CS 234
Naomi Nishimura

Python guide 2

1 Mutation and interning

Because we are concerned with the use of memory, at times it will be important to understand when Python does and doesn't adhere to the memory model used in class.

For example, although in principle each new data item is placed in a different part of memory, for efficiency Python makes of use **interning** of common strings (reusing the same memory for multiple strings). The function `id` can be used to determine the address of a data item, making it possible check when addresses are the same or different.

Relevant information includes:

- `id`

2 Classes

We will use classes extensively in the course. You should be able to create a class, create an object, and write and use methods.

Relevant information includes:

- `class`
- `__init__` to specify a function that creates an object
- `__repr__` to specify how an object is represented as a string (for tests)
- `__str__` to specify how an object is represented as a string (for printing)
- `__contains__` to define `in` for objects in this class
- `__eq__` to define == for objects in this class
- `self`
- `isinstance` to determine if an object is in a class
- `_` as the first symbol in the name of a private method so that it will not be imported by `from xx import *`

References

CS 116 notes

- Module 09: Additional Options for Organizing Data
  - Classes

Python from scratch

- Module 11: Building information into objects

Necaise

- Appendix D Classes

3 Strings

Strings will be used to create ways of printing data stored in ADTs; they also can be employed as an example of digital data.
Relevant information includes:

- `len` to determine length
- `[]` to access a particular character by position
- `[a:b]` to form a slice of a string
- `\n` to start a new line
- `+` for concatenation
- `int` to convert to an integer
- `str` to convert to a string
- `strip` to remove leading and trailing blanks
- `split` to produce a list of substrings, splitting at blanks
- forming a string from data, using {} for position numbers and `.format` to list data; `"The {0} bit the {1}.".format("man", "dog")`

References

CS 116 notes

- Module 03: Strings and Input/Output
  - Strings and their methods

Necaise

- Appendix A.5.1 Strings

4 Files

At times we will use files to store data items that we will add to an ADT. Be sure you can open and close a file, read and write lines, strip blank spaces (using the string method `strip`), and divide an input string into a list of items based on separation by blank spaces (using the string method `split`).

Relevant information includes:

- `open` to open a file to either read (as in `open(in_file, "r")`) or write (as in `open(out_file, "w")`)
- `.readlines()` to read a file that has been opened
- `.write(info)` to write `info` to the file
- `.close()` to close a file

References

CS 116 notes

- Module 10: File Input and Output

Necaise

- Appendix A.6 Text Files