Do Now

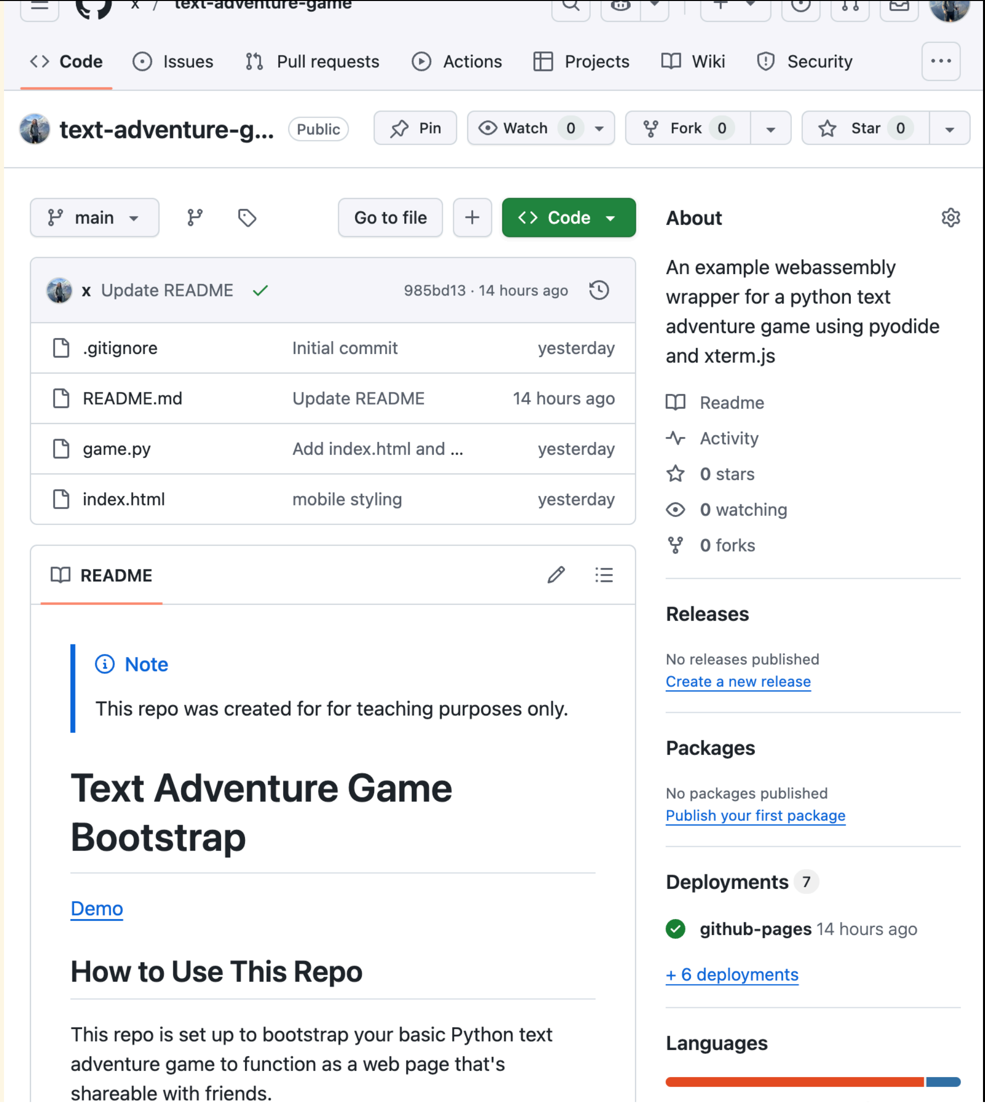
bigd103.link/function-challenges

Deploy Your Game

1. Create a Github Account
2. Go to github.com/x/text-adventure-game
3. Click "Fork" to create your own copy
4. Enable "Github Pages" in your settings
5. Copy your game into the `game.py` file

What is Github

- Like Google Docs for code
- Built on top of Git, a version control system
- A place to share and showcase projects
- Where developers collaborate worldwide
- **A simple way to host static websites**



x Update README ✓ 985bd13 · 14 hours ago

.gitignore	Initial commit	yesterday
README.md	Update README	14 hours ago
game.py	Add index.html and ...	yesterday
index.html	mobile styling	yesterday

README

Note

This repo was created for for teaching purposes only.

Text Adventure Game Bootstrap

Demo

How to Use This Repo

This repo is set up to bootstrap your basic Python text adventure game to function as a web page that's shareable with friends.

About

An example webassembly wrapper for a python text adventure game using pyodide and xterm.js

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

Deployments 7

✓ github-pages 14 hours ago

+ 6 deployments

Languages

Lists

Storing Multiple Values Together

Why Do We Need Lists?

Imagine tracking scores for 5 students:

```python
# Without lists - this gets messy fast!
student1_score = 85
student2_score = 92
student3_score = 78
student4_score = 88
student5_score = 95

# What if we had 100 students?
```

Lists let us store multiple values in one variable!

