CSI 1401 Week 1 Resources

by Allen Yan

Hello fellow students! My name is Allen Yan and I am the Master Tutor of CSI 1401 Introduction to Programming I for the Fall 2020 semester. Starting from this week, I will be making supplemental resources like this one every week to help you learn the materials in this class. These resources are intended to provide a review of the key concepts covered during the week, as well as to offer a different perspective on explaining some of the more challenging concepts. Do please note that these resources are not intended to replace the course's textbook and your professor, as they are still your most helpful resources and should be the first place you go to when running into problems.

In addition to the resources, I will be leading a group tutoring session on CSI 1401 every Wednesday from 7:00 pm to 8:00 pm (central time). The session will be conducted online via Microsoft Meetings, where I will be available to provide interactive help to students. If the above time window does not work for you, or if you need additional help, Baylor's Tutoring Center also provides 1-on-1 online tutoring appointments. For more information on signing up for group tutoring or individual tutoring, please visit https://www.baylor.edu/tutoring.

Week 1 Important Topics

Chapter 3 - Types

- Strings
- Lists
- Sets
- Dictionary

1. Strings

String is Python's (and many other programming languages) way of storing any "text phrases". A string is any sequence of characters wrapped in single or double quotes. The characters in strings are internally labeled with numeral "index" by Python. For example, consider the example below with the string "Hello 123":

Character	H	e	l	l	o		1	2	3
Index	0	1	2	3	4	5	6	7	8

There are a couple things worth noting: First, the sequence of index for the characters in strings start at the number 0, so any strings handled by Python will always have the first character labeled with index 0 and the last character labeled index (length – 1). Second, the empty space in the middle of the string is also treated as a character by Python, this is also true for other characters with empty space-like appearances, such as the newline character. ('\n')

If you had prior experience programming in other languages, you may be unaccustomed to one feature of Python strings – they cannot be modified. Therefore, operations like the following are not allowed:

```
my_string = ‘hello’
my_string[0] = ‘H’
```

As an FYI, to achieve string modification similar to other programming languages, such as C++, you can a store a string in a list (lists are mentioned in the next section), and then process them. Like this:

```
my_string = list(‘hello’)
my_string[0] = ‘H’
```

When you need to print a string stored as a list, you can simply convert it back into a string by using the join function, like below:

```
print(‘’.join(my_string))
```

One last important utility for strings is that they can be concatenated by using the + operator. It is useful for combining multiple strings into a single one:

```
string1 = ‘hello ’
string2 = ‘world’
string3 = string1 + string2
# string3 is now ‘hello world’
```

2. Lists

List is a container in Python used to store collections of data in a sequential manner. Any data stored in a list is internally labeled with indices similar to how characters are indexed in strings. Lists provide users with a quick and easy way to handle large collections of similar or related data. Each individual piece of data stored in the list is called an element. One nifty feature of Python lists is that elements of different types can be stored in the same list, which is something not supported by equivalent containers in many other common programming languages (like arrays in C++ and Java). For example, list declarations like the following are allowed:

```
stuff = [1, 3.14159, ‘lol’]
```

Python list comes with a list of handy operations to help the user (no pun intended), they are shown in the table below:

Operation	**Description**
`list[i]`	Returns list element at index i
`list.append(val)`	Adds val to the end of the list
`list.pop(i)`	Removes list element at index i
`list.remove(val)`	Removes the first element with value equal to val
`len(list)`	Returns length of the list
`list1 + list2`	Returns new list with list2 concatenated to the end of list1
`min(list)`	Returns element with the smallest value in list
`max(list)`	Returns element with the largest value in list

`sum(list)`	Returns sum of all elements in the list, requires all elements to be numbers
`list.index(val)`	Find the index of the first element in the list matching the value val
`list.count(val)`	Returns the number of elements in the list matching the value val

3. Set

Set is another useful container in Python for organizing data. Sets differ from lists in two ways: first, the elements in a set are not ordered by indices; and second, no duplicating elements can be stored in the same set, if inserting a duplicating element is attempted, the set simply ignores it.

Sets can be declared in two ways:

```
set1 = {1, 2, 3}
set2 = set([1, 2, 3])
```

Sets are useful in situations where the user wants to keep a record of unique elements. A table of useful set operations is provided below.

Operation	**Description**
`len(set)`	Returns length of the set
`set1.update(set2)`	Adds the elements in set2 into set1, discards duplicates
`set.add(val)`	Inserts val into the set
`set.remove(val)`	Removes val from the set. Raises KeyError if val is not found
`set.pop()`	Removes a random element from the set
`set.clear()`	Empties the set
`set.intersection(set1, set2, set3…)`	Returns new set containing only elements shared between set and all provided sets
`set.union(set1, set2, set3…)`	Joins elements of set and all provided sets in a new set, duplicates are discarded
`set.difference(set1, set2, set3…)`	Returns new set containing only elements of set not found in any of the provided sets
`set1.symmetric_difference(set2)`	Returns new set containing only elements that only appears in set1 or set2, but not both

4. Dictionary

Dictionary is one of the most powerful basic containers in Python. (Its equivalent in C++ and Java, the hashmap, are usually not introduced until advanced classes.) Dictionaries allow users to store data using associative relationships: insertions in dictionaries must be done using key-value pairs, the key is a term that describes it's associated value and will be used to access the stored value in the dictionary later, and the value will be the actual data stored.

Dictionaries declaration has a somewhat complicated syntax, refer to the example below:

```
presidents = {
    'George W. Bush': 2001
    'Barack Obama': 2009
    'Donald Trump': 2017
}
```

The above example creates a dictionary named presidents with names of U.S. presidents as the key and their year of inauguration as the value.

A table of common dictionary operations is listed below.

Operation	**Description**
`dict[k] = val`	Inserts value val into the dictionary with key k, if k already exists in the dictionary, then the associated value is updated to val
`del dict[k]`	Deletes the key k and its associated value from the dictionary

Useful Resources

Professor Fry and your zyBooks for this course are immensely helpful resources for this class. I would always recommend them over the resources I provide. Having said that, I will list some resources below that I find helpful when learning to program in Python.

- Basic Python Tutorial on GeeksforGeeks: https://www.geeksforgeeks.org/python-programming-language/
    - This page provides links to detailed explanations to many entry-level Python concepts along with examples. I encourage taking a look at it if the examples in your textbook were not clear enough.
- Official Python Documentation: https://docs.python.org/3/
    - This may be a bit heavy-handed for a beginner-level programmer since the official Python documentation is very thorough and technical. However, learning how to read official documentations is crucial to becoming a good programmer, because the official documentation contains information on everything you can find about Python elsewhere and more. Therefore, I encourage slowly easing yourself into learning how to read the official documentation
- Python Installation for Windows (recommended by Professor Fry): https://youtu.be/AUiM1UaRCPc
- Python Installation for Mac (recommended by Professor Fry): https://youtu.be/oyzH4M6X6F4