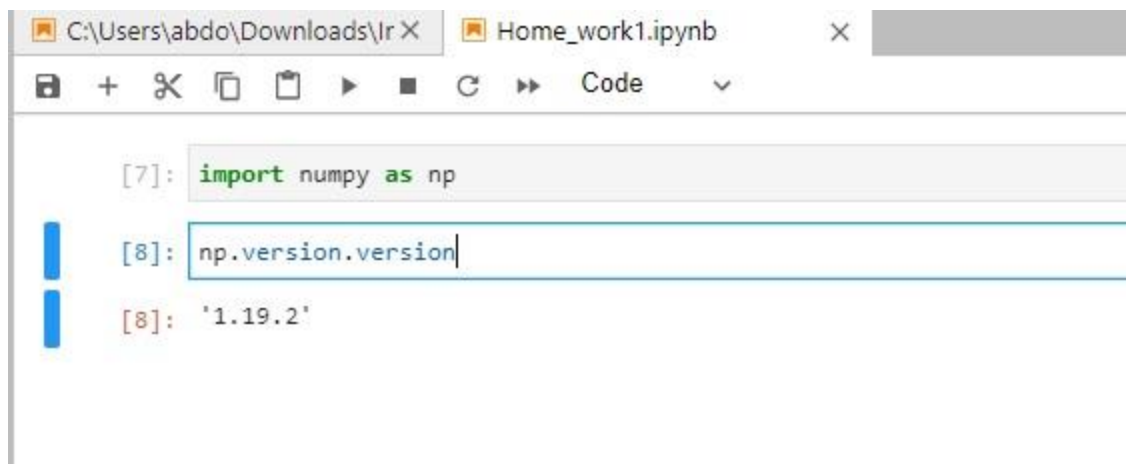


Home work 1 (Codes)

1- Write a NumPy code line(s) to get and print your numpy library version

Sol :

```
import numpy as np  
np.version.version
```



The screenshot shows a Jupyter Notebook window with the title 'Home_work1.ipynb'. The notebook contains two code cells. The first cell, labeled '[7]:', contains the code 'import numpy as np'. The second cell, labeled '[8]:', contains the code 'np.version.version'. The output of the second cell is displayed below it as '[8]: '1.19.2''. The notebook interface includes a toolbar with icons for saving, adding, deleting, and running code, as well as a 'Code' dropdown menu.

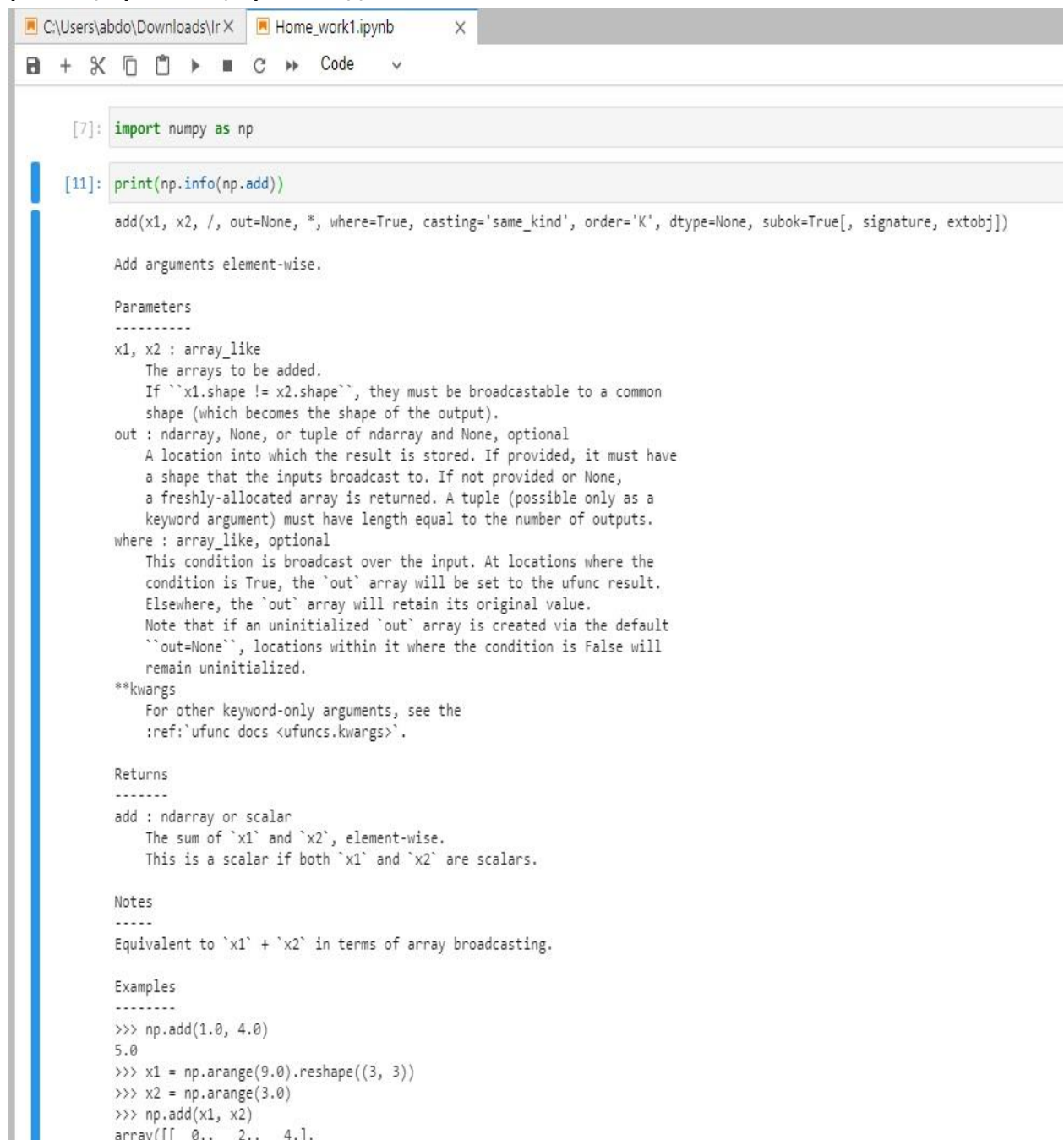
```
[7]: import numpy as np  
[8]: np.version.version  
[8]: '1.19.2'
```

2- Write a NumPy code line(s) to get help on the “add” function.

Sol:

```
import numpy as np
```

```
print(np.info(np.add))
```



```
[7]: import numpy as np

[11]: print(np.info(np.add))

add(x1, x2, /, out=None, *, where=True, casting='same_kind', order='K', dtype=None, subok=True[, signature, extobj])

Add arguments element-wise.

Parameters
-----
x1, x2 : array_like
    The arrays to be added.
    If ``x1.shape != x2.shape``, they must be broadcastable to a common
    shape (which becomes the shape of the output).
out : ndarray, None, or tuple of ndarray and None, optional
    A location into which the result is stored. If provided, it must have
    a shape that the inputs broadcast to. If not provided or None,
    a freshly-allocated array is returned. A tuple (possible only as a
    keyword argument) must have length equal to the number of outputs.
where : array_like, optional
    This condition is broadcast over the input. At locations where the
    condition is True, the `out` array will be set to the ufunc result.
    Elsewhere, the `out` array will retain its original value.
    Note that if an uninitialized `out` array is created via the default
    ``out=None``, locations within it where the condition is False will
    remain uninitialized.
**kwargs
    For other keyword-only arguments, see the
    :ref:`ufunc docs <ufuncs.kwargs>`.

Returns
-----
add : ndarray or scalar
    The sum of `x1` and `x2`, element-wise.
    This is a scalar if both `x1` and `x2` are scalars.

Notes
-----
Equivalent to `x1` + `x2` in terms of array broadcasting.

Examples
-----
>>> np.add(1.0, 4.0)
5.0
>>> x1 = np.arange(9.0).reshape((3, 3))
>>> x2 = np.arange(3.0)
>>> np.add(x1, x2)
array([[ 0.,  2.,  4.],
       [ 1.,  3.,  5.],
       [ 2.,  4.,  6.]])
```

