CS65: Introduction to Computer Science

Sequence (List)

Md Alimoor Reza
Assistant Professor of Computer Science

Announcement

Lab #4 is due by tonight October 16th Thursday for all sections

Content quiz#2 will take place today

- Topics:
  - Print formatting
  - `while` loop

Text Reading

https://analytics.drake.edu/~reza/teaching/cs65_fall25/readings/Chapter9.html

Introduction to Lists

Today's Plan

- Value for loop vs. Index For Loop
  - Hands-on Exercise 5 (parts a-e)
- Nested For Loop
  - Hands on exercise
- Sequence
  - String
  - List
  - Tuple

Review: Value `for loop` vs Index `for loop`

common practice is
the index variables w

- So far we have seen the syntax of **value** `for loop`

```
for var in [10, 20, 30, 40, 50]:
    print(var)
```

- There is another form called **index** `for loop`

```
my_list = [10, 20, 30, 40, 50]
length = len(my_list)
for i in range(length):
    print( "location ", i, " value is ", my_list[i] )
```

Review: **Value `for` loop** vs **Index `for` loop**

- **value** `for loop`
  - directly assigns a value to the variable from the sequence
  - don't keep track of the indices
  - we have access to only value
    - **good**

- **index** `for loop`
  - generates all the indices of all elements in the list
  - each element can be accessed indirectly by that index
  - we have access to both *i) index* and *ii) value*
    - **better**

Review: Nested **for** loops

- Putting one loop inside another
  - The first loop is called the outer loop
  - The second loop is called the inner loop

- Here is simpler version:

```
for i in range(3):
    for j in range(3):
        print("i: ", i, "j: ", j)
```

Review: Thonny output: nested **for** loop

```
# nested for loop
for i in range(3):
    print("Enters outer loop")
    for j in range(3):
        print("\tInner: i ->", i, " j ->", j)
```

```
>>> %Run lec10_demo.py

  Enters outer loop
          Inner: i -> 0  j -> 0
          Inner: i -> 0  j -> 1
          Inner: i -> 0  j -> 2
  Enters outer loop
          Inner: i -> 1  j -> 0
          Inner: i -> 1  j -> 1
          Inner: i -> 1  j -> 2
  Enters outer loop
          Inner: i -> 2  j -> 0
          Inner: i -> 2  j -> 1
          Inner: i -> 2  j -> 2
```

Review: Exercise#3

- Build a left-facing-triangle in the shell output that looks like this:

```
>>> %Run left_facing_triangle.py

  *
  **
  ***
  ****
  *****
  ******
  *******
  ********
  *********
  **********
```

Side length = 10

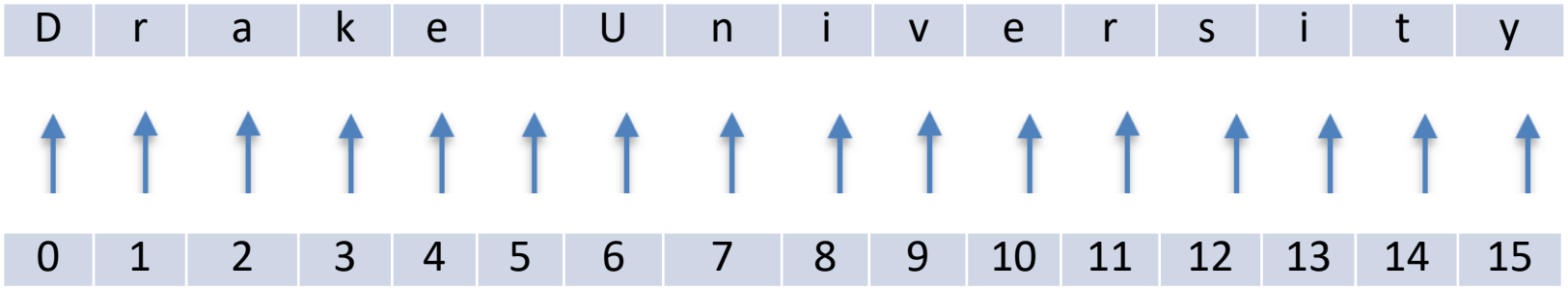
```
>>> %Run left_facing_triangle.py

  *
  **
  ***
  ****
  *****
  ******
  *******
  ********
  *********
  **********
  ***********
  ************
  *************
  **************
  ***************
  ****************
  *****************
  ******************
  *******************
  ********************
```

Side length = 20

- You can use any special character of your choice as a brick, and my favorite is the '*' character :)

Review: Sequence: Strings

- Sequence is an ordered group of elements (numbers, characters, etc)
- String is a sequence of characters
  - "Drake University"
  - "cs65:introduction_to_computer_science!"
- *length* of a String is the total number of characters
- Each position in a sequence is marked with an **index** or **position**
  - Starts (from left) at position *0* and ends at position (*length-1*)
  - Start indexing from the *left* to *right*



Review: Demo of Length of a Sequence

- How can you find the length of a string?
  - Use built-in ***len()*** function

```
my_string1 = "hello@world"
my_string2 = "Hi there!"
my_string3 = ""

print("Length of \"hello@world\" is: ", len(my_string1))
print("Length of \"Hi there!\" is: ", len(my_string2))
print("Length of \"\" is: ", len(my_string3))
```

```
Shell
Python 3.7.9 (bundled)
>>> %cd /Users/reza/Class_and_Resea
    slides/lecture10
>>> %Run lec10_demo.py

  Length of "hello@world" is:  11
  Length of "Hi there!" is:  9
  Length of "" is:  0

>>>
```

Review: Accessing Sequence Items with **Positive** Index
Left — — —> Right

- String is a sequence of characters

```
my_string1 = "Drake University"
```

- Access a specific item by appending ***brackets []*** containing an index

```
my_string1[0]      to access D
my_string1[1]      to access r
my_string1[2]      to access a
…
…
…
my_string1[15]     to access y
```

Review: Accessing Sequence Items with **Negative** Index

Left <— — — Right

- String is a sequence of characters and negative indexing begins at the end with a **-1** (not zero anymore)

