

Classes and Iterators

October 7th, 2022 | Sandra Diaz Pier

Introduction to Classes

- In object oriented programming, classes represent objects with attributes (variables) and possible actions they can perform or you can perform on them (methods).
- Classes encapsulate functionality in well defined units with a purpose, scope and well defined interaction capabilities (via its methods)

Introduction to Classes

- A class is defined using the **class** keyword, and the class definition usually contains a number of class method definitions (a function in a class).
- Each class method must have an argument `self` as its first argument. This object is a self-reference.
- Special class methods:
 - `__init__`:
 - `__str__`
 - <http://docs.python.org/3/reference/datamodel.html#special-method-names>

Introduction to iterators

- Iterable objects are objects which can be accessed element-wise using an iterator
- They implement the `__iter__` method
- Examples of iterable objects: lists, strings, dictionaries, files, etc
- `for`, `while` use iterators to access elements in the iterable objects

Introduction to iterators

- To obtain an iterator from an object, one can use the **iter** function
- The **next** method is used to access the next element in the object

Generators

- Define a function but use *yield* instead of *return* statement
- *yield* pauses the function and saves the internal state so it can be resumed later on
- Saves space in memory but slower than list comprehensions
- Only runs when executed with *next()*
- Elements are not accessible anymore after yielding

References

- (1) Based on the work by J.R. Johansson <http://jrjohansson.github.io>
- (2) <http://anandology.com/python-practice-book/iterators.html>