Classes and methods

Wednesday, May 29, 2019 3:54 PM

Classes and methods

- Sometimes, there are functions that apply only to specific data.
- One can file them along with their input data in what is called a "class".
- Once a function is moved into a class, it is called a *method* instead.
- The syntax changes a bit when one changes a function int a method.

Example: the debts database.

```
class Debts:
    data = []
    def people(self):  # a class method
        out = set()
        for d in self.data:
            out.add(d[0])
            out.add(d[1])
        return sorted(list(out))
```

Several things to note:

- Data in the class are listed inside the declaration.
- Functions no longer need to list that data.
- Instead, they contain an argument self (not a reserved name, can be anything) that refers to the class.

- Dot notation refers to data inside a class.
  - `debts.data:` the raw data.
  - `debts.people()`: a method call.

Using this:

```
debts = Debts()
debts.data = [("Alva", "Frank", 10),
              ("Fred", "George", 3),
              ("Amy", "George", 2),
              ("Frank", "Fred", 4),
              ("Frank", "Amy", 5)]
print(debts.people())
```

Vocabulary:

`Debts` is a *class*.

`debts` is a *class instance*: an object satisfying the conditions for being in the class.

`debts.data` is *class data.*

`debts.people` is a class method.

Constructors

- There is a much neater way to encode this.
- The reserved method `__init__` can do the copying.
- This -- in turn -- means one can define a Debts object in one statement.

For example:

Inside `Debts`, write:

```
def __init__(self, data):
```

```
        self.data = data
```
Then outside `Debts`, one can write
```
debts = Debts(data)
```

Stringification:

- This ugly word refers to the idea of making a human-readable (string) presentation of an object.
- The class method `__str__` is supposed to convert the class instance to a string.
- This is what is printed whenever you try to print the class.
- Thus, if you write:
  ```
  def __str__(self):
      return "a list of debts"
  ```
  inside the class, then
  ```
  print(debts)
  ```
  outside the class will actually print:
  ```
  a list of debts
  ```
- Of course, there are much more useful things to print in response.