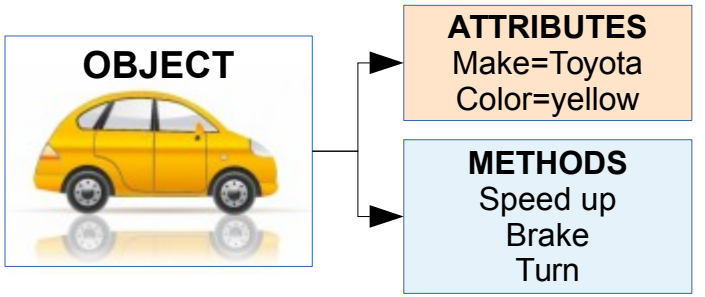
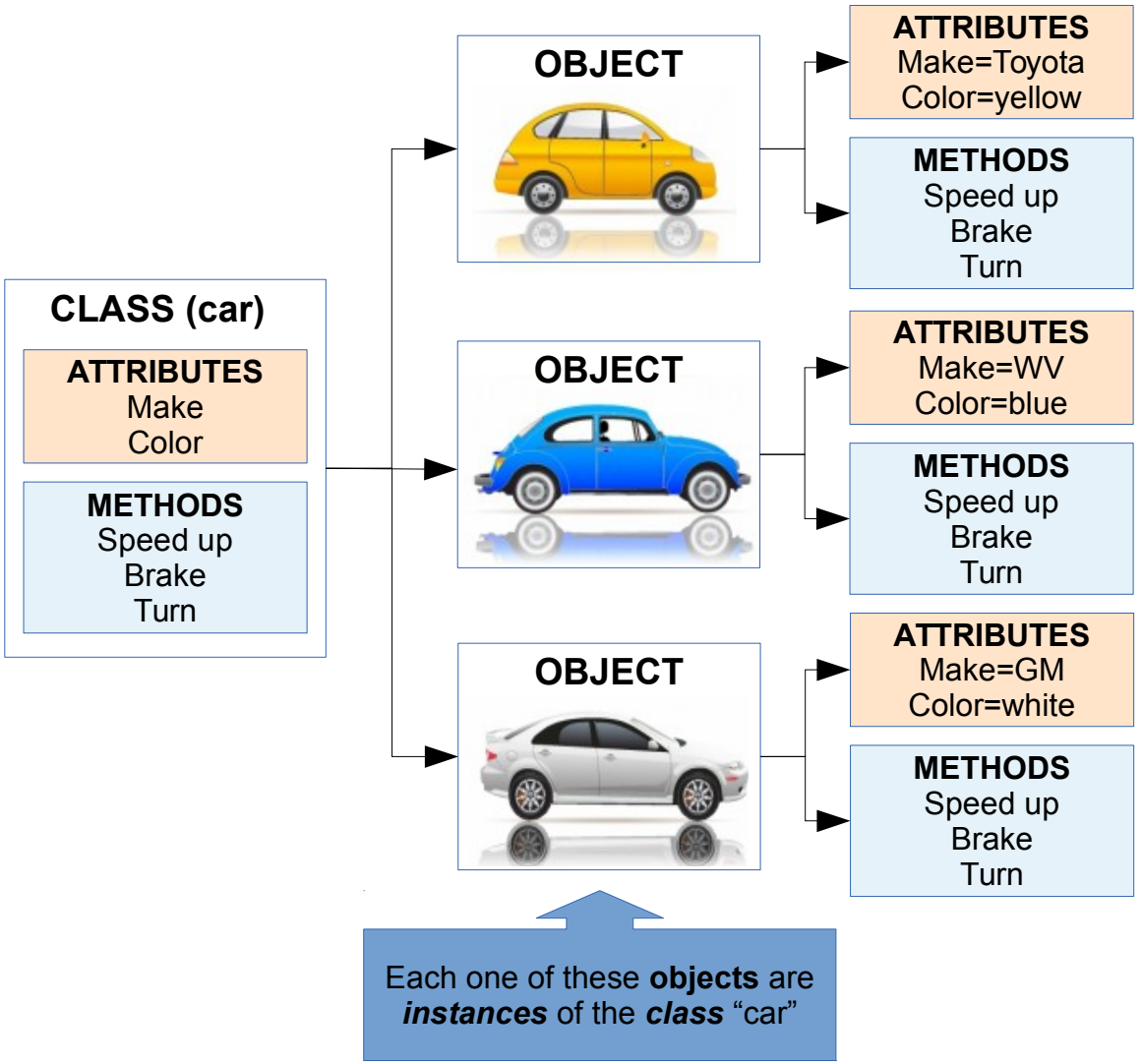
Objects

An ***object*** is just an entity (structure) that holds data and a set of related functionalities. The data is contained in variables called ***attributes*** and functionalities are encoded in the form of functions that are called ***methods***.





