

# CMSC201

## Computer Science I for Majors

### Lecture 06 – Strings (and Decisions Continued)

# Last Class We Covered

- Control structures
- Conditional operators
  - Comparison operators
  - Logical operators
- Boolean data types
- One-way and two-way decision structures
  - **if** and **if-else** statements

# Any Questions from Last Time?

# Today's Objectives

- Review control structures & conditional operators
- Understand more decision structures
  - Multi-way, using **if-elif-else** statements
- Practice implementing algorithms
- To better understand the string data type
  - Learn how they are represented
  - Learn about and use some of their built-in functions

# Example – Dangerous Dinosaurs

- You have just been flown to an island where there are a wide variety of dinosaurs
- You are unsure which are dangerous so we have come up with some rules to figure out which are dangerous and which are not



Time for...

**LIVECODING!!!**


# Multi-Way Selection Structures

# Bigger (and Better) Decision Structures

- One-way and two-way structures are useful
- But what if we have to check multiple exclusive conditions?
  - **Exclusive** conditions do not overlap with each other
  - *e.g.*, value of a playing card, letter grade in a class
- What could we use?



# Multi-Way Code Framework

```
if <condition1>:  
    <case1 statements>  
elif <condition2>:  
    <case2 statements>  
elif <condition3>:  
    <case3 statements>  
# more "elif" statements if needed  
else:   
    <default statements>
```

“else” statement  
is optional

# Multi-Way Selection Example

- A computer science professor gives a five-point quiz at the beginning of every class
- Possible grades are as follows:

5 points: A	3 points: C	1 point: F
4 points: B	2 points: D	0 points: F
- To print out the letter grade based on the raw points, what would the code need to look like?

# Multi-Way Selection Solution

```
def main():  
    score = int(input("Your quiz score out of 5: "))  
    if score == 5:  
        print("You earned an A")  
    elif score == 4:  
        print("You earned a B")  
    elif score == 3:  
        print("You earned a C")  
    elif score == 2:  
        print("You earned a D")  
    else:  
        print("You failed the quiz")
```

```
main()
```

# Multi-Way Selection Solution

```
def main():  
    score = int(input("Your quiz score out of 5: "))  
    if score == 5:  
        print("You earned an A")  
    elif score == 4:  
        print("You earned a B")  
    elif score == 3:  
        print("You earned a C")  
    elif score == 2:  
        print("You earned a D")  
    else:  
        print("You failed the quiz")
```

these are five  
separate statements

since this is an  
if-elif-else  
block, only one of the  
five statements  
will be executed

main()

# Nested Selection Structures

# Nested Selection Structures

- Up until now, we have only used a single level of decision making
- What if we want to make decisions within decisions?
- These are called *nested* selection structures
  - We'll first cover nested **if-else** statements

# Nested Selection Structure Examples

- For example, we may
  - Ask the user if they have a pet
  - **if** they have a pet
    - Ask the user what type of pet
    - **if** they have a dog, take it for a walk
    - **elif** they have a cat, clean the litter box
    - **else** clean the cage/stable/tank

# Nested Selection Structures Code

```
if condition1 == True:
    if condition2 == True:
        execute codeA
    elif condition3 == True:
        execute codeB
    else:
        execute codeC
else:
    execute codeD
```



# Nested Selection Structures Code

```
if condition1 == True:  
    if condition2 == True:  
        execute codeA  
    elif condition3 == True:  
        execute codeB  
    else:  
        execute codeC  
else:  
    execute codeD
```

this is the main level  
of our program:  
an if-else block

this is the next level,  
inside the first  
if statement

codeA, codeB, and codeC  
are separate statements

since this is an  
if-elif-else  
block, only one of them  
will be executed

if our first if  
statement was  
false, we would  
skip here and  
execute codeD

# Nested Selection Structure Example

- You recently took a part-time job to help pay for your student loans at a local cell phone store
- If you sell at least \$1000 worth of phones in a pay period, you get a bonus
  - Your bonus is 3% if you sold at least 3 iPhones, otherwise your bonus is only 2%

