Using Functions

Introduction

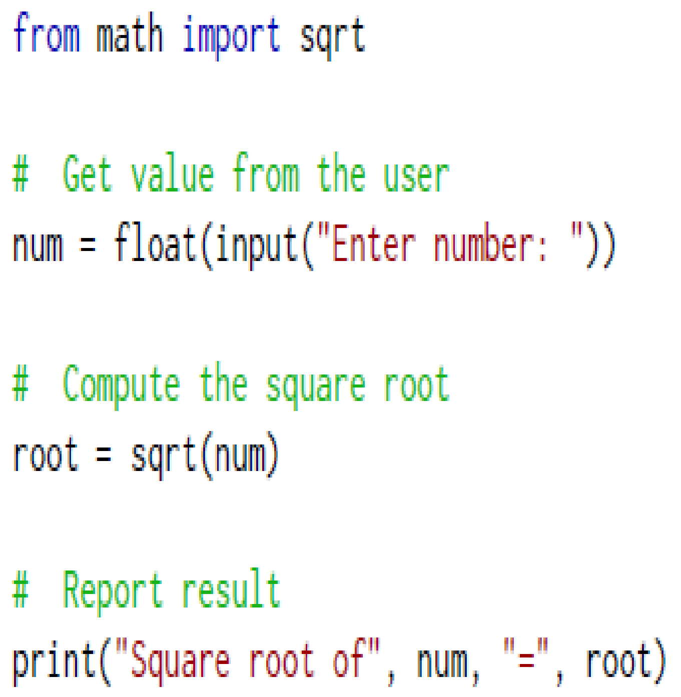
- One way to make code more reusable is by packaging it in functions.
- A function is a unit of reusable code.
- Python provides a collection of standard functions stored in libraries called modules.
- Programmers can use the functions from these libraries within their own code to build sophisticated programs.
- The Python standard library includes functions such as:

  print, input, int, float, str, and type

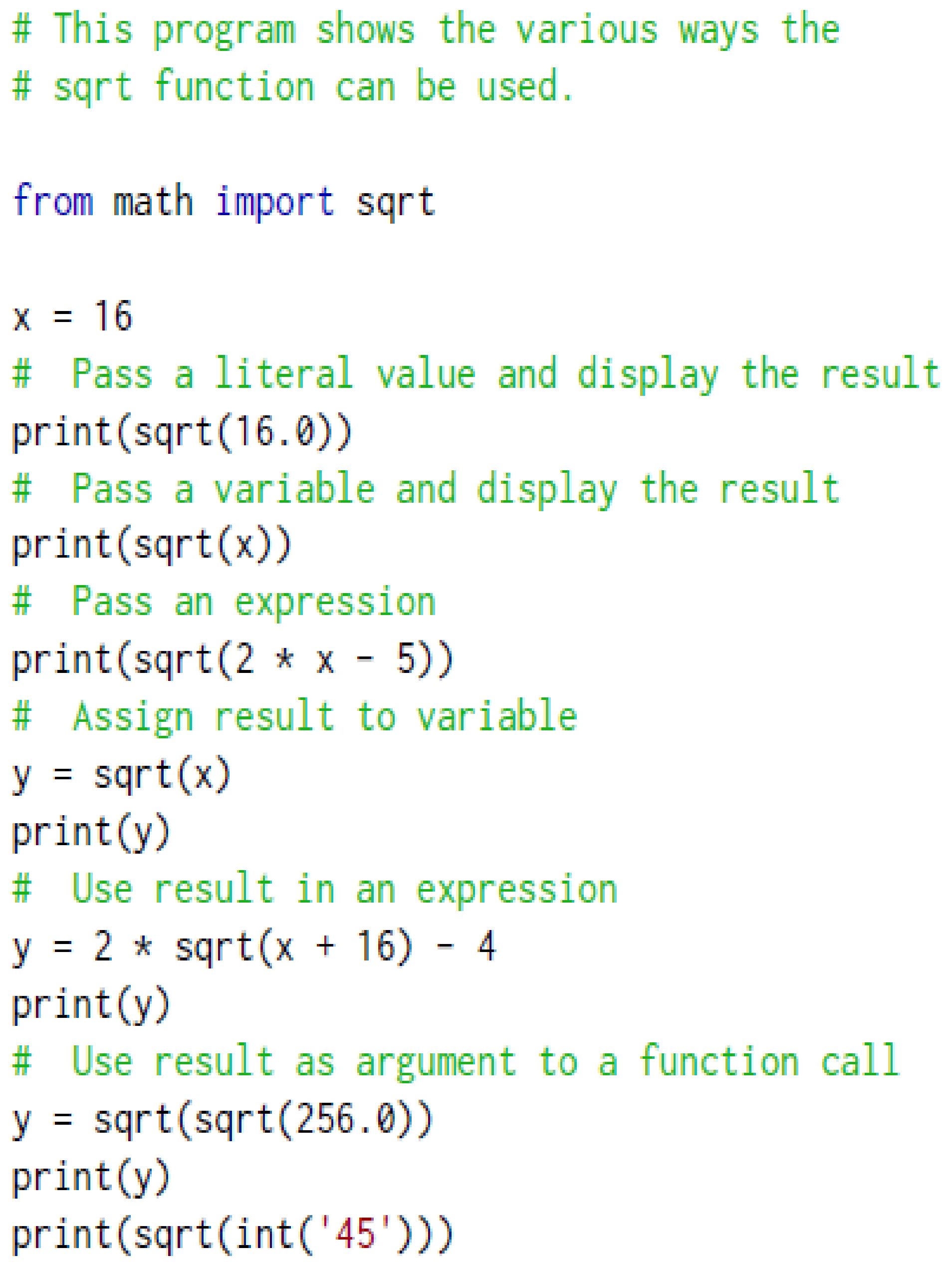
Introduction

- In mathematics, a function computes a result from a given value.
- A function in Python works like a mathematical function.
- In Python, a function is a named block of code that performs a specific task.
- If an executing program needs to perform such a task, it calls upon the function to do the work.

Examples



```python
from math import sqrt

#  Get value from the user
num = float(input("Enter number: "))

#  Compute the square root
root = sqrt(num)

#  Report result
print("Square root of", num, "=", root)
```



```python
# This program shows the various ways the
# sqrt function can be used.

from math import sqrt

x = 16
#  Pass a literal value and display the result
print(sqrt(16.0))
#  Pass a variable and display the result
print(sqrt(x))
#  Pass an expression
print(sqrt(2 * x - 5))
#  Assign result to variable
y = sqrt(x)
print(y)
#  Use result in an expression
y = 2 * sqrt(x + 16) - 4
print(y)
#  Use result as argument to a function call
y = sqrt(sqrt(256.0))
print(y)
print(sqrt(int('45')))
```

Parts of a function

- Name

  Every function has a name that identifies the code to be executed.

- Parameters

  A function must be called with a certain number of parameters, and each parameter must be the correct type.

- Result type

  A function returns a value to its caller. Generally a function will compute a result and return the value of the result to the caller.

Functions and Modules

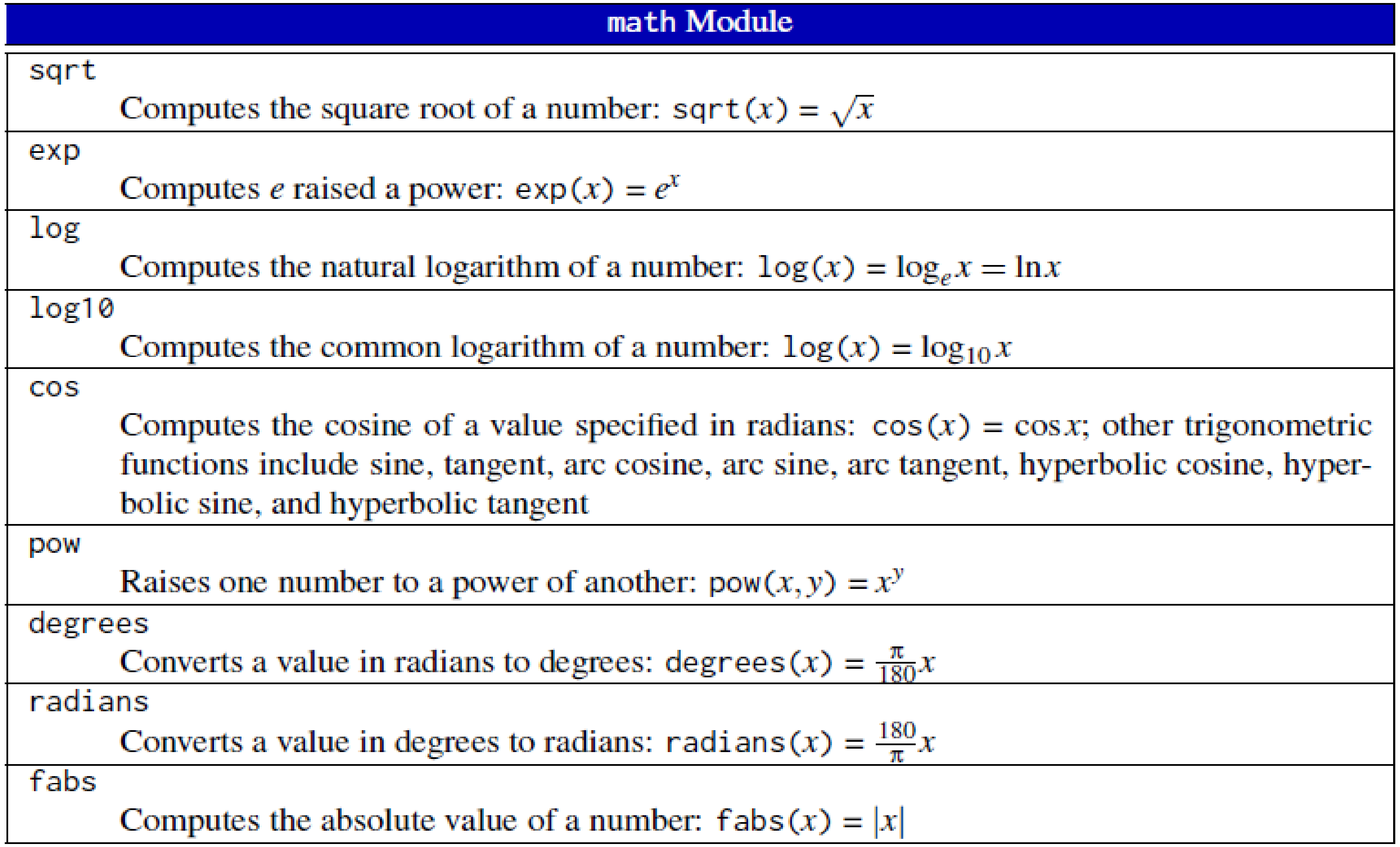
- A Python module is simply a file that contains Python code.
- The name of the file dictates the name of the module.
- The Python standard library contains thousands of functions distributed throughout more than 230 modules.
- One of the modules, known as the built-ins module (actual name __builtins__), contains all the functions we have been using : print, input, etc

Functions and Modules

```
from module import function list
```

```
import module list
```

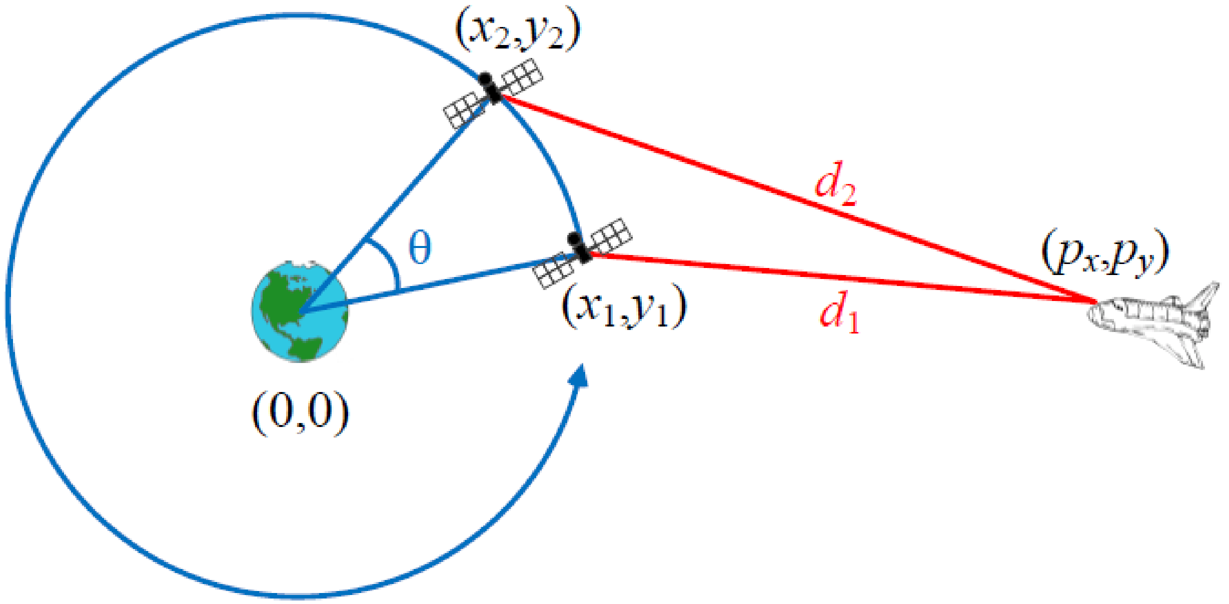
Standard Mathematical Functions



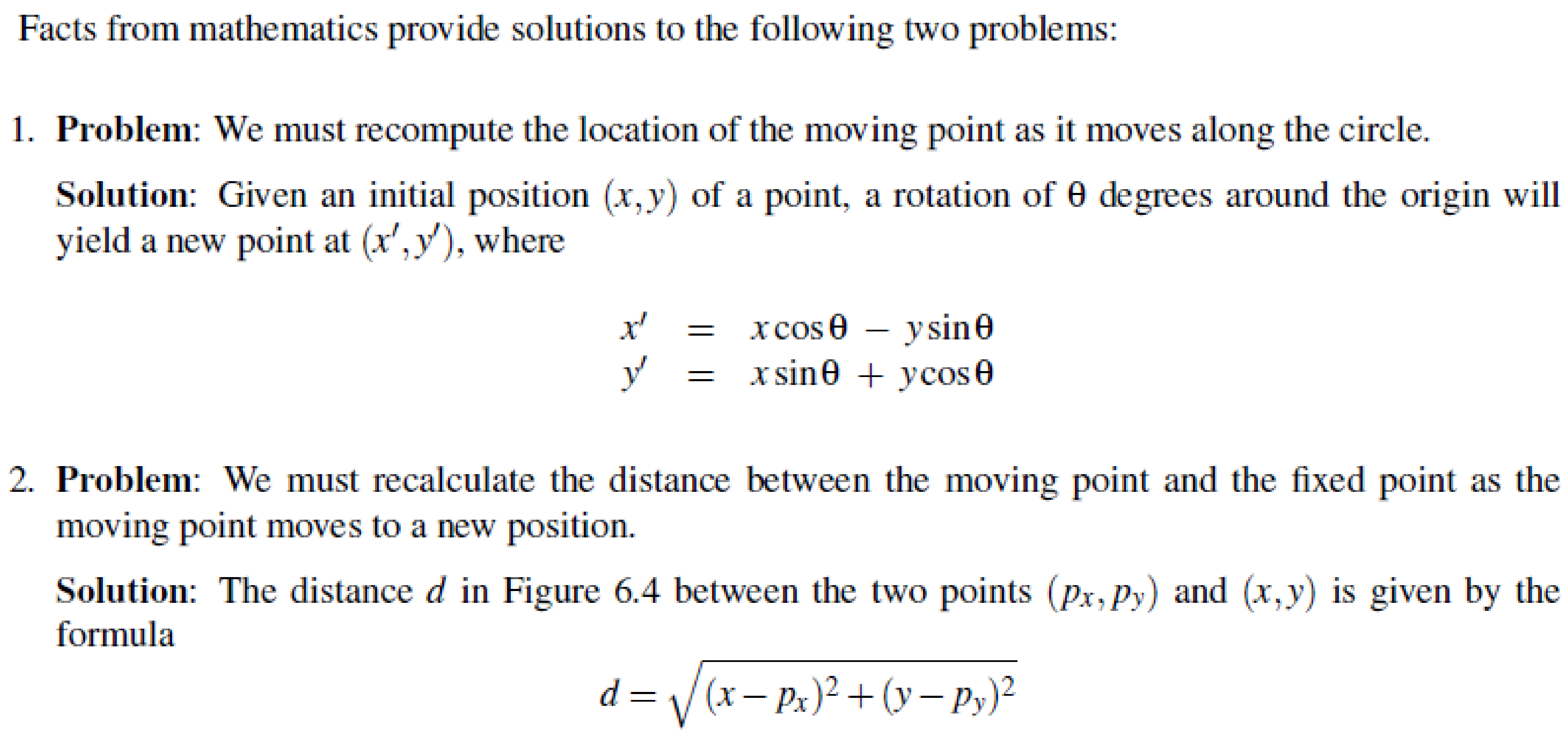
math Module	
`sqrt`	Computes the square root of a number: `sqrt`$(x) = \sqrt{x}$
`exp`	Computes e raised a power: `exp`$(x) = e^x$
`log`	Computes the natural logarithm of a number: `log`$(x) = \log_e x = \ln x$
`log10`	Computes the common logarithm of a number: `log`$(x) = \log_{10} x$
`cos`	Computes the cosine of a value specified in radians: `cos`$(x) = \cos x$; other trigonometric functions include sine, tangent, arc cosine, arc sine, arc tangent, hyperbolic cosine, hyperbolic sine, and hyperbolic tangent
`pow`	Raises one number to a power of another: `pow`$(x,y) = x^y$
`degrees`	Converts a value in radians to degrees: `degrees`$(x) = \frac{\pi}{180}x$
`radians`	Converts a value in degrees to radians: `radians`$(x) = \frac{180}{\pi}x$
`fabs`	Computes the absolute value of a number: `fabs`$(x) = \lvert x \rvert$

