10_9_23python_functions

September 10, 2023

[1]:
```
x="awesome"

def myfun():
    x="fantastic"
    #local scope
    print("python is "+x)
myfun()
print(x)
```

```
python is fantastic
awesome
```

[6]:
```
x=20

def add():
    y=30
    print("local variable y=",y)

    print("global variable x=",x)
    z=x+y
    print(Z)

def sub():
    m=10
    print("local variable m=",m)
    print("global variable x=",x)
    z=x-y
    print(z)
```

[7]:
```
add()
```

```
local variable y= 30
global variable x= 20
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[7], line 1
----> 1 add()
```

```
Cell In[6], line 9, in add()
      7 print("global variable x=",x)
      8 z=x+y
----> 9 print(Z)

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

[8]:
```
#python function arguements
```

[10]:
```
def add_numbers(a,b):
    sum=a+b
    print("sum",sum)
```

[11]:
```
add_numbers(2,5)
```

```
sum 7
```

[16]:
```
def emp_name(name):
    print("myname is",name)
```

[17]:
```
emp_name('shamn')
```

```
myname is shamn
```

[18]:
```
def add_numbers(a=4,b=6):
    sum=a+b
    print("sum",sum)
```

[19]:
```
add_numbers()
```

```
sum 10
```

[22]:
```
def emp_name(name='shamn'):
    print("myname is",name)
```

[23]:
```
emp_name()
```

```
myname is shamn
```

[24]:
```
#keyword arguemnets
```

[26]:
```
def display_info(first_name,last_name):
    print("first name", first_name)
    print("last name", last_name)
```

[30]:
```
#display_info(last_name='skills',first_name='pw',)
```

[28]:
```
#default arguments
```

[29]:
```
def add_numbers(a,b):
    sum=a+b
    print("sum",sum)
```

[31]:
```
add_numbers(a=4,b=2)
```

```
sum 6
```

[32]:
```
#args and kargs
```

[37]:
```
def find_sum(*numbers):
    result=0

    for num in numbers:
        result=result+num
    print("sum",result)
```

[41]:
```
find_sum()
```

[39]:
```
find_sum(1,2,3)
```

```
sum 1
sum 3
sum 6
```

[42]:
```
def simple(*x):
    print(x)
```

[43]:
```
simple()
```

```
()
```

[44]:
```
simple(1,2,3)
```

```
(1, 2, 3)
```

[45]:
```
simple(1,2,3,4,5,6)
```

```
(1, 2, 3, 4, 5, 6)
```

[52]:
```
def intro(**data):
    print("data type of arguemet", type(data))

    for key,value in data.items():
            print(key,value)
```

[53]:
```
intro(firstname='pw',lastname='skills')
```

```
data type of arguemet <class 'dict'>
firstname pw
lastname skills
```

[48]:
```
x={'a':1,'b':2}
```

[49]:
```
x.items()
```

[49]:
```
dict_items([('a', 1), ('b', 2)])
```

[54]:
```
for keys,value in x.items():
    print(keys,value)
```

```
a 1
b 2
```

[56]:
```
def simple(**x):
    print(x)
```

[58]:
```
simple(name='shamn',age=23,id_no=23)
```

```
{'name': 'shamn', 'age': 23, 'id_no': 23}
```

[59]:
```
def shownumbers(*x):
    print(x)
```

[60]:
```
shownumbers(1)
```

```
(1,)
```

[61]:
```
shownumbers(2,3,4,5,6)
```

```
(2, 3, 4, 5, 6)
```

[63]:
```
def sk(**y):
    print(y)
```

[64]:
```
sk(x=1,y=2)
```

```
{'x': 1, 'y': 2}
```

[65]:
```
#lamda function
```

[66]:
```
#lambda arguments(s):expression
```

[67]:
```
x=lambda:print("hello world")
```

[68]:
```
y=lambda:print("pwskills")
```

[69]:
```
y()
```

```
pwskills
```

[70]:
```
addnum=lambda a,b:a*b
```

[72]:
```
multi=lambda a,b:a*b
```

[73]:
```
multi(2,4)
```

[73]:
```
8
```

[75]:
```
a=int(input("enter first number"))
b=int(input("enter second number"))
square=lambda a,b:(a**2,b**2)
print(square(a,b))
```

```
enter first number 2
enter second number 3
(4, 9)
```

[78]:
```
def evenodd(n):
    if n%2==0:
        print("number is even")
    else:
        print("number is odd")
```

[80]:
```
evenodd(56)
```

```
number is even
```

[81]:
```
x=lambda n:"number is even"if n%2==0 else "number is odd"
```

[82]:
```
x(3)
```

[82]:
```
'number is odd'
```

[83]:
```
#write a lambda function that accepts 2 arguements and return the greater␣
 ↪amongest them
```

[84]:
```
g_num=lambda a,b:aif a>b else b
```

