CSE 160 26wi Final Exam Reference Sheet

```
# if / elif / else syntax
if condition1:
    # statements
elif condition2:
    # other statements
else:
    # more statements
```

```
# for loop syntax
for i in sequence:
    # statements

# function definition syntax
def function_name(param1, param2, …):
    # statements
```

Sequences and Lists

range([***start***,] ***stop*** [, ***step***])	Returns a sequence of numbers from ***start*** (inclusive) to ***stop*** (exclusive) incrementing by ***step***
len(lst)	Returns the number of elements in ***lst***
lst = []	Creates an empty list
lst[***idx***]	Returns the element in ***lst*** at index ***idx***
lst[***start***:***end***:***step***]	Returns a slice of ***lst*** from index ***start*** (optional) to index ***end*** (exclusive), incrementing by ***step*** (optional)
lst.append(***elmt***)	Adds the ***elmt*** to the end of ***lst***, returns **None**
lst.extend(***sequence***)	Adds each of the elements in ***sequence*** to the end of ***lst***, returns **None**
lst.index(***elmt***)	Returns the index of the first occurrence of ***elmt*** in ***lst***, errors if ***elmt*** is not in ***lst***
lst.count(***elmt***)	Returns the number of times ***elmt*** occurs in ***lst***
lst.remove(***elmt***)	Removes the first occurrence of ***elmt*** from ***lst***, errors if ***elmt*** is not in ***lst***, returns **None**
lst.pop(***idx***) lst.pop()	Removes and returns the element at index ***idx*** in ***lst*** With no parameter, removes the last element in ***lst***
lst.insert(idx, elmt)	Inserts ***elmt*** into ***lst*** at index ***idx***, returns **None**

File I/O

```
file = open(filepath, "r")

# read entire file at once
file.read()
file.close()
```

```
file = open(filepath, "r")

# read line by line
for line in file:
    # process line
```

```
file = open(filepath, "w")

# write entire file at once
file.write(string_to_write)
file.close()
```

Dictionaries

`my_dict = {}` `my_dict = dict()`	Creates a new, empty dictionary
***my_dict*[*key*]**	Returns the value associated with the given ***key*** in ***my_dict***
`del` ***my_dict*[*key*]**	Remove ***key*** (and its associated value) from ***my_dict***
`list(`***my_dict***`.keys())`	Returns a list of keys in ***my_dict***
`list(`***my_dict***`.values())`	Returns a list of values in ***my_dict***
`list(`***my_dict***`.items())`	Returns a list of tuples of the form (***key***, ***value***)
`sorted(`***my_dict***`)`	Returns a sorted list of the keys in ***my_dict***

Sorting

`sorted(`***collection*** `[, `***key***`, `***reverse***`])`	Returns a sorted copy of ***collection***, based on the optional sort key (***key***) and optional order preference (***reverse***)
`lst.sort([key, reverse])`	Sorts the given list (***lst***), based on the optional sort key (***key***) and optional order preference (***reverse***), and returns **None**
key	A function to determine how to compare two values

Classes

`class` ***Name***`:`
`    # class methods`
`    def method(self [, args]):`
`        # method body`
	Defines a new class named ***Name*** with the subsequently defined methods (and attributes)

`def __init__(self [, args]):` `    # method body`	Function that is called during the creation of an instance of the class (e.g. ***Name()***)
self	Required parameter for all methods of a class. Refers to the specific instance of the class. Can hold any number of arbitrary variables (attributes), as in ***self*.my_attribute**
n `= Name()`	Instantiates (creates) a new instance of the ***Name*** class and assigns a reference to it to the variable ***n***
***n*.*method*([*args*])**	Calls ***method*** on the instance of the class (***n***), optionally passing in any required arguments. Inside the function, **self** is assigned to ***n***