Example : Orbiting Satellite

- Suppose a spacecraft is at a fixed location in space some distance from a planet. A satellite is orbiting the planet in a circular orbit. We wish to compute how much farther away the satellite will be from the spacecraft when it has progressed q degrees along its orbital path.



Example : Orbiting Satellite



Facts from mathematics provide solutions to the following two problems:

1. **Problem**: We must recompute the location of the moving point as it moves along the circle.

   Solution: Given an initial position (x,y) of a point, a rotation of θ degrees around the origin will yield a new point at (x',y'), where

$$\begin{aligned} x' &= x\cos\theta - y\sin\theta \\ y' &= x\sin\theta + y\cos\theta \end{aligned}$$

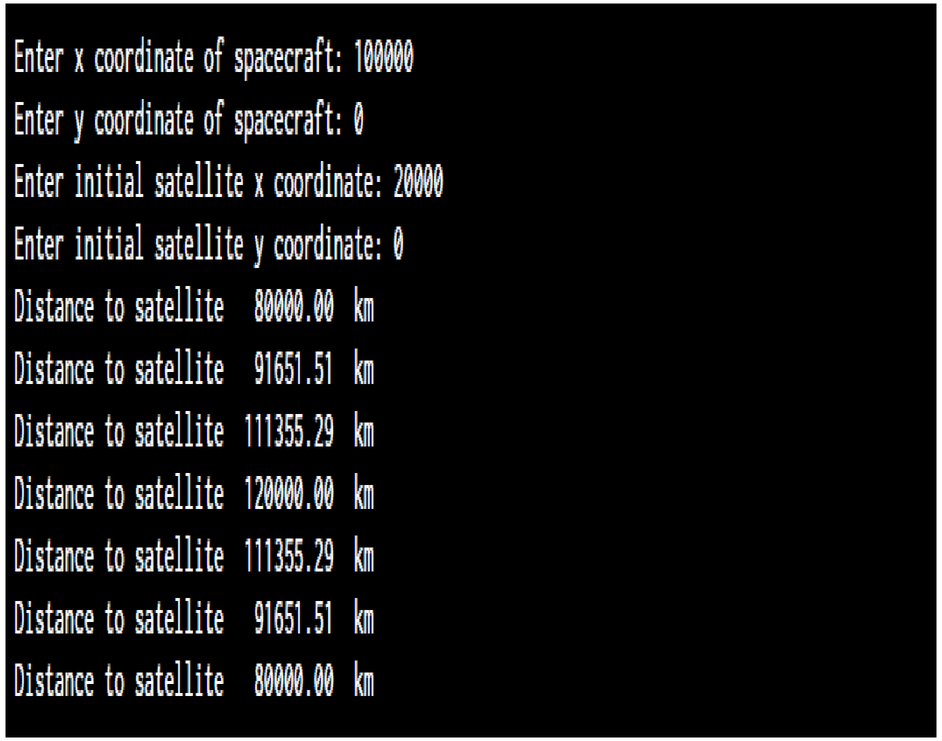
2. **Problem**: We must recalculate the distance between the moving point and the fixed point as the moving point moves to a new position.

   Solution: The distance d in Figure 6.4 between the two points (p_x, p_y) and (x,y) is given by the formula

$$d = \sqrt{(x - p_x)^2 + (y - p_y)^2}$$

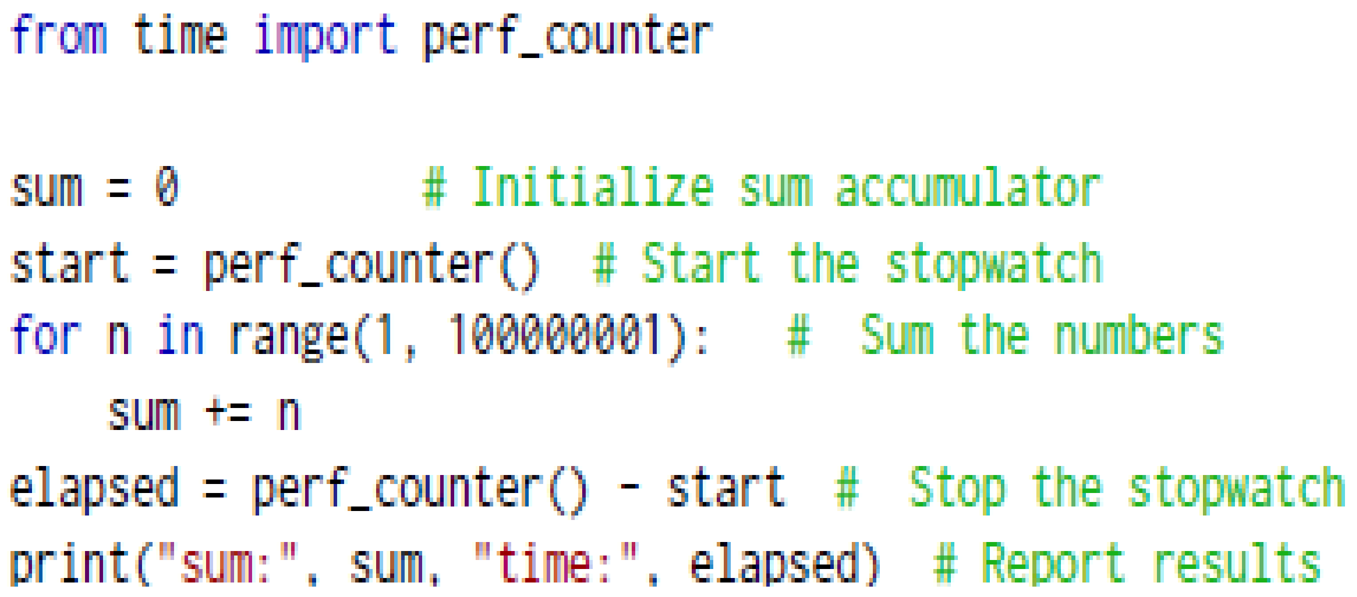
Example : Orbiting Satellite

```python
#  Use some functions and values from the math module
from math import sqrt, sin, cos, pi, radians

#  Get coordinates of the stationary spacecraft, (px, py)
px = float(input("Enter x coordinate of spacecraft: "))
py = float(input("Enter y coordinate of spacecraft: "))

#  Get starting coordinates of satellite, (x1, y1)
x = float(input("Enter initial satellite x coordinate: "))
y = float(input("Enter initial satellite y coordinate: "))

#  Convert 60 degrees to radians to be able to use the trigonometric functions
rads = radians(60)

#  Precompute the cosine and sine of the angle
COS_theta = cos(rads)
SIN_theta = sin(rads)

#  Make a complete revolution (6*60 = 360 degrees)
for increment in range(0, 7):
    # Compute the distance to the satellite
    dist = sqrt((px - x)*(px - x) + (py - y)*(py - y))
    print('Distance to satellite {0:10.2f}  km'.format(dist))
    # Compute the satellite's new (x, y) location after rotating by 60 degrees
    x, y = x*COS_theta - y*SIN_theta, x*SIN_theta + y*COS_theta
```

