Namespaces

A namespace is a space that holds names(identifiers).Programmatically speaking, namespaces are dictionary of identifiers(keys) and their objects(values)

There are 4 types of namespaces:

- Builtin Namespace
- Global Namespace
- Enclosing Namespace
- Local Namespace

Scope and LEGB Rule

- LEBB => Local,Enclosing,Global,Builtin

A scope is a textual region of a Python program where a namespace is directly accessible.

The interpreter searches for a name from the inside out, looking in the local, enclosing, global, and finally the built-in scope. If the interpreter doesn't find the name in any of these locations, then Python raises a NameError exception.

```
# local and global
a=2
def temp():
    b=3
    print(b)

temp()
print(a)
print(b)
```

```
3
2
```

```
----------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
Cell In[7], line 9
      7 temp()
      8 print(a)
----> 9 print(b)

NameError: name 'b' is not defined
```

```
# local and global -> same name
a=2
def temp():
    a=3
    print(a)
```

```
temp()
print(a)
```

```
3
2
```

```
# local and global -> local does not have but global has
# agar local me nahi hai to wo global ko print kara dega but vice
versa is not true
a=2
def temp():

    print(a)

temp()
print(a)
```

```
2
2
```

```
a=2
def temp():
    b=5
    print(b)

temp()
print(b)
```

```
5
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[20], line 7
      4     print(b)
      6 temp()
----> 7 print(b)

NameError: name 'b' is not defined
```

```
# # local and global ->editing global
a=2
def temp():
    a+=3                  # a global hai to hum usme localy update nahi
kar sakte hain
    print(a)
```

```
temp()
print(a)

---------------------------------------------------------------------------
UnboundLocalError                         Traceback (most recent call last)
Cell In[22], line 7
      4     a+=3
      5     print(a)
----> 7 temp()
      8 print(a)

Cell In[22], line 4, in temp()
      3 def temp():
----> 4     a+=3
      5     print(a)

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

```
a=2
def temp():

    global a   # abb yaha bata rahe hain ki global bale a me jake change karo to ye change ho jayega(ye basically nahi karna chahaiye)
    a+=3
    print(a)

temp()
print(a)

5
5
```

```
# local and global -> global created inside local

def temp():

    global a
    a=1       # humne globaly koi a name ka variable banaya hi nahi but hum glabal likh ke local ke andar ek variable bana rahe hai to
    print(a)   #  wo variable local hoga ya global

temp()
print(a)

# ye global variable banta hai

1
1
```

```
# local and global -> function parameter is local
def temp(z):
    #local var
    print(z)

a=5
temp(a)
print(a)
print(z) # ye z ko access nahi kar payega qki z local variable hai
```

```
5
5

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[32], line 9
      7 temp(a)
      8 print(a)
----> 9 print(z) # ye z ko access nahi kar payega qki z local variable hai

NameError: name 'z' is not defined
```

built-in scope

```
# built-in scope  => jo bhi pre-defined function hai wo sab kuch built-in scope ka part hai-> ye global se bhi upar hai ,
# ye sare jaise hi python suru hota hai to pahle se hi scope me aa jata hai ki ye sare program mr to use honge hi
print('hello')
```

