CS 150 Spring 2020 - Midterm "Cheat Sheet"

Modules

`turtle` module

- **`forward(dist), backward(dist):`** Move the turtle forward or backward by the specified length `dist`
- **`right(angle) left(angle):`** Turn the turtle right/left by `angle` (in degrees)
- **`goto(x, y):`** Move turtle to position `x, y`
- **`setheading(angle):`** Set the turtle's heading to angle
- **`circle(radius):`** Draw a circle with specified `radius`; the center is `radius` units left of the turtle
- **`penup():`** Pull the pen up – no drawing when moving
- **`pendown():`** Put the pen down – drawing when moving
- **`fillcolor(color):`** Change the fill color to `color`, where `color` is a string
- **`begin_fill(), end_fill():`** Start and end filling shapes with fill color

`random` module

- **`randint(a, b):`** Return a random integer `N` such that `a ≤ N ≤ b`
- **`uniform(a, b):`** Return a random floating point number `N` such that `a ≤ N ≤ b`

`math` module

- **`sqrt(num):`** Return the square root of `num`

Strings

The following functions are built-in

- **`len(string):`** Returns the number of characters in the string
- **`int(string), float(string):`** Converts numeric string to int or float
- **`str(object):`** Converts object, e.g., int or float to a string
- **`sorted(string):`** Returns the characters of the string as a list in sorted order

String object methods

- **`S.upper(), S.lower(), S.capitalize():`** Returns a new string, upper or lower-cased, or capitalized
- **`S.find(some_string):`** Returns the first index that **`some_string`** occurs at in string S or -1 if not found
- **`S.find(some_string, index):`** Same as above, but starts searching at index
- **`S.replace(old, new):`** Return a copy of string S with all occurrences of old substituted with new
- **`S.startswith(prefix):`** Returns **`True`** if string S starts with prefix, False otherwise
- **`S.endswith(suffix):`** Returns **`True`** if string S ends with suffix, False otherwise
- **`S.strip():`** Returns a copy of string S with only the leading and trailing whitespace removed
- **`S.split():`** Return a list of the words in string S using whitespace as the delimiter
- **`S.isalpha():`** Return **`True`** if all characters in string S are alphabetical and S has at least one character

String operators

- **`string1 + string2:`** Returns a new string that is the concatenation of string1 and string2
- **`string * int:`** Returns a new string that is string repeated int times
- **`substr in string:`** Returns True if substr is a substring of string, False otherwise

Lists

Creating new lists

- **`[]`** creates empty list
- **`[object1, object2, ...]`** creates list containing objects
- **`list(iterable)`** creates a list from any iterable object (e.g., range, string)

The following functions are built-in

- **`len(list):`** Returns the number of elements in `list`
- **`sum(list), min(list), max(list):`** Returns the sum, min, or max of elements in `list`
- **`sorted(list):`** Returns a new copy of the list in sorted order

List object methods

- **`L.append(x):`** Adds `x` to the end of list L, no return value
- **`L.extend(other_list):`** Adds all elements of `other_list` the end of list L, no return value
- **`L.index(item):`** Returns the index of the first occurrence of `item` in list L or error if it does not occur
- **`L.insert(index, x):`** Insert `x` at `index` in list L, no return value
- **`L.pop():`** Removes the item at the end of list L and returns it
- **`L.pop(index):`** Removes item at `index` from list L and returns it
- **`L.reverse():`** Reverses the elements in list L in place, no return value
- **`L.sort():`** sorts the elements in the list in place, no return value

List operators

- **`list1 + list2:`** Returns a new list that contains the elements of `list1` followed by the elements of `list2`
- **`list * int:`** Returns a new list that contains the items in `list` repeated `int` times
- **`item in list:`** Returns True if `item` is an element of `list`, False otherwise

Other

Range

- **`range(stop):`** Equivalent to range(0, stop, 1)
- **`range(start, stop[, step]):`** Create sequence of integers from inclusive **`start`** to exclusive **`stop`** by **`step`**

Slicing

- **`seq[start[:stop[:step]]:`** Slice sequence **`seq`** from inclusive **`start`** to exclusive **`stop`** by **`step`**

Reading input from the user

- **`input(message):`** Displays message to the user and returns what the user typed as a string

Reading from a file with a for-loop

Spring 2020 midterm will not include questions about files

```
with open(filename, "r") as file:
    for line in file:
        # do something with line (a string)
```