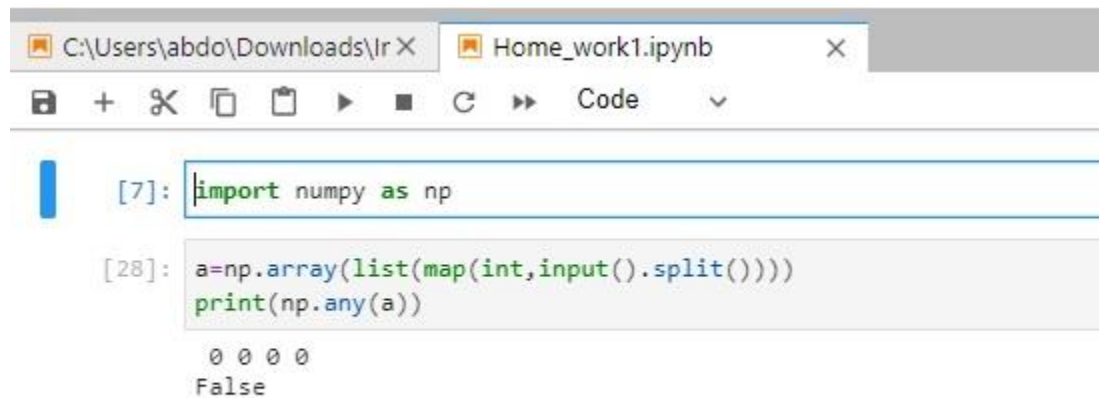
3- Write a NumPy code line(s) to test whether any of the elements of an input array is non-zero

Sol :

```
import numpy as np
```

```
a=np.array(list(map(int,input().split())))
```

```
print(np.any(a))
```



A screenshot of a Jupyter Notebook window titled 'Home_work1.ipynb'. The notebook has two tabs: 'C:\Users\abdo\Downloads\lr X' and 'Home_work1.ipynb'. The code cell [28] contains the following Python code:

```
import numpy as np
a=np.array(list(map(int,input().split()))
print(np.any(a))
```

 The output of the code is displayed below the cell:

```
0 0 0 0
False
```



A screenshot of a Jupyter Notebook window titled 'Home_work1.ipynb'. The notebook has two tabs: 'C:\Users\abdo\Downloads\lr X' and 'Home_work1.ipynb'. The code cell [29] contains the following Python code:

```
import numpy as np
a=np.array(list(map(int,input().split()))
print(np.any(a))
```

