CSE 160 24au Midterm 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*** (default 0) to index ***end*** (exclusive, default ***len(lst)***), incrementing by ***step***
lst.append(***elmt***)	Adds the element ***elmt*** to the end of ***lst***. Returns **None**.
lst.extend(***other***)	Adds each of the elements in the list ***other*** 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 ***lst*** at index ***idx***. Returns **None**.
lst.sort()	Sorts the given list ***lst***. Returns **None**.
lst.reverse()	Reverses the order of elements in the list ***lst***. 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`

`with open(`***`filepath`***`) as `***`f`***`:`
`    # read file`
	Opens the file with given ***`filepath`*** for reading via the file object ***`f`*** in the body of the “with” statement.

```
# 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`` = {}`
`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`***`]`	Removes the ***`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)`***

```
# Process each key-value pair together:
for key, value in my_dict.items():
    # process key and value
```

```
# Process one key at a time
for key in my_dict:
    # use dictionary’s key
```

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

Sets

Function	Description
`s1 = set()`	Creates a new empty set
`s1 = set([...])`	Create a new set containing all of the elements from the given list.

`s1 | s2`
`s1.union(s2)`
	Evaluates to the union of ***s1*** and ***s2***

`s1 & s2`
`s1.intersection(s2)`
	Evaluates to the intersection of ***s1*** and ***s2***

`s1 - s2`
`s1.difference(s2)`
	Evaluates to the difference of ***s1*** and ***s2***

`s1 ^ s2`
`s1. symmetric_difference(s2)`
	Evaluates to the symmetric difference of ***s1*** and ***s2***

`s1.add(elem)`	Adds ***elem*** to the set ***s1***
`s1.remove(elem)`	Removes ***elem*** from the set ***s1*** if it exists, but throws a KeyError if ***elem*** does not exist
`s1.discard(elem)`	Removes ***elem*** from the set ***s1***
`s1.pop()`	Removes an arbitrary element from the set ***s1***