Your First List

A list is an ordered collection of values:

```python
# Creating a list
scores = [85, 92, 78, 88, 95]
print(scores)

# Lists can hold any type
names = ["Alice", "Bob", "Charlie"]
print(names)

# Even mixed types!
mixed = [42, "hello", 3.14, True]
print(mixed)
```

```
[85, 92, 78, 88, 95]
['Alice', 'Bob', 'Charlie']
[42, 'hello', 3.14, True]
```

Key points:

- Square brackets `[]` create a list

Accessing List Items

Lists are **indexed** starting at 0:

```python
fruits = ["apple", "banana", "cherry", "date"]

# First item is at index 0
print(fruits[0])  # apple

# Second item is at index 1
print(fruits[1])  # banana

# Last item
print(fruits[3])  # date

# What happens here?
# print(fruits[4])  # Error! Index out of range
```

```
apple
banana
date
```

Negative Indexing

Python lets you count from the end:

```python
colors = ["red", "green", "blue", "yellow"]

# Last item
print(colors[-1])   # yellow

# Second to last
print(colors[-2])   # blue

# First item (wraps around)
print(colors[-4])   # red
```

```
yellow
blue
red
```

Most of the time, you'll just use `-1` to get the last item without knowing the list length.

Getting the Length

Use the function `len()` to find how many items:

```python
numbers = [10, 20, 30, 40, 50]

length = len(numbers)
print(f"The list has {length} items")

# Common pattern: check if empty
if len(numbers) == 0:
    print("List is empty")
else:
    print("List has items")

# Shorter way
if numbers:  # Empty lists are "falsy"
    print("List has items")
```

```
The list has 5 items
List has items
List has items
```

Modifying Lists

Lists are **mutable** - you can change them:

```python
grades = [85, 92, 78]
print(f"Original: {grades}")

# Change an item
grades[0] = 90
print(f"After change: {grades}")

# Add to the end
grades.append(88)
print(f"After append: {grades}")

# Remove a specific value
grades.remove(78)
print(f"After remove: {grades}")
```

```
Original: [85, 92, 78]
After change: [90, 92, 78]
After append: [90, 92, 78, 88]
After remove: [90, 92, 88]
```

Adding to a List

We can add to a list using a **method** called `append()`:

```python
tasks = ["homework", "dishes", "laundry"]

# Add a new task
tasks.append("grocery shopping")

print(f"Updated tasks: {tasks}")
```

```
Updated tasks: ['homework', 'dishes', 'laundry', 'grocery shopping']
```

The `append()` **method** is a special function that belongs to the `list` **instance** and modifies it in place.

Classes, Instances, and Methods

Classes

- A class is a blueprint that defines what data can be stored and what actions can be performed
- Example: The `list` class defines how lists work in Python

Instances

- An instance is a specific example created from a class, with its own data
- Example: `tasks = ["homework", "dishes"]` creates an instance of the list class

Methods

- A method is a function that belongs to an instance
- Called using dot notation: `instance.method()`
- Example: `tasks.append("laundry")` calls the `append` method on our list instance

List Methods

Lists have many more built-in methods to modify them:

```python
# Start with a list
numbers = [10, 20, 30, 40]

# Remove an item
numbers.remove(20)
print(numbers)  # [10, 30, 40]

# Insert at a specific position
numbers.insert(1, 15)  # Insert 15 at index 1
print(numbers)  # [10, 15, 30, 40]

# Remove last item and return it
last = numbers.pop()
print(f"Removed last item: {last}")  # 40
print(numbers)  # [10, 15, 30]
```

```
[10, 30, 40]
[10, 15, 30, 40]
Removed last item: 40
[10, 15, 30]
```

List Functions Summary

Method	What it does	Example
`append(item)`	Add to end	`lst.append(5)`
`remove(item)`	Remove first occurrence	`lst.remove(5)`
`pop()`	Remove & return last	`last = lst.pop()`
`insert(i, item)`	Insert at position	`lst.insert(0, 5)`
`clear()`	Remove all items	`lst.clear()`
`index(item)`	Find position	`pos = lst.index(5)`
`count(item)`	Count occurrences	`n = lst.count(5)`

Slicing Lists

Get a portion of a list with slicing:

```
letters = ['a', 'b', 'c', 'd', 'e', 'f']

# Get items 1-3 (not including 4)
print(letters[1:4])  # ['b', 'c', 'd']

# From beginning to index 3
print(letters[:3])   # ['a', 'b', 'c']

# From index 3 to end
print(letters[3:])   # ['d', 'e', 'f']
```

```
['b', 'c', 'd']
['a', 'b', 'c']
['d', 'e', 'f']
```

Iterating Over Lists

Method 1: Direct iteration

Best when you just need the values!

```python
fruits = ["apple", "banana", "cherry"]

for fruit in fruits:
    print(f"I like {fruit}")
```

```
I like apple
I like banana
I like cherry
```

