Numpy – Part 2

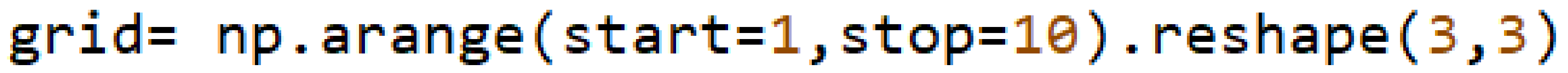
GITAA
Transforming careers

In this lecture

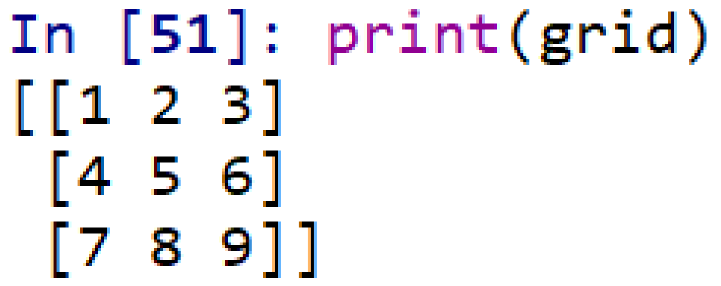
- Reshape an array
- Numpy operations
- Access elements from an array

Reshaping an array

- **reshape()**- recasts an array to new shape without changing it's data



```
grid= np.arange(start=1,stop=10).reshape(3,3)
```



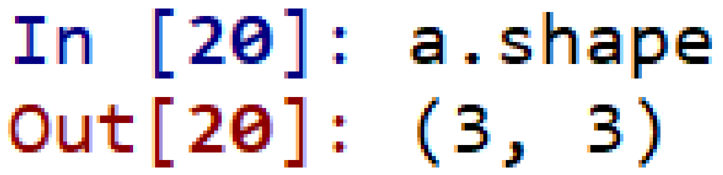
```
In [51]: print(grid)
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

Array dimensions

- Create an array ***a***

```
a=np.array([[1,2,3],[4,5,6],[7,8,9]])
```

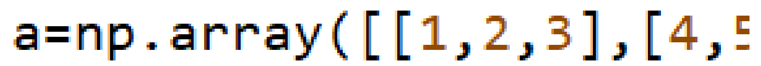
- `shape()`- returns dimensions of an array
- Syntax: `array_name.shape`



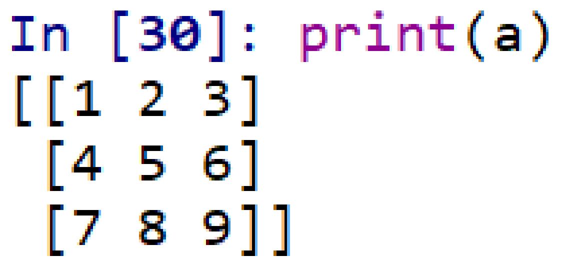
```
In [20]: a.shape
Out[20]: (3, 3)
```

Numpy addition

- numpy.sum()- returns sum of all array elements or sum of all array elements over a given axis
- Syntax: `numpy.sum(array,axis)`
- In the above syntax,
  - array() - input array
  - axis() - axis along which sum should be calculated
- Create an array ***a***



```
a=np.array([[1,2,3],[4,5
```



```
In [30]: print(a)
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

Numpy addition

- Calculate overall sum (axis=None)

```
In [27]: np.sum(a)
Out[27]: 45
```

```
In [28]: a.sum()
Out[28]: 45
```

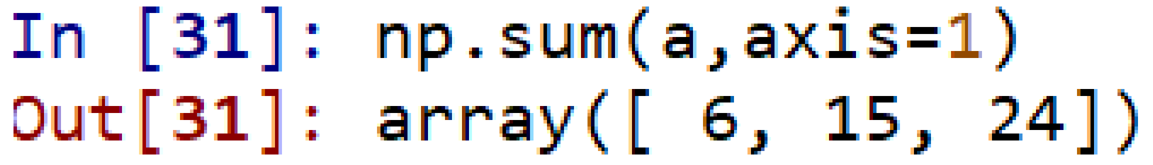
- Calculate sum along the column (axis=0)

```
In [29]: np.sum(a,axis=0)
Out[29]: array([12, 15, 18])
```

