Objectives

- Continuing text processing, manipulation
  - String operations, processing, methods

1

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?
  - What if we want to combine a string and an integer? What do we need to do?
- How can you tell which string comes first alphabetically?
  - What are some limitations to that approach?
- What is an API?
- What are methods?
- How do we call methods on an object?

2

Review: String Comparisons

- Same operations as with numbers:
  - ==, !=
  - <, <=
  - >, >=

$\left.\begin{array}{l} ==,\ != \\ <,\ <= \\ >,\ >= \end{array}\right\}$ Alphabetical comparison

- Use in conditions in `if` statements

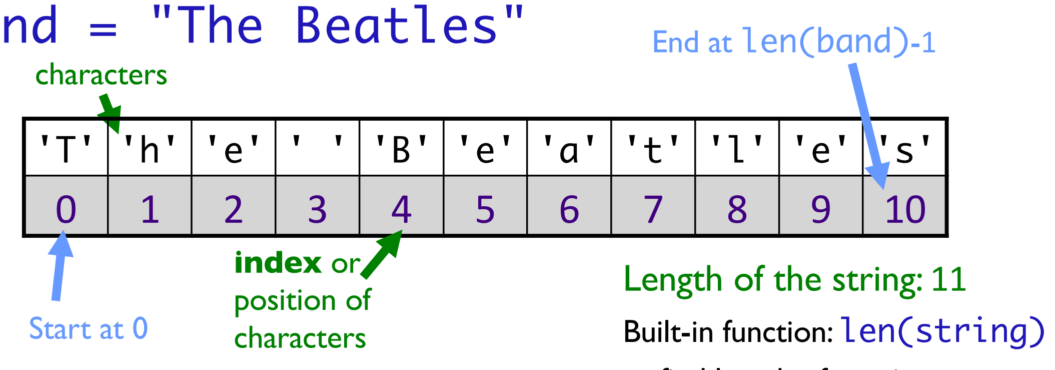
```
if courseChoice == "CSCI111":
    print("Good choice!")
else:
    print("Maybe next semester")
```

string_compare.py

Strings

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

band = "The Beatles"



'T'	'h'	'e'	' '	'B'	'e'	'a'	't'	'l'	'e'	's'
0	1	2	3	4	5	6	7	8	9	10

Length of the string: 11

Built-in function: `len(string)` to find length of a string

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"

Feb 27, 2023 Sprenkle - CSCI111 Python interpreter 5

5

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.

Feb 27, 2023 Sprenkle - CSCI111 Examples in interpreter 6

6

Substrings Operator: []

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

Expression	Result
`band[0]`	
`band[3]`	
`band[len(band)]`	
`band[len(band)-1]`	
`band[-1]`	

Feb 27, 2023

Substrings Operator: []

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

First thing you should do:

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

Expression	Result
`band[0]`	
`band[3]`	
`band[len(band)]`	
`band[len(band)-1]`	
`band[-1]`	

Feb 27, 2023

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
`band[0]`	`"T"`
`band[3]`	`" "`
`band[len(band)]`	`IndexError`
`band[len(band)-1]`	`"s"`
`band[-1]`	`"s"`

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

Summary: Iterating Through a String

- For each *character* in the string

string of length 1

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

What comes *after* in 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

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

Substrings Operator: [:]

- Select a substring (one or more characters)
- 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[0:2]	
filename[0:]	
filename[:3]	
filename[8:]	
filename[-2:]	

Feb 27, 2023

13

13

Substrings Operator: [:]

- Select a substring (one or more characters)
- 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[0:2]	"pr"
filename[0:]	"program.py"
filename[:3]	"pro"
filename[8:]	"py"
filename[-2:]	"py"

Feb 27, 2023

14

14

Testing for Substrings

- Using the `in` operator
  - Used `in` before in `for` loops
- Syntax: `substring in string`
  - Evaluates to `True` or `False`
- Example: simple Web search

```
if searchTerm in documentText:
        print(document, "contains", searchTerm)
```