Objects are generated from a generic template called ***class***. The class defines what attributes and methods an object has. Thus a ***class*** is just a template (or blueprint) from which we can generate ***instances*** of that particular class.

In python, everything including data types (str,int,float,list,dictionary,…) and functions, are objects of a given class.

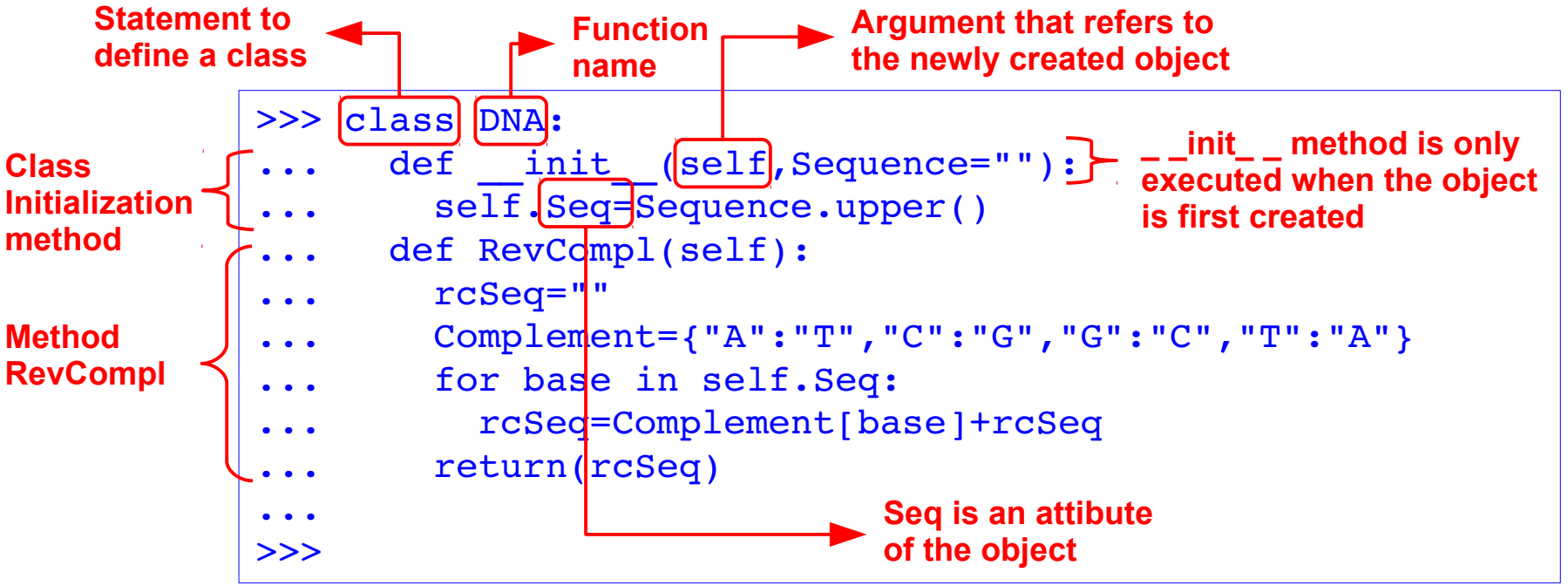
```
>>> MySeq="TCCAGAACCA"
>>> type(MySeq)
>>> dir(str)
>>> dir(MySeq)
```

Defining classes (I)

A ***class*** is a template that create ***instances*** of an object. The idea is to pack together attributes and methods that are useful for instancess of the class.

```
>>> MySeq1=DNA("ATGGCCTGA")
```

Functions must be defined before they can be invoked




```
>>> MySeq1=DNA("ATGGCCTGA")
```

Functions must be defined before they can be invoked

Unit6_Objects_example1.py

Defining classes (I)

