In [1]:
```
###assignment-1-smartinternz
```

In [2]:
```
#_1-Write a Python program to calculate the area of a rectangle given its length and width.
length = int(input())
width = int(input())
if length>0 and width>0:
    area = length*width
    print(area)
else:
    print("No length and width")
```

```
7
6
42
```

In [3]:
```
#_2-Write a program to convert miles to kilometers
miles = int(input())
#1mile  = 1.60934
kilometers = miles*1.60934
print(kilometers)
```

```
7
11.26538
```

In [5]:

```python
#_3-Write a function to check if a given string is a palindrome.
def is_palindrome(num):
    num_str = str(num)

    reversed_str = num_str[::-1]

    if num_str == reversed_str:
        return True
    else:
        return False

num = int(input("Enter a number: "))

if is_palindrome(num):
    print(num, "is a palindrome")
else:
    print(num, "is not a palindrome")
```

```
Enter a number: 121
121 is a palindrome
```

In [6]:

```python
#_4-Write a Python program to find the second largest element in a list.
list1 = [10, 20, 20, 4, 45, 45, 45, 99, 99]

list2 = list(set(list1))
list2.sort()

print("Second largest element is:", list2[-2])
```

```
Second largest element is: 45
```

In [7]:

```
#_5-Explain what indentation means in Python

#INDENTATION:
#Indentation refers to the spaces at the beginning of a code line.

#Where in other programming languages the indentation in code is for readability only, the indentation

#Python uses indentation to indicate a block of code.
#sample program to explain an indentation error
if 5 > 2:
print("Five is greater than two!")
```

```
  Cell In[7], line 11
    print("Five is greater than two!")
    ^
IndentationError: expected an indented block after 'if' statement on line 10
```

In [8]:

```
#_6-Write a program to perform set difference operation.
set1 = {1, 2, 3, 4, 5}
set2 = {3, 4, 5, 6, 7}
difference_set = set1 - set2
print("Set Difference:", difference_set)
```

```
Set Difference: {1, 2}
```

In [28]:

```python
#_7-Write a Python program to print numbers from 1 to 10 using a while loop.
num = 1
while num <= 10:
    print(num)
    num += 1
```

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

In [16]:

```python
#_8-Write a program to calculate the factorial of a number using a while loop.
n = int(input())
fact =1
itern = 1
while itern<=n:
    fact = fact*itern
    itern = itern+1
print(fact)
```

```
5
120
```

In [17]:

```python
#_9-Write a Python program to check if a number is positive, negative, or zero using if-elif-else
n = int(input())
if n ==0:
    print("Zero")
elif n>=0:
    print("Positive number")
else:
    print("Negative number")
```

```
343
Positive number
```

In [21]:

```python
#_10-Write a program to determine the largest among three numbers using conditional statements
num1 = int(input())
num2 = int(input())
num3 = int(input())

if (num1 >= num2) and (num1 >= num3):
   largest = num1
elif (num2 >= num1) and (num2 >= num3):
   largest = num2
else:
   largest = num3

print("The largest number is", largest)
```

```
5
6
7
The largest number is 7
```

In [22]:

```python
#_11-Write a Python program to create a numpy array filled with ones of given shape
import numpy as np
rows = int(input("Enter the number of rows: "))
columns = int(input("Enter the number of columns: "))
ones_array = np.ones((rows, columns))
print("Array filled with ones of shape", one_array.shape, ":")
print(one_array)
```

```
Enter the number of rows: 3
Enter the number of columns: 5
Array filled with ones of shape (3, 5) :
[[1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]]
```

In [23]:

```python
#_12-Write a program to create a 2D numpy array initialized with random integers.
import numpy as np
rows = int(input("Enter the number of rows: "))
columns = int(input("Enter the number of columns: "))
random_array = np.random.randint(low=0, high=100, size=(rows, columns))
print("2D NumPy array initialized with random integers:")
print(random_array)
```

```
Enter the number of rows: 5
Enter the number of columns: 6
2D NumPy array initialized with random integers:
[[39 60  3 38 50 17]
 [ 1  0 20 55  7  0]
 [91 26 11 28 75 18]
 [55 58 35 52  8  6]
 [71 33 32 27 29 56]]
```

In [24]:
```
#_13-Write a Python program to generate an array of evenly spaced numbers over a specified range using
import numpy as np
start = int(input("Enter the start value: "))
stop = int(input("Enter the stop value: "))
num_elements = int(input("Enter the number of elements: "))
evenly_spaced_array = np.linspace(start, stop, num_elements)

print("Array of evenly spaced numbers over the range [{}, {}]:".format(start, stop))
print(evenly_spaced_array)
```

```
Enter the start value: 3
Enter the stop value: 45
Enter the number of elements: 23
Array of evenly spaced numbers over the range [3, 45]:
[ 3.          4.90909091  6.81818182  8.72727273 10.63636364 12.54545455
 14.45454545 16.36363636 18.27272727 20.18181818 22.09090909 24.
 25.90909091 27.81818182 29.72727273 31.63636364 33.54545455 35.45454545
 37.36363636 39.27272727 41.18181818 43.09090909 45.        ]
```

In [25]:
```
#_14- Write a program to generate an array of 10 equally spaced values between 1 and 100 using linspace
import numpy as np
equally_spaced_array = np.linspace(1, 100, 10)
print("Array of 10 equally spaced values between 1 and 100:")
print(equally_spaced_array)
```

```
Array of 10 equally spaced values between 1 and 100:
[  1.  12.  23.  34.  45.  56.  67.  78.  89. 100.]
```

In [26]:

```python
#_15-Write a Python program to create an array containing even numbers from 2 to 20 using arange.
import numpy as np
even_array = np.arange(2, 21, 2)
print("Array containing even numbers from 2 to 20:")
print(even_array)
```

```
Array containing even numbers from 2 to 20:
[ 2  4  6  8 10 12 14 16 18 20]
```

In [27]:

```python
#_16-Write a program to create an array containing numbers from 1 to 10 with a step size of 0.5 using a
import numpy as np
array_with_step = np.arange(1, 10.5, 0.5)
print("Array containing numbers from 1 to 10 with a step size of 0.5:")
print(array_with_step)
```

```
Array containing numbers from 1 to 10 with a step size of 0.5:
[ 1.   1.5  2.   2.5  3.   3.5  4.   4.5  5.   5.5  6.   6.5  7.   7.5
  8.   8.5  9.   9.5 10. ]
```

In [ ]: