1.Write a program to find the area of circle given its radius

In [57]:
```
# Importing math package
# Getting radius input from the user
# caluculating area
# Print the output

import math

radius = int(input("Please provide the radius to find the area of circ
area_of_cricle = math.pi * radius ** 2
print(f"Area of circle with radius {radius} is : {area_of_cricle}")
```

```
Please provide the radius to find the area of circle: 5
Area of circle with radius 5 is : 78.53981633974483
```

2.Given a list [8, 2, 9, 4, 6], write a Python program to reverse the list without using any

built-in functions.

In [14]:
```
# Arranged list
# Initialising an empty list
# Using for loop to traverse the list
# Adding each element of the arranged list to the empty
# Printing the reversed list

arranged_list = [8,2,9,4,6]
reversed_list = []
for i in arranged_list:
    reversed_list= [i] + reversed_list
print(reversed_list)
```

```
[6, 4, 9, 2, 8]
```

3. Write a Python code to find whether a number is prime or not.

In [58]:
```
# Get the number to check from the input
# if input number is less than or equal to 1, it is not prime
# if number is greater then 1 and if it divisible by 1 and on its own
#  Hsving more than 2 factors then not prime

number_to_check = int(input("Please input a number to check whether it

if number_to_check<=1:
    print("Number is not prime")
elif number_to_check>1:
    # checking for factors
    for i in range(2,number_to_check):
        if (number_to_check % i) == 0:
            print(number_to_check,"is not a prime number")
            print(i,"times",number_to_check//i,"is",number_to_check)
            break
    else:
        print(number_to_check,"is a prime number")
```

```
Please input a number to check whether it is prime or not: 6
6 is not a prime number
2 times 3 is 6
```

4. Write a Python program to calculate the factorial of a given number using a function.

(Without recursion)

In [59]:
```
fact_number = int(input("Enter a number to find the facorial number: '
# Factorial number started with 1
factorial_number = 1
for i in range(1,fact_number+1):
    factorial_number  = factorial_number * i
print(f"Factorial of {fact_number} is {factorial_number}")
```

```
Enter a number to find the facorial number: 4
Factorial of 4 is 24
```

5. Explain the concept of lambda functions in Python. Provide a code as an example using a lambda function.

Lambda functions in Python are small, anonymous functions defined using the lambda keyword. Unlike standard functions declared with def, lambda functions can have any number of arguments but only one expression.

The expression is evaluated and returned when the function is called. Lambda functions are useful for short, simple operations that are used temporarily and don't require a formal function definition.

Syntax

lambda arguments : expression

A lambda function has the following characteristics:

1.It can only contain expressions and can't include statements in its body. 2.It is written as a single line of execution. 3.It does not support type annotations. 4.It can be immediately invoked (IIFE).

No Statements

A lambda function can't contain any statements. In a lambda function, statements like return, pass, assert, or raise will raise a SyntaxError exception. Here's an example of adding assert to the body of a lambda:

In [60]:
```
lambda x: assert x == 2
```

```
  Cell In[60], line 1
    lambda x: assert x == 2
              ^
SyntaxError: invalid syntax
```

In [61]:
```
# Code as an example
# Here we are passing x and y as arguments and calculating the sum usi
add = lambda x, y: x + y
print(add(5, 3))
```

```
8
```

In [62]:
```
# Second example with filter() to get even numbers from a list
nums = [1, 2, 3, 4, 5, 6, 7, 8, 9]
even_nums = list(filter(lambda x: x % 2 == 0, nums))
print(even_nums)
```

```
[2, 4, 6, 8]
```

6a. Explain three features of Python that make it suitable for AI development.

Python is a popular programming language for AI development due to its versatility, simplicity, and a rich ecosystem of libraries and frameworks. Here are some features of Python that make it well-suited for AI development:

Ease of Learning and Readability:

Python has a simple and clean syntax that makes it easy to learn and read. This is advantageous for developers working on AI projects, as it allows them to focus on the logic and algorithms rather than dealing with complex syntax.

Extensive Libraries and Frameworks:

Python has a vast collection of libraries and frameworks that are specifically designed for machine learning and AI development. Notable examples include TensorFlow, PyTorch, scikit-learn, Keras, and NLTK. These libraries provide pre-built functions and tools that simplify the implementation of AI algorithms. Community and Support:

Platform Independence:

Python is platform-independent, meaning that code written in Python can run on various operating systems without modification. This feature is crucial for AI development, as it allows for seamless deployment across different platforms. Integration with Other Languages:

Data Science and Visualization Tools:

Python is widely used in the field of data science, and it provides excellent tools for data manipulation and analysis. Libraries like NumPy, Pandas, and Matplotlib are extensively used for data handling, exploration, and visualization, which are essential components of AI projects. Diversity of Applications:

Developers can quickly test and iterate on their AI models, making the development process more efficient.

6b. Explain the role of the NumPy library in Python. Provide an example of creating and performing operations on a NumPy array.

1. NumPy (Numerical Python) is an open source Python library that’s used in almost every field of science and engineering. It’s the universal standard for working with numerical data in Python, and it’s at the core of the scientific Python and PyData ecosystems.
2. The NumPy library contains multidimensional array and matrix data structures It provides ndarray, a homogeneous n-dimensional array object, with methods to efficiently operate on it. NumPy can be used to perform a wide variety of mathematical operations on arrays.
3. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices.
4. NumPy arrays are faster and more compact than Python lists. An array consumes less memory and is convenient to use. NumPy uses much less memory to store data and it provides a mechanism of specifying the data types. This allows the code to be optimized even further.
5. If you are using Anaconda the distribution comes with the numpy installed so, there is no need of installing numpy separately simply we can import like below

   import numpy as np

In [63]:
```
# Impoprting numpy as np, we can call what ever alias as needed there
import numpy as np

#  creating an array
a = np.array([1, 2, 3, 4, 5, 6])
a
```

Out[63]:
```
array([1, 2, 3, 4, 5, 6])
```

In [64]:
```
# Accessing an element from the array
a[3]
```

Out[64]:
```
4
```

In [65]:
```
array_a = np.array([[[0, 1, 2],
                            [4, 5, 6]],

                           [[0, 1, 2],
                            [4, 5, 6]],

                           [[0 ,1, 3],
                            [4, 5, 7]]])
```

In [66]:
```
array_a
```

Out[66]:
```
array([[[0, 1, 2],
        [4, 5, 6]],

       [[0, 1, 2],
        [4, 5, 6]],

       [[0, 1, 3],
        [4, 5, 7]]])
```

In [67]:
```
# ndim will tell you the number of axes, or dimensions, of the array.H
array_a.ndim
```

Out[67]:
```
3
```

In [68]:
```
# size will tell you the total number of elements of the array. This i
array_a.size
```

Out[68]:
```
18
```

In [69]:
```
# shape will display a tuple of integers that indicate the number of e
# If, for example, The above is a 3-D array with 2 rows and 3 columns,
array_a.shape
```

Out[69]:
```
(3, 2, 3)
```

In [70]:
```
# Array data type
array_a.dtype
```

Out[70]:
```
dtype('int64')
```

```
In [71]: a = np.array([1,4])
```

```
In [72]: b = np.array([3,4])
```

```
In [73]: # Adding 2 different arrays
         a + b

Out[73]: array([4, 8])
```

```
In [74]: # arange funtion
         np.arange(12)

Out[74]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])
```

```
In [75]: # arange func with low and high values
         np.arange(2,19)

Out[75]: array([ 2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 1
         7, 18])
```

```
In [76]: # checking funtion docstring by using syntax np.functionname?
         np.broadcast?
```

```
In [ ]:
```