Open a terminal move to the folder containning the examples for this lesson (with the command *cd*) and then invoke the python3 interpreter from that folder and run the following commands.

```
>>> MySeq1=DNA2('atggcctga')
>>> from Unit6_Objects_example2 import DNA2
>>> help(DNA2)
>>> MySeq1=DNA2('atggcctga')
>>> MySeq1
>>> MySeq1.Seq
>>> MySeq1.RevCompl()
>>> MySeq2='gcgtat'
>>> MySeq2.RevCompl()
>>> MySeq2=DNA2('gcgtat')
>>> MySeq2.RevCompl()
```

Press “q” to exit help

In a text editor (or in IDLE) open `Unit6_Objects_example2.py` and explore the code

Unit6_Objects_example2.py

Open `Unit6_Objects_example3.py` with IDLE and Unit6_Objects_class_definitions3 in a text editor

Unit6_Objects_example3.py

Defining classes (I)

```
>>> MySeq1=DNA2('atggcctga')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'DNA2' is not defined
>>> from Unit6_Objects_example2 import DNA2
>>> help(DNA2)
Help on class DNA2 in module Unit6_Objects_example2:
class DNA2(builtins.object)
 |  Generates an object of class DNA
 |
>>> MySeq1=DNA2('atggcctga')
>>> MySeq1
<Unit6_Objects_example2.DNA2 object at 0x7f7007b951d0>
>>> MySeq1.Seq
'ATGGCCTGA'
>>> MySeq1.RevCompl()
'TCAGGCCAT'
>>> MySeq2='gcgtat'
>>> MySeq2.RevCompl()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'str' object has no attribute
'RevCompl'
>>> MySeq2=DNA2('gcgtat')
>>> MySeq2.RevCompl()
'ATACGC'
```

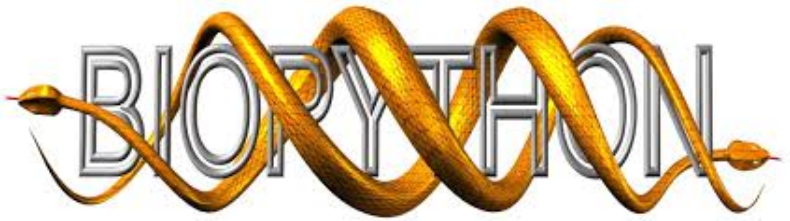
We can define a class in our script or import the class definitions from other scripts (modules)

We access to class attributes and methods with the *dot notation*:

class_name.attribute_name

class_name.method_name

Biopython



The Biopython Project is an international association of developers of freely available Python (http://www.python.org/) tools for computational molecular biology.

- The ability to parse bioinformatics files into Python utilizable data structures, including support for the following formats: Blast output, Clustalw, **FASTA**, GenBank, PubMed and Medline, ExPASy files, SCOP, UniGene, SwissProt
- Code to deal with popular on-line bioinformatics destinations such as: NCBI & ExPASy
- Interfaces to common bioinformatics programs such as:Standalone Blast from NCBI, Clustalw alignment program, EMBOSS command line tools
- A standard sequence class that deals with sequences, ids on sequences, and sequence features.
- **Tools for performing common operations on sequences, such as translation, transcription and weight calculations**.
- Code to perform classification of data using k Nearest Neighbors, Naive Bayes or Support Vector Machines.
- Code for dealing with alignments, including a standard way to create and deal with substitution matrices.
- Code making it easy to split up parallelizable tasks into separate processes.
- GUI-based programs to do basic sequence manipulations, translations, BLASTing, etc.
- Extensive documentation and help with using the modules, including this file, on-line wiki documentation, the web site, and the mailing list.
- Integration

Biopython examples

Biopython Tutorial:

- Section 3.7 (page 25):

```
>>> from Bio.Seq import Seq
>>> from Bio.Alphabet import IUPAC
>>> my_seq = Seq("GATCGATGGGCCTATATAGGATCGAAAATCGC", IUPAC.unambiguous_dna)
>>> my_seq
>>> my_seq.complement()
>>> my_seq.reverse_complement()
```

- Section 3.8 (page 26):

```
>>> coding_dna = Seq("ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG",
IUPAC.unambiguous_dna)
>>> messenger_rna = coding_dna.transcribe()
>>> messenger_rna
```

- Section 3.9 (page 27)

```
>>> messenger_rna.translate()
```

- Section 3.10 (page 29)

```
>>> from Bio.Data import CodonTable
>>> standard_table = CodonTable.unambiguous_dna_by_name["Standard"]
>>> print(standard_table)
```

Open `Unit6_Biopython_example1.py` with IDLE and run it

Unit6_Biopython_example1.py