

CSE 160 24au 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
<code>range([start,] stop [, step])</code>	Returns a sequence of numbers from start (inclusive) to stop (exclusive) incremented by step
<code>len(Lst)</code>	Returns the number of elements in Lst
<code>"string".split([delimiter])</code>	Returns a list where the elements are the result of separating " string " by [delimiter] (which defaults to all whitespace if not given).

Lists

Function	Description
<code>lst = []</code>	Creates an empty list
<code>lst[idx]</code>	Returns the element in Lst at index idx
<code>lst[start : end]</code>	Returns a sublist of Lst from index start to index end (exclusive)
<code>lst[start : end : step]</code>	Returns a sublist of Lst from index start (default 0) to index end (exclusive, default len(Lst)), incrementing by step
<code>lst.append(eLmt)</code>	Adds the element eLmt to the end of Lst . Returns None .
<code>lst.extend(other)</code>	Adds each of the elements in the list other to the end of Lst . Returns None .
<code>lst.index(eLmt)</code>	Returns index of the first occurrence of eLmt in Lst , error if eLmt is not in Lst
<code>lst.count(eLmt)</code>	Returns the number of times eLmt occurs in Lst
<code>lst.remove(eLmt)</code>	Removes first occurrence of eLmt from Lst , error if eLmt is not in Lst . Returns None .
<code>lst.pop(idx)</code> <code>lst.pop()</code>	Removes and returns the element at index idx in Lst . With no parameter, removes the last element in Lst
<code>lst.insert(idx, eLmt)</code>	Inserts an element eLmt in Lst at index idx . Returns None .
<code>Lst.sort()</code>	Sorts the given list Lst . Returns None .

File I/O

Function	Description
<code>my_file = open(<i>filepath</i>)</code>	Opens the file with given <i>filepath</i> for reading, returns a file object
<code>my_file.close()</code>	Closes file <i>my_file</i>
<code>with open(<i>filepath</i>) as <i>f</i>: # read file</code>	Opens the file with given <i>filepath</i> for reading via the file object <i>f</i> 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
<code>my_dict = {} my_dict = dict()</code>	Creates a new, empty dictionary
<code>my_dict[<i>key</i>]</code>	Returns the value associated with the given <i>key</i> in <i>my_dict</i>
<code>del my_dict[<i>key</i>]</code>	Removes the <i>key</i> (and its associated value) from <i>my_dict</i>
<code>list(my_dict.keys())</code>	Returns a list of keys in <i>my_dict</i>
<code>list(my_dict.values())</code>	Returns a list of values in <i>my_dict</i>
<code>list(my_dict.items())</code>	Returns a list of tuples of the form (<i>key</i> , <i>value</i>)
<code>sorted(my_dict)</code>	Returns a sorted list of the keys in <i>my_dict</i>

```
# 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
```

Sorting

Function	Description
<code>sorted(<i>collection</i> [,key=<i>sort_key</i>, reverse=<i>bool_val</i>])</code>	Returns a sorted copy of <i>collection</i> , based on optional sort key (<i>key</i>) and optional order preference (<i>reverse</i>)
<code><i>Lst</i>.sort([key=<i>sort_key</i>, reverse=<i>bool_val</i>])</code>	Sorts the given list <i>Lst</i> , based on optional sort key (<i>key</i>) and optional order preference (<i>reverse</i>), and returns None
<code><i>sort_key</i></code>	A reference to a function to determine what

	value to use when comparing two items
--	---------------------------------------

Graphs

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

Sets

Function	Description
<code>s1 = set()</code>	Creates a new empty set
<code>s1 = set([...])</code>	Create a new set containing all of the elements from the given list.
<code>s1 s2</code> <code>s1.union(s2)</code>	Evaluates to the union of <code>s1</code> and <code>s2</code>
<code>s1 & s2</code> <code>s1.intersection(s2)</code>	Evaluates to the intersection of <code>s1</code> and <code>s2</code>
<code>s1 - s2</code> <code>s1.difference(s2)</code>	Evaluates to the difference of <code>s1</code> and <code>s2</code>
<code>s1 ^ s2</code> <code>s1.symmetric_difference(s2)</code>	Evaluates to the symmetric difference of <code>s1</code> and <code>s2</code>
<code>s1.add(eLem)</code>	Adds <code>eLem</code> to the set <code>s1</code>
<code>s1.remove(eLem)</code>	Removes <code>eLem</code> from the set <code>s1</code> if it exists, but throws a <code>KeyError</code> if <code>eLem</code> does not exist
<code>s1.discard(eLem)</code>	Removes <code>eLem</code> from the set <code>s1</code>
<code>s1.pop()</code>	Removes an arbitrary element from the set <code>s1</code>

Classes

Function	Description
<pre>class Name: # class methods, for example: def method(self, [args]): # method body</pre>	Defines a new class named Name with the subsequently defined methods.
<pre>def __init__(self): # method body</pre>	The function that is called during class construction/creation, as in Name() .
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
<pre>n = Name()</pre>	Instantiates (creates/constructs) a new instance of the Name class and assigns a reference to it in the variable n .
<pre>n.method([args])</pre>	Calls the method function on the instance defined in n , optionally passing in any required arguments.

Common Errors

IndexError – Index out of range

AssertionError – Assert failed

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