

## Objectives

- Text process, manipulation
  - String operations, processing, methods

## Review

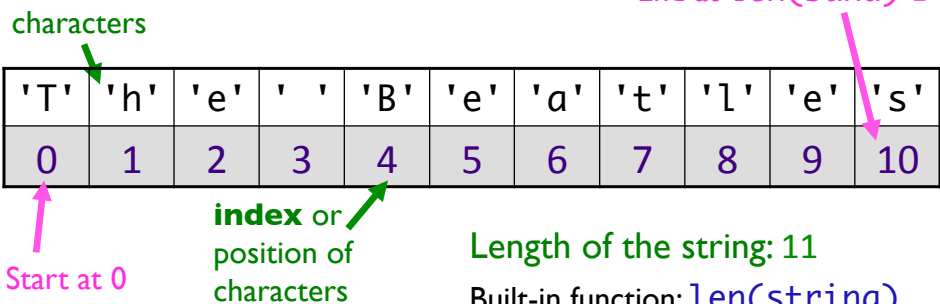
- How do we represent text?
- How can we represent really long text?
- How can we combine strings?
- How can we combine strings multiple times?
- How can you tell which string comes first alphabetically?
  - What are some limitations to that approach?
- How do you find out long a string is?

## Strings

- A **sequence** of one-character strings

➤ Example:

band = "The Beatles"



Oct 23, 2017

Sprenkle - CSCI111

3

## Iterating Through a String

- Use a **for** loop to iterate through **characters** in a string

string of length 1

```
for char in string:  
    print(char)
```

➤ Read as "for each character in the string"

Oct 23, 2017

Sprenkle - CSCI111

Python shell

4

## Substrings Operator: []

Literally, **not** optional

- Look at a particular character in the string
  - Syntax: `string[<integer_expression>]`
  - [Positive value]: index of character
  - [Negative value]: count backwards from end
- Examples:
  - `<sequence>[0]` returns the first element/char
  - `<sequence>[-1]` returns the last element/char

We will deal with sequences  
beyond strings later.

Examples in interpreter

Oct 23, 2017

Sprenkle - CSCI111

5

## Substrings Operator: []

- Look at a particular character in the string
  - Syntax: `string[<integer_expression>]`
- Examples with `band = "The Beatles"`

T	h	e		B	e	a	t	l	e	s
0	1	2	3	4	5	6	7	8	9	10

Expression	Result
<code>band[0]</code>	
<code>band[3]</code>	
<code>band[len(band)]</code>	
<code>band[len(band)-1]</code>	
<code>band[-1]</code>	

Oct 23

6

## Substrings Operator: []

- Look at a particular character in the string
  - Syntax: `string[<integer expression>]`
- Examples with `band = "The Beatles"`

T	h	e		B	e	a	t	l	e	s
0	1	2	3	4	5	6	7	8	9	10

Expression	Result
<code>band[0]</code>	"T"
<code>band[3]</code>	" "
<code>band[len(band)]</code>	<b>IndexError</b>
<code>band[len(band)-1]</code>	"s"
<code>band[-1]</code>	"s"

Oct 23

7

## Iterating Through a String

- Alternatively, can iterate through the *positions* in a string
  - Could write as a **while** loop as well

An integer

```
for pos in range(len(string)):
    print(string[pos])
```

Index into the string

`string_iteration.py`

Oct 23, 2017

Sprenkle - CSCI111

8

## Summary: Iterating Through a String

- For each character in the string

string of length 1

```
for char in mystring:  
    print(char)
```

Determines loop's  
behavior

- For each position in the string

An integer

```
for pos in range(len(mystring)):  
    print(mystring[pos])
```

Index into the string

Oct 23, 2017

Sprenkle - CSCI111

9

## Substrings Operator: [:]

- Select a substring (zero or more characters) using the `[]` and `:`
- `<sequence>[<start>:<end>]`
  - returns the subsequence from **start** up to and **not** including **end**
- `<sequence>[<start>:]`
  - returns the subsequence from **start** to the end of the sequence
- `<sequence>[:<end>]`
  - returns the subsequence from the first element up to and **not** including **end**
- `<sequence>[:]`
  - returns a copy of the entire sequence

Oct 23, 2017

Sprenkle - CSCI111

10

## Substrings Operator: [:]

- Select a substring (one or more characters) using the [] and :
- Examples: filename = "program.py"

p	r	o	g	r	a	m	.	p	y
0	1	2	3	4	5	6	7	8	9

Expression	Result
filename[0:]	
filename[0:2]	
filename[:3]	
filename[8:]	
filename[-2:]	

Oct 23, 2

11

## Substrings Operator: [:]

- Select a substring (one or more characters) using the [] and :
- Examples: filename = "program.py"

p	r	o	g	r	a	m	.	p	y
0	1	2	3	4	5	6	7	8	9

Expression	Result
filename[0:]	"program.py"
filename[0:2]	"pr"
filename[:3]	"pro"
filename[8:]	"py"
filename[-2:]	"py"

Oct 23, 2

12

## Testing for Substrings

- Using the **in** operator
  - Used **in** before in **for** loops

- Syntax:

```
substring in string:
```

- Evaluates to **True** or **False**

- Example:

```
if "cat" in name:  
    print(name, "contains 'cat'")
```

## String Search Comparison

- What do the two **if** statements test for?

```
PYTHON_EXT = ".py"  
  
filename = input("Enter a filename: ")  
  
if filename[-(len(PYTHON_EXT)):] == PYTHON_EXT:  
    # Appropriate output  
if PYTHON_EXT in filename:  
    # Appropriate output
```

How would the program execution change if it were an **if-elif**?

