Objectives

- Continuing text processing, 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 how long a string is?
- How do we find the character at a particular position of a string?
- How do we iterate over the characters in a string?

String Comparisons

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

  } Alphabetical comparison
- Use in conditions in `if` statements

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


Strings

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

```
band = "The Beatles"
```

characters

End at `len(band)-1`

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

Start at 0

index or position of characters

Length of the string: 11

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

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

string_iteration.py

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

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) 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:]`	

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"`

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'~~

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
    - Why couldn't we solve this before?
      - What are valid choices for numbers?

USING THE STR API

Review

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

`str` Methods

- `str` is a ***class*** or a ***type***

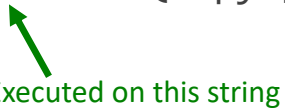
- **Methods**: available operations to perform on `str` objects
  - Provide common functionality

- 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")`



Common `str` Methods

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

Common `str` Methods

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

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])`	Return 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()`	Return a copy of the string with the leading and trailing whitespace removed
`join(<sequence>)`	Return a string which is the concatenation of the strings in the sequence with the string this is called on as the separator
`swapcase()`	Return a copy of the string with uppercase characters converted to lowercase and vice versa.

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

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?

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?

Lab 6: Pair Programming

Every lab, pairs will change

Alice	Callie	Hayden	Andrew
Andrew	Hayden	Jake	--
August	Dan	James	Ellis
Bobby	--	Jenna	Danny
Callie	Alice	Karel	Cat
Cat	Karel	Kassi	Mike
Charlotte	Nate	Laurie	--
Dan	August	Matt	Melissa
Danielle	--	Melissa	Matt
Danny	Jenna	Mike	Kassi
Ellis	James	Natalie	Giovanni
Giovanni	Natalie	Nate	Charlotte

Alphabetically by first name

Extra Credit Opportunities

- TODAY Gabriel Dance, 4:30 p.m., Stackhouse
  - "Finding Fake Followers and Watching the Watchers: New Approaches to Investigative Journalism"
- Friday, Chelsea Barabas, 5 p.m., Northen
  - "DODGING SILVER BULLETS: UNDERSTANDING THE ROLE OF TECHNOLOGY IN SOCIAL CHANGE"

- Write up on Sakai:
  - For talks recommended by Professor Sprenkle, you can earn up to 10 points extra credit for attending the talk and writing a summary containing:
  - Your interest score on a scale of 0 to 9
  - The three most important points
  - How the talk related to computer science
  - How the talk relates to our class
  - A lingering question you have

Looking Ahead

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