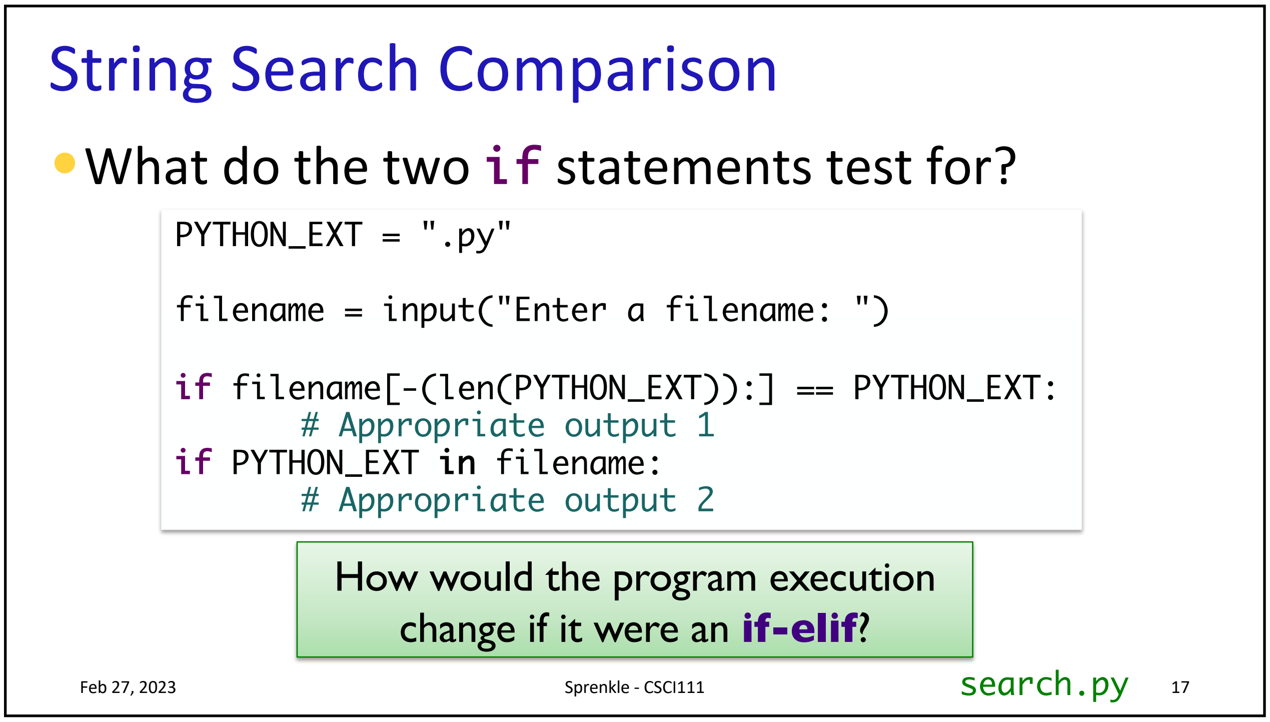
String Search Comparison

- 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 1
if PYTHON_EXT in filename:
        # Appropriate output 2
```

Provide some examples for `filename` and state how code would execute

search.py



String Search Comparison

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

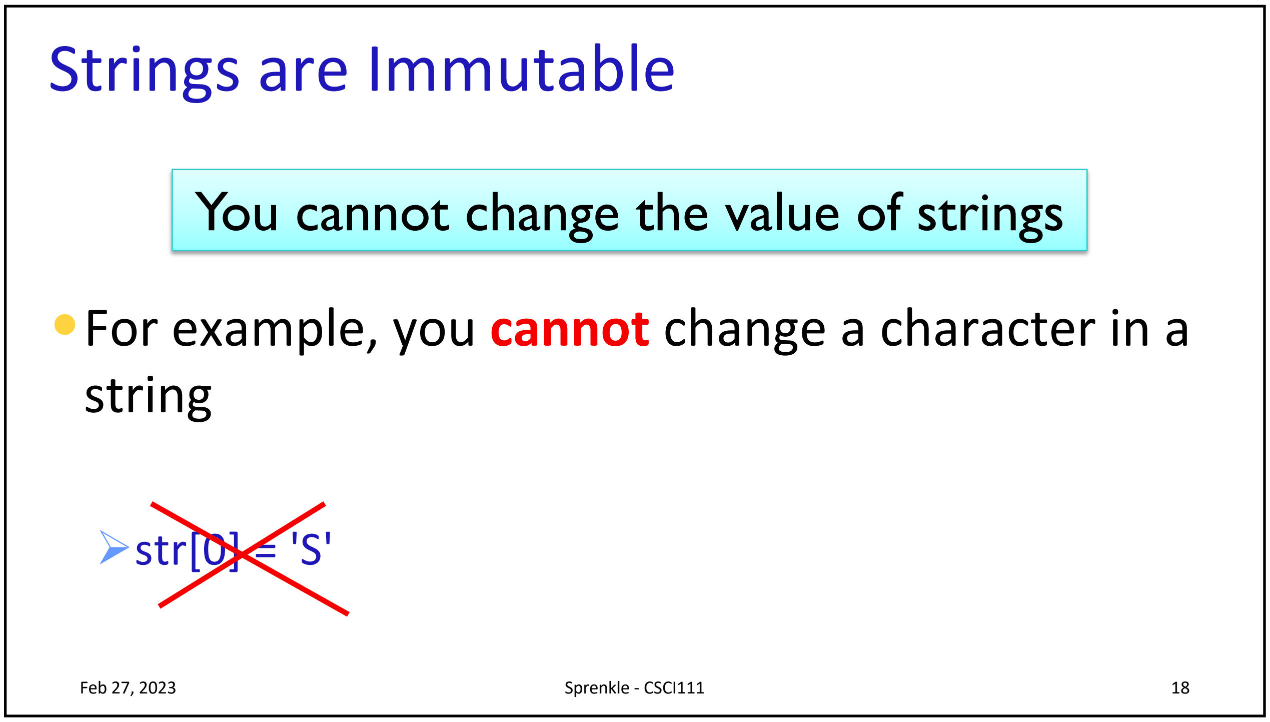
```
PYTHON_EXT = ".py"

filename = input("Enter a filename: ")

if filename[-(len(PYTHON_EXT)):] == PYTHON_EXT:
        # Appropriate output 1
if PYTHON_EXT in filename:
        # Appropriate output 2
```

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

search.py

Feb 27, 2023 Sprenkle - CSCI111 17

17



Strings are Immutable

You cannot change the value of strings

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

  - ~~str[0] = 'S'~~

Feb 27, 2023 Sprenkle - CSCI111 18

18

USING THE STR API

Review

- What is an API?
- What are methods?
- 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)`

21

`str` Methods

- Example method: `find(substring)`
  - Finds the first 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")`

find method executed on this string

22

Common str Methods

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

Feb 27, 2023 Sprenkle - CSCI111 `string_methods.py` 23

23

Common str Methods

Review: What do the square brackets in APIs mean?

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

Feb 27, 2023 Sprenkle - CSCI111 `string_methods.py` 24

24

Common `str` Methods

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

Understanding the API: What Does This Code Do?

```
sentence = input("Enter a sentence to mangle: ")

length = len(sentence)

print("*", sentence.center(int(length*1.5)), "*")

upperSentence = sentence.upper()
print(upperSentence)
print(sentence)

print("Uppercase: ", sentence.upper())
print()
print("Lowercase: ", sentence.lower())
print()

print("Did sentence change?: ", sentence)
```

Functions vs Methods (with Strings)

Functions

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

Methods

- Associated with a *class* or *type*
- Input comes from arguments ***and*** the string the method was called on
- Example:
  - `strobj.upper()`

27

How to Use APIs

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

28

Wheel of Fortune

- Determine how many of a certain letter are in a given phrase
- How would we solve this, regardless of case?

```
def getNumberOfLetters( phrase, letter ):
```

Example Test Cases:

```
test.testEqual( getNumberOfLetters("abracadabra", "a"), 5)
test.testEqual( getNumberOfLetters("Abracadabra", "a"), 5)
test.testEqual( getNumberOfLetters("abracadabra", "A"), 5)
```

Looking Ahead

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