CSE 160 23wi Final 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*** to index ***end*** (exclusive)
`lst[start : end : step]`	Returns a sublist of ***lst*** from index ***start*** to index ***end*** (exclusive), incrementing by ***step***
`lst.append(elmt)`	Adds the element ***elmt*** 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()
```

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 list of keys in ***my_dict***
my_dict`.values()`	Returns list of values in ***my_dict***

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**

Common Error Names

IndexError – Index out of range
KeyError – Key not found in dictionary
IndentationError – Invalid indentation
TypeError – Operation applied to invalid combination of types
ValueError – Function gets properly typed argument, but invalid value
SyntaxError – Invalid Python syntax
NameError – Variable name not found
FloatingPointError – Floating point operation fails
RuntimeError – Otherwise Unknown Error

Graphs

Function	Description
`import networkx as nx`	Imports the graph library and aliases the library name to “nx”, usable as “nx.Graph()”
g = nx.Graph()	Creates a new graph and assigns the variable g to reference it.
g.add_edge(“A”, “B”)	Adds an edge between nodes “A” and “B”, creating the nodes if needed.
g.add_node(“A”)	Adds node “A” to the graph
g.neighbors(“A”)	Returns a collection of the neighbors of node “A”
g.nodes() ***g.edges()***	Returns sets of nodes and edges, respectively, in the graph.

Classes

Function	Description
`__init__(self)`	The function that is called during class construction, as in Class().
self	Required parameter for all class methods (functions). Refers to the specific instance of the class. Can hold any number of arbitrary variables, as in **self.name**