CSE 160 Exam Cheat 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
```

Function	Description
`range([start,] stop [, step])`	Returns a sequence of numbers from ***start*** (inclusive) to ***stop*** (exclusive) incremented by ***step***
`len(lst)`	Returns the number of elements in ***lst***

Lists

Function	Description
`lst = []`	Creates an empty list
`lst[idx]`	Returns the element in ***lst*** at index ***idx***
`lst[start:end]`	Returns a sublist of ***lst*** from index ***start*** (inclusive) to index ***end*** (exclusive)
`lst.append(elmt)`	Adds the element ***elmt*** to the end of ***lst***. Returns **None.**
`lst.extend(mylst)`	Extend ***lst*** by appending the elements in ***mylst*** to the end of ***lst***. Returns **None.**
`lst.index(elmt)`	Returns index of the first occurrence of ***elmt*** in ***lst***. Error if ***elmt*** is not in ***lst***
`lst.count(elmt)`	Returns the number of times ***elmt*** occurs in ***lst***
`lst.remove(elmt)`	Removes first occurrence of ***elmt*** from ***lst***, Error 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 an element ***elmt*** in list at index ***idx***. Returns **None.**

File I/O

Function	Description
`my_file = open(filepath)`	Opens the file with given ***filepath*** for reading, returns a file object
`my_file.close()`	Closes file `my_file`

```
# Process one line at a time:
for line_of_text in my_file:
    # process line_of_text
```

```
# Process entire file at once
all_data_as_a_big_string = my_file.read()
```

Sets

Function	Description
{*elmt*(s)}, set(*lst*)	Constructs a set of provided *elmt*(s), or of elements in *lst*
my_set.add(*elmt*)	Adds *elmt* to *my_set*. Returns **None.**
my_set.remove(*elmt*)	Removes an element from *my_set* if present, otherwise error. Returns **None.**
my_set.discard(*elmt*)	Removes an element from *my_set* (no errors thrown). Returns **None.**
my_set.pop()	Removes and returns random element from *my_set*

Set Operation	Description
&	Intersection, or logical AND
|	Union, or logical OR
^	XOR
-	Difference

Dictionaries

Function	Description
my_dict = {}	Creates a new, empty dictionary
my_dict[*key*]	Returns the value associated with the given key in *my_dict*
my_dict.keys()	Returns an iterator over the keys in *my_dict*. Can be used in a for loop as is or converted to a list with list().
my_dict.values()	Returns an iterator over the values in *my_dict*. Can be used in a for loop as is or converted to a list with list().
my_dict.items()	Returns an iterator over the (key, value) tuples in *my_dict*. Can be used in a for loop as is or converted to a list with list().

Sorting

Function	Description
sorted(*collection* [,key=*sort_key*, reverse=*bool_val*])	Returns a sorted copy of *collection*, based on optional sort key (**key**) and optional order preference (**reverse**)
lst.sort([key=*sort_key*, reverse=*bool_val*])	Sorts the given list *lst*, based on optional sort key (**key**) and optional order preference (**reverse**), and returns **None**