rishav sharma

class 6th (functions)

built in functions- which are defined and pre coded in python. min(),max(), len(), sum(), type()

user defined- def function_name(parameters):

```
        code and statements
rules-
create a fxn using the def keyword
```

In [14]:
```
def my_func (param1='default'):
    """docstring goes here
    """
    print(param1)
```

In [15]:
```
my_func()
```

```
default
```

In [16]:
```
my_func("new param")
```

```
new param
```

In [17]:
```
my_func(param1='vishal')
```

```
vishal
```

In [22]:
```
def square(x):
    print (x**2)
square (3)
```

```
9
```

In [23]:
```
def square(x):
    return (x**2)
square (3)
```

Out[23]:
```
9
```

In [24]:
```
def square(x):
    return x**2
square (3)
```

Out[24]: 9

In [26]:
```
def name(fname, lname):
    print("hello", fname, lname)
name("abhinav", "bindra")
```

```
hello abhinav bindra
```

In [27]:
```
#default  arguments
def pizza(size,name= 'mushroom pizza'):
    print(' i like',size , name)
pizza("regular")
```

```
 i like regular mushroom pizza
```

In [28]:
```
 def pizza(size,name= 'mushroom pizza'):
    print(' i like',size , name)
pizza("regular", "paneer pizza")
```

```
 i like regular paneer pizza
```

In [31]:
```
# required argument
def pizza(size, name, loc):
    print("i like", size, name ,"pizza of", loc)
pizza("regular","mushroom","dominos")
```

```
i like regular mushroom pizza of dominos
```

In [33]:
```
#variable length argument:
def pizza (*t): #work as tuple
    print("i like", t[0], t[1],"pizza of ", t[2] )
pizza("regular","mushroom", "dominos")
```

```
i like regular mushroom pizza of  dominos
```

In [36]:
```
 #  ** will be defined as dictionary,and *  will work as tuple
def pizza(**t):
    print("i like", t["size"], t["name"], "pizza of", t["loc"])
pizza(size= "regular", name="mushroom", loc="dominos")
```

```
i like regular mushroom pizza of dominos
```

In [37]:
```
# return statement
def pizza(size, name, loc):
    return "i like" +" "+ size+" "+name+"pizza of "+ loc
pizza("regular", "mushroom", "dominos")
```

Out[37]: 'i like regular mushroompizza of dominos'

In [50]:
```
#recursion function
def factorial(num):
    if(num==1 or num ==0):
        return 1
```

```
    else:
        return(num*factorial(num-1))
num=7
print("factorial:",factorial(num))
```

```
factorial: 5040
```

In [52]:
```
# Q1 find the largest item from a given list=[10, 40, 50, 264, 389, 1, 4, 10, 29]
```

In [53]:
```
max([10, 40, 50, 264, 389, 1, 4, 10, 29])
```

Out[53]:
```
389
```

In [54]:
```
min([10, 40, 50, 264, 389, 1, 4, 10, 29])
```

Out[54]:
```
1
```

In [56]:
```
#create a  function name eligibility to check its  marks in maths and science is great
#is more than 70 is elegible for test. note: marks of maths and science should be ente
m=int(input("enter marks in maths:"))
s=int(input("enter marks in science:"))
a=int(input("enter your attendance":))

def eligibility (m,s,a):
    if m>70 and s>70 :
        print("you are eligible")
else:
    print("you are not eligible")
eligiblity(m,s,a)
```

```
  Cell In [56], line 5
    a=int(input("enter your attendance":))
                                        ^
SyntaxError: invalid syntax
```

In [57]:
```
def times(var):
    return var *2
times(5)
```

Out[57]:
```
10
```

In [58]:
```
lambda var: var*2
```

Out[58]:
```
<function __main__.<lambda>(var)>
```

In [60]:
```
lambda var: var*2
times(5)
```

Out[60]:
```
10
```

In [62]:
```
#python lambda function using list comprehension
x=lambda a,b: a if (a>b) else b
print(x(10,15))
```

```
15
```

In [65]:
```
#map and filter functions
def times(var):
    return var*2
seq = [1,2,3,4,5]

map(times,seq)
list(map(times,seq))
```

Out[65]:
```
[2, 4, 6, 8, 10]
```

In [67]:
```
#2q find out all the
li=[5,7,22,97, 54, 62,77,23, 73, 61]
list(filter(lambda x:x%2!=0,li))
```

Out[67]:
```
[5, 7, 97, 77, 23, 73, 61]
```

In [68]:
```
# filter all peple having age more than 18, using lambdaa and  filter functions
a= [13, 90,17, 59, 21, 60,5]
list(filter(lambda x:x>18,a))
```

Out[68]:
```
[90, 59, 21, 60]
```

In [70]:
```
#python modules
#which contain python tools
import math
print("sin(0)=",math.sin(0))
print("cos(30)=",math.cos(math.pi/6))
print("tan(45)=",math.tan(0))
```

```
sin(0)= 0.0
cos(30)= 0.8660254037844387
tan(45)= 0.0
```

In [74]:
```
#creating and using import operations
import operations
num1=int(input("first number:"))
num2=int(input("second number:"))

print("add", operations.add(num1, num2))
print("sub", operationns.sub(num1,num2))
print("square", operations.square(num1))
print("cube", operations.cube(num2))
```

```
Traceback (most recent call last):

  File /opt/conda/lib/python3.10/site-packages/IPython/core/interactiveshell.py:3378
in run_code
    exec(code_obj, self.user_global_ns, self.user_ns)

  Cell In [74], line 2
    import operations

  File ~/work/operations.py:2
    a+b
    ^
IndentationError: expected an indented block after function definition on line 1
```

In [75]:
```
# using an alias
    import operations as op
num1=int(input("first number:"))
num2=int(input("second number:"))

print("add", op.add(num1, num2))
print("sub", op.sub(num1,num2))
print("square",op.square(num1))
print("cube", op.cube(num2))



#using an asterisk means importing all the functions. from file.
#dir() function names (or varibales names) in module.
```

```
  Cell In [75], line 2
    import operations as op
    ^
IndentationError: unexpected indent
```

In [ ]: