CS 1112: Introduction To Programming

Tuples

Dr. Nada Basit

basit[*at*]Virginia[*dot*]edu

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*
- Remember to always be **kind**, **respectful**, **supportive**, **compassionate** and **mindful of others**! ☺
- 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
  - *Contact us!* *We will work with you!*
  - Get some rest ☺
  - View the recorded lectures – *please allow 24-48 hours to post*

Announcements

- **Quiz 5 (Loops and Strings)** is due by 11:00pm on 3/17 (***Monday -- Today***)!
- **PA04** is due by 11:00pm on 3/19 (***Wednesday***)!
  - 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 (up to 10 pts)** going on! Deadline: March 26, 2025 (11:00pm)
  - **Exam 1 grader error** (submit form) Deadline: March 26, 2025 (11:00pm)

Properties of some collections

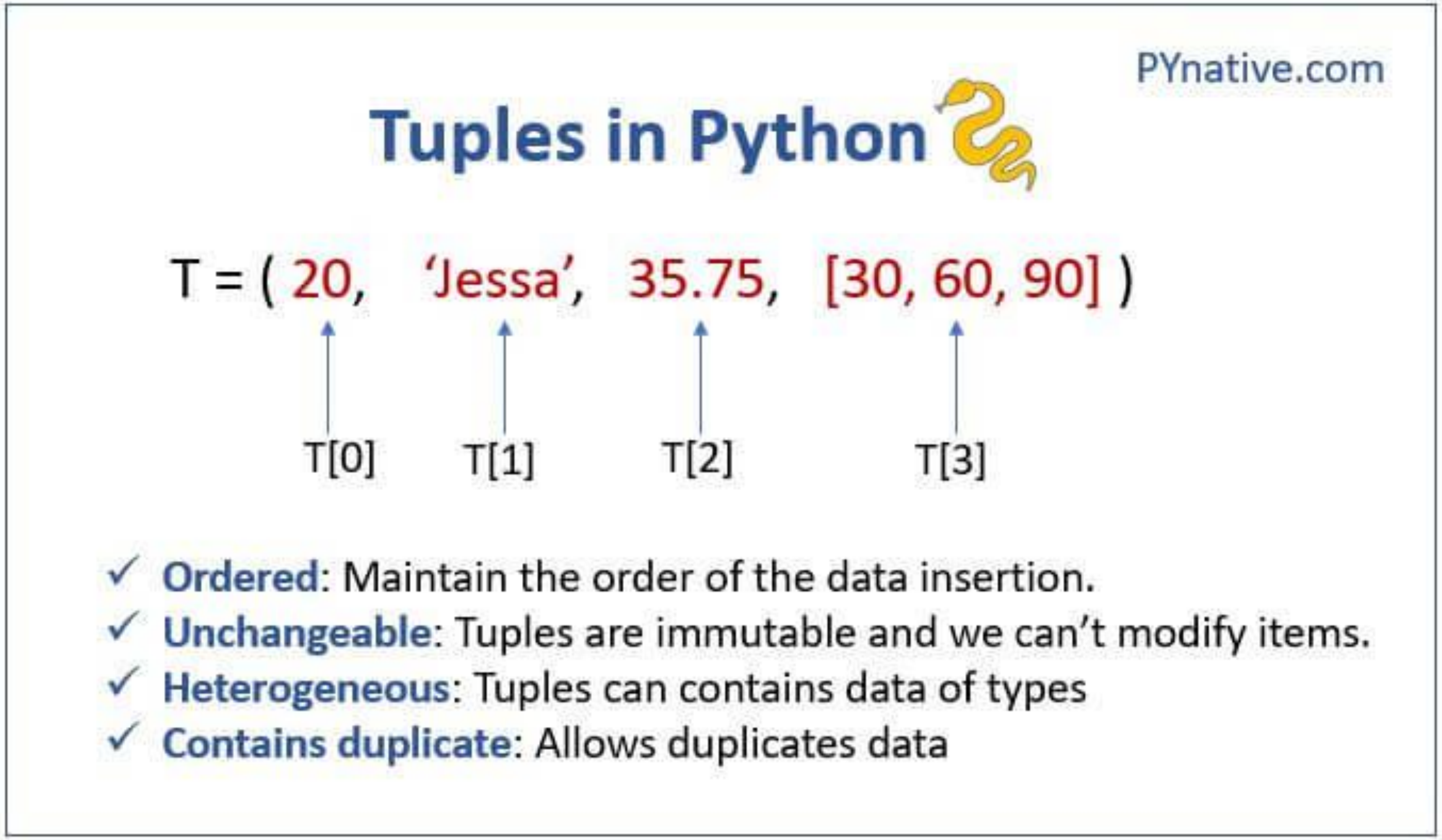
Type	**Stores**	**Syntax**
Range	ints	`range(3,7)`
String	characters	`"Hello", "abc 123"`
List	anything	`[1,2,3,6,"hello"]`
Tuple	anything	`(1,2,3,6,"hello")`

Today

Next Week

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

Tuples in Python


PYnative.com

Tuples in Python

T = (20, 'Jessa', 35.75, [30, 60, 90])

- ✓ **Ordered**: Maintain the order of the data insertion.
- ✓ **Unchangeable**: Tuples are immutable and we can't modify items.
- ✓ **Heterogeneous**: Tuples can contains data of types
- ✓ **Contains duplicate**: Allows duplicates data

Tuples

Tuples are very similar to the List data structure – but have one main difference!

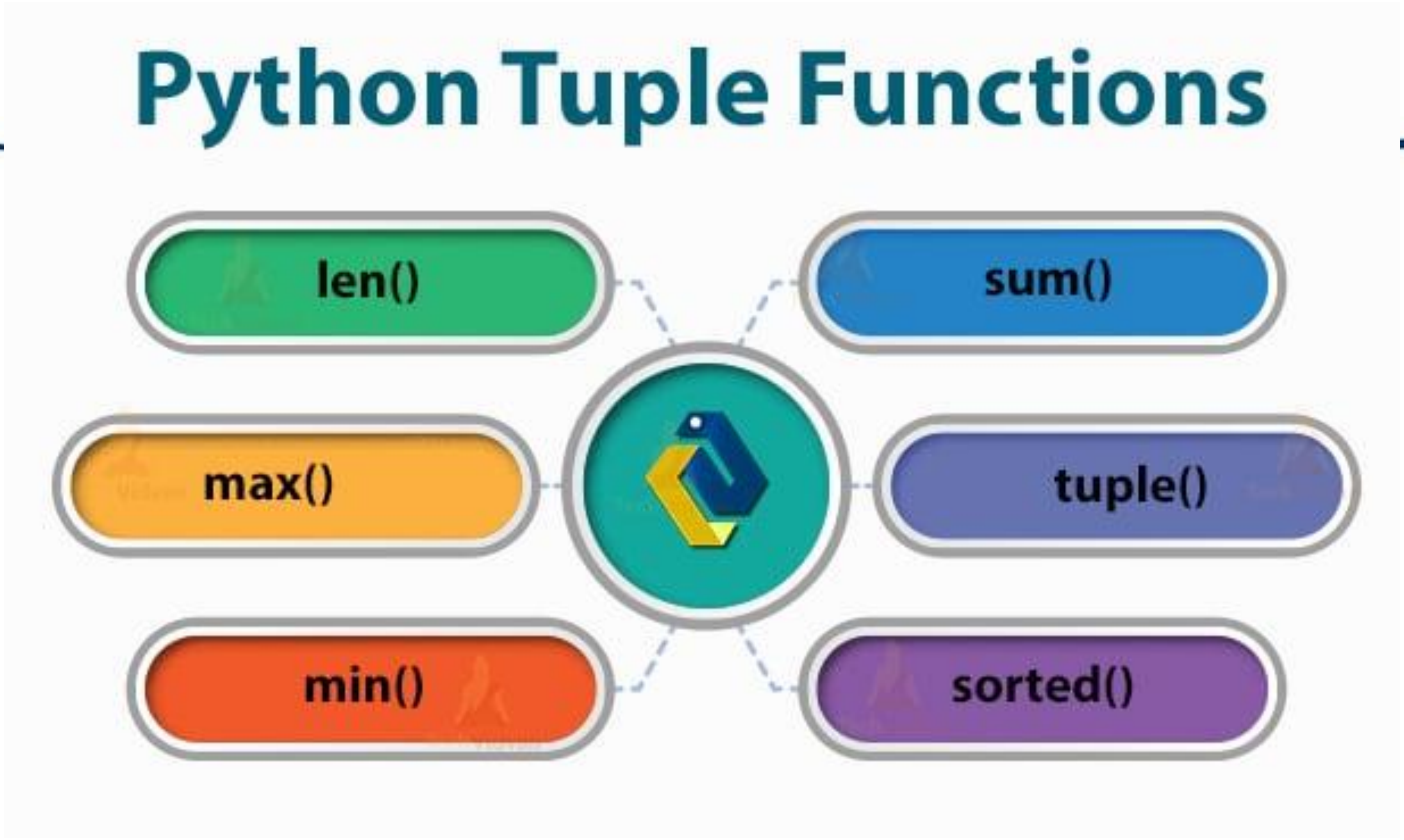
-- Tuples are *immutable* (Tuples are unchangeable, which means we cannot add/delete/etc)