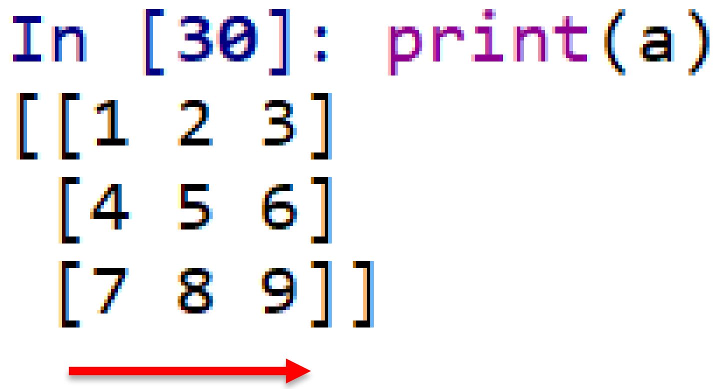
```
In [30]: print(a)
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

Numpy addition

- Calculate sum along the row (axis=1)



```
In [31]: np.sum(a,axis=1)
Out[31]: array([ 6, 15, 24])
```



```
In [30]: print(a)
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

Numpy addition

- `numpy.add()`- performs elementwise addition between two arrays
- Syntax: `numpy.add(array_1,array_2)`
- Create two arrays ***a*** and ***b***

```
a=np.array([[1,2,3],[4,5,6],[7,8,9]])

b= np.arange(start=11,stop=20).reshape(3,3)
```

Numpy addition

- Print ***a*** and ***b***

```
In [16]: print(a)
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

```
In [17]: print(b)
[[11 12 13]
 [14 15 16]
 [17 18 19]]
```

```
np.add(a,b)
```

$\rightarrow$

```
Out[23]:
array([[12, 14, 16],
       [18, 20, 22],
       [24, 26, 28]])
```

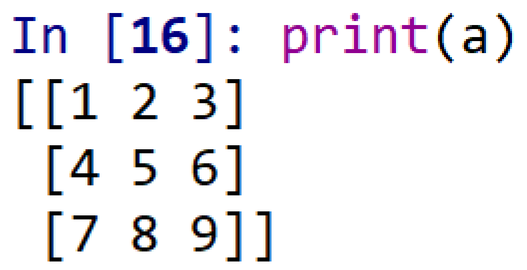
Numpy multiplication

- `numpy.multiply()`- performs elementwise multiplication between two arrays
- Syntax: `numpy.multiply(array_1,array_2)`
- Consider the same arrays ***a*** and ***b***

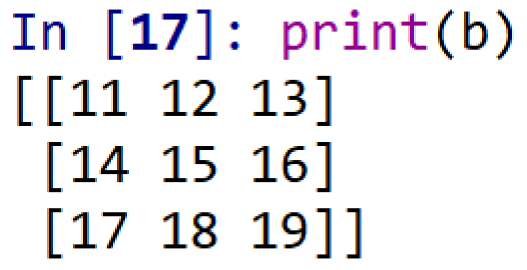
```
a=np.array([[1,2,3],[4,5,6],[7,8,9]])

b= np.arange(start=11,stop=20).reshape(3,3)
```

Numpy multiplication

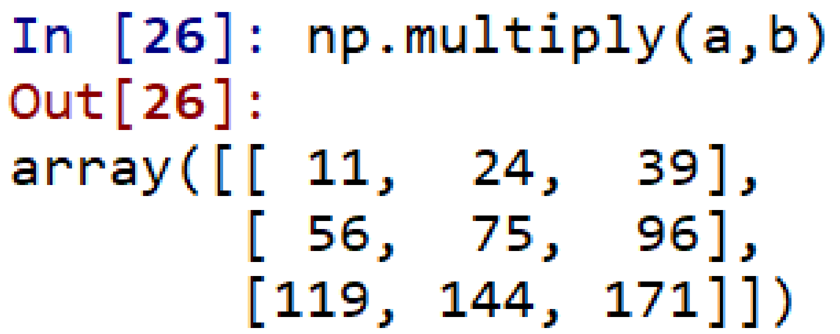
- Print ***a*** and ***b***