```
hello
```

```
# how to see all built-ins
import builtins
print(dir(builtins))
```

```
['ArithmeticError', 'AssertionError', 'AttributeError', 'BaseException', 'BaseExceptionGroup', 'BlockingIOError', 'BrokenPipeError', 'BufferError', 'BytesWarning', 'ChildProcessError', 'ConnectionAbortedError', 'ConnectionError', 'ConnectionRefusedError', 'ConnectionResetError', 'DeprecationWarning', 'EOFError', 'Ellipsis', 'EncodingWarning', 'EnvironmentError', 'Exception', 'ExceptionGroup', 'False', 'FileExistsError', 'FileNotFoundError', 'FloatingPointError', 'FutureWarning', 'GeneratorExit', 'IOError', 'ImportError', 'ImportWarning', 'IndentationError', 'IndexError', 'InterruptedError', 'IsADirectoryError', 'KeyError', 'KeyboardInterrupt', 'LookupError', 'MemoryError', 'ModuleNotFoundError', 'NameError', 'None',
'NotADirectoryError', 'NotImplemented', 'NotImplementedError',
'OSError', 'OverflowError', 'PendingDeprecationWarning',
'PermissionError', 'ProcessLookupError', 'RecursionError',
'ReferenceError', 'ResourceWarning', 'RuntimeError', 'RuntimeWarning',
'StopAsyncIteration', 'StopIteration', 'SyntaxError', 'SyntaxWarning',
'SystemError', 'SystemExit', 'TabError', 'TimeoutError', 'True',
'TypeError', 'UnboundLocalError', 'UnicodeDecodeError',
'UnicodeEncodeError', 'UnicodeError', 'UnicodeTranslateError',
'UnicodeWarning', 'UserWarning', 'ValueError', 'Warning',
'WindowsError', 'ZeroDivisionError', '__IPYTHON__', '__build_class__',
'__debug__', '__doc__', '__import__', '__loader__', '__name__',
'__package__', '__spec__', 'abs', 'aiter', 'all', 'anext', 'any',
'ascii', 'bin', 'bool', 'breakpoint', 'bytearray', 'bytes',
'callable', 'chr', 'classmethod', 'compile', 'complex', 'copyright',
'credits', 'delattr', 'dict', 'dir', 'display', 'divmod', 'enumerate',
'eval', 'exec', 'execfile', 'filter', 'float', 'format', 'frozenset',
'get_ipython', 'getattr', 'globals', 'hasattr', 'hash', 'help', 'hex',
'id', 'input', 'int', 'isinstance', 'issubclass', 'iter', 'len',
'license', 'list', 'locals', 'map', 'max', 'memoryview', 'min',
'next', 'object', 'oct', 'open', 'ord', 'pow', 'print', 'property',
'range', 'repr', 'reversed', 'round', 'runfile', 'set', 'setattr',
'slice', 'sorted', 'staticmethod', 'str', 'sum', 'super', 'tuple',
'type', 'vars', 'zip']
```

```
L=[1,2,3]
print(L)
```

```
[1, 2, 3]
```

```
# renamining built-ins
L=[1,2,3]

def max():    # jo pre-definede(built-ins) max hai usse hum rename kar
de rahe hain and ek function ka name bana de rahe hain
    print('hello')

max(L)

# error aayega qki abb jo max function rename hua hai usme 0 parametr
pass required hai but aap 1 bhej rahe ho
```

```
----------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
Cell In[41], line 7
      4 def max():    # jo pre-definede(built-ins) max hai usse hum
rename kar de rahe hain and ek function ka name bana de rahe hain
      5     print('hello')
----> 7 max(L)
```

```
TypeError: max() takes 0 positional arguments but 1 was given
```

```
L=[1,2,3]
print(max(L))   # yaha se 3 print ho jayega qki abhi rename (function
nahi ncreate hua hai) nahi hua hai
def max():
    print('hello')

max(L)
```

```
3
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call
last)
Cell In[21], line 6
      3 def max():
      4     print('hello')
----> 6 max(L)

TypeError: max() takes 0 positional arguments but 1 was given
```

```
# koi bhi data ko dhundhne ka rule kuch iss parkar hota hai ki pahle
wo local me dhundhta hai , phie enclosed me , phir global me ,
# and then phir built-ins me agar data local me mil gya toh uske
according kaam karega nahi to enclosed me dhundhega, nahi to global me
and last me
# built-ins me search karta hai

# yaha pe max jab global me mil gya to ye max global ke according use
hoga and ye built-ins max ko search nahi karega
```

Enclosing

```
# enclosing scope -> ye basically nested ke andar dekhne milta hai jab
bhi functions ke andar functions hota hai

def outer():
    def inner():
        print('inner function')
    inner()
    print('outer function')

outer()
print('main program')
```

```
inner function
outer function
main program
```

```
# basically jo sabse andar ka jo function/data hota hai usse local
bola jata (yaha pe inner functio) hai and jo main program ka scope
# hai usse global and jo yaha pe outer function hai usse hi enclosing
bola jata hai aur kabhi kabhi usse non-local bhi bola jata hai



def outer():
    a=3
    def inner():
        a=4
        print(a)                          # sabse pahle LEGB rule ke
according local me a ko khoja mil gya to usi ko print kara dya
        print('inner function')
    inner()
    print('outer function')

