CS 1112: Introduction To Programming

Lists

Dr. Nada Basit
basit[*at*]Virginia[*dot*]edu

Spring 2024

Friendly Reminders

- Your ***safety*** and ***comfort*** is important!
  - If you choose to wear a mask you are welcome to do so
  - *We will interpret wearing a mask as being considerate and caring of others in the classroom (not that you are sick), and realize that some may choose to mask to remain distanced*

- Be an ***active*** participant in your learning!
  You're welcome and ***encouraged*** to ask questions during class!

- If you feel ***unwell***, or think you are, please stay home
  - *We will work with you!*
  - Get some rest ☺
  - View the recorded lectures – *please allow 24-48 hours to post*
  - *Contact us!*

Announcements

- **Quiz 5** is due by 11:00pm on 3/18 (***Monday***)
- **PA04** is due by 11:00pm on Wednesday (***3/20***)!
  - Submit on Gradescope: your .py file, and a PDF of your reflection
  - Please be mindful about submitting the right kind of files (.py file and .pdf file) as well as submitting the .py file that is named correctly (see assignment document for full details)
  - A note about submitting on Gradescope: you can submit an **UNLIMITED** number of times prior to the deadline. Look at the score you got, if you have some points taken off, that's ok, go back and fix your code and resubmit! Do this as often as you like BEFORE the assignment deadline. You cannot resubmit after the deadline.
  - ***REMEMBER ALSO***: You have a grace period of 24 hours to submit your PAs!
- **Exam 1**: if you notice grader-error, don't hesitate to let me or the TAs know!
  - Exam 1 error corrections going on!

Properties of some collections

Today

Type	**Stores**	**Syntax**
Range	ints	`range(3,7)`
String	characters	`"Hello", "abc 123"`
List	anything	`[1,2,3,6,"hello"]`
Tuple	anything	`(1,2,3,6,"hello")`

Next Week

Dictionary	key:value pairs	`{17:"hi", 29:"bye"}`
Set	anything	`{1,2,6,"hi"}`

Kinds of Collections (the word “collection” in python technically has a much more strict meaning)

Sequence types

```
str     # string
range
list
tuple
```

- Order Matters
- Repetition of Items OK
- Counting starts at 0
- Collection[index] gives a specific value from the collection

Non-Sequence types

```
dict  # dictionary
set
```

- Ordering - not really?
  - first item is not at index 0
  - dict - insertion order only
  - set - no ordering
- No Repetition of Items Allowed

Lists

- A way of collecting together a bunch of things, and giving them all one **name**
- Looks like this: [*the*, *things*, *separated*, *by*, *commas*] ☺

```
my_list1 = [1, 2, 3, 4]

my_list2 = ['one', 'two', 'three', 'four']

my_list3 = [1, 'two', 3, 'four']

len(my_list3) # gives the number of things in the list (in this case 4)

my_list2[i] # gives the ith thing in the list (starts at 0)
```

Lists

One of the most popular and versatile data structures!

(Yes, it is completely *mutable*)

Some common **list** methods

```
my_list = ['apple', 'banana']

my_list.append('watermelon’)

# Where does append put the new item?
```

Some common **list** methods

```
my_list = ['apple', 'banana']

my_list.append('watermelon’)

# ['apple', 'banana', 'watermelon']
```

Some common **list** methods

```
# ['apple', 'banana', 'watermelon']

my_list.insert(1, 'orange')
```

Some common **list** methods

```
# ['apple', 'banana', 'watermelon']

my_list.insert(1, 'orange')

# ['apple', 'orange', 'banana', 'watermelon']
```

Some common **list** methods

```
# ['apple', 'orange', 'banana', 'watermelon']

my_list.remove('apple')
```

Some common **list** methods

```
# ['apple', 'orange', 'banana', 'watermelon']

my_list.remove('apple')

# [~~'apple',~~ 'orange', 'banana', 'watermelon’]

# ['orange', 'banana', 'watermelon']
```

Some common **list** methods

```
my_list.index('banana')

# ['orange', 'banana', 'watermelon']
```

Some common **list** methods

```
my_list.index('banana')

# ['orange', 'banana', 'watermelon’]

# Answer:  1
```

Some common **list** methods

```
# ['orange', 'banana', 'watermelon’]

my_list.sort()
```

Some common **list** methods

```
# ['orange', 'banana', 'watermelon’]

my_list.sort()

# ['banana', 'orange', 'watermelon’]
# Since strings: sorts in alphabetical order
```

Some common **list** methods

```
my_list = ['apple', 'banana']
my_list.append('watermelon') # ['apple', 'banana', 'watermelon']
my_list.insert(1, 'orange’)
     # ['apple', 'orange', 'banana', 'watermelon']
my_list.remove('apple') # ['orange', 'banana', 'watermelon']
my_list.index('banana') # 1
my_list.sort() # ['banana', 'orange', 'watermelon']
```

Tuples

Tuples

- Similar to lists, but can’t be **modified** (more on this in future classes)
  - Python can access them *faster* than lists *(but this only matters for really, really, big lists/tuples)*
- Some functions that we will use return tuples instead of returning lists
- We can **create a list** from a tuple with type-casting
  - `new_list = list(old_tuple)`
- When given the option between creating a list and creating a tuple, we will almost always choose a **list**

```
my_tuple1 = (1,2,3,4)
my_tuple2 = ('one', 'two', 'three', 'four')
my_tuple3 = (1, 'two', 3, 'four')
len(my_tuple3) # gives the number of things in the tuple(in this case 4)
my_tuple2[i] # gives the ith thing in the tuple (starts at 0)
```