When given the option between creating a list and creating a tuple, we will almost always choose a **list**

Tuples

```
my_tuple0 = tuple((‘Nina’,30, 5.75, [7, 11]))
      # creating a tuple using the tuple constructor

my_tuple1 = (1,2,3,4)  # creating a tuple using parenthesis ()

my_tuple2 = ('one', 'two', 'three', 'four’)  # a tuple of strings (same type)

my_tuple3 = (1, 'two', 3, 'four’)  # a tuple of heterogenous items (types)

### Given tuples are immutable, there’s no point in creating an empty tuple! ☺

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)
```

(Main Tuple functions, but there are others…)



Because of its *immutability*, we cannot use some of the methods like:

1. `append()` and `insert()`, because we cannot add elements to the tuple
2. `remove()` and `pop()`, because we cannot delete elements of a tuple

Cannot use mutating methods…!

```
my_tuple = () # Empty tuple


my_tuple = ('Virginia', 'California')
my_tuple.append('Indiana') !!!!!

# Tuples are immutable – so, cannot append!
Also, not helpful to create an empty tuple!
```

Can use **indexing** as we've seen before!

```
my_tuple = ('Purple', 'Green', 'Yellow')

print("First element:", my_tuple[0])

# First element: Purple
```

Can use **indexing** as we’ve seen before!

```
my_tuple = ('Purple', 'Green', 'Yellow')
print("Last element:", my_tuple[-1])

# Last element: Yellow
```

Some common **tuple** methods [index]

```
my_tuple = ('Purple', 'Green', 'Yellow')

my_tuple.index('Yellow')
```

Some common **tuple** methods [**index**]

```
my_tuple = ('Purple', 'Green', 'Yellow')

my_tuple.index('Yellow')

# Answer:  2
```

Some common **tuple** methods [**count**]

```
my_tuple = (22, 45, 23, 78, 22, 22, 77)
my_tuple.count(22)


alpha = (“a”, “b”, “a”, “c”, “d”)
alpha.count(“b”)
```

Some common **tuple** methods [**count**]

```
my_tuple = (22, 45, 23, 78, 22, 22, 77)

my_tuple.count(22)

# Answer:  3
```

Some common **tuple** methods [**count**]

```
alpha = (“a”, “b”, “a”, “c”, “d”)

alpha.count(“b”)

# Answer:  1
```

Some common **tuple** methods [`min/max`]

```
my_tuple = (22, 45, 23, 78, 22, 22, 77)
print(min(my_tuple))


my_tuple = (22, 45, 23, 78, 22, 22, 77)
print(max(my_tuple))
```