a=1
outer()
print('main program')
```

```
4
inner function
outer function
main program
```

```
def outer():
    a=3
    def inner():

        print(a)                          # sabse pahle LEGB rule ke
according local me a ko khoja nahi mila to enclosing me gya and print
karaya
        print('inner function')
    inner()
    print('outer function')

a=1
outer()
print('main program')
```

```
3
inner function
outer function
main program
```

```
def outer():

    def inner():

        print(a)                          # sabse pahle LEGB rule ke
```

```
according local me a ko khoja nahi mila to enclosing me gya waha bhi
nahi mila and
        print('inner function')           #  global me gya and print
kara dya
    inner()
    print('outer function')

a=1
outer()
print('main program')
```

```
1
inner function
outer function
main program
```

```
# agar function ke andar function ke andar function rahe to -> sabse
andar bala local , uske bahar bala enclosing-1 and uske bahar bala
enclosing-2 ...

# nonlocal keyword

def outer():
    a=2
    def inner():
        a+=2
        print(a)
        print('inner function')
    inner()
    print('outer function')


outer()
print('main program')
```

```
----------------------------------------------------------------------
-----
UnboundLocalError                         Traceback (most recent call
last)
Cell In[14], line 13
      9     inner()
     10     print('outer function')
---> 13 outer()
     14 print('main program')

Cell In[14], line 9, in outer()
      7     print(a)
      8     print('inner function')
----> 9 inner()
     10 print('outer function')
```

```
Cell In[14], line 6, in outer.<locals>.inner()
      5 def inner():
----> 6     a+=2
      7     print(a)
      8     print('inner function')

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

```python
def outer():
    a=2
    def inner():
        nonlocal a    # nonlocal keyword ke help se hum enclosing ko inner scope se change kar sakte hain
        a+=2
        print('inner',a)
    inner()
    print('outer ',a)


outer()
print('main program')
```

```
inner 4
outer  4
main program
```

Decorators

A decorator in python is a function that receives another function as input and adds some functionality(decoration) to it and returns it.

This can happen only because python functions are 1st class citizens.

