CS177 Recitation

Python Sequences
Strings

• A string is a list of characters
• Strings are created by surrounding a sequence of characters with quotation marks and assigning them to a variable

```python
>>> myString = "hello"
>>> print(myString)
hello
```

• Concatenate strings using the plus operator

```python
>>> print("hello"+" world")
hello world
```
Strings

• There are some special characters in strings that are not printed in the typical way
• Two popular special characters:
  – \n  : means to print a new line
  – \t : means to print a tab character
• >>> print(“one\ttwo\ttthree”)
onetwothreethree
• >>> print(“line 1\nline 2\nline 4”)
line 1
line 2
line 4
Special Characters

- We can encode and decode ASCII

```python
>>> ord('b')
98
>>> chr(98)
'b'
```
Strings Indexing

- You can get a substring, or slice, from a string by using a range expression index.
- A range expression uses a colon to indicate the range.

```python
>>> myString = "hello"
>>> print(myString[1:4])
ell
```
Strings methods

• Every variable in Python is an object
• Objects may have methods
• Methods are like functions except they apply to the object
• To execute a method on an object we use the dot notation
• The dot notation takes the form `object.method`

```python
>>> myString = "hello"
>>> myString.capitalize()
'Hello'
```
Strings methods

• Change the case of a string:

```python
>>> myString = "Hello"
>>> myString = myString.lower()
>>> myString
'hello'
>>> myString.upper()
'HELLO'
```
Strings methods

• Searching a string:

```python
>>> myString = "hello"
>>> myString.find("ell")
1
>>> myString.find("ll")
2
>>> myString.find("b")
-1
```
Lists

• We can also store more complex elements into an list.
• $y$ is an example of a list of string. Each element is a string. We could expand it as follows:

\[
y = ["ABCD", "BCD", "CD", "D"]
\]

• As you can see each element can be a different length. They can also be different types:

\[
y = ["ABCD", [1,2,3], "CD", "D"]
\]

• Suppose we wanted to extract the value 3

\[
y[1][2]
\]
List methods

• Similar as string methods, we need to use the dot notation `object.method`. ‘Object’ now is the list.

```python
>>> c = [1, 2, 3, 4, 5]
>>> c.append(6)
```

– Results in `c` having an additional element:

```
[1, 2, 3, 4, 5, 6]
```
List methods

```python
>>> L = [0, 1, 2, 0]
>>> L.reverse()
>>> print(L)
[0, 2, 1, 0]
>>> L.remove(0)
>>> print(L)
[2, 1, 0]
>>> L.remove(0)
>>> print(L)
[2, 1]
```
List methods

```python
>>> L = [0, 1, 2]
>>> L.extend([4, 5, 6])
>>> print(L)
[0, 1, 2, 4, 5, 6]
>>> L.extend(["Hello"])
>>> print(L)
[0, 1, 2, 4, 5, 6, "hello"]
>>> L.insert(0, "a")
>>> print(L)
["a", 0, 1, 2, 4, 5, 6, "hello"]
>>> L.insert(2, "a")
>>> print(L)
["a", 0, "a", 1, 2, 4, 5, 6, "hello"]
```
Files

• A file is a sequence of data that is stored in secondary memory (disk drive).
• Different types of files:
  – Text files: store text, can be read by humans
  – Binary files: store data, cannot be read directly
• A file usually contains more than one line of text.
• Python uses the standard newline character (\n) to mark line breaks.
File processing

• Create or open, read or write, close
• For example

    infile = open(someFile, "r")
    # file processing codes
   (infile.close()
File methods

- `<file>.read()` – returns the entire remaining contents of the file as a single (possibly large, multi-line) string
- `<file>.readline()` – returns the next line of the file. This is all text up to and including the next newline character
- `<file>.readlines()` – returns a list of the remaining lines in the file. Each list item is a single line including the newline characters.
File methods

• An example:
  ```python
  infile = open(someFile, "r")
  for i in range(5):
      line = infile.readline()
      print (line[:-1])
  ```
  This reads the first 5 lines of a file
  Slicing is used to strip out the newline characters at the ends of the lines

• Another example:
  ```python
  infile = open(someFile, "r")
  for line in infile.readlines():
      # Line processing here
  infile.close()
  ```
File methods

• Write into a file: `<file>.write()`
• An example:

```python
>>> file = open("Letter.txt", "w")
>>> file.write("Dear friend:
")
>>> file.write("I love you.
")
>>> file.write("Sincerely,
")
>>> file.write("XXX")
>>> file.close()
```
File methods

• Letter.txt
Dear friend:
I love you.

Sincerely,

XXX
Any question?