Some common **tuple** methods [min/max]

```
my_tuple = (22, 45, 23, 78, 22, 22, 77)
print(min(my_tuple))  # Answer:  22


my_tuple = (22, 45, 23, 78, 22, 22, 77)
print(max(my_tuple))  # Answer:  78
```

Some common **tuple** methods [**min/max**]

```
my_tuple = (-1,'s',5,6,'t’)

print(max(my_tuple))
```

Some common **tuple** methods [**min/max**]

```
my_tuple = (-1,'s',5,6,'t’)

print(max(my_tuple))
```

Be careful when you have different data types! Values of different data types cannot be compared.

```
# TypeError: '>' not supported between instances of 'str' and 'int'
```

Some common **tuple** methods [**sum**]

```
my_tuple = (22, 45, 23, 78, 22, 22, 77)

print(sum(my_tuple))
```

Some common **tuple** methods [sum]

```
my_tuple = (22, 45, 23, 78, 22, 22, 77)

print(sum(my_tuple))

# Answer:  289
```

Some common **tuple** methods [**sorted**]

```
my_tuple = (“m”,“p”,“n”,“d”,“k”,“s”,“w”)

print(sorted(my_tuple))
```

Some common **tuple** methods [**sorted**]

```
my_tuple = (“m”,“p”,“n”,“d”,“k”,“s”,“w”)

print(sorted(my_tuple))

# Answer:  ['d', 'k', 'm', 'n', 'p', 's', 'w’]
# NOTICE: what did we get back???
```

Some common **tuple** methods [**sorted**]

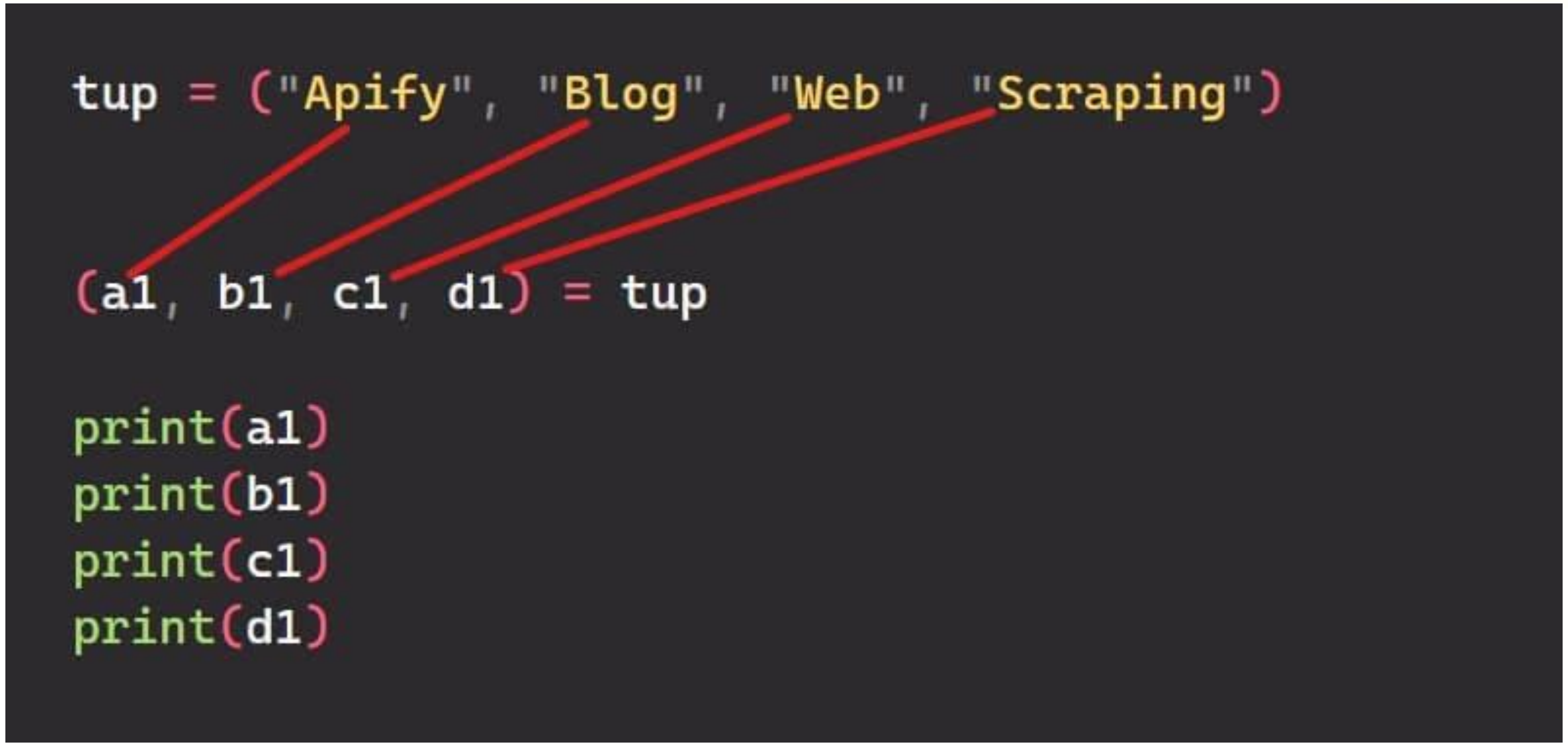
```
my_tuple = (“m”,“p”,“n”,“d”,“k”,“s”,“w”)

print(sorted(my_tuple))

# Answer:  ['d', 'k', 'm', 'n', 'p', 's', 'w’]
# NOTICE: you get a list back, with the elements
in the tuple sorted (obviously the tuple doesn’t
get sorted since it is immutable!) So, if you
print my_tuple afterwards, it would be the SAME.
```

