Global Keyword

- The global keyword is used inside a function to indicate that it refers to global variable not local variable

In [4]:
```
x=10
def update():
    x=x+6
    return x
print(x)
update()
```

```
10
---------------------------------------------------------------------------
UnboundLocalError                         Traceback (most recent call last)
Cell In[4], line 6
      4     return x
      5 print(x)
----> 6 update()

Cell In[4], line 3, in update()
      2 def update():
----> 3     x=x+6
      4     return x

UnboundLocalError: cannot access local variable 'x' where it is not associated with
a value
```

In [6]:
```
x=10
def update():
    global x
    x=x+6
    return x
print(x)
update()
```

```
10
```

Out[6]: 16

Globals()

- globals() is a built-in function that returns a dictionary representing the current global symbol table.
- It allows you to access and modify global variables programmatically.

In [9]:
```
x=10
def update():
    globals()['x']=x+15 #acess & modify the global variable using dictionary
    return x
```

```
print(x)
update()
```

```
10
```

Out[9]: 25

Passing List to a Function

In [14]:
```
# Count of Even & Odd numbers in a list using Functions
def count(lst):
    even=0
    odd=0
    for i in lst:
        if i%2==0:
            even+= 1
        else:
            odd+= 1
    print('EvenCount:',even)
    print('OddCount:',odd)
nums=[1,7,8,17,54,18,2,6,10]
count(nums)
```

```
EvenCount: 6
OddCount: 3
```

In [16]:
```
# Fibonnaci Sequence
def fib(n):
    a,b=0,1
    if n<=1:
        return a
    else:
        print(a)
        print(b)
        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 [20]:
```
# Factorial
def fact(n):
```

```
    res=1
    for i in range(1,n+1):
        res=res*i
    return res
fact(5)
```

Out[20]: 120

In [33]:
```
# Factorial
def fact(n):
    if n==0:
        return 1
    else:
        return n*fact(n-1)
fact(5)
```

Out[33]: 120

Recursive Function

- calling function itself

In [23]:
```
def fun():
    print('Hello')
fun()
```

```
Hello
```

In [25]:
```
def fun():
    print('Hello')
    fun()
fun()
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
```

```
Hello
Hello
Hello
Hello
Hello
Hello
```

```
---------------------------------------------------------------------------
RecursionError                            Traceback (most recent call last)
Cell In[25], line 4
      2     print('Hello')
      3     fun()
----> 4 fun()

Cell In[25], line 3, in fun()
      1 def fun():
      2     print('Hello')
----> 3     fun()

Cell In[25], line 3, in fun()
      1 def fun():
      2     print('Hello')
----> 3     fun()

    [... skipping similar frames: fun at line 3 (2970 times)]

Cell In[25], line 3, in fun()
      1 def fun():
      2     print('Hello')
----> 3     fun()

Cell In[25], line 2, in fun()
      1 def fun():
----> 2     print(       )
      3     fun()

File ~\AppData\Roaming\Python\Python312\site-packages\IPython\core\interactiveshell.
py:3038, in InteractiveShell._tee.<locals>.write(data, *args, **kwargs)
   3036 if not data:
   3037     return result
-> 3038 execution_count = self.execution_count
   3039 output_stream = None
   3040 outputs_by_counter = self.history_manager.outputs

File ~\AppData\Roaming\Python\Python312\site-packages\traitlets\traitlets.py:687, in
TraitType.__get__(self, obj, cls)
    685     return self
    686 else:
--> 687     return t.cast(G, self.get(obj, cls))

File ~\AppData\Roaming\Python\Python312\site-packages\traitlets\traitlets.py:666, in
TraitType.get(self, obj, cls)
    664     raise TraitError("Unexpected error in TraitType: default value not set p
roperly") from e
    665 else:
--> 666     return t.cast(G, value)

RecursionError: maximum recursion depth exceeded
```

In [27]:
```
import sys
sys.getrecursionlimit()
```

Out[27]: 3000

In [29]:
```
sys.setrecursionlimit(45)
```

In [31]:
```
i=0
def fun():
    global i
    i+=1
    print('Hello',i)
    fun()
fun()
```

```
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
HelloUnexpected exception formatting exception. Falling back to standard exception
```

```
Traceback (most recent call last):
  File "C:\Users\HP\AppData\Roaming\Python\Python312\site-packages\IPython\core\inte
ractiveshell.py", line 3667, in run_code
    exec(code_obj, self.user_global_ns, self.user_ns)
  File "C:\Users\HP\AppData\Local\Temp\ipykernel_4632\1900560786.py", line 7, in <mo
dule>
    fun()
  File "C:\Users\HP\AppData\Local\Temp\ipykernel_4632\1900560786.py", line 6, in fun
    fun()
  File "C:\Users\HP\AppData\Local\Temp\ipykernel_4632\1900560786.py", line 6, in fun
    fun()
  File "C:\Users\HP\AppData\Local\Temp\ipykernel_4632\1900560786.py", line 6, in fun
    fun()
  [Previous line repeated 15 more times]
  File "C:\Users\HP\AppData\Local\Temp\ipykernel_4632\1900560786.py", line 5, in fun
    print('Hello',i)
  File "C:\Users\HP\AppData\Roaming\Python\Python312\site-packages\IPython\core\inte
ractiveshell.py", line 3038, in write
    execution_count = self.execution_count
                      ^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\HP\AppData\Roaming\Python\Python312\site-packages\traitlets\traitle
ts.py", line 687, in __get__
    return t.cast(G, self.get(obj, cls))  # the G should encode the Optional
                     ^^^^^^^^^^^^^^^^^^
  File "C:\Users\HP\AppData\Roaming\Python\Python312\site-packages\traitlets\traitle
ts.py", line 666, in get
    return t.cast(G, value)
           ^^^^^^^^^^^^^^^^
RecursionError: maximum recursion depth exceeded