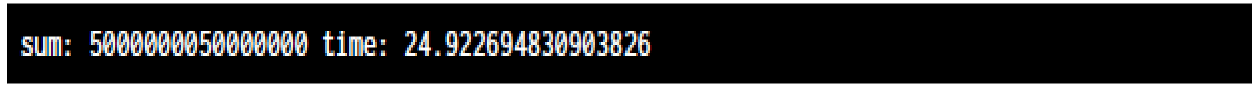


```
Enter x coordinate of spacecraft: 100000
Enter y coordinate of spacecraft: 0
Enter initial satellite x coordinate: 20000
Enter initial satellite y coordinate: 0
Distance to satellite   80000.00  km
Distance to satellite   91651.51  km
Distance to satellite  111355.29  km
Distance to satellite  120000.00  km
Distance to satellite  111355.29  km
Distance to satellite   91651.51  km
Distance to satellite   80000.00  km
```

time Functions



```python
from time import perf_counter

sum = 0             # Initialize sum accumulator
start = perf_counter()  # Start the stopwatch
for n in range(1, 100000001):   #  Sum the numbers
    sum += n
elapsed = perf_counter() - start  #  Stop the stopwatch
print("sum:", sum, "time:", elapsed)  # Report results
```

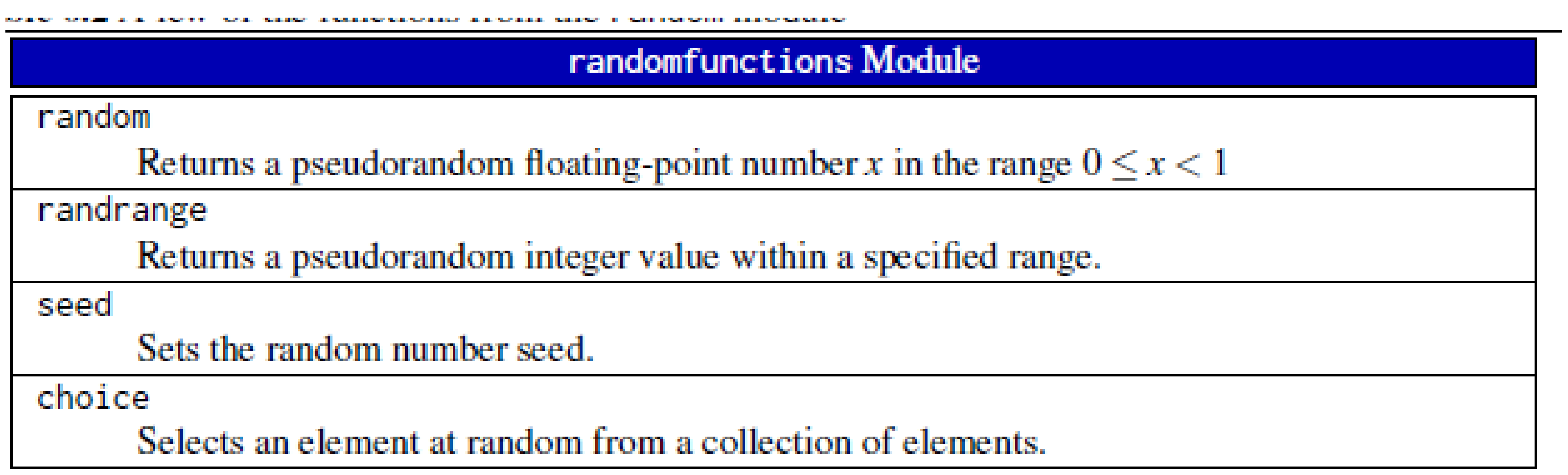


```
sum: 5000000050000000 time: 24.922694830903826
```

