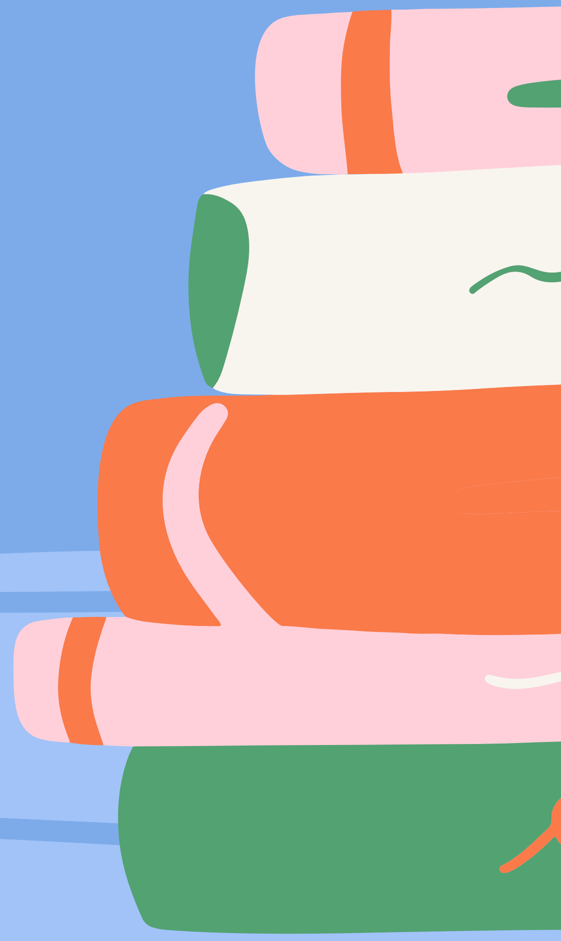
CODE OS

DEVELOPED BY STUDENTS

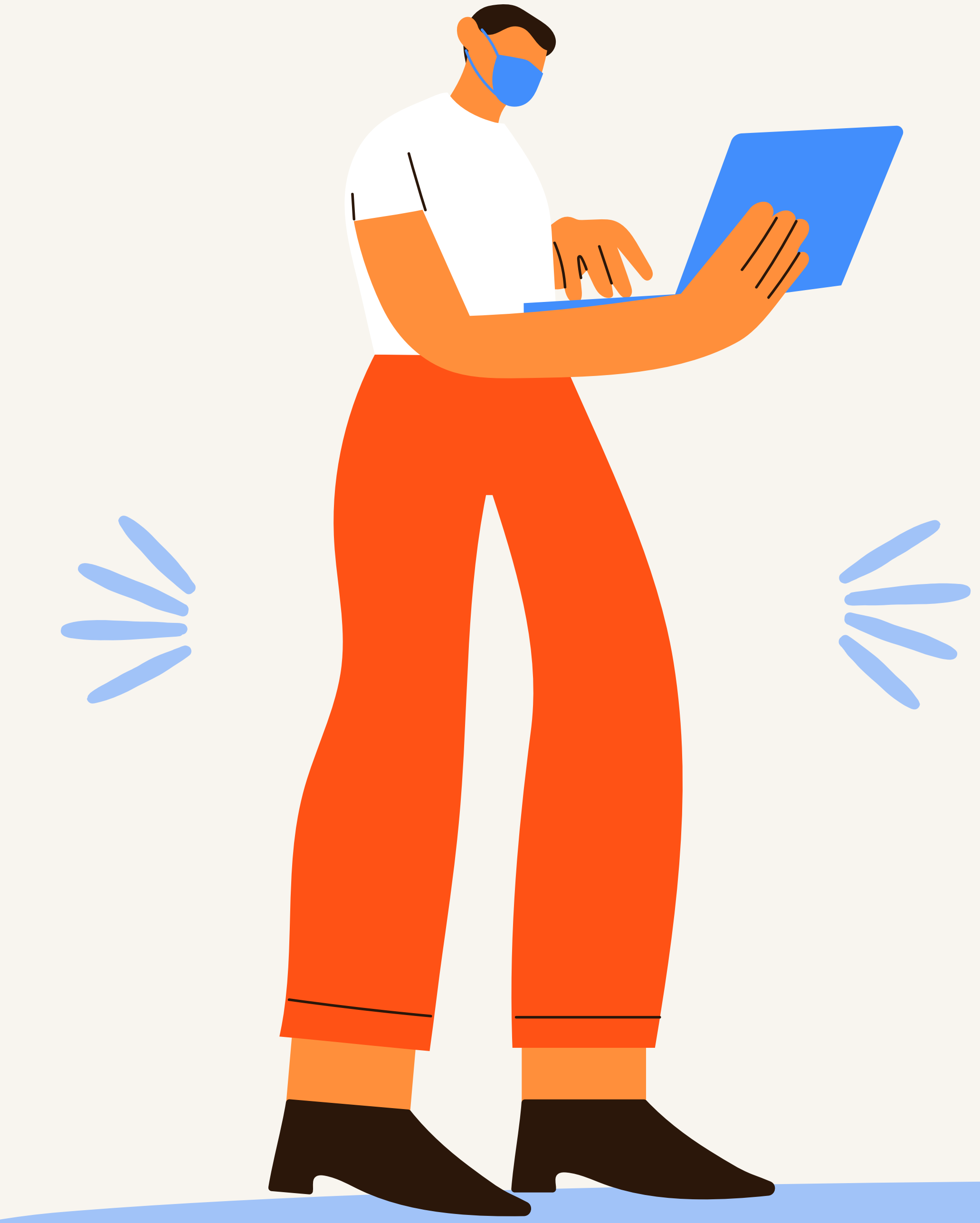
Modules in Python

6 days Python Bootcamp



Modules!

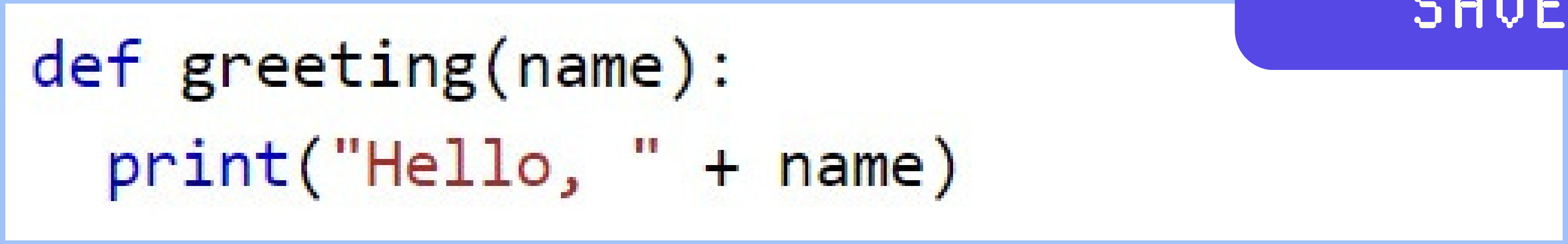
Modules are simply files with the “. py” extension containing Python code that can be imported inside another Python Program. In simple terms, we can consider a module to