```
In [16]: print(a)
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```



```
In [17]: print(b)
[[11 12 13]
 [14 15 16]
 [17 18 19]]
```



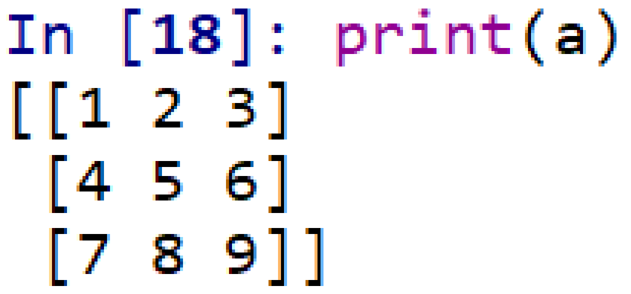
```
In [26]: np.multiply(a,b)
Out[26]:
array([[ 11,  24,  39],
       [ 56,  75,  96],
       [119, 144, 171]])
```

Other numpy functions

Function name	Description
numpy.subtract	performs element wise subtraction between two arrays
numpy.divide	returns an element wise division of inputs
numpy.remainder	Return element-wise remainder of division

Accessing components of an array

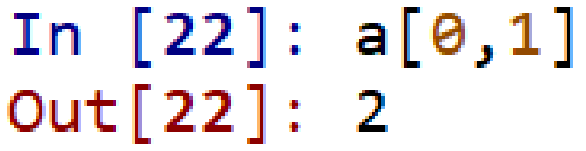
- Components of an array can be accessed using index number



```
In [18]: print(a)
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

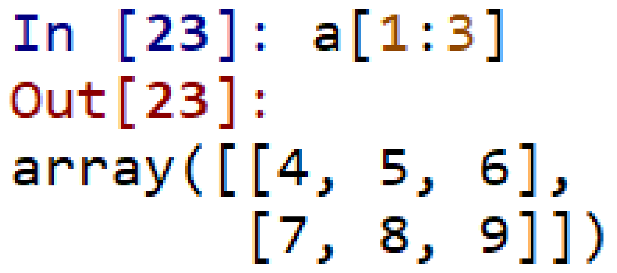
- Extract element with index (0,1) from ***a***



```
In [22]: a[0,1]
Out[22]: 2
```

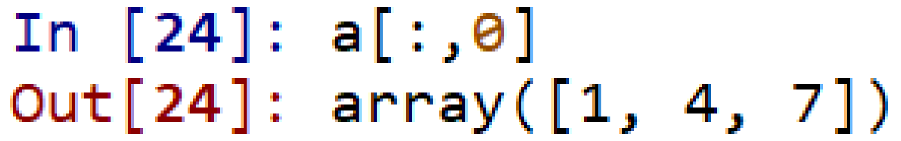
Accessing components of an array

- Extract elements from second and third row of array ***a***



```
In [23]: a[1:3]
Out[23]:
array([[4, 5, 6],
       [7, 8, 9]])
```

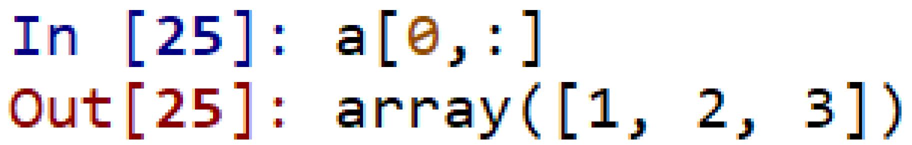
- Extract elements from first column of array ***a***



```
In [24]: a[:,0]
Out[24]: array([1, 4, 7])
```

Accessing components of an array

- Extract elements the first row of array ***a***



```
In [25]: a[0,:]
Out[25]: array([1, 2, 3])
```

Subset of arrays

- Array ***a***

```
In [18]: print(a)
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

- Subset a 2x2 array from the original array ***a***
- Consider the first two rows and columns from ***a***

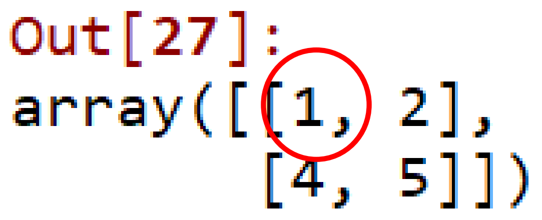
```
In [26]: a_sub=a[:2,:2]
```

- Print subset array ***a_sub***

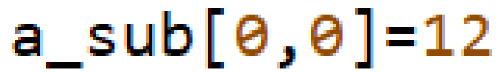
```
Out[27]:
array([[1, 2],
       [4, 5]])
```

