Carnegie Mellon University
Computer Science Academy

CS1: Unit 10 Review

Lists

Lists

- A list is a Python type where we can store multiple values
  - Ex. `animalList = [ 'dog', 'cat', 'cow' ]`, `len(animalList)` is `3`
- We can loop over a list using a `for` loop to access individual values in the list
  - Ex. `for animal in animalList`, variable `animal` is set to each value in `animalList`
- Lists are powerful because they allow us to store multiple values together and change those values later
- Unlike a group, shapes remain in a list even when they are invisible
- We can use an index to access values in a list, just like in a string
  - Ex. `animalList[0]` is `'dog'`, `animalList[1]` is `'cat'`, `animalList[2]` is `'cow'`
- We can loop over a list using indexes
  - Ex. `for i in range(len(animalList))`, `animalList[i]` can be used to access each value in `animalList`
- We can change a value in a list by assigning into that index
  - Ex. if we set `animalList[0] = 'blue'`, `animalList` will be `[ 'dog', 'cat', 'cow' ]`

List Methods

List methods

- We use `append` to add a value to a list
  - Ex. If `animalList = [ 'dog', 'cat', 'cow' ]` and we run `animalList.append('pig')`, `animalList` will be `[ 'dog', 'cat', 'cow', 'pig' ]`
- `append()` is a method but not a function
  - We call `animalList.append(value)` but not `append(animalList, value)`
- We use `pop` to remove the last value from a list
  - Ex. If `animalList = [ 'dog', 'cat', 'cow', 'pig' ]` and we run `animalList.pop()`, `animalList` will be `[ 'dog', 'cat', 'cow' ]`, since `'pig'` was removed (popped)
- `pop()` removes and returns the last value in the list
  - Ex. If `animalList = [ 'dog', 'cat', 'cow', 'pig' ]` and we run `animal = animalList.pop()`, `animal` will be `'pig'`, the last value in the list
- We can also provide an index of which value to pop
  - Ex. `animalList.pop(0)` will remove the first value in the list, `'dog'`
- We use `choice` to get a random value from a list
  - Ex. If `animalList = [ 'dog', 'cat', 'cow']`, `animal = choice(animalList)`, `animal` will be `'dog'`, `'cat'`, or `'cow'` by chance
- `choice()` is a function, not a method
  - We call `choice(animalList)` but not `animalList.choice()`

Return Values

More functions that return values

- group.hitTest()
    - A method that takes (x, y) position and returns the topmost shape in the group. If no shape in the group is hit, then group.hitTest(x, y) returns None
- app.group.hitTest()
    - If the group is not specified, shapes by default are added to the top-level group, app.group
    - We can use app.group.hitTest() to test if any shapes in that top-level group are hit

Writing functions that return values

- Why helper functions help
    - The code is easier to read in smaller pieces, and the function name clearly indicates what it does
    - We can also reuse the helper function by calling it from multiple places
- Using multiple returns
    - If a function does not return a value, then Python automatically returns the value None
    - A function can use return any number of times
    - As soon as any one of the returns is reached, the function returns that value and stops running
- Returning a shape
    - A function can return any kind of value, including a shape such as a Circle, or even a Group of shapes