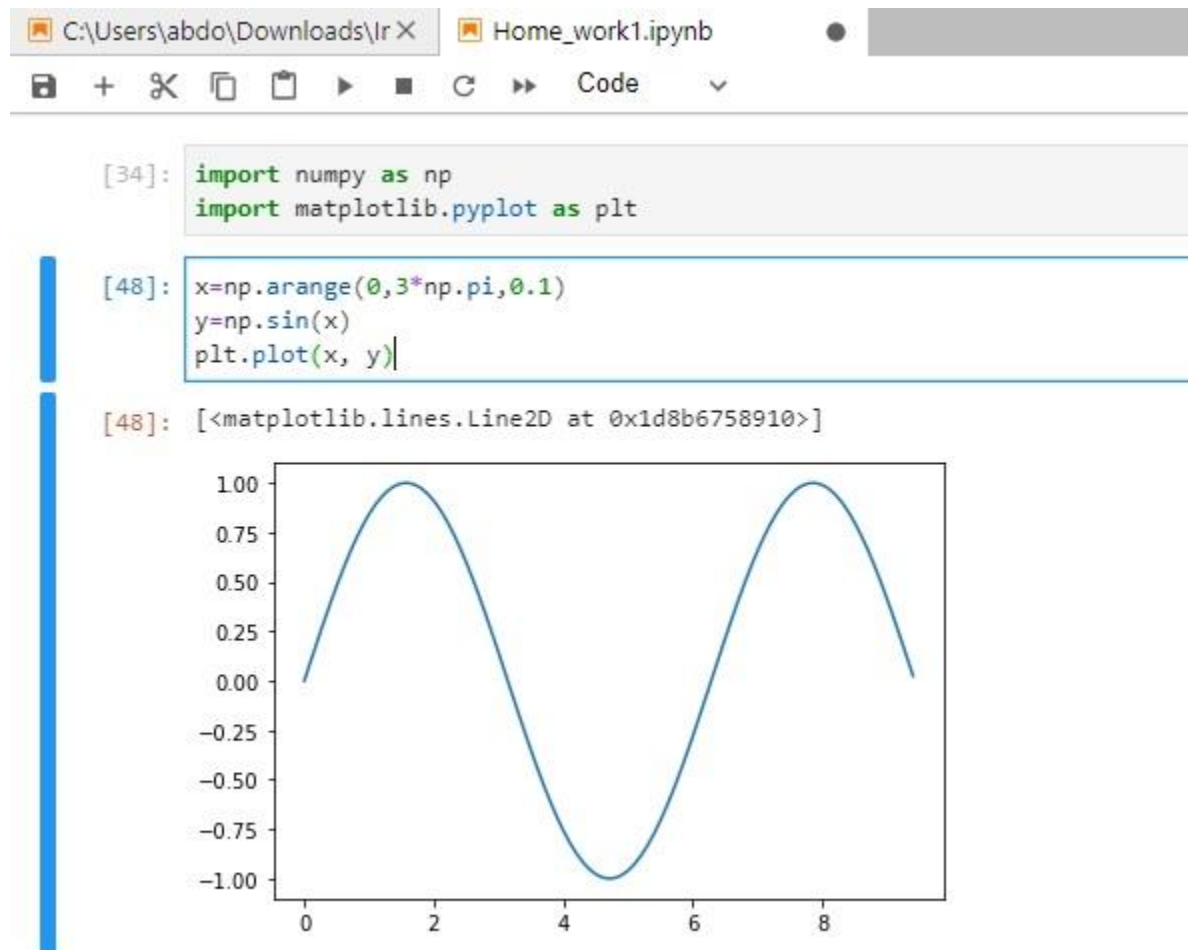
 The output of the code is displayed below the cell:

```
0 0 1 0 0
True
```

4- Write a NumPy code line(s) to compute the x and y coordinates for points on a sine curve and plot the points using matplotlib.

Sol:

