CMSC201
Computer Science I for Majors

Lecture 12 – Tuples

Last Class We Covered

- Modularity
  - Meaning
  - Benefits
- Program design
  - Top Down Design
  - Top Down Implementation
  - Bottom Up Implementation

Any Questions from Last Time?

Today's Objectives

- To learn about the tuple data structure in Python
- To be able to perform basic operations with tuples including:
  - Creation, conversion, slicing, traversing

- To discuss casting in detail
- To discuss the membership "**`in`**" operator

Tuples

The Tuple Data Structure

- Tuples are sequences of objects
  - Just like lists!

- Lists use square brackets [ ]
- Tuples use parentheses ( )

- They have some other differences, but we'll discuss those later in the semester

Creating a Tuple

- To create a tuple, simply use parentheses around a comma-separated list of items

```
classTup = (201, 202, 341, 313)
```

- How do you think you create an empty tuple?

```
emptyTup = ()
```

- What about a tuple with one element?

One-Element Tuples

- At first glance, you might think that it's this:

```
oneItem = (201)
```

- But this won't work – it's an int (why?)

- Parentheses are used for more than just tuples
  - Also used for order of operations

- To create a one element tuple, use a comma

```
oneItemTuple = (201,)
```

Tuple Operators and Properties

- Tuples are ordered, and can contain more than one type of variable, just like lists
  - They can even contain another tuple!

- Tuples can also be indexed, concatenated, sliced, and can use the `len()` function
  - Just like strings and lists

Tuple Exercises

- You are given the following tuple:

```
hounds = ("Ibizan", "Afghan", "Serbian", "Bassett")
```

- Write pieces of code that do the following:
  1. Print out each element followed by " Hound"
  2. Use **`len()`** to print how many hounds there are
  3. Use slicing to create a tuple called **`oldBreeds`** that includes only the Afghan and Serbian hound
  4. Print out each element in ***reverse*** order

Types and Casting

Finding a Variable's Type

- This is a bit of a review, but we haven't covered this in detail before
- To find what type a variable is, use **`type()`**
- Example:

```
>>> a = 3.0
>>> type(a)
<class 'float'>
```

```
>>> b = "moo"
>>> type(b)
<class 'str'>
```

Casting to a Type

- We can change a value from one type to another using something called ***casting***

- Example:

```
>>> e = 2.718
>>> int(e)
2
>>> str(e)
'2.718'
```

The type you want to cast to, then the variable whose value you want to cast

This code means:
"show what e is as an integer"

Casting to a Type: Assignment

- Casting alone doesn't change the variable's type

```
>>> courseNum = "201"
>>> int(courseNum)
201
>>> type(courseNum)
<class 'str'>
```

cast courseNum's value to an integer

type is still a string (!?)

- To make an actual change, you need to "save" it with the assignment operator

Casting to a Type: Assignment

- Use the assignment operator (=) to actually change the variable's type

```
>>> courseNum = "201"
>>> type(courseNum)
<class 'str'>
>>> courseNum = int(courseNum)
>>> type(courseNum)
<class 'int'>
```

this is what actually causes the variable's type to change

Membership Operator

Types of Operators in Python

- Arithmetic Operators
- Assignment Operators
- Comparison Operators
- Logical Operators
- Membership Operators
- Bitwise Operators
- Identity Operators

what we're covering now

Membership Operator

- The membership operator is very powerful
- What do you think this code does?

```
hounds = ("Ibizan", "Afghan", "Serbian", "Bassett")

guess = input("Please enter a dog: ")
while guess not in hounds:
    print("You guessed wrong!")
    guess = input("Guess again: ")
```

- Runs until the user guesses a dog `in` the tuple

Membership "`in`" Operator

- Syntax:

```
element in sequence
```

element to look for

"in" keyword

may be a list, tuple, or string

- Checks to see if element exists in sequence
  - Evaluates to either `True` or `False`
  - Use it together with `while`, `if`, or `elif`
- Can also use `not in` to test for absence

It's Time For…

LIVECODING!!!

The "Tuple of Secrets"

- Write a function that takes in a tuple, and asks the user to guess what is in the tuple
  - Counting the number of correct guesses made
  - Use a sentinel loop to let them keep guessing
  - Return number of correct guesses
- You'll want to use:
  - Actual parameters and **`return`**
  - While loops and membership "**`in`**"

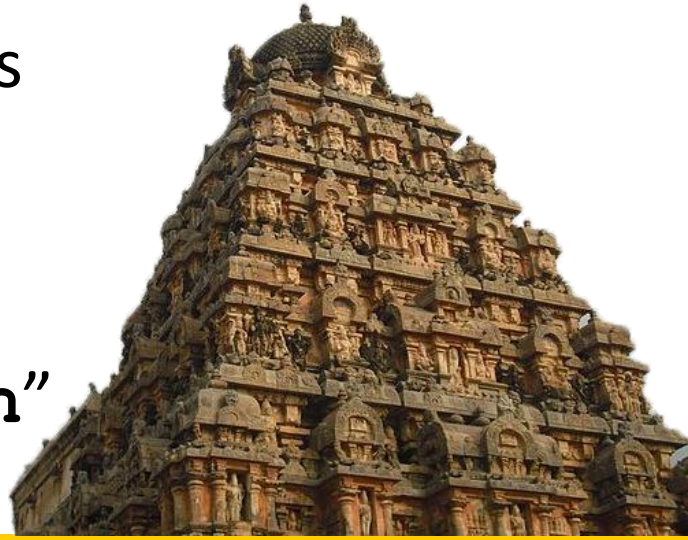

 Image from wikimedia.org

"Tuple of Secrets" Sample Run

- With a tuple whose contents are set in **`main()`**

```
bash-4.1$ python fxnPrac.py
Please enter your guess (stop to quit): hello
        hello is NOT in the tuple of secrets
Please enter your guess (stop to quit): Hrabowski?
        Hrabowski? is NOT in the tuple of secrets
Please enter your guess (stop to quit): dogs are great
        dogs are great is in the tuple of secrets!
Please enter your guess (stop to quit): cats are great
        cats are great is NOT in the tuple of secrets
Please enter your guess (stop to quit): stop

You got 1 correct guesses
```

Announcements

- Project 1 is out on Blackboard now
  - Must use the design (posted on Blackboard now)
  - Design due by Saturday (March 11th) at 8:59:59 PM
  - Project due by Friday (March 17th) at 8:59:59 PM

- Midterm will be next week
  - We'll have an in-class review on Monday/Tuesday
  - Review worksheet only available <u>in class!</u>
  - Start studying on your own now!