Subset of arrays

- Here the value 1 should be updated to 12

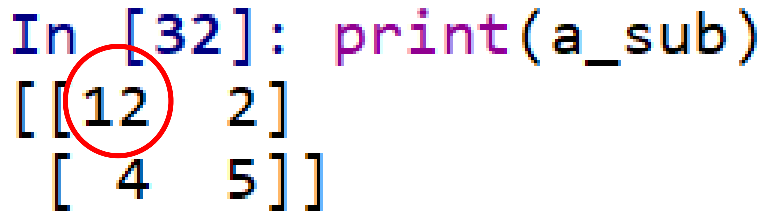


```
Out[27]:
array([[1, 2],
       [4, 5]])
```



```
a_sub[0,0]=12
```

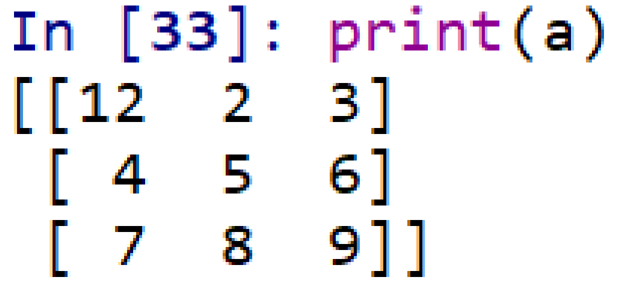
- Print the updated sub array



```
In [32]: print(a_sub)
[[12  2]
 [ 4  5]]
```

Subset of arrays

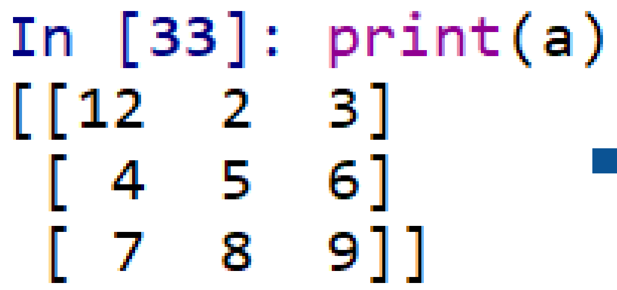
- Modifying the subset will automatically update the original array as well



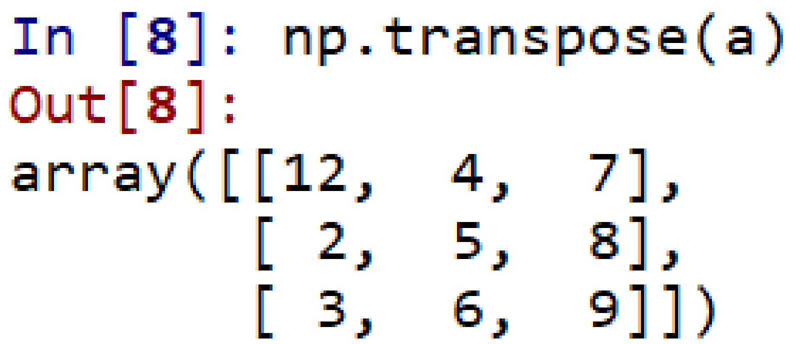
```
In [33]: print(a)
[[12  2  3]
 [ 4  5  6]
 [ 7  8  9]]
```

Modifying array using transpose ( )

- `numpy.transpose()`- permute the dimensions of array
- Syntax: `numpy.transpose(array)`



