Modules, Part 1

Info 206

Niall Keleher

21 September 2017

Today's Quiz: http://bit.ly/2yra2R5

Today's Outline

Modules

Modules

- Modules consist of *attributes*
- The `import` statement imports a module and all if its attributes
- Attributes, when imported, assume a place in the program's namespace
- The `from` statement imports specific attributes from a module
- *Packages* - A directory that contains a collection of modules. Packages are essentially directories that has a special empty file named `__init.py__`

Numpy

Numpy

- Import individual attributes

```
import numpy.matlib
```

- In this example `numpy` is the package and `matlib` is the module within the package `numpy`
- The package `numpy` has a empty file named `__init.py__` that tells Python that numpy is a package

Numpy

- Import individual attributes

```
import numpy.matlib
```

- In this example `numpy` is the package and `matlib` is the module within the package `numpy`
- The package `numpy` has a empty file named `__init.py__` that tells Python that numpy is a package
- Import entire package

```
import numpy
numpy.__version__
```

Numpy

- Import individual attributes

```
import numpy.matlib
```

- In this example `numpy` is the package and `matlib` is the module within the package `numpy`
- The package `numpy` has a empty file named `__init.py__` that tells Python that numpy is a package
- Import entire package

```
import numpy
numpy.__version__
```

- Using an alias import statement

```
import numpy as np
```

PyData

Numpy

An efficient n-dimensional array representation

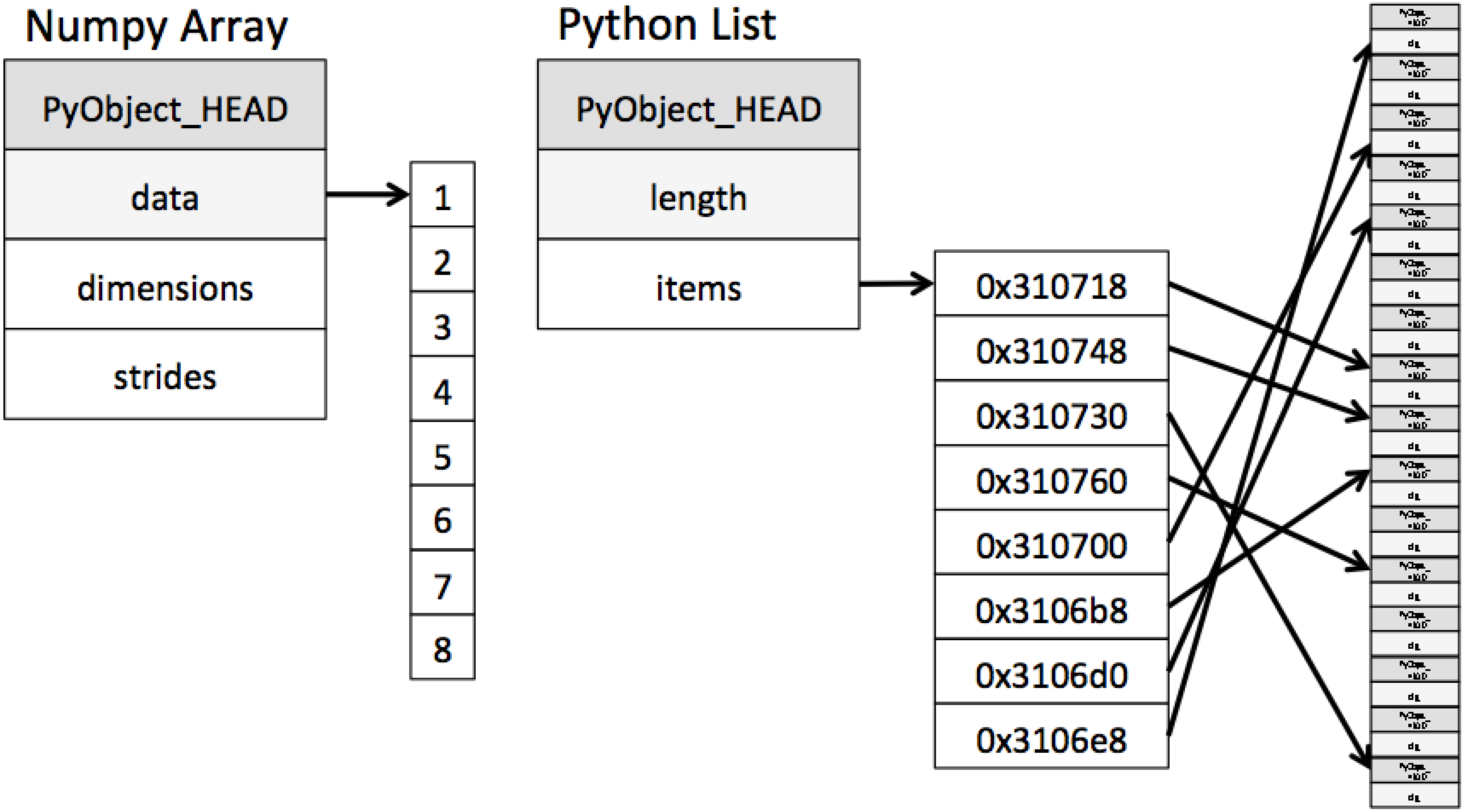
Object types and revisiting lists

```
L = list(range(10))
print(L)
print(type(L[0]))
```

Object types and revisiting lists

```
L = list(range(10))
print(L)
print(type(L[0]))
```

```
L2 = [True, "2", 3.0, 4]
print([type(item) for item in L3])
```



Numpy's efficiency

Creating arrays

- Python's built-in array package

```
import array
L = list(range(10))
A = array.array('i', L)
print(A)
```

- Numpy

```
A2 = np.array(L)
print(A2)
print(type(A2))
```

Nested multi-dimensional arrays

- Generated using a list comprehension

```
np.array([range(i, i + 3) for i in [2, 4, 6]])
```

NumPy Array Attributes

```python
import numpy as np
np.random.seed(0)  # seed for reproducibility

x1 = np.random.randint(10, size=6)  # One-dimensional array
x2 = np.random.randint(10, size=(3, 4))  # Two-dimensional array
x3 = np.random.randint(10, size=(3, 4, 5))  # Three-dimensional arra
```

NumPy Array Attributes

```python
import numpy as np
np.random.seed(0)  # seed for reproducibility

x1 = np.random.randint(10, size=6)  # One-dimensional array
x2 = np.random.randint(10, size=(3, 4))  # Two-dimensional array
x3 = np.random.randint(10, size=(3, 4, 5))  # Three-dimensional arra
```

```python
print("x3 ndim: ", x3.ndim)
print("x3 shape:", x3.shape)
print("x3 size: ", x3.size)
```

Array Indexing

- one-dimensional arrays

```
print(x1)
print(x1[0])
print(x1[4])
```

- multi-dimensional arrays

```
print(x2)
print(x2[0, 0])
print(x2[2, 0])
```

Array Slicing: Subarrays

- one-dimensional arrays

```
x = np.arange(10)
print(x)
print(x[:5])   # first five elements
```

- multi-dimensional arrays

```
print(x2[:2, :3]   # two rows, three columns)
```

Group Project

Design Document Guidelines

Team Meetings

Team I

Exercises

Meeting 9: Module Exercises

Exercises

- Instructions in the Github course-exercise repository
- Meeting 9 - Due at the end of the day on Thursday (Sept 28)

End of Meeting #9

For next meeting

- Videos: N/A
- Readings:
    - Lutz Chapter 24: Module Packages
    - Lutz Chapter 25: Advanced Module Topics s