CS 150 - Midterm “Cheat Sheet” (Version 1.10)

- 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

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

- 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`**
- Built-in functions
  `abs(a):` Return absolute value of number a

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
  `upper(), lower(), capitalize():` Returns a new upper or lower-cased, or 1st letter upper-cased string
  `find(some_string):` Returns the first index that **`some_string`** occurs at in the string or -1 if not found
  `find(some_string, index):` Same as above, but starts searching at index
  `replace(old, new):` Return a copy of the string with all occurrences of old substituted with new
  `startswith(prefix):` Returns **`True`** if the string starts with prefix, False otherwise
  `endswith(suffix):` Returns **`True`** if the string ends with suffix, False otherwise
  `strip():` Returns a copy of the string with only the leading and trailing whitespace removed
  `split():` Return a list of the words in the string using whitespace as the delimiter
  `isalpha():` Return **`True`** if all characters in string are alphabetical and the string 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
  `append(x):` Adds `x` to the end of the list
  `extend(other_list):` Adds all elements of `other_list` the end of the list
  `index(item):` Returns the index of the first occurrence of `item` in the list or error if it does not occur
  `insert(index, x):` Insert `x` before `index` in the list

`pop()`: Removes the item at the end of the list and returns it
`pop(index)`: Removes item at `index` from the list and returns it
`reverse()`: Reverses the elements in the list in place
`sort()`: sorts the elements in the list in place

- 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

Modules

- **`turtle`** module
  `forward(dist), backward(dist)`: Move the turtle forward/backward by the length `dist`. Doesn't change heading.
  `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 turtles heading to angle
  `circle(radius)`: Draw a circle with specified `radius`; the center is `radius` units left of the turtle
  `dot(size)`: Draw a filled circle with diameter `size` centered on current position 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