## Strings are Immutable

You cannot change the value of strings

- For example, you **cannot** change a character in a string

~~➤ str[0] = 'S'~~

Oct 23, 2017

Sprenkle - CSCI111

15

## Testing for Substrings

- Using the **in** operator
  - Used **in** before in **for** loops
- Syntax:

```
substring in string:
```

➤ Evaluates to **True** or **False**

- Example:

```
if "cat" in name:  
    print(name, "contains 'cat'")
```

Oct 23, 2017

Sprenkle - CSCI111

16



## String Search Comparison

- What do the two **if** statements test for?

```
PYTHON_EXT = ".py"

filename = input("Enter a filename: ")

if filename[-(len(PYTHON_EXT)):] == PYTHON_EXT:
    # Appropriate output
if PYTHON_EXT in filename:
    # Appropriate output
```

How would the program execution change if it were an **if-elif**?

Oct 23, 2017

Sprenkle - CSCI111

[search.py](#)

17

## Revised Pick4 Game

- To play: pick 4 numbers between 0 and 9
- To win: select the numbers that are selected by the magic ping-pong ball machine
- Done previously: Simulate the magic ping-pong ball machines
- Additional Functionality:
  - Determine if the user picks the winning number

Oct 23, 2017

Sprenkle - CSCI

[pick4winner.py](#)

3

Extra credit on lab 6

## Revised Pick4 Numbers

- Tell the user how many numbers they got right
  - Get prizes for having some numbers right
- Examples:

Pick4 Num	User's Pick	Num Correct
"7737"	"1234"	1
"0204"	"1234"	2
"1234"	"1234"	4

Oct 23, 2017

Sprenkle - CSCI pick4num\_places.py 9

## USING THE STR API

Oct 23, 2017

Sprenkle - CSCI111

20

## Review

- What is an API?
- How do we call methods on an object?


## str Methods

- **str** is a *class* or a *type*
- **Methods**: available operations to perform on **str** objects
  - Provide common functionality
- To see all methods available for **str** class
  - `help(str)`

## str Methods

- Example method: **find(substring)**
  - Finds the index where substring is in string
  - Returns -1 if substring isn't found
- To call a method:
  - `<str_obj>.methodname([arguments])`
  - Example: `filename.find(".py")`

Executed on this string



Oct 23, 2017

Sprenkle - CSCI111

23

## Common str Methods

Method	Operation
<code>center(width)</code>	Returns a copy of string centered within the given number of columns
<code>count(sub[, start [, end]])</code>	Return # of non-overlapping occurrences of substring <code>sub</code> in the string.
<code>endswith(sub), startswith(sub)</code>	Return <code>True</code> iff string ends with/starts with <code>sub</code>
<code>find(sub[, start [, end]])</code>	Return first index where substring <code>sub</code> is found
<code>isalpha(), isdigit(), isspace()</code>	Returns <code>True</code> iff string contains letters/digits/whitespace only
<code>lower(), upper()</code>	Return a copy of string converted to lowercase/uppercase

Oct 23, 2017

Sprenkle - CSCI111 [string\\_methods.py](#)

## Common `str` Methods

Method	Operation
<code>replace(old, new[, count])</code>	Returns a copy of string with all occurrences of substring <code>old</code> replaced by substring <code>new</code> . If <code>count</code> given, only replaces first <code>count</code> instances.
<code>split([sep])</code>	Return a list of the words in the string, using <code>sep</code> as the delimiter string. If <code>sep</code> is not specified or is None, any whitespace string is a separator.
<code>strip()</code>	Return a copy of the string with the leading and trailing whitespace removed
<code>join(&lt;sequence&gt;)</code>	Return a string which is the concatenation of the strings in the sequence with the string this is called on as the separator
<code>swapcase()</code>	Return a copy of the string with uppercase characters converted to lowercase and vice versa.

Oct 23, 2017

Sprenkle - CSCI111

25

## String Methods vs. Functions

### Functions

- All input comes from arguments/parameters
- Example: `len` is a built-in function
  - Called as `len(strobj)`

### Methods

- Input comes from `s` arguments *and* the string the method was called on
- Example:
  - `strobj.upper()`

Oct 23, 2017

Sprenkle - CSCI111

26

## Using the APIs

- Given a problem, break down the problem
  - Can any of the parts of the problem be solved using a method in the API?

Oct 23, 2017

Sprenkle - CSCI111

27

## Are You Smarter Than a 5th Grader?

- Problem in spelling from the show: How many a's are in abracadabra?
  - Solve using `str` methods
- Silly problem but can generalize to other problems
  - How many a's are in a given word?
  - How many of a certain letter are in a given word?

Oct 23, 2017

Sprenkle - CSCI111

28

## Verifying User Input

- How can we verify that the user entered the lottery number in the correct format?

`pick4winner_better_error_handling.py`

Oct 23, 2017

Sprenkle - CSCI111

29

## Looking Ahead

- Lab 6 Prep due tomorrow
- Lab 6 tomorrow!
- Broader Issue Friday

Oct 23, 2017

Sprenkle - CSCI111

30