CS 1112: Introduction To Programming

Strings

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

- **PA04** is due by 11:00pm on Wednesday (***3/20***)!
  - Will come out after the break.
  - Submit on Gradescope: your .py file, and a PDF of your reflection
- **Quiz 5** will come out the week after Spring Break on Friday (3/15) and due Monday (***3/18***)
- **Exam 1** is being graded!
- Drop deadline is 10/10

Strings

Type	Values	Operators	Fun Fact
`int` *Integer*	-1, 0, 1, 234932, 7	+, -, *, /, **, //,%	An integer can be as large as your computer has memory for
`float` *Floating point value*	0.5, 1.0, -3.1, 1.56456345, 2.2222222222222222222 22	+, -, *, **, /	Float values are often not exact
`str` ***String***	**'Hello', "UVA", "7011", "CS 1112"**	**+, <indexing> [ ]**	A string that equals the reverse of itself is called a "***palindrome***" i.e., racecar
`bool` *Boolean*	True, False	not, and, or	George Boole died from lecturing (kind of)
variables	Have names: [a-z,A-Z,_][a-z,A-Z,_,0-9]	= # an assignment statement	Variables usually don't vary within a math problem
function	i.e. print(), input()	(): # do that named action	The first three letters of "function" spell FUN!

Python Strings: Basics

- A String is an **immutable data type** – you could also consider it a **data structure** in Python that represents *a sequence of characters*
  - **Immutable**: once you have created a string, you cannot change it
- Strings are used widely in many different applications
  - Storing and manipulating text data
  - Representing names, addresses, and other types of data that can be represented as text
- Python does **not** have a "*character*" data type
  - A single character is simply a string with a length of 1
  - "University of Virginia" vs. "X"
- Simple string **assignment**: `my_string = "hello"`
- **Printing** strings: `print(my_string)` or `print("Hello!")` or `print("hi " + name)`

Python Strings: Basics

```
String1 = ‘This is a string – single quotes’

String2 = “This is a string – double quotes”

String3 = ‘Here I am “mixing” quotes’

String4 = ‘’’This is a string on multiple
                lines – triple quotes’’’
```

Python Strings: Basics

- Strings as collections:
  - Collection of characters
  - **Order** matters
  - Repetition (of characters) is ok
  - Each character is assigned a particular index, starting at index zero (0)

hello
0 1 2 3 4

0	1	2	3	4	5	6
A	S	T	R	I	N	G

-7	-6	-5	-4	-3	-2	-1
A	S	T	R	I	N	G

```
X = "hello"
X[i] gives the ith
character in the
string (starts at 0)

X[0] = first character
     = "h"
```

Indexing details

- (Using the example, s = ‘wilson’)
- Start counting at 0 (**positive** indices)
  - `s[0]` is ‘w’
    - Read as “s sub 0” or “s of 0”
    - “*The zeroth character of s*”
  - `s[1]` is ‘i’
  - `s[2]` is ‘l’
  - (Indices are always **integers**, if you tried `s[1.0]` it will give you an error)
- `len(s)` gives the number of things in the collection
  - `len(s)` is 6
- Use **negative** indices to count from the ***end***.
  - `s[-1]` is ‘n’
  - `s[-2]` is ‘o’

Wilson the Volleyball
From the movie **Cast Away** (2000)

More on String Indexing

- Consider the string "Bananas"

Positive Index	0	1	2	3	4	5	6
Letter	**B**	**A**	**N**	**A**	**N**	**A**	**S**
Negative Index	-7	-6	-5	-4	-3	-2	-1

- The table shows the positive and negative index of each letter

- The **largest** positive index of a string is the length of the string **minus 1**
  - Example, in the above, the largest index of the string is **6**
- The **smallest** index is the negative value of the length
  - Example, in the above, the smallest index is **-7**

Slicing

- **Indexing** is how to get one thing out of a collection
- **Slicing** is how to get a chunk out of a collection (multiple contiguous items)
  - Give it a **start** index and an **end** index
  - Evaluates to a new collection from **start** (*inclusive*) to **end** (*exclusive*)

  - If S is a string, here's the syntax:

```
S[start:stop:step]
```

Start position *End position* *The increment*

  - This returns the portion of the string from index **start** to index **stop**, at a step size **step**.
  - **Fun trick** with slicing: **Reversing a string**: `my_string[: : -1]` # full index range, backwards!
  - … more on Slicing later!