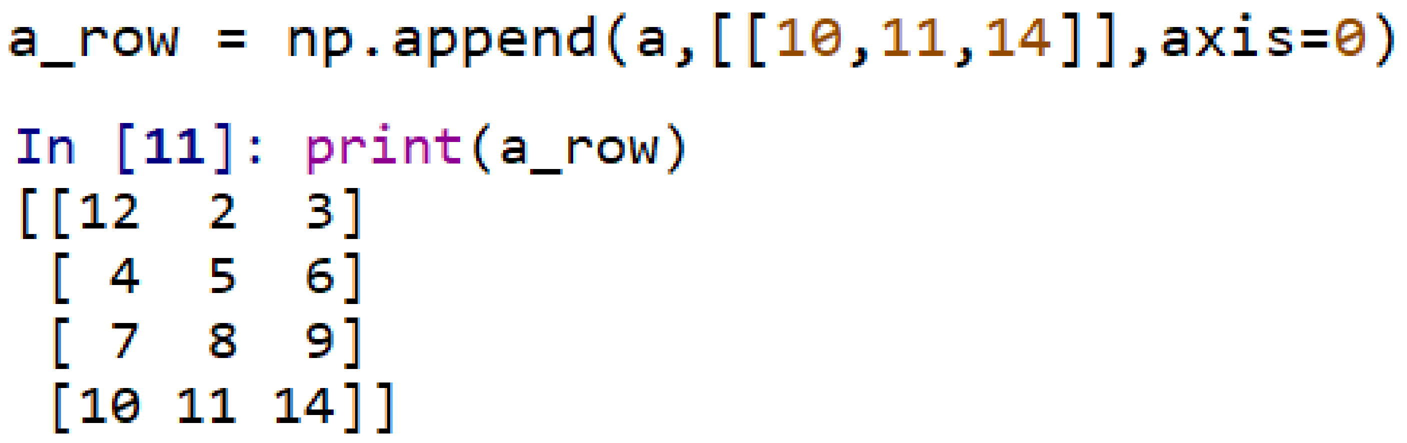
```
In [33]: print(a)
[[12  2  3]
 [ 4  5  6]
 [ 7  8  9]]
```



```
In [8]: np.transpose(a)
Out[8]:
array([[12,  4,  7],
       [ 2,  5,  8],
       [ 3,  6,  9]])
```

Modifying array using append( )

- **append()**- adds values at the end of the array
- Syntax: `numpy.append(array,axis)`
- Adding the new array to ***a*** as a row

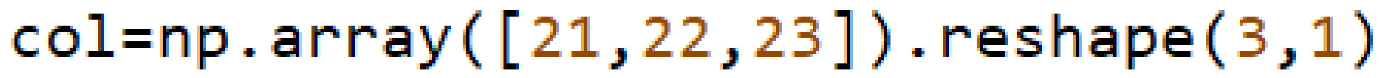


```
a_row = np.append(a,[[10,11,14]],axis=0)
```

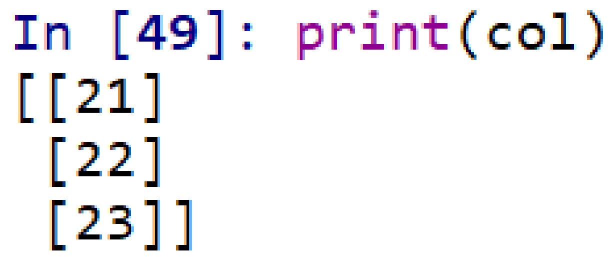
```
In [11]: print(a_row)
[[12  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 14]]
```

Modifying array using append( )

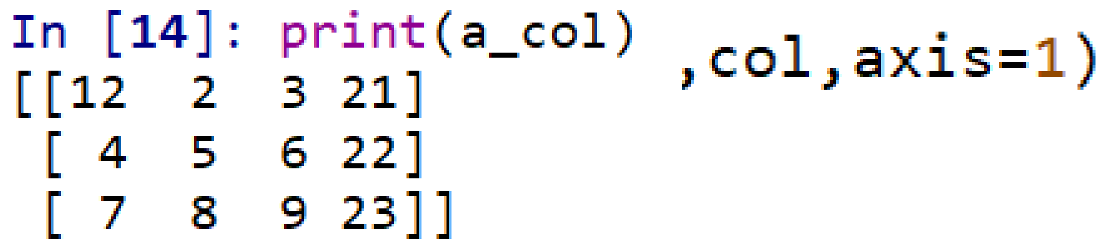
- Adding the new array to ***a*** as a column
- Create an array and reshape to column array



```
col=np.array([21,22,23]).reshape(3,1)
```



```
In [49]: print(col)
[[21]
 [22]
 [23]]
```