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "C:\Users\HP\AppData\Roaming\Python\Python312\site-packages\IPython\core\inte
ractiveshell.py", line 2176, in showtraceback
    stb = self.InteractiveTB.structured_traceback(
          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\HP\AppData\Roaming\Python\Python312\site-packages\IPython\core\ultr
atb.py", line 1182, in structured_traceback
    return FormattedTB.structured_traceback(
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\HP\AppData\Roaming\Python\Python312\site-packages\IPython\core\ultr
atb.py", line 1053, in structured_traceback
    return VerboseTB.structured_traceback(
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\HP\AppData\Roaming\Python\Python312\site-packages\IPython\core\ultr
atb.py", line 861, in structured_traceback
    formatted_exceptions: list[list[str]] = self.format_exception_as_a_whole(
                                            ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\HP\AppData\Roaming\Python\Python312\site-packages\IPython\core\ultr
atb.py", line 746, in format_exception_as_a_whole
    records = self.get_records(etb, context, tb_offset) if etb else []
              ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\HP\AppData\Roaming\Python\Python312\site-packages\IPython\core\ultr
atb.py", line 848, in get_records
    res = list(stack_data.FrameInfo.stack_data(etb, options=options))[tb_offset:]
```

```
          ^^^
```

```
                                ^^^^^^^^^^^^^^^^^
RecursionError: maximum recursion depth exceeded
```

Anonymous or Lambda Function

- a function without a name called as anonymous or lambda function

In [42]:
```
# Normal Function
def square(n):
    return n*n
square(6)
```

Out[42]: 36

In [44]:
```
# Lambda Function
s=lambda n:n*n #argument,operation
s(6)
```

Out[44]: 36

In [46]:
```
add=lambda a,b:a+b
add(18,42)
```

Out[46]: 60

filter()

- filter() is used to filter elements from an iterable based on a function that returns True or False.

In [51]:
```
nums=[1,5,7,6,8,6,3,8,5,9,0]
```

In [55]:
```
def is_even(n):
    return n%2==0
list(filter(is_even,nums))
```

Out[55]: [6, 8, 6, 8, 0]

In [57]:
```
list(filter(lambda n:n%2==0,nums))
```

Out[57]: [6, 8, 6, 8, 0]

In [59]:
```
def is_odd(n):
    return n%2!=0
list(filter(is_odd,nums))
```

Out[59]: [1, 5, 7, 3, 5, 9]

In [61]:
```
list(filter(lambda n:n%2!=0,nums))
```

Out[61]: [1, 5, 7, 3, 5, 9]

map()

- map() applies a function to all items in an iterable and returns a map object, which can be converted into a list or another iterable type.

In [64]:
```
nums=[1,2,3,4,5]
def add2(n):
    return n+2
list(map(add2,nums))
```

Out[64]: [3, 4, 5, 6, 7]

In [66]:
```
list(map(lambda n:n*2,nums)) #square of nums
```

Out[66]: [2, 4, 6, 8, 10]

reduce()

- reduce() is part of the functools module and is used to apply a function cumulatively to the elements of an iterable.

In [69]:
```
# we have to import reduce from functoolsn
from functools import reduce
def add_all(a,b):
    return a+b

reduce(add_all,nums)
```

Out[69]: 15

In [71]:
```
reduce(lambda a,b:a+b,nums)
```

Out[71]: 15

In [73]:
```
reduce(lambda a,b:a*b,nums)
```

Out[73]: 120

Decorator

- A decorator in Python is a design pattern that allows you to modify the behavior of a function or method without changing its actual code. It wraps a function inside another function to extend its functionality.

In [77]:
```
def decorator_function(original_function):
    def wrapper_function():
        print("Wrapper executed before", original_function.__name__)
        return original_function()
    return wrapper_function

@decorator_function
def display():
    print("Display function executed")
display()
```

```
Wrapper executed before display
Display function executed
```

In [79]:
```
def my_decorator(func):
    def wrapper():
        print("Something is happening before the function is called.")
        func()
        print("Something is happening after the function is called.")
    return wrapper

@my_decorator
def say_hello():
    print("Hello!")
say_hello()
```

```
Something is happening before the function is called.
Hello!
Something is happening after the function is called.
```

In [ ]:

In [ ]: