17thseptember'23_forloop_lambda_map_reduce_filter_recursion

October 19, 2023

[3]:
```
'''type()
print()
len()
range()'''
```

[3]: 'type()\nprint()\nlen()\nrange()'

[7]:
```
def test():
    print("this is my first fun")
```

[8]:
```
test()
```

```
this is my first fun
```

[10]:
```
def test1():
    print("test")
```

[11]:
```
test1()+"sudh"
```

```
test
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[11], line 1
----> 1 test1()+"sudh"

TypeError: unsupported operand type(s) for +: 'NoneType' and 'str'
```

[12]:
```
type(test1())
```

```
test
```

[12]: NoneType

[13]:
```
def test2():
    print(1+2)
```

[14]:
```
test2()+10
```

```
3
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[14], line 1
----> 1 test2()+10

TypeError: unsupported operand type(s) for +: 'NoneType' and 'int'
```

[17]:
```
def test3():
    return 1+2
```

[18]:
```
test3()+10
```

[18]:
```
13
```

[19]:
```
type(test3())
```

[19]:
```
int
```

[22]:
```
def test4():
    return 2,"sudh",True,[1,2,3,4,5]
```

[23]:
```
test4()
```

[23]:
```
(2, 'sudh', True, [1, 2, 3, 4, 5])
```

[24]:
```
a,b,c,d=1,"sha",34,45
```

[25]:
```
a=1
b="sha"
c=34
d=45
```

[26]:
```
type(test4())
```

[26]:
```
tuple
```

[27]:
```
m,n,o,p=test4()
```

[28]:
```
m
```

[28]:
```
2
```

[29]:
```
n
```

[29]:
```
'sudh'
```

[30]:
```
o
```

[30]:
```
True
```

[31]:
```
p
```

[31]:
```
[1, 2, 3, 4, 5]
```

[32]:
```
def test5(x):
    return x
```

[33]:
```
test5(5)
```

[33]:
```
5
```

[34]:
```
test5("sham")
```

[34]:
```
'sham'
```

[35]:
```
test5([2,3,4,5,])
```

[35]:
```
[2, 3, 4, 5]
```

[37]:
```
test5(2,5)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[37], line 1
----> 1 test5(2,5)

TypeError: test5() takes 1 positional argument but 2 were given
```

[40]:
```
def test6(x,y):
    return x,y
```

[41]:
```
test6("shamnu",[1,2,3,4,5])
```

[41]:
```
('shamnu', [1, 2, 3, 4, 5])
```

[42]:
```
def test7(a:int,b:int):
    return a,b
```

[44]:
```
test7(4,5)
```

[44]:
```
(4, 5)
```

[45]:
```
test7("shamnu",[4,5,6,7])
```

[45]:
```
('shamnu', [4, 5, 6, 7])
```

[48]:
```
def test8(x:int,y:int):
    return x/y
```

[49]:
```
test8(4,5)
```

[49]:
```
0.8
```

[1]:
```
test8("sudh","kumar")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[1], line 1
----> 1 test8("sudh","kumar")

NameError: name 'test8' is not defined
```

[2]:
```
def test9(c,d):
    """this is my function which can take two input and return addition or␣
 ↪concatination
    """
    return c+d
```

[3]:
```
test9(4,5)
```

[3]:
```
9
```

[4]:
```
def test10(*args):
    return args
```

[5]:
```
test10()
```

[5]:
```
()
```

[6]:
```
test10(2,3,4,5,6,6)
```

[6]:
```
(2, 3, 4, 5, 6, 6)
```

[7]:
```
def test11(*sha):
    return sha
```

[8]:
```
test11(3,4,5,45,6)
```

[8]:
```
(3, 4, 5, 45, 6)
```

[9]:
```
def test12(**kwargs):
    return kwargs
```

[10]:
```
test12(a=5,b=3,c=5)
```

[10]:
```
{'a': 5, 'b': 3, 'c': 5}
```

[11]:
```
test12(n={'a': 5, 'b': 3, 'c': 5})
```

[11]:
```
{'n': {'a': 5, 'b': 3, 'c': 5}}
```

[12]:
```
def test13(b,a=10,c=20):
    return a,b,c
```

[13]:
```
test13(69)
```

[13]:
```
(10, 69, 20)
```

[14]:
```
test13(34,a=45)
```

[14]:
```
(45, 34, 20)
```

[15]:
```
v=test13(4,5,67)
```

[16]:
```
v
```

[16]:
```
(5, 4, 67)
```

[10]:
```
def test14(a,b):
    return a+b
```

[11]:
```
test14(5,6)
```

[11]:
```
11
```

[12]:
```
m=lambda a,b: a+b
```

[13]:
```
m
```

[13]:
```
<function __main__.<lambda>(a, b)>
```

[14]:
```
test14
```

[14]:
```
<function __main__.test14(a, b)>
```

[5]:
```
m(3,4)
```

[5]:
```
7
```

[15]:
```
def check_even(n):
    if n%2==0:
        return "given number is an even number"
    else:
        return "given number not an even number"
```

[16]:
```
check_even(3)
```

[16]:
```
'given number not an even number'
```

[17]:
```
3%2
```

[17]:
```
1
```

[18]:
```
b=lambda n:"given number is an even number" if n%2==0else "given number not an␣
↪even number"
```

[19]:
```
b(4)
```

[19]:
```
'given number is an even number'
```

[20]:
```
b(3)
```

[20]:
```
'given number not an even number'
```

[21]:
```
3%2
```

[21]:
```
1
```

[22]:
```
l=[3,4,5,6,7,5,4,21,4]
min_list=lambda li:min(li)
```

[23]:
```
min_list(l)
```

[23]:
```
3
```

[24]:
```
div=lambda a,b:a/b
```

[25]:
```
div(5,6)
```

[25]:
```
0.8333333333333334
```

[26]:
```
upper=lambda s:s.upper()
```

[30]:
```
upper("shamnu")
```

[30]:
```
'SHAMNU'
```

[33]:
```
l=[1,2,3,5,6,56,6,52,6,2]

def sum_odd(b):
    l1=[]
    for i in b:
        if i%2!=0:
            l1.append(i)
    return sum(l1)
```

[34]:
```
sum_odd(l)
```

[34]:
```
9
```

[35]:
```
[i for i in l if i%2!=0]
```

[35]:
```
[1, 3, 5]
```

[36]:
```
sum([i for i in l if i%2!=0])
```

[36]:
```
9
```

[37]:
```
sum_off_lambda= lambda b: sum([i for i in b if i%2!=0])
```

[38]:
```
sum_off_lambda(l)
```

[38]:
```
9
```

[39]:
```
sum_odd(l)
```

[39]:
```
9
```

[40]:
```
def fact_number(n):
    if n== 0:
        return 1
    else:
        return n*fact_number(n-1)
```

[41]:
```
fact_number(4)
```

[41]: 24

[42]:
```
fact_number(0)
```

[42]: 1

[2]:
```
def fact_num(n):
    a=1
    for i in range(1,n+1):
        a=i *a
    return a
```

[3]:
```
fact_num(4)
```

[3]: 24

[5]:
```
fact_num(0)
```

[5]: 1

[6]:
```
def fact_number(n):
    if n== 0:
        return 1
    else:
        return n*fact_number(n-1)
```

[7]:
```
fact_num_lambda=lambda n:1 if n==0 else n*fact_num_lambda(n-1)
```

[8]:
```
fact_num_lambda(5)
```

[8]: 120

[10]:
```
l=[5,6,7,8,9]

l1=[]
for i in l:
    l1.append(i+2)
```

[11]:
```
l1
```

[11]: [7, 8, 9, 10, 11]

[13]:
```
def add_list(l):
    l1=[]
    for i in l:
        l1.append(i+2)
    return l1
```

