110
```

```
hello 111
hello 112
hello 113
hello 114
hello 115
hello 116
hello 117
hello 118
hello 119
hello 120
hello 121
hello 122
hello 123
hello 124
```

```
---------------------------------------------------------------------------
RecursionError                            Traceback (most recent call last)
Cell In[36], line 10
      8     i += 1
      9     wish()
---> 10 wish()

Cell In[36], line 9, in wish()
      7 print('hello',i)
      8 i += 1
----> 9 wish()

Cell In[36], line 9, in wish()
      7 print('hello',i)
      8 i += 1
----> 9 wish()

    [... skipping similar frames: wish at line 9 (122 times)]

Cell In[36], line 9, in wish()
      7 print('hello',i)
      8 i += 1
----> 9 wish()

Cell In[36], line 7, in wish()
      5 def wish():
      6     global i
----> 7     print('hello',i)
      8     i += 1
      9     wish()

File ~\anaconda3\Lib\site-packages\ipykernel\iostream.py:664, in OutStream.write(sel
f, string)
    655 def write(self, string: str) -> Optional[int]:  # type:ignore[override]
    656     """Write to current stream after encoding if necessary
    657
    658     Returns
   (...)
    662
    663     """
--> 664     parent = self.parent_header
    666     if not isinstance(string, str):
    667         msg = f"write() argument must be str, not {type(string)}"  # type:ig
nore[unreachable]

RecursionError: maximum recursion depth exceeded
```

factorial using recursion

In [97]:
```
# 5*4*3*2*1  1*2*3*4*5           #
def fact(n):
    if n == 0:
        return 1
    return n*fact(n-1)
```

```
result = fact(5)
result


'''-----------------------------------------------
normal factorial
-----------------------------------------------
def fact(n):
    f = 1
    for i in range(1,n+1):
        f = f*i
    return f


fact(5)
    '''
```

Out[97]:
```
'-----------------------------------------------\nnormal factorial\n--------------
-----------------------------------\ndef fact(n):\n    f = 1\n    for i in range
(1,n+1):\n        f = f*i\n    return f\n        \n    \nfact(5)\n    '
```

anonymous Function| lambda function

Function wothout name is called - Anonymous function or lambda function

In [53]:
```
'''
with lambda function we can reduce the lines of code let say from 7 to 4
'''
```

Out[53]:
```
' \nwith lambda function we can reduce the lines of code let say from 7 to 4 \n'
```

In [98]:
```
add3 = lambda x: x + 3   # take x and returns x+3
add3(7)  # returns 10
''' if you want to use it multiple time then store it in the variable '''
```

Out[98]:
```
' if you want to use it multiple time then store it in the variable '
```

In [56]:
```
def square(x):
    return x*x
square(5)
```

Out[56]:
```
25
```

In [58]:
```
square = lambda x: x*x
square(5)
```

Out[58]:
```
25
```

In [59]:
```
f = lambda a, b : a + b
f1 = lambda a, b : a - b

result = f(1,4)
result1 = f1(4,1)

print(result)
print(result1)
```

```
5
3
```

how to use the lambda function in other function like filter ,map ,reduce

In [79]:
```
# filter
'''
filter function takes the list and keeps only items which pass the test condition.e

num  = [1,2,3,4,5,6]
evens = list(filter(lambda x:x%2==0,num))
evens
```

Out[79]: [2, 4, 6]

In [62]:
```
def is_even(n):
    return n % 2 == 0

nums = [3,2,6,8,4,6,2,9]

evens = list(filter(is_even, nums))  # is_even is not an inbuilt function
print(evens)
```

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

In [65]:
```
num = [1,2,3,4,5,6,7,8,9,10]
odds = list(filter(lambda x : x %2 != 0,num))

odds
```

Out[65]: [1, 3, 5, 7, 9]

In [74]:
```
def is_odd(x):              # in order to us efunction in map filter reduce,the fun
                                # case should be pass
    return x % 2 != 0
nums1 = [1,2,5,6,4,8,9]
odd1 = list(filter(is_odd,nums1))   # filter without lambda also
print(odd1)
```

```
[1, 5, 9]
```

In [75]:
```
nums2 = [1,5,9,8,4,5,6,8,755]
```

```
# lambda helps us to reduce the code for better optimization


odds5 = list(filter(lambda x: x%2 != 0,nums2 ))
odds5
```

Out[75]: [1, 5, 9, 5, 755]

map function will be use to transform the list using some function ex : square each element

In [77]:
```
num3 = [1,2,3,4,56]
squared = list(map(lambda x : x*x,num3))
squared
```

Out[77]: [1, 4, 9, 16, 3136]

In [78]:
```
nums = [3,2,6,8,4,6,2,9]

evens = list(filter(is_even, nums))

double = list(map(lambda n : n*2, evens))
double_ = list(map(lambda n : n+2, evens))
doubble_1 = list(map(lambda n : n-2, evens))

print(evens)
print(double)
print(double_)
print(doubble_1)
```

```
[2, 6, 8, 4, 6, 2]
[4, 12, 16, 8, 12, 4]
[4, 8, 10, 6, 8, 4]
[0, 4, 6, 2, 4, 0]
```

reduce function keeps applynig some function till only one value lasts till end

In [83]:
```
nums5= [1,2,3,4,5,67,8,9,]
from functools import reduce
def add(a,b):
    return a + b
result5 = reduce(add,nums5)
result5
```

Out[83]: 99

In [84]:
```
re = reduce(lambda a,b:a-b,nums5)
re
```

Out[84]: -97

sorted() function which sort the list by taking key parameter. e.g. fruits using lenghth of string

In [95]:
```
# sorted using the parameter of the lenghth of the string
fruits = ['apple','pineapple','banana','orange','watermelon']
sorted_fruits = sorted(fruits,key = lambda x : len(x))
sorted_fruits
```

Out[95]: ['apple', 'banana', 'orange', 'pineapple', 'watermelon']

In [92]:
```
# sorted using the last digit of number
num6 = [28,14,32,46,9]
sorted_num6 = sorted(num6,key = lambda x: x % 10)
sorted_num6
```

Out[92]: [32, 14, 46, 28, 9]

In [ ]: