CS 234
Naomi Nishimura

Python session 1

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 Python basics

To try:

- Before looking at the sample solution, try to write a function `distance` that consumes four numbers representing the coordinates of two points (the x- and y-coordinates of the first point followed by the x- and y-coordinates of the second point) and produces the distance between the two points. Don't worry if you don't remember the exact formula - the purpose of this exercise is to remember how to write a function.
- The formula that you use should square two numbers. Try to write this operation in at least two different ways.
- See what happens if you remove the return statement (you can do this by making it into a comment).

Sample solution: `sess1q1distance.py`
Python syntax to notice:

- A module is imported using `import` and dot notation is used for functions from the module. (Note: We will use `from xx import *` when we have defined our own module `xx.py`.)
- A function is defined using `def`.
- The body of the function is indented.
- A value is returned using `return`.
- A function is called using its name with parameters in parentheses.
- When no value is returned, `None` is produced.
- Squares can be computed in at least three different ways.

Python style to notice:

- Identifiers are in lower case, with words connected by underscores.
- Lines are indented four spaces.
- Aspects of the design recipe appear in comments; examples and tests are missing.

2 Creating a testing file

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

- Add a statement that imports `sess1q1distance.py` in such a way that you do not need to use a prefix to use the function `distance`.
- Add a statement that imports `check.py` in such a way that you use a prefix `check` to use the function `within`.
- Create a set of tests to test your solution `sess1q1distance.py`.
- Remember that `distance` produces a floating point number.

Sample solution: `sess1q2distanceuse.py`
Python syntax to notice:

- Two different types of `import` statements are used for the two modules.
- Each `check.within` statement has a name, a test, an expected answer, and a tolerance.
- Each expected answer must be a floating point number in order to be compared to a floating point number produced by the function.
- When using `check.expect`, you need to supply a name, a test, and an expected answer, but no tolerance.

3 Booleans, conditionals, branching

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

- Before looking at the sample solution, try to write a function `assess` that consumes a value that is either the Boolean `False` or an integer, and produces one of the following strings: `"False"` (when `False` is consumed), `"Even"` (when an even integer is consumed), or `"Odd"` (when an odd integer is consumed).
- Try writing your solution twice, once using nested branching and once using `elif`.
- Write tests using the module `check`.
- Make sure that your solution works correctly on the input `0`.

Sample solution: `sess1q3assess.py`
Python syntax to notice:

- Colons are used after branching statements.

- Indentation is used for each branch.
- The remainder on division by 2 is calculated using `%`.
- To obtain the correct solution on input `0`, `is` is used to compare the input and `"False"`.

4 Iteration

To try:

- Before looking at the sample solution, try to write a function `print_triangle` that consumes an integer `rows` and prints a triangle of asterisks (∗) with that many rows such that the first row has one asterisk and each subsequent row has two more asterisks than the previous row. The asterisk in the first row should be aligned to be in the middle of the triangle.
- Try using `for` for iteration, and then once you have that working, try using `while` for iteration.
- Now add a conditional statement with a `break` so that the printing is terminated early, based on when the condition is satisfied.
- Change `break` to `continue` and see what changes.

Sample solution: `sess1q4printtriangle.py`

Python syntax to notice:

- Colons are used with `for` and `while` loops.
- Values from 0 to `rows`−1 are obtained using `range`.
- Changes in looping can be created by using `break` and `continue`.
- Copies of a string can be formed by using ∗.

Python style to notice:

- The identifier `STAR` is fully capitalized to show that it is a constant.
- Comments are used to show effects and requirements.
- Comments are used to explain local variables and constants.

5 Recursion

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

- Before looking at the sample solution, try to write a function `num_fives` that consumes a nonnegative integer and produces the number of times the digit 5 occurs in the integer. For example, `num_fives(5435425)` should produce 3.
- Your function should make use of recursion, at each step breaking the number into the last digit and a number made from the remaining digits.
- Make sure that your tests cover the base case.

Sample solution: `sess1q5numfives.py`
Python syntax to notice:

- The function `num_fives` calls itself.
- The function `num_fives` has a non-recursive base case.
- For integer division, `//` is used.
- To obtain an integer from a non-integer, `math.floor` is used.

6 Random and Python lists

To try:

- Before looking at the sample solution, try to write a function `random_change` that consumes a list of integers, a lower bound, and an upper bound. Your function should choose an integer at random from the range from the lower bound to the upper bound and then multiply each of the integers in the first half of the list (or, more strictly speaking, when the length of the list is odd, each of the integers that precede the middle integer) by the randomly chosen integer.
- You should import the module `random`.

Sample solution: `sess1q6randomchange.py`
Python syntax to notice:

- The length of the list is determined by using `len`.
- The item in a particular position in the list can be accessed by using `[]`.
- Random numbers can be generated using `random.randint`.

7 Python data types

There is a small question using Python data types in Python session 2; make sure that you exercise your understanding of these types sufficiently to be able to understand their use.