Modules

To extend the base functionality of Python, extension modules can be imported. Many useful modules are delivered with the standard package, but many of those we will use have to be installed from external sources, like for example pyTorch, TensorFlow, numpy, scikit-learn and so on, which we will use at a later point. The preferred way to do this is to use a package manager. Starting with native modules, `math` offers many useful tools:

In [3]:

```python
import math

print(math.sin(math.pi))
print(math.cos(math.pi))
print(math.exp(0.7))
```

```
1.2246467991473532e-16
-1.0
2.0137527074704766
```

Sometimes directly importing modules clutters the workspace. If you only need a small subset of the functions, classes, or constants provided by a module, you can restrict the import to those:

In [2]:

```python
from math import sin, cos, exp, pi

print(sin(pi))
print(cos(pi))
print(exp(0.7))
```

```
1.2246467991473532e-16
-1.0
2.0137527074704766
```

Much nicer. In many cases, you do need a lot of different functions from a module and sometimes modules have long names. The workaround to this is to give an imported module a shorter alias. This is the preferred method of importing modules. It also helps avoiding masking issues, where two functions from different packages have the same name and in the end you don't know which of them is used in your script. An example would be the sine functions from `math` and `numpy`.

In [4]:

```python
import math as m
import numpy as np

print(m.sin(0))
print(np.sin(0))
```

```
0.0
0.0
```