Modules and Packages

1. Python Modules

- A **module** in Python is just a file that ends with `.py`.
- It can have **functions**, **variables**, or **classes** that you want to use in other programs.

How to Use a Module

You use the `import` keyword to bring in (import) a module into your code, like

```
import math
```

Built-in Modules

Python comes with many **built-in modules** (like `math`, `random`, etc.).
You can check the full list in the **Python Standard Library**.

What Happens When You Import a Module?

- The **first time** you import a module, Python **runs the code inside it once**.
- If you **import it again** somewhere else in your program, Python **does not run it again**.
- This means the module is like a **singleton** – it gets loaded just once, and its variables are shared wherever you use it.

If we want to import module math, we simply import the module:

In [1]:
```
# import the library
import numpy
import math
```

In [2]:
```
# use it (ceiling rounding)
math.ceil(2.4)
```

Out[2]: 3

Exploring built-in modules

While exploring modules in Python, two important functions come in handy - the dir and help functions. dir functions show which functions are implemented in each module. Let us see the below example and understand better.

In [3]:
```
print(dir(math))
```

```
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'a
sin', 'asinh', 'atan', 'atan2', 'atanh', 'cbrt', 'ceil', 'comb', 'copysign', 'cos',
'cosh', 'degrees', 'dist', 'e', 'erf', 'erfc', 'exp', 'exp2', 'expm1', 'fabs', 'fact
orial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose',
'isfinite', 'isinf', 'isnan', 'isqrt', 'lcm', 'ldexp', 'lgamma', 'log', 'log10', 'lo
g1p', 'log2', 'modf', 'nan', 'nextafter', 'perm', 'pi', 'pow', 'prod', 'radians', 'r
emainder', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc', 'ulp']
```

When we find the function in the module we want to use, we can read about it more using the help function, inside the Python interpreter:

In [4]:
```
help(math.ceil)
```

```
Help on built-in function ceil in module math:

ceil(x, /)
    Return the ceiling of x as an Integral.

    This is the smallest integer >= x.
```

How to Write our own modules ?

Writing Python modules is very simple. To create a module of your own, simply create a new .py file with the module name, and then import it using the Python file name (without the .py extension) using the import command.

---

2. Python Packages

Packages are namespaces which contain multiple packages and modules themselves. They are simply directories, but with a twist.

The twist is, each package in Python is a directory which MUST contain a special file called **`__init__.py`**. This file can be empty, and it indicates that the directory it contains is a Python package, so it can be imported the same way a module can be imported.

If we create a directory called foo, which marks the package name, we can then create a module inside that package called bar. We also must not forget to add the **`__init__.py`** file inside the foo directory.

To use the module bar, we can import it in two ways:

In [ ]:
```
```

In [ ]:
```
# Just an example, this won't work
import foo.bar
```

In [ ]:
```
# OR could do it this way
from foo import bar
```

In the first method, we must use the foo prefix whenever we access the module bar. In the second method, we don't, because we import the module to our module's namespace.

The **`__init__.py`** file can also decide which modules the package exports as the API, while keeping other modules internal, by overriding the **`__all__`** variable, like so:

In [ ]:
```
__init__.py:

__all__ = ["bar"]
```