Lists vs. Tuples

This code ...

```python
# list vs. tuple comparison
my_list = [1, 2.7, 'wahoo', True]
my_tuple = (1, 2.7, 'wahoo', True)

for thing in my_list:
   print(thing)

for thing in my_tuple:
   print(thing)

print(my_list[2])
print(my_tuple[2])

print(my_list)
my_list[0] = 100
print(my_list)

print(my_tuple)
my_tuple[0] = 100  # Error - tuples are
immutable
print(my_tuple)
```

Produces this output

```
1
2.7
wahoo
True

1
2.7
wahoo
True

wahoo
wahoo

[1, 2.7, 'wahoo', True]
[100, 2.7, 'wahoo', True]


(1, 2.7, 'wahoo', True)
my_tuple[0] = 100
TypeError: 'tuple' object does not support
item assignment
```

Operating on Collections

- Let’s practice adding to collections -
  - **Adding <u>one</u> item to the collection**
    - Strings:
    - Lists:
    - Tuples:
    - Ranges:
  - **Adding <u>several</u> items to the collection**
    - Strings:
    - Lists:
    - Tuples:
    - Ranges:

Operating on Collections

- Let’s practice adding to collections -
  - **Adding <u>one</u> item to the collection**
    - Strings: `my_string = “hello” + “w”`
    - Lists: `my_list.append(thing)`
    - Tuples: you should probably use a list
    - Ranges: can’t do this
  - **Adding <u>several</u> items to the collection**
    - Strings: `my_string = “hello” + “goodbye”`
    - Lists: `my_list = list1 + list2`
    - Tuples: `my_tuple = tuple1 + tuple2`
    - Ranges: can’t do this

Collection types we know: strings, lists, ranges, tuples

```python
# a program to look at collections
string1 = "1234"
tuple1 = (1,2,3,4)
list1 = [1,2,3,4]
range1 = range(1,5)

my_group = [string1, tuple1, list1, range1]

def collection_info(the_collection):
   print('-----------------')
   print(type(the_collection), the_collection)
   for each in the_collection:
      print(type(each), each)

print("my_group - ", my_group)
for item in my_group:
   collection_info(item)
```

```
my_group -  ['1234', (1, 2, 3, 4), [1, 2, 3, 4], range(1,
5)]
-----------------
<class 'str'> 1234
<class 'str'> 1
<class 'str'> 2
<class 'str'> 3
<class 'str'> 4
-----------------
<class 'tuple'> (1, 2, 3, 4)
<class 'int'> 1
<class 'int'> 2
<class 'int'> 3
<class 'int'> 4
-----------------
<class 'list'> [1, 2, 3, 4]
<class 'int'> 1
<class 'int'> 2
<class 'int'> 3
<class 'int'> 4
-----------------
<class 'range'> range(1, 5)
<class 'int'> 1
<class 'int'> 2
<class 'int'> 3
<class 'int'> 4
```

Code

```
# list vs. tuple comparison
my_list = [1, 2.7, 'wahoo', True]
my_tuple = (1, 2.7, 'wahoo', True)
for thing in my_list:
   print(thing)
for thing in my_tuple:
   print(thing)
print(my_list[2])
print(my_tuple[2])
print(my_list)
my_list[0] = 100
print(my_list)
print(my_tuple)
my_tuple[0] = 100  # Error - tuples
are immutable
print(my_tuple)
```

Output

```
1
2.7
wahoo
True
1
2.7
wahoo
True
wahoo
wahoo
[1, 2.7, 'wahoo', True]
[100, 2.7, 'wahoo', True]
(1, 2.7, 'wahoo', True)
my_tuple[0] = 100
TypeError: 'tuple' object does not support
item assignment
```

Quick & Fun Survey Questions

Get to know your peers! ☺ **WOULD YOU RATHER…**

Have the ability to see 10 minutes into the future <u>or</u> 150 years into the future??

Python Demonstration

Let's jump on PyCharm!

`lists.py` - Lots of List examples!

```
        mirror_mod.use_y = True
        mirror_mod.use_z = False
    elif _operation == "MIRROR_Z":
        mirror_mod.use_x = False
        mirror_mod.use_y = False
        mirror_mod.use_z = True

        #selection at the end -add back the deselec
    mirror_ob.select= 1
    modifier_ob.select=1
    bpy.context.scene.objects.active = modifier_ob
    print("Selected" + str(modifier_ob)) # modifier
        #mirror_ob.select = 0
```

In-Class "lab" Activity!

Activity for Today!

- In pairs or groups up to three work on the following activity.
- **lists_ica.py**
- *Practice writing code that uses a list. Also practice with sorting and searching*

Remember to check-in with a TA before leaving class today!

Reminder: CS Laptop Loaner Program

- This course requires students to have a **laptop**
- I realize that not everybody might have one (nor necessarily need one for their desired major / path…)
- If you do not have a laptop for any reason… ***not to worry!***
- The CS department's Systems staff has a notebook / laptop loaner program and will be able to loan you a notebook / laptop computer for the duration of the semester if you don't have one or if you cannot afford one.
  - Also available if your laptop is broken and under repair, we can arrange for you to receive a loaner laptop for a week or two until your own laptop is fixed

Interested? Link: https://www.cs.virginia.edu/wiki/doku.php?id=cs_laptop_loaner

I am happy to be your sponsor. Please let me know.