# Nested Selection Solution

```
def main():
    totalSales = float(input("Please enter your total sales:"))

    if totalSales >= 1000.00:
        iPhonesSold = int(input("Enter the number of iPhones sold:"))

        if iPhonesSold >= 3:
            bonus = totalSales * 0.03
        else:
            bonus = totalSales * 0.02

        print("Your bonus is $", bonus)

    else:
        print("Sorry, you do not get a bonus this pay period.")

main()
```

# Strings

# The String Data Type

- Text is represented in programs by the string data type
- A ***string*** is a sequence of characters enclosed within quotation marks (") or apostrophes (')
  - Sometimes called double quotes or single quotes
- *FUN FACT!* – *The most common use of personal computers is word processing*

# String Examples

```
>>> str1 = "Hello"  
>>> str2 = 'spam'  
>>> print(str1, str2)  
Hello spam  
>>> type(str1)  
<class 'str'>  
>>> type(str2)  
<class 'str'>
```

# Getting Strings as Input

- Using `input()` automatically gets a string

```
>>> firstName = input("Please enter your name: ")
Please enter your name: Shakira
>>> print("Hello", firstName)
Hello Shakira
>>> type(firstName)
<class 'str'>
>>> print(firstName, firstName)
Shakira Shakira
```

# Numbering in Strings

- Strings don't count their characters from 1
  - They start counting from 0!
- Strings with  $n$  characters go from 0 to  $n-1$ 
  - The string below has 5 characters, and is numbered from 0 to 4

<b>0</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>
<b>H</b>	<b>e</b>	<b>l</b>	<b>l</b>	<b>o</b>



# Accessing Individual Characters

- We can access the individual characters in a string through *indexing*
  - Characters are the letters, numbers, spaces, and symbols that make up a string
- The characters in a string are numbered starting from the left, beginning with 0

# Syntax of Accessing Characters

- The general form is

**strName [expression]**

- Where **strName** is the name of the string variable and **expression** determines which character is selected from the string

# Example String

0	1	2	3	4	5	6	7	8
H	e	l	l	o		B	o	b

```
>>> greet = "Hello Bob"
>>> greet[0]
'H'
>>> print(greet[0], greet[2], greet[4])
H l o
>>> x = 8
>>> print(greet[x - 2])
B
```

## Example String

0	1	2	3	4	5	6	7	8
H	e	l	l	o		B	o	b

- In a string of  $n$  characters, the last character is at position  $n-1$  since we start counting with 0
- So if a string is 10 characters long, the last character is at what index?
  - Index 9

## Example String

0	1	2	3	4	5	6	7	8
H	e	l	l	o		B	o	b

- Index from the right side using negative indexes

```
>>> greet[-1]
```

```
'b'
```

```
>>> greet[-3]
```

```
'B'
```

`greet[0]`

already means the  
first character, 'H'

- Why don't we start from zero?

## Substrings and Slicing

# Substrings

- Indexing only returns a single character from the entire string
- We can access a ***substring*** using a process called ***slicing***
  - Substring: a (sub)part of another string
  - Slicing: we are slicing off a portion of the string

# Slicing Syntax

- The general form is

**strName[start:end]**

- **start** and **end** must both be integers
  - The substring begins at index **start**
  - The substring ends before index **end**
    - The letter at index **end** is not included



# Slicing Examples

0	1	2	3	4	5	6	7	8
H	e	l	l	o		B	o	b

```
>>> greet[0:2]
```

```
'He'
```

```
>>> greet[5:9]
```

```
' Bob'
```

```
>>> greet[:5]
```

```
'Hello'
```

```
>>> greet[1:]
```

```
'ello Bob'
```

```
>>> greet[:]
```

```
'Hello Bob'
```

# Specifics of Slicing

- If **start** or **end** are missing, then the start or the end of the string are used instead
- The index of **end** must come after the index of **start**
  - What would the substring **greet[1:1]** be?  
  ' '
  - An empty string!

