Iterators:

Iterators:

* Iter(iterable) gives an iterator on iterable
* Next(iterator) gives an element and moves next

L1=[5,6,10,11,15]

for i in L1:

print(i,end=" ")

# 5 6 10 11 15

Iterators:

L1=[5,6,10,11,15]

it=iter(L1)

print(it)

#<list\_iterator object at 0x000001F90C4C2358>

print(next(it))

#5

print(next(it))

#6

print(next(it))

#10

print(next(it))

#11

print(next(it))

#15

print(next(it))

#StopIteration

L1=[5,6,10,11,15]

for i in range(len(L1)+1):

    print(next(it),end=" ")

    # 5 6 10 11 15

    # StopIteration

#iter(iterable) ---> gives an iterator on iterable

# Tuples

T1=(5,6,7)

it=iter(T1)

for i in range(len(T1)):

    print(next(it),end=" ")

    #5 6 7

# SETS

S1={5,6,7}

it=iter(S1)

for i in range(len(S1)):

    print(next(S1),end=" ")

#   TypeError: 'set' object is not an iterator

# Dictonary

d1={1:"one",2:"two",3:"three"}

it=iter(d1)

for i in range(len(d1)):

    print(next(it),end=" ")

    # 1 2 3

#Strings

s1="NNR"

it=iter(s1)

for i in range(len(s1)):

    print(next(it),end=" ")

    #N N R

##### range

r=range(3,6)

it=iter(r)

for i in range(len(r)):

    print(next(it),end=" ")

    #3 4 5

#next(iterator) gives an element and moves next

GENERATORS:

#################### GENERATORS

r=range(4) #0 1 2 3

it=iter(r)

print(it)

# <range\_iterator object at 0x0000029199F5C610>

# range is example of generators

# iterator is builtin

# range is generators

n=4

def myrange(n):

    i=0

    while i<n:

        yield i

        i=i+1

m=myrange(n)

for i in range(n):

print(next(m),end=" ")

#     # 0 1 2 3

# print(next(m))

# #StopIteration

def mylist(d):

    i=0

    while True:

        yield d[i]

        i=(i+1)%7

d=["Sun","Mon","Tue","Wed","Thu","Fri","Sat"]

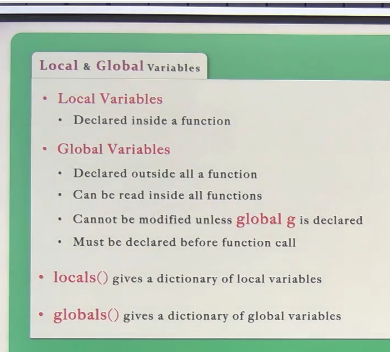
m=mylist(d)

for i in range(len(d)+5):

    print(next(m),end=" ")

# Sun Mon Tue Wed Thu Fri Sat Sun Mon Tue Wed Thu

Local and Global:



# Local and Global

def fun():

    a=10

    print(a)

fun()

#10

Prog 2:

g=5.25

print("Outside:::1",g)

def fun():

    a=10

    print(a)

fun()

# Outside:::1 5.25

# 10

Prog 3:

g=5.25

print("Outside:::1",g)

def fun():

    a=10

    print(f'local: {a}')

    print(f'global:::{g}')

fun()

# Outside:::1 5.25

# local: 10

# global:::5.25

Prog 4:

g=5.25

print("Outside:::1",g)

def fun():

    a=10

    print(f'local: {a}')

    print(f'global:::{g}')

fun()

print("Outside:::2",g)

# # Outside:::1 5.25

# # local: 10

# # global:::5.25

# # Outside:::2 5.25

Prog 5:

g=5.25

print("Outside:::1",g)

def fun():

    a=10

    g=199

    print(f'local: {a}')

    print(f'global:::{g}')

fun()

print("Outside:::2",g)

#   Outside:::1 5.25

#   local: 10

#   global:::199

#   Outside:::2 5.25

Prog 6:

g=5.25

print("Outside:::1",g)

def fun():

    a=10

    global g

    g=199

    print(f'local: {a}')

    print(f'global:::{g}')

fun()

print("Outside:::2",g)

