

CS 150 Fall 2021 – Quiz 7 “Cheat Sheet”

Built-ins

abs(x): Returns the absolute value of **x**

sum(x), **max(x)**, **min(x)**: Compute sum, max, min of list

Input/Output

- Reading input from the user
input(message): Displays message to the user and returns what the user typed as a string
- Reading from a file
with open(filename, "r") as file:
 for line in file:
 # do something with line (a string)
- Writing to a file
open(filename, "w"): Write to file (overwrite any existing content)
open(filename, "a"): Append to the end of existing contents
file.write(item): Writes item to file (e.g. string, number) w/o trailing newline
- Reading from a URLs (webpages)
import urllib.request
with urllib.request.urlopen(some_url) as web_page:
 for line in web_page:
 line = line.decode('utf-8', 'ignore')
 # do something with line (now a string)
- Command-line arguments
import sys
sys.argv: is a list containing the command-line arguments (the first element is always the program name)

Sequences

- Range
range(stop): Equivalent `range(0, stop, 1)`
range(start, stop[, step]): Create sequence from inclusive **start** to exclusive **end** by **step**
- Slicing
seq[start[:stop[:step]]]: Slice **seq** from inclusive **start** to exclusive **stop** by **step**

Strings

- The following functions are built-in and answer questions about strings
len(string): Returns the number of characters in the string
int(string), **float(string)**: Converts a string to an int or float
- String object methods
upper(), **lower()**, **capitalize()**: Returns a new upper or lower-cased, or 1st letter upper-cased string
find(some_string): Returns the first index that **some_string** occurs at in the string or -1 if not found
find(some_string, index): Same as above, but starts searching at index
replace(old, new): Return a copy of the string with all occurrences of old substituted with new
startswith(prefix): Returns **True** if the string starts with prefix, False otherwise
endswith(suffix): Returns **True** if the string ends with suffix, False otherwise
strip(): Returns a copy of the string with leading and trailing whitespace removed
split(): Return a list of the words in the string using whitespace as the delimiter
- String operators
string1 + string2: Returns a new string that is the concatenation of string1 and string2
string * int: Returns a new string that is string repeated int times
substr in string: Returns True if substr is a substring of string, False otherwise

Lists

- Creating new lists
 - `[]` creates empty list
 - `[object1, object2, ...]` creates list containing objects
 - `list(iterable)` creates a list from any iterable object (e.g., range, set, string)
- The following functions are built-in and answer questions about lists
 - `len(list)`: Returns the number of elements in `list`
 - `sum(list)`, `min(list)`, `max(list)`: Returns the sum, min, or max of elements in `list`
 - `sorted(list)`: Returns a new copy of the list in sorted order
- List object methods
 - `append(x)`: Adds `x` to the end of the list
 - `extend(other_list)`: Adds all elements of `other_list` to the end of the list
 - `index(item)`: Returns the index of the first occurrence of `item` in the list or error otherwise
 - `insert(index, x)`: Insert `x` at `index` in the list
 - `pop()`: Removes the item at the end of the list and returns it
 - `pop(index)`: Removes item at `index` from the list and returns it
 - `reverse()`: Reverses the elements in the list
 - `sort()`: sorts the elements in the list
- List operators
 - `list1 + list2`: Returns a new list that contains the elements of `list1` followed by the elements of `list2`
 - `list * int`: Returns a new list that contains the items in `list` repeated `int` times
 - `item in list`: Returns True if `item` is an element of `list`, False otherwise

Sets

- Creating new sets
 - `set()` creates empty set
 - `{elt1, elt2, ...}` creates a new set with the given elements
 - `set(iterable)` creates a set from any iterable object (e.g., string, list)
- The following functions are built-in and answer questions about sets
 - `len(set)`: Returns the number of elements in the set
- Set object methods
 - `add(elt)`: Adds `elt` to the set
 - `clear()`: Removes all elements from the set
 - `pop()`: Removes an arbitrary element from the set and returns it
 - `remove(elt)`: Removes `elt` from the set
- Set operators
 - `elt in set`: Returns True if `elt` is an element of `set`, False otherwise
 - `set1 <= set2`: Returns True if `set1` is a subset of `set2` (every element of `set1` is in `set2`), False otherwise
 - `set1 | set2`: Returns union of the two sets (new set with elements from both set)
 - `set1 & set2`: Returns intersection of the two sets (new set with only elements common to both sets)
 - `set1 - set2`: Returns set difference (new set with elements `set1` not in `set2`)

Dictionaries

- Creating new dictionaries
 - `{}` creates empty dictionary
 - `{key1:value1, key2:value2, ...}` creates a new dictionary with key-value pairs
- The following functions are built-in and answer questions about dictionaries
 - `len(dict)`: Returns the number of entries (key-value pairs) in the dictionary
- Dictionary object methods

clear(): Removes all entries from the dictionary

keys(): Returns an iterable object of all the keys in the dictionary

values(): Returns an iterable object of all the values in the dictionary

items(): Returns an iterable object of all (key, value) tuples in the dictionary

get(key[, item]): Returns value associated with **key** if in dictionary, **item** otherwise. **item** defaults to None.

- Dictionary operators

item in dict: Returns True if **item** is in the keys of **dict**, False otherwise

Tuples

- Creating new tuples

() creates empty tuple

(object1, object2, ...) creates tuple containing objects

- The following functions are built-in and answer questions about tuples

len(tuple): Returns the number of elements in the tuple

- Tuple operators

item in tuple: Returns True if **item** is contained in **tuple**, False otherwise

tuple1 + tuple2: Returns a new tuple that is the concatenation of **tuple1** and **tuple2**

Modules

- **random** module

randint(a, b): Return a random integer N such that $a \leq N \leq b$

uniform(a, b): Return a random floating point number N such that $a \leq N \leq b$

- **math** module

sqrt(num): Return the square root of **num**

- **numpy** module (**import numpy as np**)

np.array([10, 12, 14, 20]): creates 1-D vector from list

a+b a-b a*b a/b: element-wise operations on vector

a>3: element-wise comparison (returns boolean vector)

np.sqrt(a): compute element-wise sqrt

np.power(a, exp): raise **a** to the power **exp** element-wise

len(x): number of elements in a vector

np.sum(x), np.max(x), np.min(x), np.mean(x): compute sum, max, min, mean of vector

- **datascience** module (**import datascience as ds**)

ds.table().with_columns('a', [1,2], 'b', [3,4]): Create table with columns **a** and **b**

t["b"], t["b"]=: Evaluate to column named **c** in table **t** as a vector, create/assign to column named **c**

t.with_column('b', [1,2]): Return table **t** with new column named **b**

t.select(["a", "b"]): Evaluate to the subset of table **t** with just columns named **a** and **b**

t.where(expr): Extract rows of table **t** for indices at which **expr** is True

- **matplotlib** module (**import matplotlib.pyplot as plt**)

plt.plot(x, y): add data in iterables **x** and **y** to the plot

plt.show(): display the graph

plt.xlabel(string): label the x-axis with **string** (similarly **pyplot.ylabel**)

plt.title(string): set **string** as the title of the plot