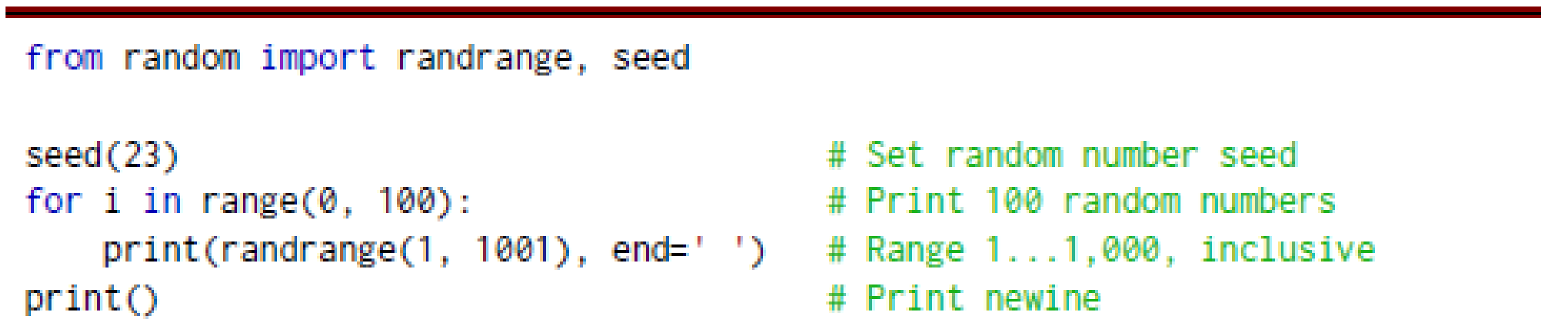
Example

```python
from math import sqrt
from time import perf_counter

max_value = 10000
count = 0
value = 2         # Smallest prime number
start = perf_counter()  # Start the stopwatch
while value <= max_value:
    # See if value is prime
    is_prime = True  # Provisionally, value is prime
    # Try all possible factors from 2 to value - 1
    trial_factor = 2
    root = sqrt(value)
    while trial_factor <= root:
        if value % trial_factor == 0:
            is_prime = False    # Found a factor
            break               # No need to continue; it is NOT prime
        trial_factor += 1       # Try the next potential factor
    if is_prime:
        count += 1              # Count the prime number
    value += 1                  # Try the next potential prime number
elapsed = perf_counter() - start       # Stop the stopwatch
print("Count:", count, "  Elapsed time:", elapsed, "sec")
```

Random Numbers



randomfunctions Module
`random` Returns a pseudorandom floating-point number x in the range $0 \leq x < 1$
`randrange` Returns a pseudorandom integer value within a specified range.
`seed` Sets the random number seed.
`choice` Selects an element at random from a collection of elements.



```python
from random import randrange, seed

seed(23)                                    # Set random number seed
for i in range(0, 100):                     # Print 100 random numbers
    print(randrange(1, 1001), end=' ')      # Range 1...1,000, inclusive
print()                                     # Print newine
```