```
  Cell In[84], line 1
    g_num=lambda a,b:aif a>b else b
                       ^

SyntaxError: invalid syntax
```

[85]:
```
g_num
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[85], line 1
----> 1 g_num

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

[86]:
```
#lambda fn of fing sum of numbers
```

[87]:
```
sum_of=lambda a,b,c:(a+b+c)
```

[88]:
```
sum_of(1,2,3)
```

[88]:
```
6
```

[91]:
```
products=[{'name':'product1','price':20}, {'name':'product2','price':
 ↪30},{'name':'product3','price':40}]
#sorted()
sorted_products=sorted(products,key=lambda x:x['price'])
```

[93]:
```
for i in sorted_products:
    print(i)
```

```
{'name': 'product1', 'price': 20}
{'name': 'product2', 'price': 30}
{'name': 'product3', 'price': 40}
```

[95]:
```
key=lambda x:x['price']
```

[96]:
```
key
```

[96]:
```
<function __main__.<lambda>(x)>
```

[97]:
```
x['price']
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
```

```
Cell In[97], line 1
----> 1 x['price']

TypeError: 'function' object is not subscriptable
```

[98]:
```
#write a lambda expression that accepts acharacter as argument and return true␣
↪if i is a vowel other false
#a,e e,i,o,u
#imran
#i a
```

[99]:
```
#filter
#map
#reduce
```

[100]:
```
vowels=['a','e','i','o','u'
```

```
  Cell In[100], line 1
    vowels=['a','e','i','o','u'
                               ^
SyntaxError: incomplete input
```

[101]:
```
y=str(x)
```

[102]:
```
y
```

[102]:
```
'<function <lambda> at 0x7fef542ccaf0>'
```

[103]:
```
type(y)
```

[103]:
```
str
```

[104]:
```
language=['python','java',',javascript']
```

[105]:
```
enumerate_prime=enumerate(language,20)
```

[106]:
```
list(enumerate_prime)
```

[106]:
```
[(20, 'python'), (21, 'java'), (22, ',javascript')]
```

[107]:
```
# list comprehension
```

[108]:
```
#[output|collection for x in range()|condition]
```

[112]:
```
lst=[1,2,3,4,5,6,7,8,9,10]
```

```
[113]: a=[x for x in lst]
```

```
[111]: a
```

```
[111]: 2
```

```
[114]: for i in lst:
           print(i)
```

```
1
2
3
4
5
6
7
8
9
10
```

```
[115]: lst=[1,2,3,4,5,6,7,8,9,10]
       a=[x+1 for x in lst]
```

```
[116]: a
```

```
[116]: [2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
```

```
[118]: lst=[1,2,3,4,5,6,7,8,9,10]
       c=[x for x in lst if x>4]
```

```
[119]: c
```

```
[119]: [5, 6, 7, 8, 9, 10]
```

```
[120]: a=[]
       for x in lst:
           if x>4:
               a.append(x)
```

```
[121]: a
```

```
[121]: [5, 6, 7, 8, 9, 10]
```

```
[123]: l=[1,2,3,4,5,6,7,8,9,10]
       result=[i for i in l if i%2!=0]
       result
```

```
[123]: [1, 3, 5, 7, 9]
```

```
[124]: l=[1,2,3,4,5,6,7,8,9,10]
       result=[i for i in l if i%2==0]
       result
```

```
[124]: [2, 4, 6, 8, 10]
```

```
[126]: lst=[1,2,3,4,5,6,7,8,9,10]
       d=[x for x in lst if x>4 if x%2==0]
```

```
[127]: d
```

```
[127]: [6, 8, 10]
```

```
[129]: l=[1,2,3,4,5,6,7,8,9,10]
       e=[x if x>4 else 'lessthan 4' for x in l]
```

```
[130]: e
```

```
[130]: ['lessthan 4', 'lessthan 4', 'lessthan 4', 'lessthan 4', 5, 6, 7, 8, 9, 10]
```

```
[131]: a=[]

       for x in lst:
           if x>4:
               a.append(x)
           else:
               a.append("less than 4")
```

```
[134]: lst=[1,2,3,4,5,6,7,8,9,10]

       f=['two' if x%2==0 else 'three' if x%3==0 else 'not a &3' for x in lst]
```

```
[135]: f
```

```
[135]: ['not a &3',
        'two',
        'three',
        'two',
        'not a &3',
        'two',
        'not a &3',
        'two',
        'three',
        'two']
```

[137]:
```python
a=[]

for x in lst:
    if x%2==0:
        a.append('two')
    elif x%3==0:
        a.append('three')
    else:
        a.append('not a&3')
```

[138]:
```python
f
```

[138]:
```
['not a &3',
 'two',
 'three',
 'two',
 'not a &3',
 'two',
 'not a &3',
 'two',
 'three',
 'two']
```

[ ]:
```

```