be the same as a code library or a file that contains a set of functions that you want to include in your application.



CAN WE CREATE MODULES?

Of course we can even create a new world to live in!

To create a module just save the code you want in a file with the file extension .py



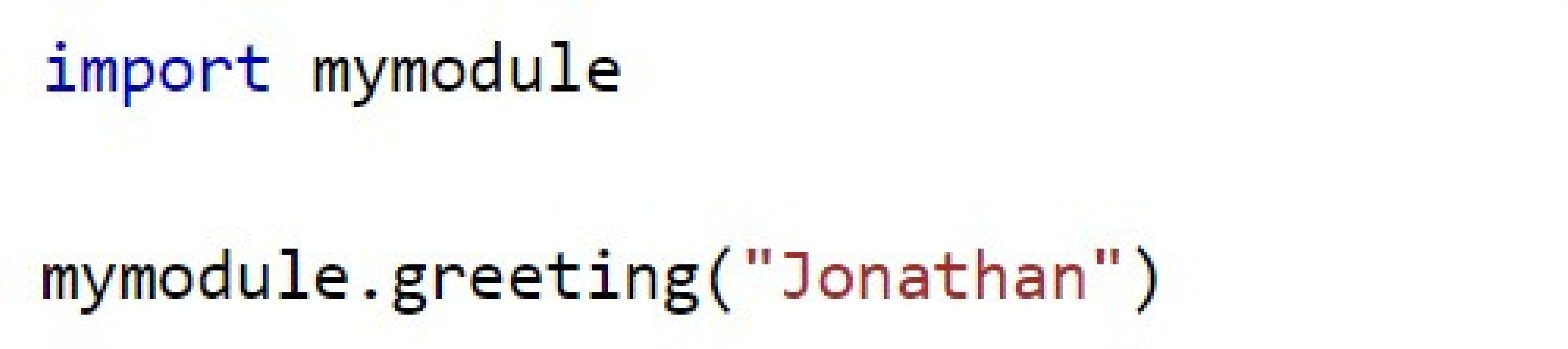
```
def greeting(name):
  print("Hello, " + name)
```

SAVE

mymodule.py

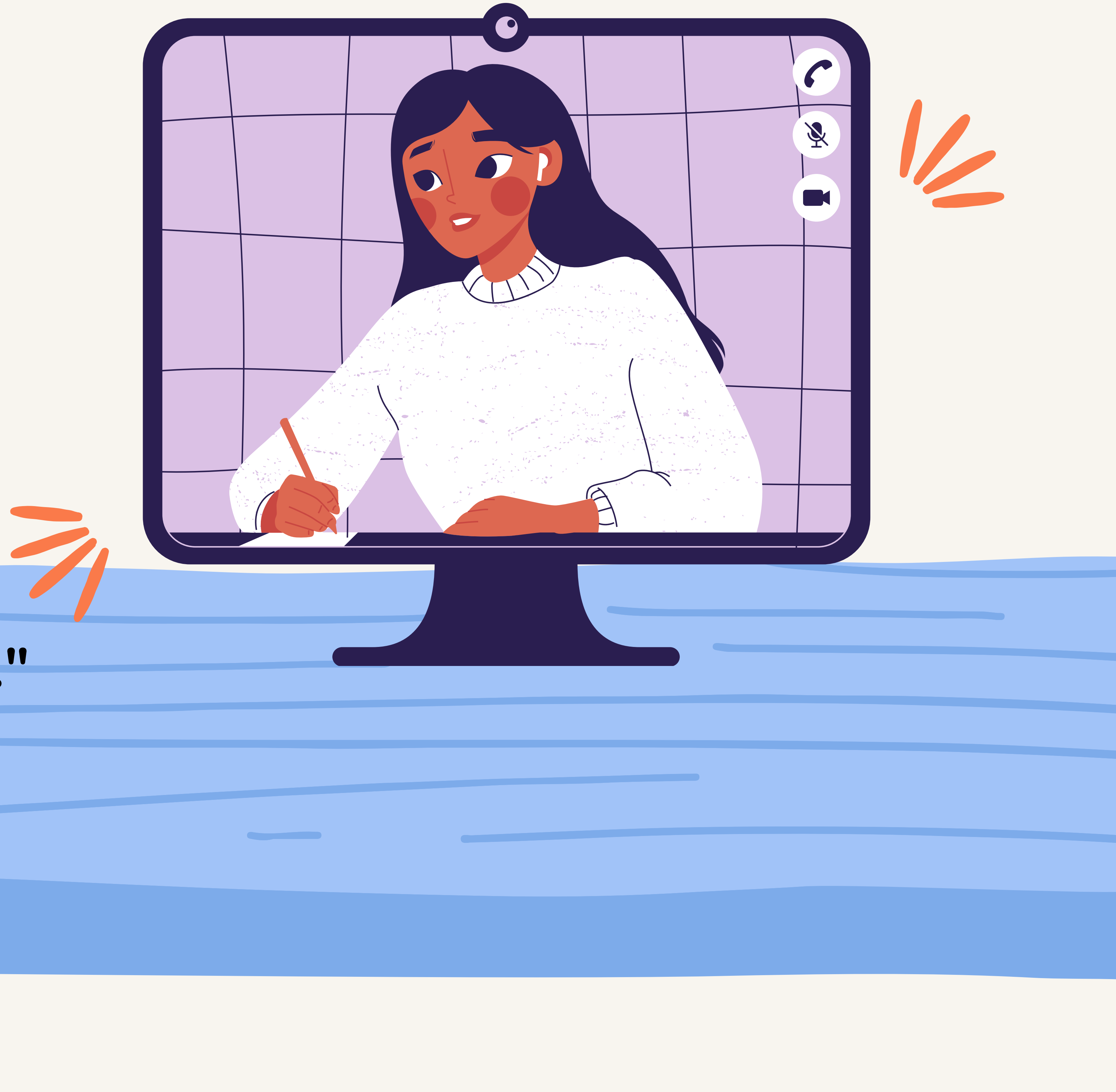
Using a Module

We can use a module using "import" statement.



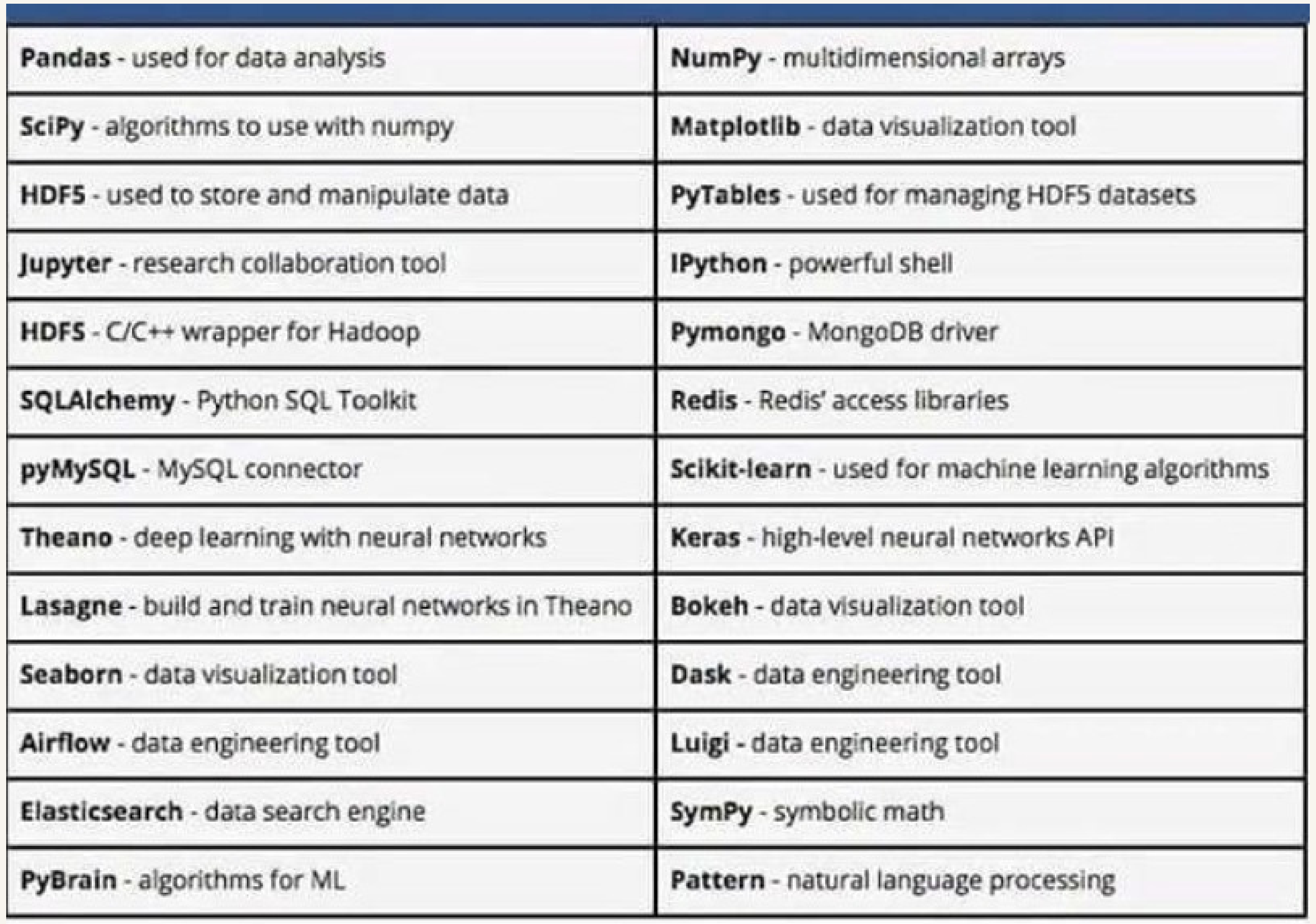
```
import mymodule

mymodule.greeting("Jonathan")
```

Then use the module anywhere you want using "." operator.



CODE OS
DEVELOPED BY STUDENTS

SOME EXAMPLES OF INBUILT MODULES!



Pandas - used for data analysis	**NumPy** - multidimensional arrays
SciPy - algorithms to use with numpy	**Matplotlib** - data visualization tool
HDF5 - used to store and manipulate data	**PyTables** - used for managing HDF5 datasets
Jupyter - research collaboration tool	**IPython** - powerful shell
HDFS - C/C++ wrapper for Hadoop	**Pymongo** - MongoDB driver
SQLAlchemy - Python SQL Toolkit	**Redis** - Redis' access libraries
pyMySQL - MySQL connector	**Scikit-learn** - used for machine learning algorithms
Theano - deep learning with neural networks	**Keras** - high-level neural networks API
Lasagne - build and train neural networks in Theano	**Bokeh** - data visualization tool
Seaborn - data visualization tool	**Dask** - data engineering tool
Airflow - data engineering tool	**Luigi** - data engineering tool
Elasticsearch - data search engine	**SymPy** - symbolic math
PyBrain - algorithms for ML	**Pattern** - natural language processing

PYTHON FROM...IMPORT STATEMENT

```
from math import pi,e
print("The value of pi is", pi)
```

we imported only the pi and attribute from the math module.

```
from math import *
print("The value of pi is", pi)
```

we have imported all the definitions from the math module. This includes all names visible in our scope except those beginning with an underscore(private definitions)

DIR() FUNCTION

Used to list in all the function names in the module

```
import math
print(dir(math))
```

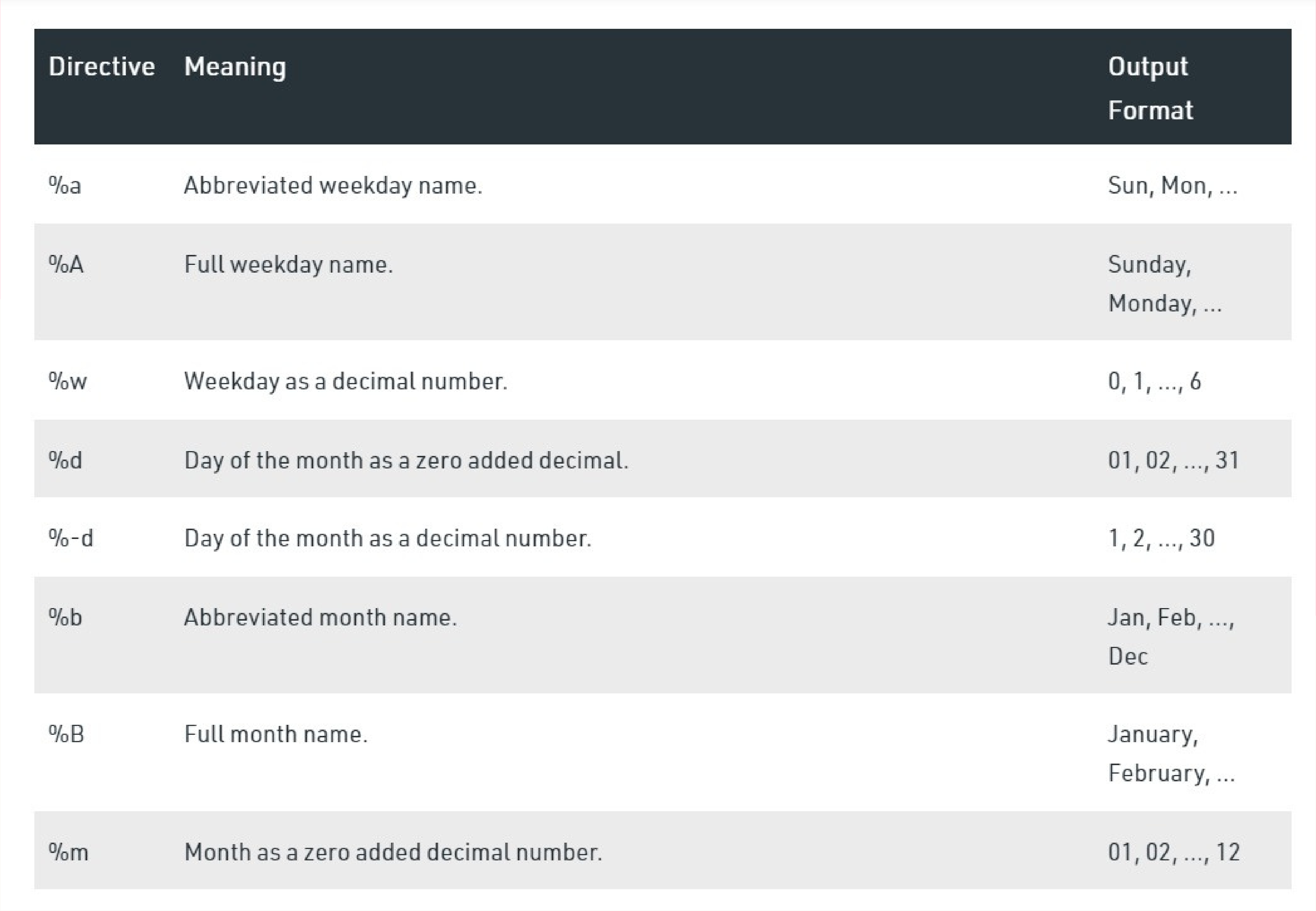
O/P

```
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin',
'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'comb', 'copysign', 'cos', 'cosh', 'degrees', 'dist',
'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma',
'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'isqrt', 'ldexp', 'lgamma', 'log',
'log10', 'log1p', 'log2', 'modf', 'nan', 'perm', 'pi', 'pow', 'prod', 'radians', 'remainder',
'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc']
```

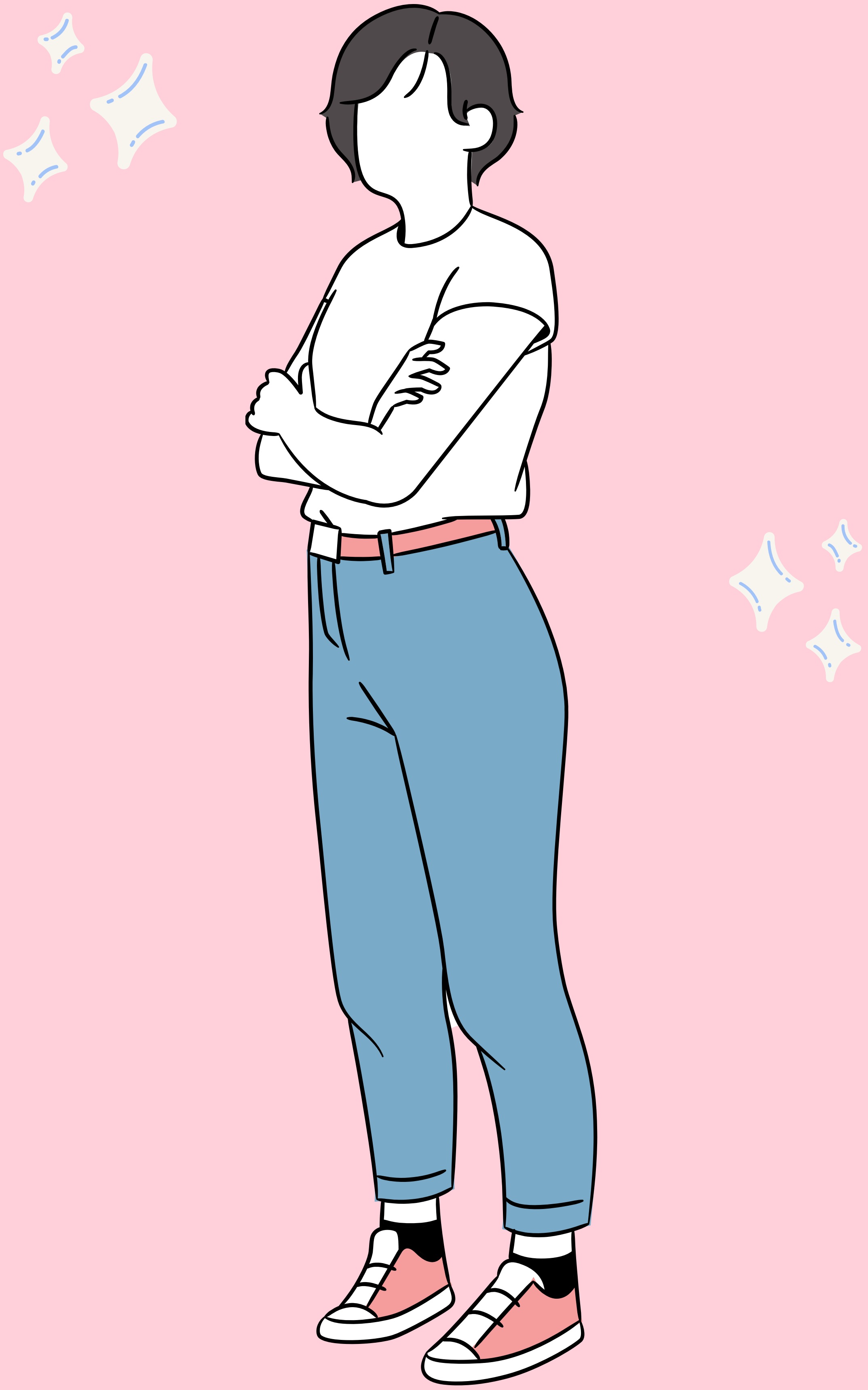
EXAMPLE

datetime module

Datetime module supplies classes to work with date and time. These classes provide a number of functions to deal with dates, times and time intervals.

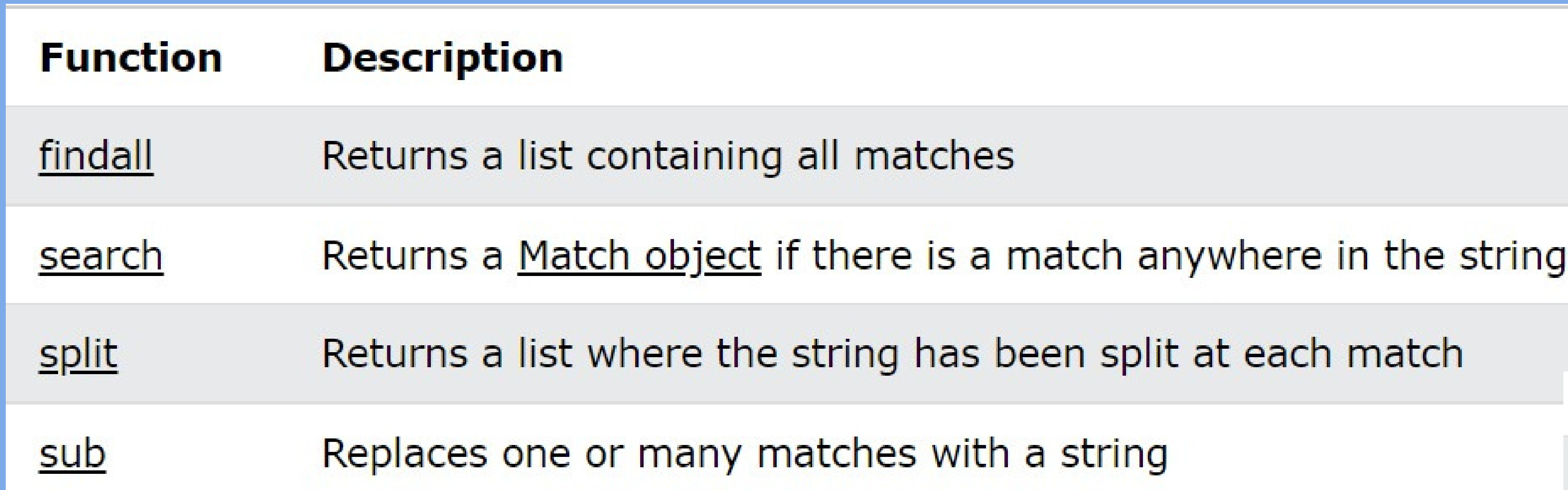


Directive	Meaning	Output Format
%a	Abbreviated weekday name.	Sun, Mon, ...
%A	Full weekday name.	Sunday, Monday, ...
%w	Weekday as a decimal number.	0, 1, ..., 6
%d	Day of the month as a zero added decimal.	01, 02, ..., 31
%-d	Day of the month as a decimal number.	1, 2, ..., 30
%b	Abbreviated month name.	Jan, Feb, ..., Dec
%B	Full month name.	January, February, ...
%m	Month as a zero added decimal number.	01, 02, ..., 12

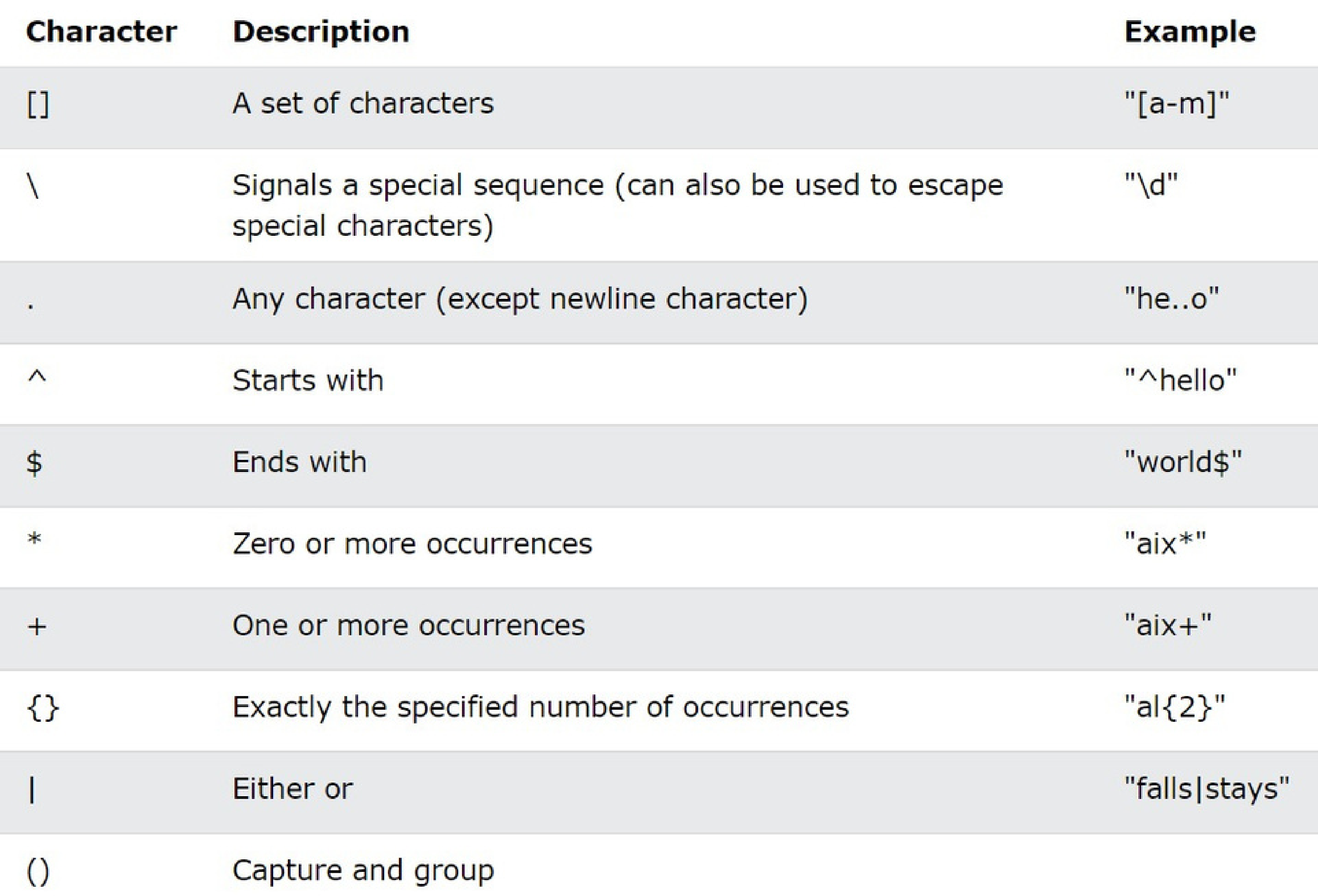


EXAMPLE

RegEx Module



Function	Description
findall	Returns a list containing all matches
search	Returns a Match object if there is a match anywhere in the string
split	Returns a list where the string has been split at each match
sub	Replaces one or many matches with a string



Character	Description	Example
[]	A set of characters	"[a-m]"
\	Signals a special sequence (can also be used to escape special characters)	"\d"
.	Any character (except newline character)	"he..o"
^	Starts with	"^hello"
$	Ends with	"world$"
*	Zero or more occurrences	"aix*"
+	One or more occurrences	"aix+"
{}	Exactly the specified number of occurrences	"al{2}"
|	Either or	"falls|stays"
()	Capture and group	