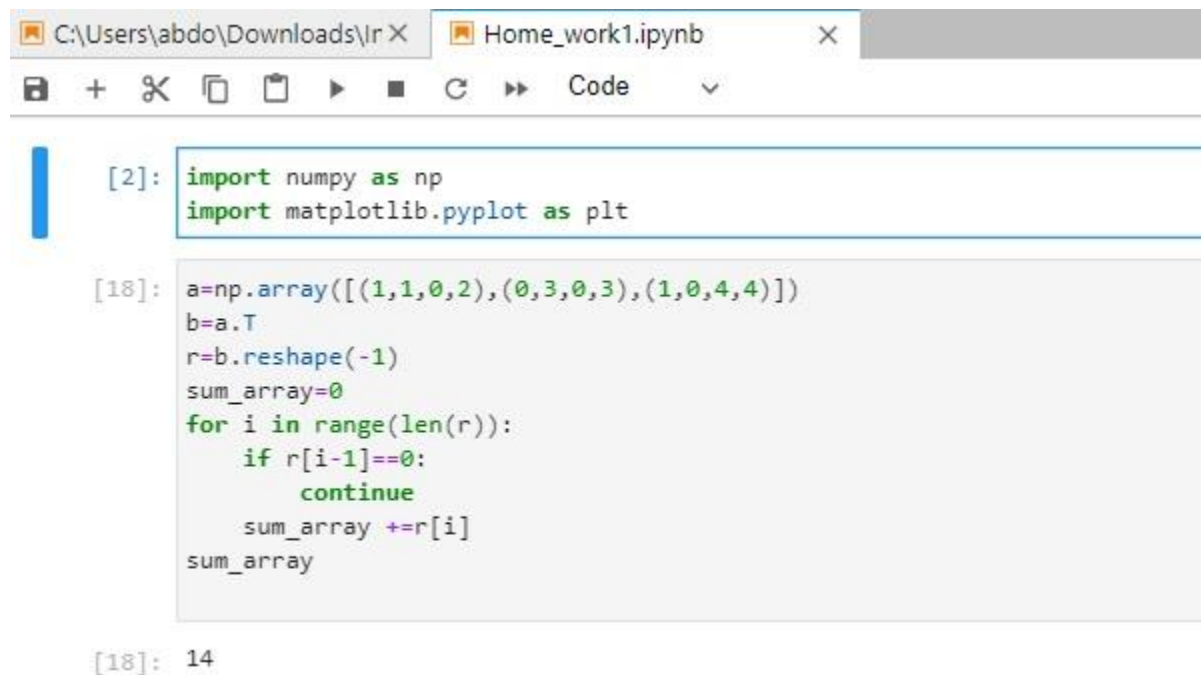
```
import numpy as np
import matplotlib.pyplot as plt
x=np.arange(0,3*np.pi,0.1)
y=np.sin(x)
plt.plot(x, y)
```



5- Write a NumPy code line(s) to add elements in a matrix. If an element in the matrix is 0, we will not add the element below this element (in red)

Sol:

```
import numpy as np
a=np.array([(1,1,0,2),(0,3,0,3),(1,0,4,4)])
b=a.T
r=b.reshape(-1)
sum_array=0
for i in range(len(r)):
    if r[i-1]==0:
        continue
    sum_array +=r[i]
sum_array
```



The screenshot shows a Jupyter Notebook window with two tabs: 'C:\Users\abdo\Downloads\lr X' and 'Home_work1.ipynb'. The 'Home_work1.ipynb' tab is active. The notebook contains two code cells. The first cell, labeled '[2]:', contains the imports: `import numpy as np` and `import matplotlib.pyplot as plt`. The second cell, labeled '[18]:', contains the main logic of the code: `a=np.array([(1,1,0,2),(0,3,0,3),(1,0,4,4)])`, `b=a.T`, `r=b.reshape(-1)`, `sum_array=0`, a `for` loop with a `continue` statement for zero values, and the final `sum_array` output. The output of the second cell is shown as `[18]: 14`.

```
[2]: import numpy as np
import matplotlib.pyplot as plt

[18]: a=np.array([(1,1,0,2),(0,3,0,3),(1,0,4,4)])
b=a.T
r=b.reshape(-1)
sum_array=0
for i in range(len(r)):
    if r[i-1]==0:
        continue
    sum_array +=r[i]
sum_array

[18]: 14
```

6- Write a NumPy code line(s) to extract all numbers which are less and greater than a specified integer in an input array