# Outside:::1 5.25

# local: 10

# global:::199

# Outside:::2 199

Prog 7:

g=5.25

def fun():

    a=10

    print(f'local: {a}')

    print(f'global:::{g}')

g=5.25

fun()

g=5.25

#NameError: name 'g' is not defined

########################

# Declared outside all function

# Can be read inside all functins

# Cannot be modified unless global g is declared

# Must be declared before function call

##############################

Prog 8:

x,y,z=5,1.25,"hi"

def fun():

    a,b,c=1,2,3

    print(locals())

    # {'a': 1, 'b': 2, 'c': 3}

    print(globals())

    #{'\_\_name\_\_': '\_\_main\_\_', '\_\_doc\_\_': None, '\_\_package\_\_': None, '\_\_loader\_\_':

    #  <\_frozen\_importlib\_external.SourceFileLoader object at 0x000001AF1B945390>,

    #  '\_\_spec\_\_': None, '\_\_annotations\_\_': {}, '\_\_builtins\_\_': <module 'builtins' (built-in)>,

    # '\_\_file\_\_': 'local\_global.py', '\_\_cached\_\_': None,

    # 'x': 5, 'y': 1.25, 'z': 'hi',

    # 'fun': <function fun at 0x000001AF1B8FC2F0>}

fun()

#  locals() gives a dictonry of local variables

# globals() gives a dictionary of global variables

Recursive:

def fun(n):

    if(n>0):

        print(n,end=" ")

        fun(n-1)

        #3 2 1

fun(3)

A diagram of a diagram

AI-generated content may be incorrect.

Factorial

def fact(n):

    if(n<=0):

        return 1

    else:

        return n \* fact(n-1)

print(fact(5))

A computer screen shot of a green rectangular object

AI-generated content may be incorrect.

FIB:

def fib(n):

    if n==0:

        return 0

    elif n==1:

        return 1

    else:

        return fib(n-1) + fib(n-2)

for i in range(8):

    print(fib(i), end=" ")

# 0 1 1 2 3 5 8 13

Prog 2:

def fib(n):

    a, b = 0, 1

    for i in range(n+1):

        yield a

        a, b = b, a + b

if \_\_name\_\_ == "\_\_main\_\_":

    for num in fib(7):

        print(num, end=" ")

# 0 1 1 2 3 5 8 13

Flatten:

def flatten(lst):

    for item in lst:

        if hasattr(item,'\_\_iter\_\_'):

            # print(f'iterable {item}')

            yield from flatten(item)

        else:

            # print(f'{item} is not iterable')

            yield item

L=[1,2,[3,4,[5,6,7],8],9,[10,11]]

flat=flatten(L)

flat\_list=list(flat)

print(flat\_list)

#$ python flatten.py

# 1 is not iterable

# 2 is not iterable

# iterable [3, 4, [5, 6, 7], 8]

# 3 is not iterable

# 4 is not iterable

# iterable [5, 6, 7]

# 5 is not iterable

# 6 is not iterable

# 7 is not iterable

# 8 is not iterable

# 9 is not iterable

# iterable [10, 11]

# 10 is not iterable

# 11 is not iterable

# [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]

Prog 1:

import calendar as cal

# print(cal.MONDAY)

# # 0

# print(cal.SUNDAY)

# # 6

# print(cal.day\_name[0])

# # Monday

# print(cal.day\_name[6])

# # Sunday

# list(cal.day\_name)

# # ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']

# print(list(cal.day\_abbr))

# #['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']

# #

# print(list(cal.month\_name))

# # ['', 'January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'December']

# print(list(cal.month\_abbr))

# # ['', 'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']

Prog 2:

import calendar

def next\_month():

    count=1

    while True:

        name=calendar.month\_name[count]

        yield name

        count=count%12 +1

        print(count)

        #2 3 4 5 6 7

it=next\_month()

print(next(it))

 # January

print(next(it))

 # February

print(next(it))

 # March

print(next(it))

 # April

print(next(it))

 # May

print(next(it))

 # June

print(next(it))

 # July