CEBT1100 Introduction to Data Analysis and Python

Lists, Dictionaries and Tuples

Lists

Creating Lists

- A list can contain mixed types of items (it's dynamic).
- In languages like Java, all list items need to be the same type.
- List definition looks very similar to a format called "JSON" which we will see later.

```
alist = [1, 42, "Hello"]
```

- A multidimensional list can be created by putting a "list within a list". We will look more at this later.

```
blist = [1, 42, "Hello", ["a", "b", "c"], 2, 3]
```

Accessing the List

- To access a member of the list, use the indexer.
- Indexer is "0 based", position 1 is equal to 0.

```
print(alist[1]) # Print second item.
```

Looping Through a List

- The "for" loop is the most basic way to iterate through a list.
- The for loop here assigns each list item to “a” and does something with that variable in the loop.
- Notice that the for terminates with a “:” and the body of the loop is indented.

```
my_numbers = [2, 4, 8, 16, 32]
for a in my_list:
  print a
```

Getting a Portion of a List (Sublist)

my_new_list = mylistp[3:] # Omit the first three list items

First Two Items	0:2
First Two Items (Alternate)	:2
Third and Fourth Items	2:4
Fourth up to the End	3:
Last two items	-2:

List Operations - Appending

Assign a value to an index

```
mylist[2] = 'new value'
```

Append to the list

```
mylist.append('another value')
longlist = mylist +[4,5,6]
```

Verify by getting the size of the list

```
length = len(mylist)
```

Creating a list with “Range”

- In case you want to create a range of numbers, there is a shortcut.
- To create a static list with a range, use the following declaration. Here we create a list from 2 to 10, skipping 2.
- Notice that range goes to 10, it does not process 11. It is not "inclusive" for the end range.

```
even_numbers = list(range(2,11,2))
```

Explanation

- *list = create a new list. Similar to x = [], or empty list.*
- *range = give the new list a range of numbers.*

List Operations - Deleting

Remove from list using value.

```
mylist.remove('abc')
```

Remove from list using the position (also gets the value).

```
print(mylist.pop(2)) # Removes 3rd item & print.
```

Remove relative to the end

```
mylist.pop(-2) # Removes 3rd to last item
```

Delete all List Items

```
mylist.clear()
```

Another way to delete a list item

```
del mylist[3] # Remove 3rd item
```

Remove relative to the end

```
del mylist[-2:] # Removes last two items
```

Exercise 1
(5 minutes)

Your input will be:

```
mylist2 = ['red', 'green', 'blue']
```

1. Add two colours to the end, 'black' and 'white'.
2. Change the third item to 'yellow'.
3. Delete the colour green by position.
4. Delete the colour red by name.
5. Make a for loop to print each remaining item, capitalized, with a line number in front of it.

Exercise 1 - Output

1 Yellow

2 Black

3 White

Exercise 2
(20 minutes)

- Create a list of 20 numbers, randomly assigned.
- To generate the list, we can use:
- random_list = [random.randrange(1, 100) for i in range(20)]

- Scan the list and display several values using a manual method:
  - The minimum, the maximum, the count and the average.

Common Aggregate Functions

1. Min
   - Get the minimum value in the list. (or alphabetical).
2. Max
   - Get the highest value in the list (or the highest alphabetical letter).
3. Sum
   - Get the sum of values in the list.
4. Len
   - The number of items in the list.

Exercise 3
(10 minutes)

- Repeat exercise 2 using the built-in functions.
  - Use the “min”, “max”, “len” and “sum” functions.

Exercise 4
(5 minutes)

Use the following list for this exercise.

```
my_list = ['A', 'B', 1, 2, 3]
```

1. Repeat the previous exercise (2B) with the provided list as an input
   - Note: This will give you an error.
2. Discuss what the error is and why it happened.

Combining “for” and “range” to make a new range.

```
squares = [value**2 for value in range(1,11)]
print(squares)
```

Here, squares is a new list being created.

Each member from 1 to 10 is squared.

The result should be 1, 4, 9, …

Exercise 5
(15 minutes)

- Create a new list with the first ten values of 2^n
- Use the "pow" (power) function for this.

pow(value, exponent)

$2^0 = 1$, $2^1 = 2$, $2^2 = 4$, $2^3 = 8$

Etc…

- The new list should look like (including 0):

[1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024]

Multidimensional Lists

- Multidimensional lists are simply list(s) within a list.

```
mylist = ["A", "B", "C"]
mylist2 = ['abc', mylist, [1, 2, 3]]
item1 = mylist2[2][2]
item2 = mylist2[0]
item3 = item2[2]
item4 = mylist2[1][1]
```

- What is the value of item1, 2, 3 and 4?

Copying a List

- To copy a list, use the (:) format.
- The : will output the list verbatim, and allow you to copy it.

```
newList = oldList
```

  - This is wrong.
  - This will create 2 copies of the same list!

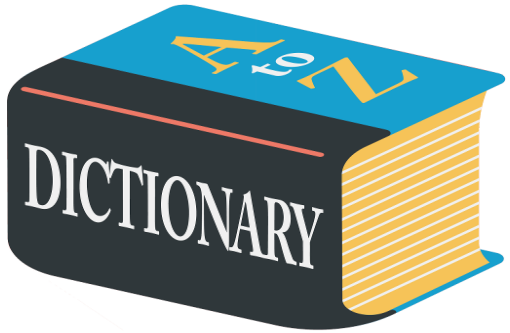
- Use "copy", or create a new list with the ":" range.

```
newList = oldList[:]
newList = oldList.copy()
```

Exercise 6
(10 minutes)

- Given an input list:
- mylist = ["A", "B", "C"]

1. Create a new list "copied_list_1" by copying the list by assignment (a = b).
2. Create a new list "copied_list_2" by copying the list with the deep copy operator (x.copy) or the (":") operator.
3. In the original my_list, change "B" to "X".
4. Display both lists using the "print" command.



Dictionaries

Accessing information using an index

Dictionary

- A dictionary allows us to store a value we need to reference later, by using a key.
- No two entries must share the same key (unique)
- The purpose is to find entries very quickly using a key.
- The list input is in the "JSON" format.

- Key for the first item is "us" and the value is "USA".

```
countries = {'us':'USA', 'fr':'France', 'uk':'Eng'}
print countries['uk']
```

A safer way to get values from the dictionary

- Using the dataset.get method, we can specify the name of the key, and the value to return if not found.

```
countries = {'us':'USA', 'fr':'France', 'uk':'Eng'}

print (countries.get('zb', 'Unknown'))
```

Result:

```
Unknown
```

Adding and Updating Items

- Add a new item, Germany

```
countries['de'] = 'germany' # New item
```

- Update an item (2 ways)

```
countries['de'] = 'Germany'
countries.update({'de':'Germany'})
```

Deleting Items

- Delete an Item

```
del countries['de']
```

- Clear all Items

```
countries.clear()
```

- Delete the dictionary completely

```
del countries
```

Dictionary Iteration

```
countries = {'us': 'USA', 'fr': 'France', 'ca': 'Canada'}
```

- for loop mechanics
  - The for loop only returns the key by default.
- Get full items

```
for key, value in my_dict.items()
    print(key, value)
```

- Combine both

```
for key in my_dict
    print(my_dict[key])
```

More about Dictionaries

- Get a list of the keys in the list.

  dict.keys()
- Get a list of (index and value) tuples

  dict.items()
- Test existence of a key value (Important)
  - This returns a Boolean true or false.

  found = 'ca' in countries

Determining if a Value is in a dictionary Example

- Use the “in” command to do a quick search.

```
countries = {'us': 'USA',
             'fr': 'France',
             'uk': 'United Kingdom’}

print("fr" in countries) TRUE
print("gr" not in countries) TRUE
print("ab" in countries) FALSE
```

- Remember Boolean expressions are “True” and “False”

Exercise 1
(20 minutes)

- Using a starter PY file, convert a price from USD to CAD and display the result in a readable way. You must get all the information from the provided dictionary files. The price and source currency is listed on top of the file.

1. Open the file "exercise_5_dictionary.py"
2. Look up the currency code exchange rate.
3. Look up the currency code name.
4. Print the original and converted prices along with the item name (see output next slide)

Exercise 1 (Output)

Item name: Bluray Movie

Original Price: $19.99 (USD)

Purchased Price: $26.74 (CAD)

Tuples

- A tuple is almost exactly like a list with one difference.
- The tuple is “immutable”, it cannot be changed once it’s created.
- Tuple format: Uses parentheses instead of square brackets.
  - `my_tuple = (“a”, “b”, 12, 13, 14)`

Tuple Use

- Accessing a tuple is exactly the same as a list (reading).
- Writing to a tuple, once created, cannot be done. You must COPY the tuple into a new list.
  - `my_list_from_tuple = list(my_tuple)`

- Try this: Create a tuple, then try to change a value.

- Why create a tuple? Because tuples are faster to work with for the computer when using large sets of data, due to their low overhead (no need to manage changes and growing lists)

UNIVERSITÉ Concordia UNIVERSITY

CONCORDIA.CA