# More Slicing Examples

0	1	2	3	4	5	6	7	8
H	e	l	l	o		B	o	b
-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> greet[2:-3]
```

```
'llo '
```

```
>>> greet[-6:-2]
```

```
'lo B'
```

```
>>> greet[-6:6]
```

```
'lo '
```

```
>>> greet[-9:8]
```

```
'Hello Bo'
```

# Forming New Strings - Concatenation

- We can put two or more strings together to form a longer string
- **Concatenation** “glues” two strings together

```
>>> "Peanut Butter" + "Jelly"  
'Peanut ButterJelly'  
>>> "Peanut Butter" + " & " + "Jelly"  
'Peanut Butter & Jelly'
```

# Rules of Concatenation

- Concatenation does not automatically include spaces between the strings

```
>>> "Smash" + "together"  
'Smashtogether'
```

- Concatenation can only be done with strings!
  - So how would we concatenate an integer?

```
>>> "CMSC " + str(201)  
'CMSC 201'
```

# Forming New Strings - Repetition

- Concatenating the same string together multiple times can be done with *repetition*

– Which operator would you use for this?

```
>>> animal = "dogs"
```

```
>>> animal*3
```

```
'dogsdogsdogs'
```

```
>>> animal*8
```

```
'dogsdogsdogsdogsdogsdogsdogsdogsdogs'
```



# Length of a String

- To get the length of a string, use `len()`

```
>>> title = "CMSC 201"
```

```
>>> len(title)
```

```
8
```

```
>>> len("Help I'm trapped in here!")
```

```
25
```

- Why would we need the length of a string?



# String Operators in Python

Operator	Meaning
<code>+</code>	Concatenation
<code>*</code>	Repetition
<code>STRING[#]</code>	Indexing
<code>STRING[#:#]</code>	Slicing
<code>len(STRING)</code>	Length
<code>for VAR in STRING</code>	Iteration

We'll cover this in a future class, when we learn **for** loops!

# Just a Bit More on Strings

- Python has many, many ways to interact with strings, and we will cover them in detail soon
- For now, here are two very useful functions:
  - `s.lower()` – copy of `s` in all lowercase letters
  - `s.upper()` – copy of `s` in all uppercase letters
- Why would we need to use these?
  - Remember, Python is case-sensitive!


## String Processing Examples

# Example: Creating Usernames

- Our rules for creating a username:
  - First initial, first 7 letters of last name (lowercase)

```
# get user's first and last names
first = input("Please enter your first name: ")
last  = input("Please enter your last name: ")

# concatenate first initial with 7 letters of last name
userName = first[0].lower() + last[:7].lower()
print("Your username is: ", userName)
```



Why is this 7?

# Example: Creating Usernames

```
>>> first = input("Please enter your first name: ")
Please enter your first name: Donna
>>> last = input("Please enter your last name: ")
Please enter your last name: Rostenkowski
```

```
>>> userName = first[0] + last[:7]
>>> print("Your username is: ", userName)
Your username is DRostenk
```

Usernames must be lowercase!

```
>>> userName = first[0].lower() + last[:7].lower()
>>> print("Your username is: ", userName)
Your username is drostenk
```

# Example: Creating Usernames

```
>>> first = input("Please enter your first name: ")
Please enter your first name: Barack
>>> last  = input("Please enter your last name: ")
Please enter your last name: Obama

>>> uname = first[0].lower() + last[:7].lower()
>>> print("Your username is: ", uname)
Your username is bobama
```

- What would happen if we did `last[7]`?
  - **IndexError** – but why does `last[:7]` work?

# Announcements

- Your Lab 3 is meeting this week!
- Homework 2 is out
  - Due by Wednesday (Sept 21st) at 8:59:59 PM
  - You must take the Academic Integrity Quiz!
- Homework 3 will come out Wednesday night
  - You must have taken the Academic Integrity Quiz!

# Practice Problems

- Create a directory inside your “201” folder, called “**practice**”; go into the new folder
- Copy this file into your new folder  
`/afs/umbc.edu/users/k/k/k38/pub/cs201/stringPractice.py`
- Complete the files according to its instructions
- Remember, the command to copy is “**cp**”:  
`cp /afs/umbc.edu/users/k/k/k38/pub/cs201/stringPractice.py .`