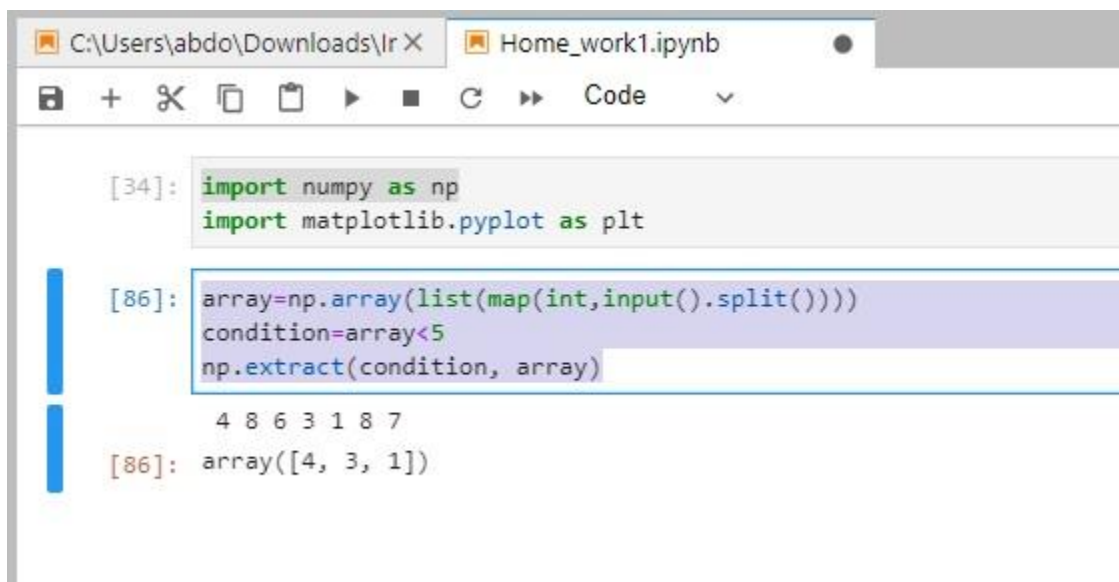
Sol:

```
import numpy as np
```

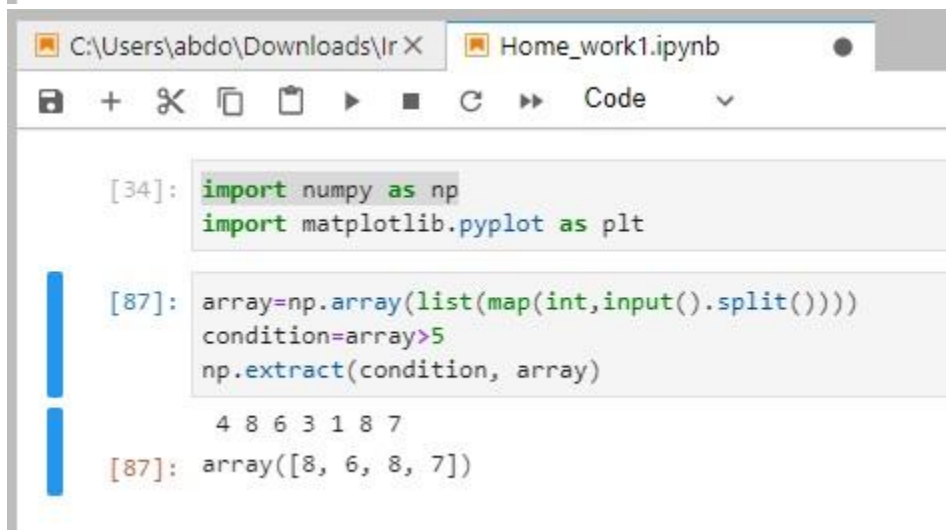
```
array=np.array(list(map(int,input().split())))
```

```
condition=array<5
```

```
np.extract(condition, array)
```



A screenshot of a Jupyter Notebook window titled 'Home_work1.ipynb'. The notebook has two cells. The first cell, labeled [34], contains the code: `import numpy as np` and `import matplotlib.pyplot as plt`. The second cell, labeled [86], contains the code: `array=np.array(list(map(int,input().split())))`, `condition=array<5`, and `np.extract(condition, array)`. Below the code in the second cell, the input '4 8 6 3 1 8 7' is shown, followed by the output `[86]: array([4, 3, 1])`.

