CS 234
Naomi Nishimura

Python session 2

Note: Although there may not be time in the session itself for full use of the design recipe, it is highly recommended that you write the purpose, effects, and contract for each function before checking the model solutions. In that way, you will have a chance to practice use of the design recipe before being marked on it in assignments.

1 Mutation and interning

Files to download: `check.py`, `sess2q1provided.py`

To try:

- Before looking at the sample solution, try to determine answers to the questions about the code fragments below. The code fragments can be found in the downloaded file `sess2q1provided.py`. You may wish to use `check.py` to create tests that check your intuition.
- If needed, create more types of data items (such as lists and strings) and try to guess when the addresses will be the same and when different (that is, when Python is using interning of common strings and when not).
- If needed, experiment further with mutation to ensure you understand which types of data are mutable and which types are not. For example, even though tuples are immutable, if an item in a tuple is mutable (such as a list), then it can in fact be changed.

Questions to answer:

- Do `my_list` and `your_list` have the same address? Do corresponding elements in the lists have the same addresses?

```
my_list = [1, 2, 3, 4]
your_list = [1, 2, 3, 4]
```

- Which pairs of variables (excluding `magic`) have the same addresses?

```
a_one = "a"
a_two = "a"
magic = "abracadabra"
a_three = magic[0:1]
```

- Do `w_one` and `w_two` have the same address?

```
w_one = "whatchamacallit"
w_front = "whatcha"
w_back = "macallit"
w_two = w_front + w_back
```

- Will changing `value` change `pair[1]`?

```
value = 2
pair = [1, value]
```

- Will changing `my_list` change `my_tuple[2]`?

```
my_list = [1, 2, 3, 4]
my_tuple = (1, 2, my_list)
```

Sample solution: `sess2q1mutation.py`
Python syntax to notice:

- Addresses of data items can be determined by using `id`.

2 Classes

Files to download: `check.py`, `contiguous.py`, `sess2q2provided.py`
To try:

- Before looking at the sample solution, try to write a class for `Point` objects, each containing an integer x-coordinate `x_value`, an integer y-coordinate `y_value`, and a string label `label`.
- Your class should make use of contiguous memory to store the values in an array of size 3. The mini-textbook contains an appendix "Python modules for data structures" that explains the use of the module `contiguous.py`, which is available from the Python page of the course website.
- Write methods to create a Point, print the information associated with a Point, determine each of the attribute values, determine if two Points have the same values for both `x_value` and `y_value`, determine if two Points are equal (have the same values for all three attributes), and determine the distance between two Points. You should use the math module to calculate the distance, using the formula from the downloaded file `sess2q2provided.py`.
- Create a testing file that imports the file containing the class definition of Point. Once your tests work, see what happens when you change the name of a method to have the prefix _ to make it into a private method.

Sample solution: `sess2q2point.py` and `sess2q2pointuse.py`
Python syntax to notice:

- A class can be defined by using `class`.
- Methods can be created using `__init__`, `__repr__`, and `__eq__`
- Operator overloading and dot notation are used to apply methods.
- Method definitions use `self` to refer to the object.

Python style to notice:

- The identifier for the name of a class is capitalized..
- A **docstring** (documentation for the class in the form of a string) is used in the class definition.
- The design recipe is used for each method.

3 Strings

Files to download: `check.py`
To try:

- Before looking at the sample solution, try to write a function `string_transform` that consumes a string and produces a string consisting of taking each word (ignoring blanks) and putting its first half (or, strictly speaking, when the length of the list is odd, each of the characters that precede the middle character) followed by a question mark, on a separate line.
- For example, `string_transform(" this is strange ")` will produce

```
th?
i?
str?
```

- If time permits, create tests for your function.

Sample solution: `sess2q3stringtransform.py`
Python syntax to notice:

- The words are extracted using `strip` and then `split`.
- The length of a word is determined by using `len`.
- Slices are used to extract partial information.
- The operation `+` is used for concatenation.
- A new line is started by using `\n`.

4 Files

To try:

- Before looking at the sample solution, try to write a function `add_line_nums` that consumes the names of input and output files and creates an output file with the same lines as the input file, but with each line starting with its line number.

Sample solution: `sess2q4addlinenums.py`, `sess2q4in.txt`, `sess2q4out.txt`
Python syntax to notice:

- A file is opened to read or write by using `open`.
- A file is closed by using `close`.
- The lines in a file are read by using `readlines`.
- It is possible to use `for` and `in` to loop through all lines.
- The function `str` is used to convert the line number into a string.

5 Creating a text file

To try:

- Before looking at the sample solution, try to write a function `make_points_text_file` that consumes the number of points, a string of information about the points, and a name for the file to create.
- The function should create a file that contains the number of points on the first line and the three pieces of information for each point (one character for each) on each of the subsequent lines. The characters should be separated by spaces.
- The name of the file should be formed by appending the name provided to `"testpoints"` and then appending `".txt"`.
- Try to create a text file using your function.

Sample solution: `sess2q5makepointstextfile.py`, `sess2q5makepointstextfileuse.py`, and `testpointssix.txt`
Python syntax to notice:

- The creation of an output file name can be achieved by using `.format`.
- The functions `open` and `close` can be used for file access.
- To write to a file, `.write` can be used.
- The function `str` is used to convert an integer to a string.
- Characters are extracted from the input string by using `[]`.
- The operator `+` and `\n` are used to create lines to write to the file.

6 Using a text file to create a linked list of Points

Files to download: `check.py`, `linked.py`, `sess2q2point.py`, `sess2q5makepointstextfile` (for testing)

To try:

- Before looking at the sample solution, try to write a function `make_points` that consumes the name of a text file and produces a linked list of Points, one Point for each point described in the file. The order in which they appear in the linked list should be the same as the order in which they are described in the text file.
- Your solution should make use of linked memory. The mini-textbook contains an appendix "Python modules for data structures" that explains the use of the module `linked.py`, which is available from the Python page of hte course website.
- Write tests to make sure that you can navigate in the linked list and extract Point information.

Sample solution: `sess2q6makepointslist.py` and `sess2q6makepointslistuse.py`

Python syntax to notice:4

- Extraction of information from the file can be accomplished using `open`, `readlines`, and `close`.
- Blanks are removed using `strip` and `split`.
- The functions `int` is used to transform a string into an integer.
- The solution uses Point methods and methods from `linked.py`.