CS1010S Finals Cheat Sheet

by otkl via cheatography.com/179565/cs/38302/

Types of Errors and Examples

IndexError	The index of a sequence is out of range.	(1,)[1]
KeyError	Key is not found in the dictionary.	cs = {10:10}; cs[s]
Syntax-Error	Invalid syntaxes found in code.	print(Hello World) (no quote marks) or for i in range(3) (no colon)
TypeError	A function/operation is applied to objects of incorrect types.	(1,2) + [3,4] or "cs" + 1010
ValueError	Function gets an argument of a correct type but improper value.	int("3.14")
NameError	The variable with the name is not found in the local & global scope.	del cs1010s
Attribute-Error	An attribute reference or assignment fails due to incorrect data type.	(1,2).append(3)
ZeroDivisionError	Second operand (denominator) of a division/module operation is zero.	1/0
Recursion-Error	An operation/runs out of memory.	def f(x): return x + f(x-1)
UnboundLocal-Error	A reference is made to a local variable in a function/method, but no value has been bound to that variable.	def f(x): ;z = y + x; y = 0
Runtim-eError	Can be due to many reasons. One of them being list/dictionary size changing during iteration.	a = [1, 2]; for ele in a: a.append(ele)

Defining Custom Errors

class CustomError:
pass

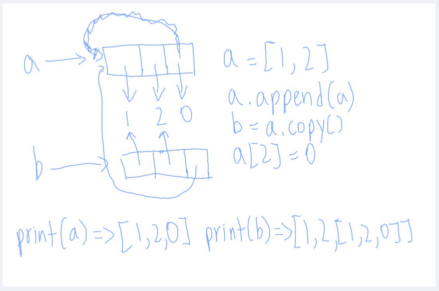
Exception Handling

try:	The code inside the try block will run until it reaches an error. Once it does, it will be handled accordingly by the respective exception statements.
except Error:	The code inside this block will only run if the encountered error is the error to be handled by this except statement.
except Exception:	The code inside this block will only run if the error has not been handled by the previous except statements.
else:	The code inside the else block will only run if no errors were encountered in the try statement.
finally:	The code inside the finally block will run no matter what.

For/While Loop Statements

break	Terminates the loop (once)
continue	Stops the current iteration of the loop, and goes on to the next iteration of the current loop
pass	Does nothing and continues the rest of the code inside the current iteration of the loop

List shallow copies (list.copy(), list[:])



List and dictionary functions

string.split(sep)	Splits the string into a list by the seperator, which by default is "".	"CS1010S".split("1") == ["CS", "0", "0S]

CS1010S Finals Cheat Sheet

by otkl via cheatography.com/179565/cs/38302/

List and dictionary functions (cont)

dict.get(key, value)	Returns the value that is paired to the key in the dictionary. If the key does not exist in the dictionary, then it will return the value instead.	d = {"CS":10, 10: "S"} ; d.get("CS", "NOT FUN") == 10; d.get("10", "GG CS") == "GG CS"
del dict[key]	Removes the key-value pair with the indicated key in the dictionary. Returns a KeyError if key is not found in the dict	d = {"CS":10, 10: "S"}; del d[10]; d = {"CS": 10}
sep.join(iterable)	Joins all the items in the iterable into a string, with the sep inbetween each item.	d = {"CS":10, "10":"S"}; " - ".join(d); "CS - 10"

Passcode locking

```
def lock(obj, passcode):
o = obj.copy()
o.clear()
o["locked"] = lambda x: o if x == passcode else False
def unlock(obj, passcode):
o = obj["locked"] (passcode)
if o != False:
obj.clear()
obj.update(o)
```

Orders of Growth (OOG)

$O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(2^n) < O(n^n)$
O(1): Indexing, replacing variable name
O(log n): Constantly halving/doubling a number (depending on direction)
O(n): Going through the whole tuple/string (for loop/recursion)
$O(n^2)$: Going through the whole tuple once for each element (Usually nested for loop)
$O(2^n)$: The tree splits into 2/x number of branches for each level (Usually for recursion tree)
Sample Answer:
Time: O(n), there is a total of n recursive calls.
Space: O(n), there is a total of n recursive calls, and each call will take up space on the stack.
Time: O(n), the loop will iterate n times.
Space: O(1), no extra memory is needed because the variables are overwritten with the new values.
String slicing and concatenation takes O(n) time as well

Boolean Values

False evaluates to 0; int(False) == 0, while True evaluates to 1; int(True) = 1
On the other hand, any empty string, tuple, list, dict etc ("", (), [], {}), value 0 and None all evaluates to False; bool(0/None/""/()/[]/{}) = False, and any other expression will evaluate to True; bool(1/-95/"CS1010S is fun"/("C", "S", "S", "U", "C", "K", "S") = True

Checking data type

1. type(value) == Type
2. isinstance(value, Type)