Objectives

- Wrap up indefinite loops
- Text processing, manipulation
    - String operations, processing, methods
- Broader Issue: Child safety online

Review

- How do we write indefinite loops in Python?
  - Why are they called *indefinite* loops?
- What are two ways to think about `while` loops?
  - What questions should you ask and how do the answers inform your solutions to these problems?
- Which are more powerful: `for` loops or `while` loops?
- Continue solving the consecutive flips problem (last problem from last time)
  - What information do we need to model/represent/keep track of?
  - What other questions are you considering?

Flipping Coins

- Problem: How many flips does it take to get 3 consecutive heads?
  - How can we simulate flipping a coin?
- Recap:
  - Have the `game` module
    - `flipCoin()` returns constant `HEADS` or `TAILS`
- Now:
  - Write solution using sentinel design pattern
  - Write solution using a `while True` loop and `break`

`game.py` `consecutiveHeads.py`

TEXT PROCESSING

Motivation: Text Processing

- Mostly focused on numbers so far
  - A little on graphics
- We can manipulate text to do useful work
  - Search: finding most relevant documents to a query
  - Understanding language
  - Analyzing web logs (who is looking at my web page?)
  - Many, many others
- **Today's Focus**: the `str` data type and what you can do with them

Strings: `str`

- Used for text
- Indicated by double quotes "" or single quotes ''
  - In general, I'll use double quotes
  - Empty string: "" or ''
- Use triple quotes """ for strings that go across multiple lines

```
"""This string
is long.
Like, really, really long"""
```

STRING OPERATIONS

String Operations

Operand	Syntax	Meaning
+	`str1 + str2`	Concatenate two strings into one string
*	`str * num`	Concatenate string `num` times

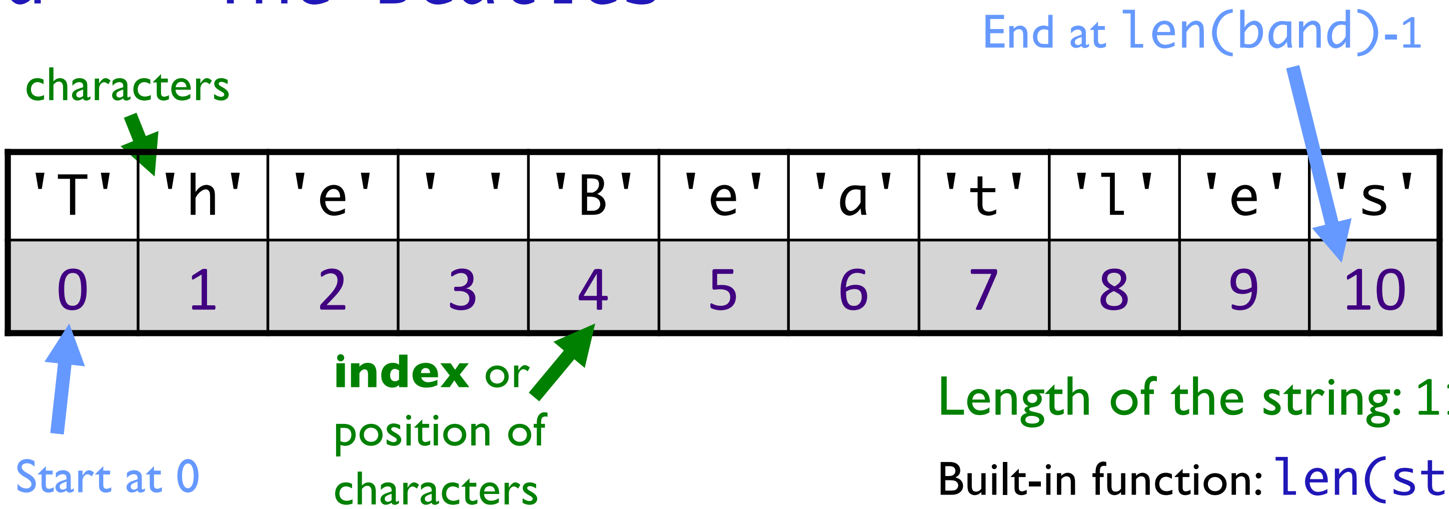
- Examples:
  - `"I feel " + "sleepy"`
    - Evaluates to `"I feel sleepy"`
  - `"Oops! " * 3`
    - Evaluates to `"Oops! Oops! Oops! "`

Recall lab 0

Strings

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

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


End at len(band)-1
characters
'T' 'h' 'e' ' ' 'B' 'e' 'a' 't' 'l' 'e' 's'
0 1 2 3 4 5 6 7 8 9 10
index or position of characters
Start at 0

Length of the string: 11

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

Index 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

Index 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]`	

Index 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]`	

Index 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"`

Strings are Immutable

You cannot change the value of strings

- For example, you **cannot** change a character in a string
    - ~~str[0] = 'S'~~

USING THE STR API

`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 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

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

`string_methods.py`

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

`string_methods.py`

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)`

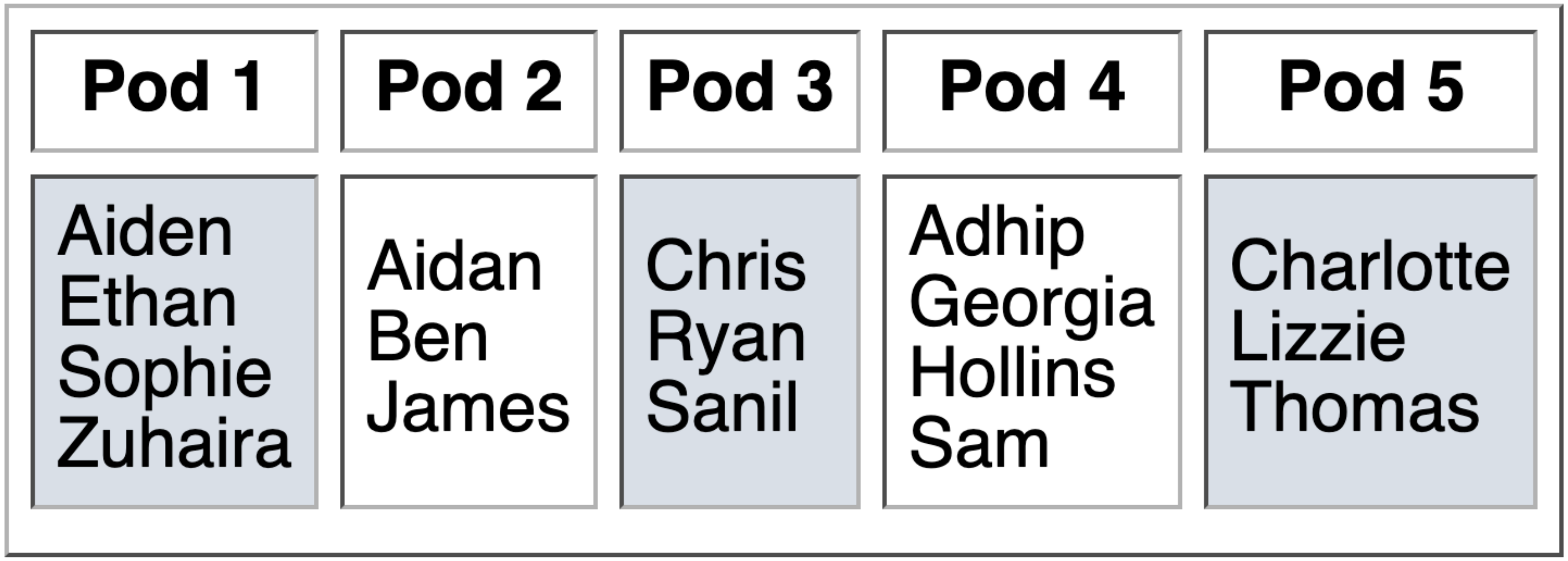
Methods

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

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?

Broader Issue Groups



Pod 1	Pod 2	Pod 3	Pod 4	Pod 5
Aiden Ethan Sophie Zuhaira	Aidan Ben James	Chris Ryan Sanil	Adhip Georgia Hollins Sam	Charlotte Lizzie Thomas

Introduce yourselves

Broader Issue Discussion

- Recap the problems and challenges, briefly
- Discuss the proposed solutions, their domains/source/actors, and their tradeoffs
  - What are the most promising?
- Reflect on these quotes:
  - "The longer we wait to come up with a solution for each of these potential issues that might happen tomorrow, the greatest chance is that it will have already happened and then we are chasing behind, and you're trying to undo harms that already happened."
  - "It's just a constant flow of new material instead of this recirculation of known material which makes the visual fingerprinting part really difficult."
  - "developers should focus on creating 'safety by design' models to mitigate the damages to children rather than having prevention measures taking a reactive response to existing threats."
- For how long will the media need to say, "X (formerly Twitter)"?

My Takeaways

- Good development practices could have improved the outcomes for the AI/algorithms
  - Better testing, thinking about what could happen
  - Find issues before the user does!

Midterm Grade Calculation

- 50% - Exam 1
- 50% - Labs

Exam 1 Reflection

- What strategies did you use to study?
    - What strategies did you use in the course in general?
- What did you do well on? What did you miss?
- What strategies should you keep? What should change?

Course Grade Overview

- (35%) Programming projects
- (30%) Two hourly exams
- (20%) A comprehensive final exam
- (7%) Writeups and discussions of Broader Issues
- (3%) Interactive textbook – prelabs
- (5%) Participation and attendance

Looking Ahead: After Break

- Lab 6 Prep Assignment: Tuesday
- Lab 6
    - Indefinite loops
    - Strings