[15]:
```
add_list([3,4,5,6,7])
```

[15]:
```
[5, 6, 7, 8, 9]
```

[16]:
```
v = lambda a : a+2
```

[17]:
```
v(10)
```

[17]:
```
12
```

[18]:
```
l
```

[18]:
```
[5, 6, 7, 8, 9]
```

[21]:
```
map(lambda a : a+2,l)
```

[21]:
```
<map at 0x7f12006d2ef0>
```

[22]:
```
list(map(lambda a : a+2,l))
```

[22]:
```
[7, 8, 9, 10, 11]
```

[23]:
```
l
```

[23]:
```
[5, 6, 7, 8, 9]
```

[24]:
```
def test16(c):
    return c+2
```

[27]:
```
map(test16 , l)
```

[27]:
```
<map at 0x7f12030a5bd0>
```

[28]:
```
list(map(test16 , l))
```

[28]:
```
[7, 8, 9, 10, 11]
```

[30]:
```
test16(5)
```

[30]:
```
7
```

[31]:
```
test16(6)
```

[31]:
```
8
```

[32]:
```
test16(8)
```

[32]:
```
10
```

[33]:
```
test16(9)
```

[33]:
```
11
```

[34]:
```
l=["sha","mnida","pwskills"]
```

[35]:
```
len("sha")
```

[35]:
```
3
```

[36]:
```
len("mnida")
```

[36]:
```
5
```

[37]:
```
len("pwskills")
```

[37]:
```
8
```

[41]:
```
[3,5,8]
```

[41]:
```
[3, 5, 8]
```

[44]:
```
l1=[]
for i in l:
    l1.append(len(i))
```

[45]:
```
l1
```

[45]:
```
[3, 5, 8]
```

[53]:
```
def test17(l):
    l1 = []
    for i in l:
        l1.append(len(i))
        return l1
```

[54]:
```
test17(l)
```

[54]:
```
[3]
```

[55]:
```
list(map(lambda c: len(c),l))
```

[55]:
```
[3, 5, 8]
```

[56]:
```
def test18(c):
    return len(c)
```

[57]:
```
list(map(test18,l))
```

[57]: [3, 5, 8]

[58]:
```
list(map(len,l))
```

[58]: [3, 5, 8]

[59]:
```
l
```

[59]: ['sha', 'mnida', 'pwskills']

[61]:
```
l[::-1]
```

[61]: ['pwskills', 'mnida', 'sha']

[67]:
```
s="shaizali"
```

[68]:
```
s[::-1]
```

[68]: 'ilaziahs'

[64]:
```
list(map(lambda a: a[::-1],l))
```

[64]: ['ahs', 'adinm', 'sllikswp']

[72]:
```
l=[2,3,1,2,1,3]
from functools import reduce
```

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

[73]: 36

[74]:
```
l
```

[74]: [2, 3, 1, 2, 1, 3]

[77]:
```
min(l)
```

[77]: 1

[78]:
```
max(l)
```

[78]: 3

[82]:
```
reduce(lambda a,b : a if a>b else b, l)
```

[82]: 3

[83]:
```
#factorial of number
```

[84]:
```
n=4
fact=reduce(lambda a,b: a*b, range(1,n+1))
```

[85]:
```
def test (a,b):
    return a*b
```

[86]:
```
fact1=reduce(test,range(1,n+1))
```

[87]:
```
fact1
```

[87]: 24

[88]:
```
l
```

[88]: [2, 3, 1, 2, 1, 3]

[90]:
```
reduce(lambda a,b: a*b, [i for i in l if i%2==0])
```

[90]: 4

[91]:
```
[i for i in l if i%2==0]
```

[91]: [2, 2]

[93]:
```
l=[1,22,34,4,51,26,7,8,9]
```

[94]:
```
list(filter(lambda a:a%2==0,l))
```

[94]: [22, 34, 4, 26, 8]

[95]:
```
s="Pw Skills"
```

[96]:
```
list(filter(lambda a:a.islower(),s))
```

[96]: ['w', 'k', 'i', 'l', 'l', 's']

[97]:
```
l=["pw","pwskills","sha","krish"]
```

[98]:
```
list(filter(lambda a:len(a)<=4,l))
```

[98]: ['pw', 'sha']

[99]:
```
l
```

[99]:
```
['pw', 'pwskills', 'sha', 'krish']
```

[100]:
```
list(filter(lambda a: a[0]=='p',l))
```

[100]:
```
['pw', 'pwskills']
```

[101]:
```
def gen_fib(n):
    if n<=1:
        return n
    else:
        return gen_fib(n-1)+gen_fib(n-2)
```

[102]:
```
gen_fib(7)
```

[102]:
```
13
```

[103]:
```
100
```

[103]:
```
100
```

[104]:
```
def sum_till_n(n):
    if n==1:
        return 1
    else:
        return n+ sum_till_n(n-1)
```

[105]:
```
sum_till_n(5)
```

[105]:
```
15
```

[ ]:
```

```