Some common **tuple** methods [**sorted**]

```
my_tuple = (“m”,“p”,“n”,“d”,“k”,“s”,“w”)

print(sorted(my_tuple))
```

Be careful if you have different data types! Values of different data types cannot be compared – and therefore, cannot be sorted.

Unpacking a Tuple… (Can be very handy!)



```
tup = ("Apify", "Blog", "Web", "Scraping")


(a1, b1, c1, d1) = tup

print(a1)
print(b1)
print(c1)
print(d1)
```

https://blog.apify.com/content/images/2023/10/Python-tuple-unpacking.jpg

Some common **tuple** methods

There are **other methods** that can be used with tuples such as **all**() or **any**(). You're welcome to look up these and other functions to discover what they are.

Remember, none of these methods will mutate the elements in the tuple *(due to immutability!)*

Additional Information

Python can access tuples faster than lists.

We can create a list from a tuple with type-casting – using list()

```
      new_list = list(old_tuple)
```

We can create a tuple from a list with type-casting – using tuple()

```
      new_tup = tuple(old_list)
```

We can do **slicing** on tuples (and lists)

We can do **membership** on tuples (and lists) – use keyword “**in**”

Lists vs. Tuples

Reasons to use LISTS

- For flexibility of handling data: adding, deleting, changing (after list is created)
  - Mutability of Lists
- Generally, **lists** are still the better choice unless we're dealing with VERY large structures where speed is important

Reasons to use TUPLES

- When using data that doesn't or shouldn't change
  - Immutability of Tuples
  - (Structure to loop through and access the data without modifying the data)
- Python can access tuples faster than lists *(better performance) due to immutability*
- Tuples are memory efficient compared to lists *due to immutability*

Tuples are immutable and therefore do not have as many methods as a list. A lot of those list methods that we have seen involved mutating the values.
For tuples, we cannot append, remove, etc.
Otherwise, lists and tuples behave in pretty much the same way!

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)
```

Iterating through the elements

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
```

Python Demonstration

Let's jump on PyCharm!

`tuples.py` - Lots of Tuple examples!

Python Demonstration

Let's jump on PyCharm!

`tuples.py` – Some live coding involving tuples! (Weather station data)

```
        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.
- **tuples_ica1.py**
- *Practice writing solutions that utilize tuples*

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.