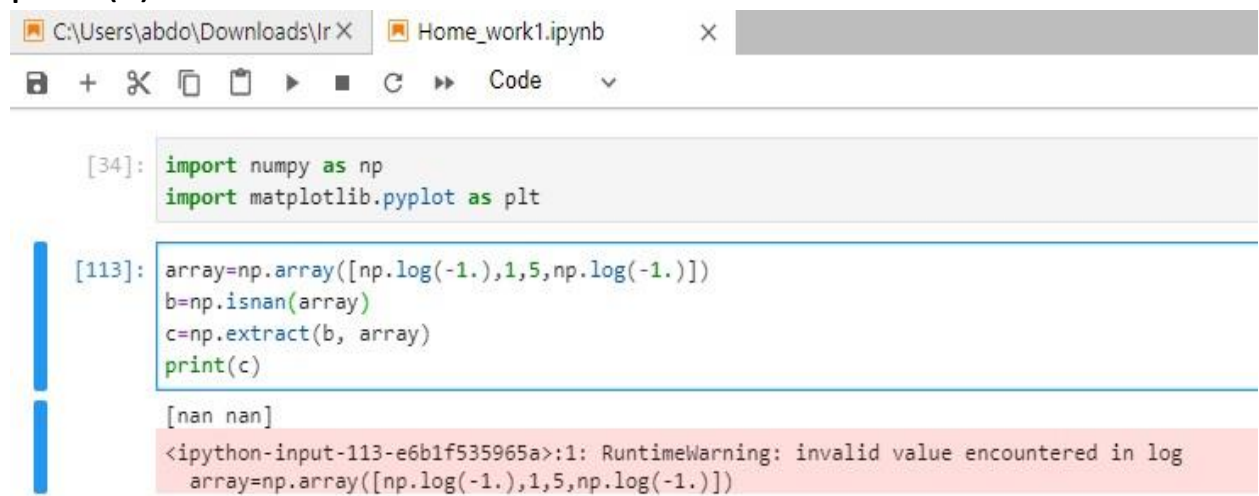


A screenshot of a Jupyter Notebook window titled 'Home_work1.ipynb'. The notebook has two cells. The first cell, labeled [34], contains the code: `import numpy as np` and `import matplotlib.pyplot as plt`. The second cell, labeled [87], contains the code: `array=np.array(list(map(int,input().split())))`, `condition=array>5`, and `np.extract(condition, array)`. Below the code in the second cell, the input '4 8 6 3 1 8 7' is shown, followed by the output `[87]: array([8, 6, 8, 7])`.

7- Write a NumPy code line(s) to find the missing (hint: undefined) data in an input array

Sol:

```
import numpy as np
array=np.array([np.log(-1.),1,5,np.log(-1.)])
b=np.isnan(array)
c=np.extract(b, array)
print(c)
```



The screenshot shows a Jupyter Notebook window with two tabs: 'C:\Users\abdo\Downloads\lr X' and 'Home_work1.ipynb'. The 'Home_work1.ipynb' tab is active. The notebook interface includes a toolbar with icons for saving, adding, deleting, and running code. Below the toolbar, there are two code cells. The first cell, labeled '[34]:', contains the code: `import numpy as np` and `import matplotlib.pyplot as plt`. The second cell, labeled '[113]:', contains the code: `array=np.array([np.log(-1.),1,5,np.log(-1.)])`, `b=np.isnan(array)`, `c=np.extract(b, array)`, and `print(c)`. The output of the second cell is `[nan nan]`. Below the output, a red warning box displays the message: `<ipython-input-113-e6b1f535965a>:1: RuntimeWarning: invalid value encountered in log` followed by the code `array=np.array([np.log(-1.),1,5,np.log(-1.)])`.

```
[34]: import numpy as np
import matplotlib.pyplot as plt

[113]: array=np.array([np.log(-1.),1,5,np.log(-1.)])
b=np.isnan(array)
c=np.extract(b, array)
print(c)

[nan nan]

<ipython-input-113-e6b1f535965a>:1: RuntimeWarning: invalid value encountered in log
array=np.array([np.log(-1.),1,5,np.log(-1.)])
```