There are 2 types of decorators available in python

- `Built in decorators` like `@staticmethod`, `@classmethod`, `@abstractmethod` and `@property` etc
- `User defined decorators` that we programmers can create according to our needs

```python
# 1st class citizen means what --> 1st class citizen wo objects hote hain kisi bhi programming language me  jiske sath aap sare operation kar sakte ho,
# like create kar sakte ho, store kar sakte ho , delete kar sakte ho , usme input de sakte ho, function usse kisi tarah se output kar sakta hai and
# python me function bhi 1st class citizen hai isse aap create, kar sakte ho delete kar sakte ho,rename kar sakte ho , kisi list ke andar daal sakte ho,
# function ko kuch input de sakte ho , function se kuch output le sakte ho
# functions are 1st class citizen
def func():
    print('hello')

a=func    # func()  -> bracket laga ke store nahi kar sakte hain kisi
another variable me

a()

hello

def func():
    print('hello')

a=func

del func
func()

---------------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
Cell In[33], line 7
      4 a=func    # func()  -> bracket laga ke store nahi kar sakte
hain kisi another variable me
      6 del func
----> 7 func()

NameError: name 'func' is not defined

# ek function ko dusre ke andar as input
def modify(func,num):
    return func(num)

def square(num):
    return num**2

modify(square,2)


4

# simple example of decorator

# suppose hum apne output ko upar and niche se star line se decorate
karna chah rahe hain

def my_decorator(func):
    def wrapper():   # generally andar ke function  ka name wrapper hi
```

```
rakhte hain
        print('*************************')
        func()
        print('*************************')
    return wrapper

def hello():
    print('hello')

a=my_decorator(hello)
a()

def disply():
    print('hello jay')

b=my_decorator(disply)
b()

*************************
hello
*************************
*************************
hello jay
*************************



#
def outer():
    a=5
    def inner():
        print(a)
    return inner

b=outer()
b()


# yaha basically a outer function ke scope me hai and jab uoter
function kuch return kar deta hai to wo function mar jata hai and uske

# andar ka bhi sabkuch mar jata hai but ye ek special case hai jaha pe
ki outerfunction ke scope ke a variable ko return ke
# baad bhi inner function access kar paa raha hai

#

# matlab ki agar aap ek function se dusre function ko return kar rahe
ho to inner function ke pass bhi ek addree hota hai sare variable ka
# jiske  karan phir se access ho paa raha hai  -> issi property ko
python mr closure bola jata hai
```

```
# closure --> inner function parent function ke marne ke baad bhi unke
chizon ko access kar sakta hai
```

```
5
```

```
# Better syntax
def my_decorator(func):
    def wrapper():
        print('**************************')
        func()
        print('**************************')
    return wrapper

@my_decorator
def hello():
    print('hello')

#a=my_decorator(hello)
#a()

hello()
```

```
**************************
hello
**************************
```

```
# meaning ful decorator

# hum ek aaisa decorator banayenge jo ki kisi bhi function ka
execution time batayega

import time

def timer(func):
    def wrapper():
        start = time.time()
        func()
        print('time taken by ',func.__name__, time.time()-start, '
secs')  #func.__name__ --> aaise likhne se function ka actual name
print ho jata hai
    return wrapper


@timer
def hello():
    print('hello world')
    time.sleep(2)

hello()    # calling

@timer
def display():
    print('displaying something')
    time.sleep(4)




display()   # calling

hello world
time taken by  hello 2.00123929977417  secs
displaying something
time taken by  display 4.001263856887817  secs

# another senario
import time

def timer(func):
    def wrapper():
        start = time.time()
        func()
        print('time taken by ',func.__name__, time.time()-start, '
secs')  #func.__name__ --> aaise likhne se function ka actual name
print ho jata hai
    return wrapper

@timer
def square(num):
    time.sleep(1)
    return num**2

square(2)


# ye islye fat gya qki jaise hi func ko square milega ye ek argument
ke sath milega and the wrapper return hoga and wrapper ke andar se jab
func() call hoga to ye to ek bhi argument
# accept hi nahi kar raha hai and hum ek arguement pass kar rahe hain
islye error aaya

----------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
Cell In[70], line 16
     13     time.sleep(1)
     14     return num**2
---> 16 square(2)

TypeError: timer.<locals>.wrapper() takes 0 positional arguments but 1
was given
```

```
 # so how we can handle this   *args ke help se

import time

def timer(func):
    def wrapper(*args):
        start = time.time()
        func(*args)
        print('time taken by ',func.__name__, time.time()-start, '
secs')  #func.__name__ --> aaise likhne se function ka actual name
print ho jata hai
    return wrapper

@timer
def square(num):
    time.sleep(1)
    print(num**2)

square(2)

@timer
def hello():
    print('hello world')
    time.sleep(2)

hello()
```

```
4
time taken by  square 1.0010876655578613  secs
hello world
time taken by  hello 2.0015687942504883  secs
```

```
# another problem to check ki ki aapko jo input mila hai kisi function
ke andar uska data type sahi hai ki nahi

def square(num):
    print(num**2)

square(2)
```

```
4
```

```
square('hehe')   # aaise problem se bachna hai to yahi check karna hai
```

```
----------------------------------------------------------------------
-----
```

```
TypeError                                 Traceback (most recent call
last)
Cell In[8], line 1
----> 1 square('hehe')

Cell In[4], line 2, in square(num)
      1 def square(num):
----> 2     print(num**2)

TypeError: unsupported operand type(s) for ** or pow(): 'str' and
'int'
```

```
def sanity_check(data_type):
    def outer_wrapper(func):
        def inner_wrapper(*args):
            if type(*args) == data_type:
                func(*args)
            else:
                raise TypeError('ye data type nahi chalega')
        return inner_wrapper
    return outer_wrapper

@sanity_check(int)
def square(num):
    print(num**2)

square(2)
```

```
4
```

```
square('hii')
```

```
----------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
Cell In[14], line 1
----> 1 square('hii')

Cell In[12], line 7, in
sanity_check.<locals>.outer_wrapper.<locals>.inner_wrapper(*args)
      5     func(*args)
      6 else:
----> 7     raise TypeError('ye data type nahi chalega')

TypeError: ye data type nahi chalega
```

```
@sanity_check(str)
def greet(name):
    print('hello',name)
```

```
greet('jay')

hello jay
```