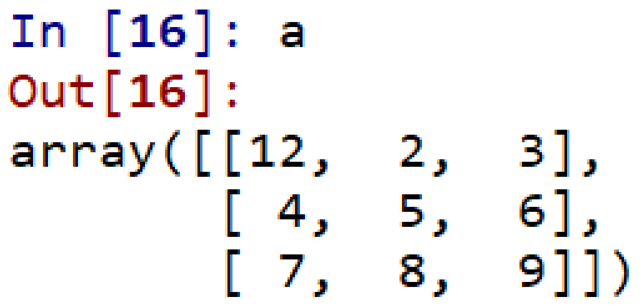
```
In [14]: print(a_col)     ,col,axis=1)
[[12  2  3 21]
 [ 4  5  6 22]
 [ 7  8  9 23]]
```

Modifying array using insert( )

- `insert()`- adds values at a given position and axis in an array
- Syntax: `numpy.insert(array,obj,values,axis)`
  - `array` - input array
  - `obj` - index position
  - `values` - array of values to be inserted
  - `axis` - axis along which values should be insert

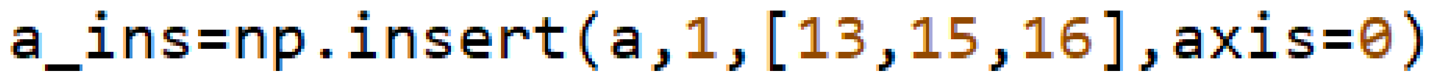
Modifying array using insert( )

- Consider array ***a***

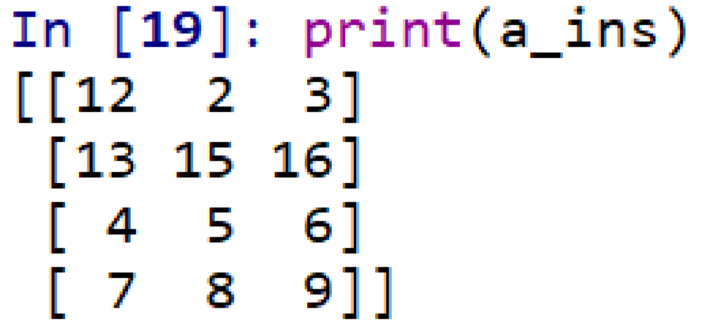


```
In [16]: a
Out[16]:
array([[12,  2,  3],
       [ 4,  5,  6],
       [ 7,  8,  9]])
```

- Insert new array along row and at the 1st index position



```
a_ins=np.insert(a,1,[13,15,16],axis=0)
```



```
In [19]: print(a_ins)
[[12  2  3]
 [13 15 16]
 [ 4  5  6]
 [ 7  8  9]]
```

Modifying array using delete( )

- `delete()`- removes values at a given position and axis in an array
- Syntax: `numpy.delete(array,obj,axis)`
  - `array` - input array
  - `obj` - indicate array to be removed or it’s position
  - `axis` - axis along which array should be removed

Modifying array using delete( )

- Delete third row from the existing array ***a_ins***

```
a_del=np.delete(a_ins,2,axis=0)
```

```
In [19]: print(a_ins)
[[12  2  3]
 [13 15 16]
 [ 4  5  6]
 [ 7  8  9]]
```

$\longrightarrow$

```
In [21]: print(a_del)
[[12  2  3]
 [13 15 16]
 [ 7  8  9]]
```

Summary

- Reshape an array
- Numpy operations
- Accessing components
- Subset of arrays
- Modifying array

```
operation == "MIRROR_X":
mirror_mod.use_x = True
mirror_mod.use_y = False
mirror_mod.use_z = False
 _operation == "MIRROR_Y":
mirror_mod.use_x = False
mirror_mod.use_y = True
mirror_mod.use_z = False
 _operation == "MIRROR_Z":
mirror_mod.use_x = False
mirror_mod.use_y = False
mirror_mod.use_z = True

#selection at the end -add
r_ob.select= 1
ier_ob.select=1
ntext.scene.objects.activ
("Selected" + str(modifier
mirror_ob.select = 0
 = bpy.context.selected_ob
data.objects[one.name].se

print("please select exactl
```

THANK YOU