DataScientest • com

Object Oriented Programming

Predefined classes

In Python, many predefined classes such as the `list`, `tuple` or `str` classes are regularly used to facilitate the developer's tasks. Like all other classes, they have their own attributes and methods that are available to the user.

One of the great interests of object oriented programming is to be able to create classes and share them with other developers. This is done through packages such as `numpy`, `pandas` or `scikit-learn`. All of these packages are actually classes created by other developers in the Python community to give us tools that will make easier to develop our own algorithms.

We will first discuss one of the most important predefined object classes, the `list` class, in order to learn how to use it to its full potential.
Next, we will briefly introduce the `DataFrame` class of the `pandas` package and learn to identify and manipulate its methods.

1. The list class

- **(a)** Use the `dir(list)` command to display all attributes and methods of the `list` class.

In [1]:
```
dir(list)
```

Out[1]:
```
['__add__',
 '__class__',
 '__class_getitem__',
 '__contains__',
 '__delattr__',
 '__delitem__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__gt__',
 '__hash__',
 '__iadd__',
 '__imul__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__mul__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__reversed__',
 '__rmul__',
 '__setattr__',
 '__setitem__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'append',
 'clear',
 'copy',
 'count',
 'extend',
 'index',
 'insert',
 'pop',
 'remove',
 'reverse',
 'sort']
```

- **(b)** Use the `help(list)` command to display the *documentation* of the `list` class. This documentation is useful to understand how to use the methods of a class.

In [1]:
```
help(list)
```

```
Help on class list in module builtins:

class list(object)
 |  list(iterable=(), /)
 |
 |  Built-in mutable sequence.
 |
 |  If no argument is given, the constructor creates a new empty list.
 |  The argument must be an iterable if specified.
 |
 |  Methods defined here:
 |
 |  __add__(self, value, /)
 |      Return self+value.
 |
 |  __contains__(self, key, /)
 |      Return key in self.
 |
 |  __delitem__(self, key, /)
```

> The `dir` and `help` commands are the first commands to run when you don't understand how to use a method of a class or when you can't remember the name of a method.

- **(c)** Using the `dir` or `help` commands, find a method that will reverse the order of the elements of the list `list_1` .

In [2]:
```
list_1 = [1, 2, 3, 4, 5, 6, 7, 8, 9]

### Insert your code here

list_1.reverse()
list_1
```

Out[2]: [9, 8, 7, 6, 5, 4, 3, 2, 1]

Hide solution

In [4]:
```
list_1 = [1, 2, 3, 4, 5, 6, 7, 8, 9]

list_1.reverse() # reverses the order of the elements of the list calling it.
                 # this method WILL MODIFY the list that calls it.

list_1
```

Out[4]: [9, 8, 7, 6, 5, 4, 3, 2, 1]

- **(d)** Using the `dir` and `help` commands, find a method that will insert the value `10` in the fifth position of the list `list_2` .

In [5]:
```
list_2 = [1, 2, 3, 4, 5, 6, 7, 8, 9]

### Insert your code here


list_2.insert(4, 10)
list_2
```

Out[5]: [1, 2, 3, 4, 10, 5, 6, 7, 8, 9]

Hide solution

In [6]:
```
list_2 = [1, 2, 3, 4, 5, 6, 7, 8, 9]

list_2.insert(4, 10) # inserts the value 10 at the index 4 (fifth position in Python) of the list.

list_2
```

Out[6]: [1, 2, 3, 4, 10, 5, 6, 7, 8, 9]

- **(e)** Using the `dir` and `help` commands, find a method that will sort the list `list_3` .

In [7]:
```
list_3 = [5, 2, 4, 9, 6, 7, 8, 3, 10, 1]

### Insert your code here

list_3.sort()
list_3
```

Out[7]: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Show solution

2. The DataFrame class

> The `pandas` package contains a class named `DataFrame` whose usefulness makes it the most used package by data scientists to manipulate data.
>
> To use the `pandas` package, you must first import it. Then, to instantiate a `DataFrame` , you must call its constructor defined in the `pandas` package.

- **(a)** Import the `pandas` package under the alias `pd` .

In [8]:
```
### Insert your code here

import pandas as pd

df = pd.DataFrame()
df
```

Out[8]:

—

Hide solution

In [ ]:
```
import pandas as pd

df = pd.DataFrame()
```

> If you run the `dir(df)` or `dir(pd.DataFrame)` instructions, you will see that the `DataFrame` class has a lot of methods and attributes. It is very difficult to remember them all, hence the usefulness of the commands `dir` and `help` .
>
> However, given the length of the documentation, it is not practical to directly use the `dir(df)` or `help (df)` commands. To have direct access to the documentation of a specific method, you can instead use the `help` function with the argument `object.method` .

- **(c)** Using the `help(pd.DataFrame)` command, build a `DataFrame` named `df1` using the list `list_4` .

In [9]:
```
help(pd.DataFrame)
```

```
Help on class DataFrame in module pandas.core.frame:

class DataFrame(pandas.core.generic.NDFrame, pandas.core.arraylike.OpsMixin)
 |  DataFrame(data=None, index: 'Axes | None' = None, columns: 'Axes | None' = None, dtype: 'Dtype | None' = None, copy: 'bool | None' = None)
 |
 |  Two-dimensional, size-mutable, potentially heterogeneous tabular data.
 |
 |  Data structure also contains labeled axes (rows and columns).
 |  Arithmetic operations align on both row and column labels. Can be
 |  thought of as a dict-like container for Series objects. The primary
 |  pandas data structure.
 |
 |  Parameters
 |  ----------
 |  data : ndarray (structured or homogeneous), Iterable, dict, or DataFrame
 |      Dict can contain Series, arrays, constants, dataclass or list-like objects. If
 |      data is a dict, column order follows insertion-order.
 |
 |      .. versionchanged:: 0.25.0
```

In [10]:
```
list_4 = [1, 5, 45, 42, None, 123, 4213 , None, 213]

### Insert your code here

df1 = pd.DataFrame({"col1" : list_4})
df1 = pd.DataFrame(list_4)
df1.head(2)
```

Out[10]:

	0
0	1.0
1	5.0

Hide solution

In [11]:
```
list_4 = [1, 5, 45, 42, None, 123, 4213, None, 213]

df1 = pd.DataFrame(data = list_4)

df1
```

Out[11]:

	0
0	1.0
1	5.0
2	45.0
3	42.0
4	NaN
5	123.0
6	4213.0
7	NaN
8	213.0

In [12]:
```
list_4.__contains__(1)
```

Out[12]: True

> By displaying the `DataFrame` `df1`, you can see that some of its values are assigned to `NaN`, which stands for *Not a Number*. In practice, this happens very often when we import a database that is unprocessed. The `DataFrame` class contains a very simple method to get rid of these missing values: the `dropna` method.

In [13]:
```
### Insert your code here

df2 = df1.dropna()
df2
```

Out[13]:

	0
0	1.0
1	5.0
2	45.0
3	42.0
5	123.0
6	4213.0
8	213.0

Hide solution

In [14]:
```
df2 = df1.dropna()
df2
```

Out[14]:

	0
0	1.0
1	5.0
2	45.0
3	42.0
5	123.0
6	4213.0
8	213.0

> Another method of the `DataFrame` class which is widely used is the `apply` method. This method allows you to apply a function passed as an argument to all the entries of the `DataFrame` calling the method.

- **(e)** Define a function named `divide2` which returns the division by 2 of a number passed as argument.
- **(f)** Create a `DataFrame` named `df3` which will contain the values of `df2` divided by 2.

In [15]:
```
# Insert your code here

def divide2(x):
    return x / 2

df3 = df2.apply(divide2)
df3
```

Out[15]:

	0
0	0.5
1	2.5
2	22.5
3	21.0
5	61.5
6	2106.5
8	106.5

Hide solution

In [16]:
```
def divide2(x):
    return x/2

df3 = df2.apply(divide2)  # applies the function divide2 to all entries of the DataFrame

df3
```

Out[16]:

	0
0	0.5
1	2.5
2	22.5
3	21.0
5	61.5
6	2106.5
8	106.5

The `DataFrame` class has many methods like `apply` or `dropna` that you will explore in more depth during your learning journey. The `list` class being too basic for the needs of data scientists, these methods make the `DataFrame` class the standard to manipulate data.

All the packages that you will be invited to use in your training will be handled as objects, i.e. you will first have to initialize an object of the class (`DataFrame` , `Scikit Model` ,

Validate