University of Toronto - Computer Science - Image

Creating Your Own Types

Reviewing Python's Modules and Built-in Types

Importing a module

The `math` module contains functions such as `math.ceil`, which returns the smallest integer that is equal than or greater to the function's argument. Importing `math` creates a variable called `math`. This creates a namespace, or a chunk of memory, that holds the information for all the functions and data variables in the `math` module.

Type `str`

Because `str` is built into Python, we already have a type named `str`. But, instead of a module, `str` refers to a class, and contains methods rather than functions. The notation for calling a `str` method is similar to the one used to call a function from the `math` module:

```
>>> math.ceil(3.2)
4
>>> str.count('syzygy', 'y')
3
```

The first argument to a method is always an instance of the class, so we also use this notation call the method:

```
>>> 'syzygy'.count('y')
3
```

Creating a Custom Class

The `str` class has lots of useful methods, however, it does not contain every possible method we might need. We will create a new class that customizes class `str`. Our class will have all the `str` methods, plus a new method that checks whether a string begins and ends with the same letter.

We start our class definition with `class WordplayStr(str)`. In this case, `WordplayStr` is the name of our new class, and the `(str)` indicates that our new class is based on the `str` class. `WordplayStr` is a *subclass* of `str`, which means that `WordplayStr` inherits all of the methods from `str`. Here is the completed class definition:

```
class WordplayStr(str):
    """A string that can report whether it has interesting properties."""


    def same_start_and_end(self):
        """ (WordplayStr) -> bool

        Precondition: len(self) != 0

        Return whether self starts and ends with the same letter.

        >>> s = WordplayStr('abracadabra')
        >>> s.same_start_and_end()
        True
        >>> s = WordplayStr('canoe')
        >>> s.same_start_and_end()
        False
        """

        return self[0] == self[-1]


if __name__ == '__main__':
    import doctest
    doctest.testmod()
```

Jennifer Campbell • Paul Gries
University of Toronto