Method 2: Index iteration

Use when you need the position too!

```python
fruits = ["apple", "banana", "cherry"]

for i in range(len(fruits)):
    print(f"{i}: {fruits[i]}")
```

```
0: apple
1: banana
2: cherry
```

Lists with Loops and Conditions

```python
numbers = [15, 8, 23, 42, 16, 4]

# Find all even numbers
evens = []
for num in numbers:
    if num % 2 == 0:
        evens.append(num)
print(f"Even numbers: {evens}")

# Count how many are over 20
count = 0
for num in numbers:
    if num > 20:
        count = count + 1
print(f"{count} numbers are over 20")
```

```
Even numbers: [8, 42, 16, 4]
2 numbers are over 20
```

Building Lists

Start with an empty list and grow it:

```python
# Collect user input
shopping_list = []

for i in range(3):
    item = input(f"Enter item {i+1}: ")
    shopping_list.append(item)

print(f"\nYour shopping list: {shopping_list}")

# Generate a list
squares = []
for num in range(1, 6):
    squares.append(num ** 2)
print(f"Squares: {squares}")
```

Common List Patterns

```python
scores = [85, 92, 78, 88, 95, 73]

# Find the sum
total = 0
for score in scores:
    total = total + score
print(f"Total: {total}")

# Find the maximum
highest = scores[0]  # Start with first
for score in scores:
    if score > highest:
        highest = score
print(f"Highest: {highest}")

# Calculate average
average = total / len(scores)
print(f"Average: {average:.1f}")
```

```
Total: 511
Highest: 95
Average: 85.2
```

Lists of Lists (2D Arrays)

Lists can contain other lists:

```python
# Tic-tac-toe board
board = [
    ['X', 'O', 'X'],
    ['O', 'X', 'O'],
    ['X', 'O', 'X']
]

# Access specific cell
print(board[0][0])  # Top-left: 'X'
print(board[1][1])  # Center: 'X'

# Print the board
for row in board:
    print(row)
```

```
X
X
['X', 'O', 'X']
['O', 'X', 'O']
['X', 'O', 'X']
```

Working with 2D Lists

```
# Grade table: [student][test]
grades = [
    [85, 92, 88],  # Student 0
    [78, 85, 90],  # Student 1
    [92, 95, 93]   # Student 2
]

# Average for student 0
total = 0
for grade in grades[0]:
    total = total + grade
avg = total / len(grades[0])
print(f"Student 0 average: {avg:.1f}")

# All grades
for i in range(len(grades)):
    for j in range(len(grades[i])):
        print(f"Student {i}, Test {j}: {grades[i][j]}")
```

```
Student 0 average: 88.3
Student 0, Test 0: 85
Student 0, Test 1: 92
Student 0, Test 2: 88
Student 1, Test 0: 78
```

Lists vs Strings

For the most part, we can treat strings like lists of single character strings:

```python
# Both can be indexed and sliced
word = "hello"
letters = ['h', 'e', 'l', 'l', 'o']

print(word[0])      # 'h'
print(letters[0])   # 'h'

# But strings are immutable!
# word[0] = 'H'     # ERROR!
letters[0] = 'H'    # This works!
print(letters)

# Convert between them
word_to_list = list("python")
print(word_to_list)
```

```
h
h
['H', 'e', 'l', 'l', 'o']
['p', 'y', 't', 'h', 'o', 'n']
```

Checking if Items Exist

Use `in` to check membership:

```python
inventory = ["sword", "shield", "potion"]

# Check if item exists
if "potion" in inventory:
    print("You have a potion!")

# Check if NOT in list
if "armor" not in inventory:
    print("You need armor!")

# In a loop
item_to_find = "shield"
found = False
for item in inventory:
    if item == item_to_find:
        found = True
        break
print(f"Found {item_to_find}: {found}")
```

```
You have a potion!
You need armor!
Found shield: True
```

Sorting Lists

```python
# Sort a list of numbers
numbers = [42, 15, 8, 23, 4]
numbers_numbers = sorted(numbers)
print(f"Sorted numbers: {numbers_numbers}")
# Sort a list of strings
names = ["Charlie", "Alice", "Bob"]
sorted_names = sorted(names)
print(f"Sorted names: {sorted_names}")
```

```
Sorted numbers: [4, 8, 15, 23, 42]
Sorted names: ['Alice', 'Bob', 'Charlie']
```

`sorted()` returns a new sorted list, leaving the original unchanged.

```python
# Sort in reverse order
numbers = [42, 15, 8, 23, 4]
reverse_sorted_numbers = sorted(numbers, reverse=True)
print(f"Reverse sorted numbers: {reverse_sorted_numbers}")
```

```
Reverse sorted numbers: [42, 23, 15, 8, 4]
```

Exercise: Shopping List App

bigd103.link/shopping-list