Contents

1. Modules, packages and libraries

Module

The module is a simple Python file that contains collections of functions, classes and global variables and with having a .py extension file.

Example of modules are Datetime, Random, Math etc.

Example (random module):

```
import random


print(random.random())
print(random.uniform(2.1, 5.4))
print(random.randint(2, 10))
print(random.randrange(1, 10, 2))


numbers = [1, 3, 5, 7, 9]
random.shuffle(numbers)
print(numbers)
```

```
0.4382587528646582
4.610697534556159
3
1
[9, 5, 7, 1, 3]
```

Let's play Guessing Game:

```
import random
print("Guessing Game: ")
magic_number = random.randint(1, 50)
guess = None
attempts = 0


while guess != magic_number:
    guess = int(input("Enter a number between [1, 50]"))
    attempts += 1

    if guess < magic_number:
        print("Try Higgher")

    elif guess > magic_number:
        print("Try Lower")

    else:
        print(f"You got correct ans i.e. {magic_number} in {attempts} attemps")
```

Example (math module):

```
import math

print(math.pi)
print(math.e)
print(math.pow(2, 5))
print(math.sqrt(25))
print(math.factorial(5))
print(math.gcd(8, 12))
```

```
3.141592653589793
2.718281828459045
32.0
5.0
120
4
```

Creating and using a module.

Creating Module **calculation.py**

```
def add(a, b):
    return a + b


def subtract(a, b):
    return a - b


def multiply(a, b):
    return a * b


def divide(a, b):
    try:
        return a / b
    except ZeroDivisionError:
        return "Denominator cannot be zero"
```

1. Using module

```
import calculation


print(calculation.add(2, 3))
print(calculation.subtract(2, 3))
print(calculation.multiply(2, 3))
print(calculation.divide(2, 0))
```

In Python, there are several ways to import modules and specific parts of modules, each with its own use cases. Here are the most common methods:

1. **Importing the Entire Module**: import calculation
2. **Importing Specific Functions or Classes**: from calculation import add, subtract
3. **Importing All Names from a Module**: from calculation import *
4. **Importing a Module with an Alias**: import calculation as cl

Importing Specific Functions or Classes

from calculation import add, subtract
 	
print(add(2, 3))
print(subtract(2, 3))

Importing All Names from a Module

from calculation import *
 	
print(add(2, 3))
print(subtract(2, 3))
print(multiply(2, 10))
print(divide(3, 4))

Importing a Module with an Alias:

import calculation as cl
 	
print(cl.add(2, 3))
print(cl.subtract(2, 3))
print(cl.multiply(2, 10))
print(cl.divide(3, 4))

Math Module docs

Math Constants

- **math.pi**: The mathematical constant π (pi), approximately 3.14159.
- **math.e**: The base of natural logarithms (Euler's number), approximately 2.71828.
- **math.tau**: The mathematical constant τ (tau), which is 2π, approximately 6.28318.
- **math.inf**: Represents positive infinity.
- **math.nan**: Represents a "Not a Number" (NaN) value.

Math Methods

- **math.sqrt(x)**: Returns the square root of x.
- **math.pow(x, y)**: Returns x raised to the power of y.

- **math.exp(x)**: Returns e raised to the power of x.
- **math.log(x, [base])**: Returns the logarithm of x to the specified base. If base is not provided, returns the natural logarithm.
- **math.log10(x)**: Returns the base-10 logarithm of x.
- **math.log2(x)**: Returns the base-2 logarithm of x.
- **math.sin(x)**: Returns the sine of x (in radians).
- **math.cos(x)**: Returns the cosine of x (in radians).
- **math.tan(x)**: Returns the tangent of x (in radians).
- **math.asin(x)**: Returns the arc sine (inverse sine) of x, in radians.
- **math.acos(x)**: Returns the arc cosine (inverse cosine) of x, in radians.
- **math.atan(x)**: Returns the arc tangent (inverse tangent) of x, in radians.
- **math.degrees(x)**: Converts angle x from radians to degrees.
- **math.radians(x)**: Converts angle x from degrees to radians.
- **math.ceil(x)**: Returns the smallest integer greater than or equal to x.
- **math.floor(x)**: Returns the largest integer less than or equal to x.
- **math.trunc(x)**: Truncates x to an integer, removing the fractional part.
- **math.fabs(x)**: Returns the absolute value of x (as a float).
- **math.factorial(x)**: Returns the factorial of x.
- **math.gcd(a, b)**: Returns the greatest common divisor of a and b.
- **math.isqrt(x)**: Returns the integer square root of x.
- **math.comb(n, k)**: Returns the number of ways to choose k items from n items without repetition (combinations).
- **math.perm(n, k)**: Returns the number of ways to arrange k items from n items (permutations).
- **math.copysign(x, y)**: Returns a float with the magnitude of x and the sign of y.
- **math.isfinite(x)**: Returns True if x is neither infinity nor NaN.
- **math.isinf(x)**: Returns True if x is infinity.
- **math.isnan(x)**: Returns True if x is NaN.
- **math.modf(x)**: Returns the fractional and integer parts of x as a tuple.
- **math.hypot(x, y)**: Returns the Euclidean norm, sqrt(x*x + y*y).

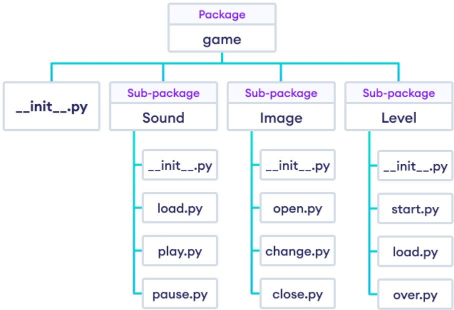
- **math.dist(p, q)**: Returns the Euclidean distance between points p and q, each given as a sequence (like a tuple).
- **math.fsum(iterable)**: Returns an accurate floating point sum of values in the iterable, avoiding intermediate rounding errors.
- **math.prod(iterable)**: Returns the product of all values in the iterable.

Packages

A package in Python is a way of organizing related modules into a directory hierarchy. Packages help you organize your code more logically and manage namespaces effectively, especially in larger projects. A directory that contains multiple modules and an optional special file __init__.py.

A typical package:

```
mypackage/
│
├── __init__.py
├── module1.py
└── module2.py
```



Creating a package:

```
geometry/
│
├── __init__.py
├── calculations.py
└── area.py
```

```
calculations.py
```

```
def add(a, b):
    return a + b


def subtract(a, b):
    return a - b

def multiply(a, b):
    return a * b

def divide(a, b):
    try:
        return a / b
    except ZeroDivisionError:
        return "Denominator cannot be zero"
```

area.py

```
import math

DEFAULT_RADIUS = 1

def area_of_circle(radius):
    return math.pi * radius ** 2

def area_of_rectangle(length, breath):
    return length * breath
```

example.py

```
from geometry.area import area_of_circle, DEFAULT_RADIUS
from geometry.calculations import add

a = 10
b = 20
radius = 2.4
print(f"Sum = {add(a, b)}")
print(f"Area of circle = {area_of_circle(radius)}")
print(f"Area of circle = {area_of_circle(radius=DEFAULT_RADIUS)}")
```

Library:

A library is a collection of modules that provides a set of functions, classes, and tools to perform a specific set of tasks or solve particular problems. Libraries can be thought of as large packages or sets of packages that offer a broader range of functionality.

Third-Party vs. Standard Libraries:

- Standard Libraries: Python comes with a set of built-in libraries as part of the Python Standard Library. Examples include math, datetime, os, and json.
- Third-Party Libraries: These are developed and maintained by the Python community or organizations and are not included in the standard library. Examples include numpy, pandas, requests, and Django.

2. The standard library and library functions

The Python Standard Library is a collection of modules and packages that are included with Python and provide a wide range of functionality. These modules and packages cover various programming tasks, such as file I/O, system operations, data manipulation, and more.

No need to install standard library separately; it comes bundled with the Python installation.

Here are some commonly used modules in the Python Standard Library along with their key functions:

1. math
   a. math.sqrt(x): Returns the square root of x.
   b. math.factorial(x): Returns the factorial of x.
   c. math.pi: Constant for the value of π.
2. datetime
   a. datetime.datetime.now(): Returns the current local date and time.
   b. datetime.date.today(): Returns the current date.
   c. datetime.timedelta(days=5): Represents a duration of 5 days.
3. os
   a. os.listdir(path): Returns a list of files and directories in path.
   b. os.mkdir(path): Creates a directory named path.
   c. os.path.join(a, b): Joins two or more pathname components.

```
from datetime import datetime, timedelta, timezone

print(datetime.now())
dt = datetime(2024, 8, 12, 14, 30, 0)
future_date = dt + timedelta(days= 5)
print(future_date)
utc_now = datetime.now(timezone.utc)
print(utc_now)
print(utc_now.astimezone())
```

```
import os

cur_dir = os.getcwd()
print(cur_dir)
files = os.listdir('.')
print(files)
get_file = os.path.exists('myfile.txt')
print(get_file)
path = os.path.join('folder', 'subfolder', 'file.txt')
print(path)
```

The Python Standard Library includes over 200 modules and contains thousands of functions across its modules.

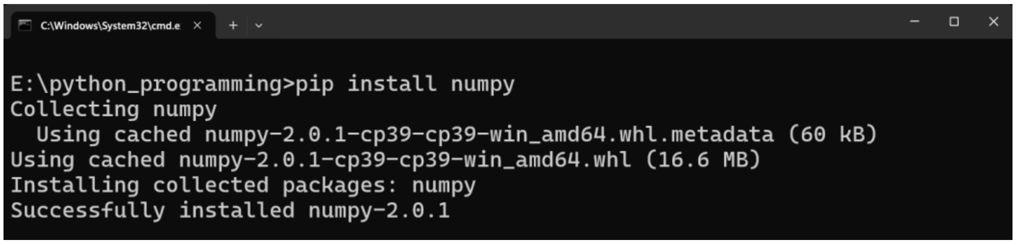
3. Adding more python libraries

To expand your Python environment with additional libraries beyond the standard library, you can use the Python Package Index (PyPI), which hosts a vast collection of third-party libraries.

Using pip to Install Libraries

```
Syntax:
        pip install library_name
```

Example: pip install numpy



```
E:\python_programming>pip install numpy
Collecting numpy
  Using cached numpy-2.0.1-cp39-cp39-win_amd64.whl.metadata (60 kB)
Using cached numpy-2.0.1-cp39-cp39-win_amd64.whl (16.6 MB)
Installing collected packages: numpy
Successfully installed numpy-2.0.1
```

4. Numpy Library

NumPy (Numerical Python) is an open-source Python library that’s widely used in **science and engineering**. The NumPy library contains multidimensional array data structures, such as the homogeneous, N-dimensional ndarray, and a large library of functions that operate efficiently on these data structures.

Creating, Indexing, and Slicing NumPy Arrays

```
import numpy as np


array1 = np.array([1, 2, 3, 4, 5])
print(array1)
array2 = np.zeros((2, 3))
print(array2)
array3 = np.ones((3, 2))
print(array3)
array4 = np.arange(10)
print(array4)
array5 = np.array([[1, 2, 3],
                   [4, 5, 6]])
print(array5)
print(array5.shape)
print(array5.ndim)
print(array5.size)
print(array5.dtype)
```

Creating NumPy Arrays

Creating arrays with NumPy involves using functions like np.array(), np.zeros(), np.ones(), and np.arange()

```
#indexing and slicing
import numpy as np
array = np.array([10, 20, 30, 40, 50])
print(array[4])
matrix = np.array([[1, 2, 3],
                   [4, 5, 6]])
print(matrix[1, 2])
print(array[0:4])
print(matrix[0:2, 1:3])
```

Copying and Editing NumPy Arrays

```
import numpy as np

array = np.array([1, 2, 3, 4, 5])
array_copy_1 = array            #shallow
array_copy_2 = array.copy()     #deep
print(array_copy_1)
print(array_copy_2)
array[2] = 20
print(array_copy_1)
print(array_copy_2)
```

Stacking and Restructuring NumPy Arrays

Combining multiple arrays into a single array using functions like np.vstack() and np.hstack(). Changing the shape of an array using np.reshape().

```
import numpy as np
a = np.array([[1, 2], [3, 4], [5, 6]])
b = np.array([[7, 8], [9, 10], [11, 12]])
```

```
v_stack = np.vstack((a, b))
print(v_stack)

h_stack = np.hstack((a, b))
print(h_stack)
print(a)
reshaped = a.reshape((2, 3))
print(reshaped)
```

Q. Convert 1D array to 2D Array

Arithmetic Operations with NumPy Arrays

```
import numpy as np
a = np.array([[1, 2],
              [3, 4]])
b = np.array([[7, 8],
              [9, 10]])

print(a + b)
print(a * b)
print(a * 2)
```

Operations with NumPy Arrays of Different Shapes

```
import numpy as np
a = np.array([[1, 2],
              [3, 4],
              [5, 8]])
b = np.array([[7],
              [8],
              [9]])
print(a + b)       # Broadcasting addition
```

Concatenation, Reversion, and Persistence of NumPy Arrays

```
import numpy as np

a = np.array([1, 2])
b = np.array([3, 4, 8])
concat = np.concatenate((a, b))
print(concat)
reversed_array = a[::-1]
print(reversed_array)

np.save('array.npy', b)
loaded_array = np.load('array.npy')
print(loaded_array)
```

Applications of numPy Random number generation

```
import numpy as np

random_numbers = np.random.random(5)
print(random_numbers)

random_integers = np.random.randint(1, 10, size=5)
print(random_integers)
```

Applications of numPy Statistics

```
import numpy as np

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

mean = np.mean(array)
print(f"Mean: {mean}")
```

```

std_dev = np.std(array)
print(f"Standard Deviation: {std_dev}")

median = np.median(array)
print(f"Median: {median}")
```

Applications of numPy Linear algebra

```
'''Consider the following system of linear equations:
                    2x+3y=13
                    4x+9y=30
    A system of linear equations can be represented in matrix form as:
                    A·X=B'''

import numpy as np
a = np.array([[2, 3],
                      [4, 9]])
b = np.array([13 , 30])
x = np.linalg.solve(a, b)
print(f'Solutions = {x}')
```