Examples and images from *learnbyexample*

Some String functions

```
my_string = "Hello World "
len(my_string)  # 12
"lo" in my_string  # True
"hi" in my_string  # False
my_string.lower()  # "hello world "
my_string.upper()  # "HELLO WORLD "
my_string.startswith("Hello")  # True
my_string.startswith("Steve")  # False
my_string.endswith("World ")  # True
my_string.endswith("World")  # False
```

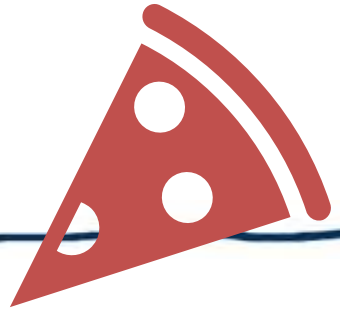
Length .....
Contains ....
Lowercase ...
Uppercase....
Starts with .
Ends with ...

None of these functions change the value of my_string

Instead, they return a new value

Strings are **immutable**

What is printed?



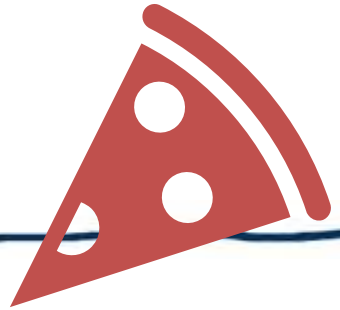
```
my_string = "I love pizza"
print(len(my_string))
```

```
my_string = "I love pizza"
print(my_string[1])
```

```
my_string = "I love pizza"
print(my_string[12])
```

```
my_string = "I love pizza"
print(len(my_string[1]))
```

What is printed?



```
my_string = "I love pizza"
print(len(my_string))
```

12

```
my_string = "I love pizza"
print(my_string[1])
```

<space character>

```
my_string = "I love pizza"
print(my_string[12])
```

IndexError: string index out of range

```
my_string = "I love pizza"
print(len(my_string[1]))
```

1

What is printed?

my_string = **"I love pizza"**

print(my_string.**upper()**)

What is printed?

my_string = **"I love pizza"**

print(my_string.**upper()**)

I LOVE PIZZA

What is printed?

my_string = **"I love pizza"**

print(**“L”** **in** my_string)

What is printed?

my_string = **"I love pizza"**

print(**“L”** **in** my_string)

False

Other String Functions you should know

- **my_string.replace(str1, str2)**
  - Returns a copy of the string where all occurrences of str1 are replaced by str2

- **my_string.strip()**
  - returns a copy of the string with the leading and trailing characters (whitespace characters, by default) removed

```
my_string = "Hello World "
my_string.replace("World", "Class")  # "Hello Class
"
my_string.strip()  # "Hello World"
```

Other String Functions you should know

- **my_string.count(str1)**
  - total number of occurrences of str1 in my_string

- **my_string.find(str1)**
  - returns the lowest index in the string where str1 is found

```
my_string = "Hello World "
my_string.count("l")  # 3
my_string.find("rl")  # 8
```

Quick & Fun Survey Questions

Get to know your peers! ☺

Sweet or Savory?

What is printed?

my_string = **"I love pizza"**

print(my_string.**replace(“z”, “bb”)**)

What is printed?

my_string = **"I love pizza"**

print(my_string.**replace(“z”, “bb”)**)

I love pibbbba

What is printed?

my_string = “ **I love\n pizza \n** "

print(my_string.**strip()**)

What is printed?

my_string = “ **I love\n pizza \n** "

print(my_string.**strip()**)

I love
pizza

What is printed?

my_string = “ **I love\n pizza \n** "

print(my_string.**strip()**)

I love
pizza

No leading or
trailing whitespace

What is printed?

my_string = “ **I love\n pizza \n** "

print(my_string.**find(“zz”)**)

What is printed?

my_string = **“ I love\n pizza \n "**

print(my_string.**find(“zz”)**)

What is printed?

my_string = “ **I love\n pizza \n** "

print(my_string.**find(“zz”)**)

Key Points:

1. Start from 0
2. What space counts
3. \n counts as 1 character only

Other String Functions you should know

- **my_string.join()**
  - returns a string which is the concatenation of the strings in the argument, connected by my_string

```
my_string = "Hello World "
my_string.join(["a", "b", "c"])  # "aHello World bHello World c"
```

Other String Functions you should know

- **my_string.join()**
  - returns a string which is the concatenation of the strings in the argument, connected by my_string

```
my_string = "Hello World "
my_string.join(["a", "b", "c"])  # "aHello World bHello World c"
```

We'll talk about what these "square brackets" mean soon (we've seen them!)

Python Demonstration

Let's jump on PyCharm!

`strings.py` - Many examples about strings, string functions, and more!

In-Class "lab" Activity!

Activity for Today!

- In pairs or groups up to three work on the following activity.
- **strings_ica1.py**
- *Practice solving small problems using string indexing and string functions*

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.