**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*** to index ***end*** (exclusive) |
| 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() |

**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 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** |