CS 231
Naomi Nishimura

Python session 2

1 Recursion

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

- Before looking at the sample solution, try to write a function `factorial` that produces the factorial of a nonnegative integer input.
- Remember that $0! = 1$ and that for any $n > 0$, $n! = n \cdot n - 1 \cdots 1$.
- Your solution should use recursion.

Sample solution: `sess2q1factorial.py`
Python syntax to notice:

- The function `factorial` calls itself
- The function `factorial` has a non-recursive base case

2 Recursion using lists

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

- Before looking at the sample solution, try to write a function `merge` that consumes two sorted lists of integers and produces the sorted list of all the integers in the two lists.
- The output list can be formed by determining which of the two list has the smallest item, putting that item first and then merging the remaining lists

Sample solution: `sess2q2merge.py`
Python syntax to notice:

- The function `merge` calls itself
- The function `merge` has two non-recursive base cases
- The operation `+` is used to concatenate two lists

3 Creating a class

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

- Before looking at the sample solution, try to write a class definition `Pair` such that each object in the class consists of two fields: `item` (Any) and `value` (Int).
- Your class definition should ensure that you can create a new object by using the method `Pair`.
- Optional: Create a method `__repr__` that represents a `Pair` as a string.
- Optional: Create a method `__eq__` that determines if two pairs have the same values in both the `item` and `value` fields.
- Optional: Use `isinstance` to check that an object you have created is a `Pair`.

Sample solution: `sess2q3pair.py` with tests in `sess2q3pairuse.py`
Python syntax to notice:

- Use of `class`
- Indentation in the class definition
- Use of `__init__`
- Use of `__repr__`
- Use of `self`
- Use of `str` to convert values to strings in `__repr__`
- Use of `isinstance` to check if an object is in a class

4 Creating a class and methods

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

- Before looking at the sample solution, try to write a class definition `ThreeD` such that each object in the class consists of four fields: `entries` (a list of Grids), `dim_one`, `dim_two`, and `dim_three` (all integers). Your objects can serve as three-dimensional versions of grids, storing items in locations specified by three values `x`, `y`, and `z` indicating the position, where the values of `x` range from `0` to `dim_one` - 1, the values of `y` range from `0` to `dim_two` - 1, and the values of `z` range from `0` to `dim_three` - 1.
- Your class definition should ensure that you can create a new object by using the method `ThreeD`.

- Now add methods `__repr__`, `access`, and `enter` to your class definition `ThreeD`.
- The first method should create a string representing your object. There are many possible options available to you.
- The method `access` will consume three values `x`, `y`, and `z`, representing a position in three dimensions, and will produce the data item stored in that location.
- The method `enter` will consume four values `x`, `y`, `z` and `item`, representing a position in three dimensions, and a data item, and will enter the data item in that location.

Sample solution: `sess2q4threeD.py` with tests in `sess2q4threeDuse.py`
Python syntax to notice:

- Placement of methods inside the class definition
- Use of `__repr__`
- Use of `self`

5 Sorting pairs

Files to download: `check.py`, `sess2q3pair.py`
To try:

- Before looking at the sample solution, try to write a function `pairs_sort_up` that consumes a list of pairs and produces a list of the pairs in nondecreasing order by value.
- You may wish to write a helper function that extracts the value from a pair.
- Next, try to write a function `pairs_sort_down` that consumes a list of pairs and produces a list of the pairs in nonincreasing order by value.

Sample solution: `sess2q5pairsort.py` with tests in `sess2q5pairsortuse.py`
Python syntax to notice:

- Use of `key`
- Use of `reverse`

6 Using random

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

- Before looking at the sample solution, try to write a function `random_select` that consumes a non-empty list and produces a random item in the list.
- Be careful in selecting the lower and upper bounds of the random integer.

Sample solution: `sess2q6randomselect.py` with tests in `sess2q6prandomselectuse.py`
Python syntax to notice:

- Use of `random.randint`