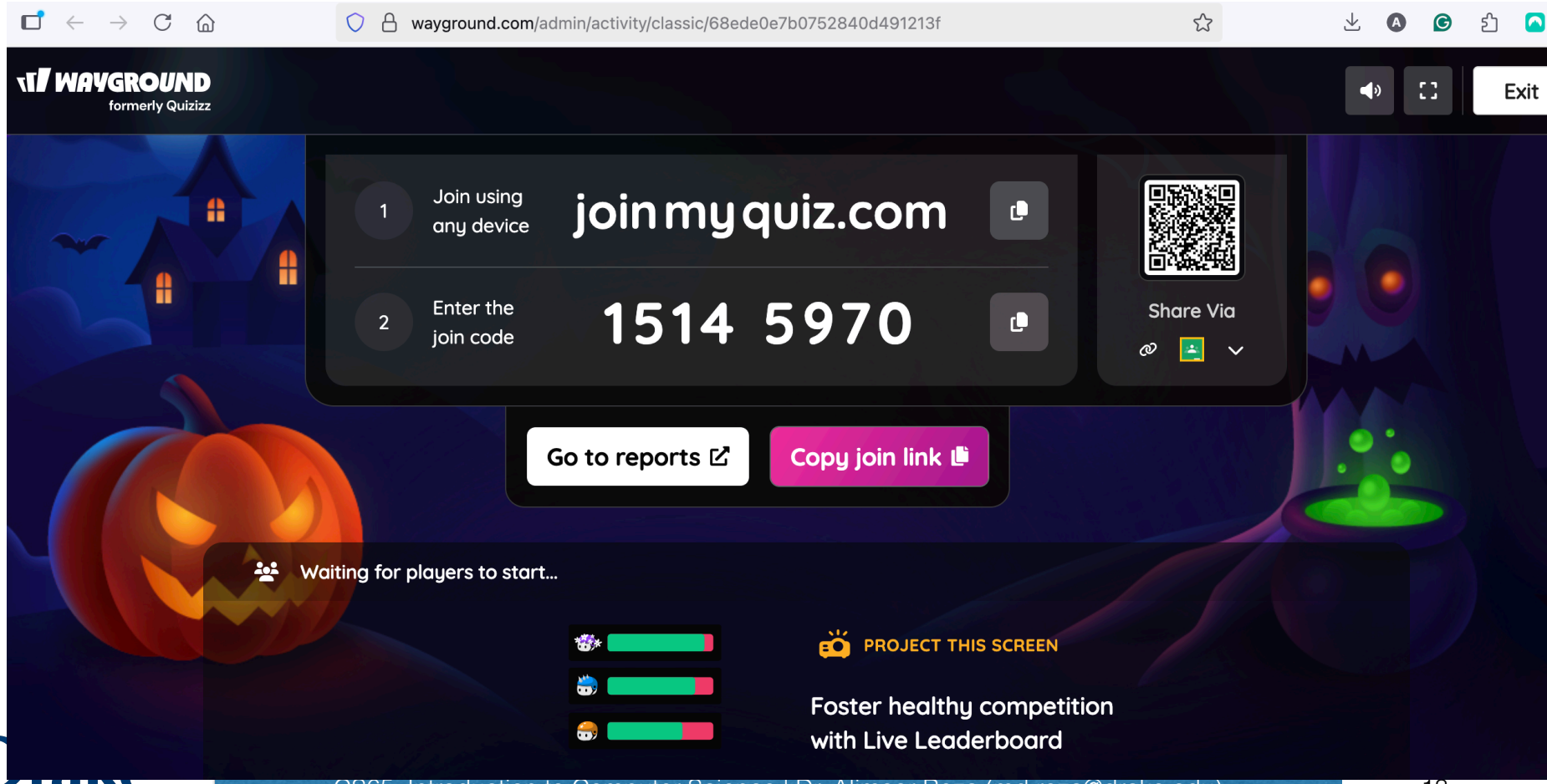
```
my_string1 = "Drake University"
```

- Access a specific item by appending ***brackets []*** containing an index

```
my_string1[-1]    to access  y
my_string1[-2]    to access  t
my_string1[-3]    to access  i
…
my_string1[-16]   to access  D
```

Poll: String and index

- Please participate in the poll below
  - https://wayground.com/join?gc=15145970



Today's Plan

- Sequence
  - String
  - List
  - Tuple

Intro to Lists

Containers

Some types in Python are used for storing a collection of values. These types are called *containers*.

List: a *container* type that stores values in a ordered sequence

List Examples

```
james_bond_actors = ["Connery","Niven","Lazenby","Moore","Dalton","Brosnan","Craig"]
rainfall_amounts = [0.0, 0.3, 0.71, 0.0, 0.21, 1.1, 0.4]
dates = ["March 14", 2021, "4/1"]
empty_list = []
```

values in a list are called **items** or **elements**,

e.g., `"Connery", 0.3`

List syntax: start with `[`, end with `]`, separate with `,`

Wait… why use lists?… couldn’t we just use variables?

Accessing items in a list

You can access individual items in a list using their **index**, which is their *position* in the list

indices start at 0

```
1  # T. Urness
2  # intro to lists
3
4  beatles = ["John", "Paul", "Ringo", "George"]
5  print(beatles[2])
6
7  rainfall_amounts = [0.0, 0.3, 0.71, 0.0, 0.32, 1.1, 0.4]
8  print(rainfall_amounts[1])
```

Lists2.py

Lists *index* notation

You can treat an item in a list just like any other variable:

```
rainfall_amounts = [0.1, 0.3, 0.71, 0.0, 0.32, 1.1, 0.4]
total_weekend_rainfall = rainfall_amounts[0] + rainfall_amounts[6]
print(total_weekend_rainfall)
```

Lists3.py

List indices

You will get an error if you try to access an item using an index that is too big for the list.

```
# You will get an error if you try to access an item using an index
# that is too big for the list.
beatles = ["John", "Paul", "Ringo", "George"]
print(beatles[5])
```

```
Traceback (most recent call last):
  File "/Users/000617123/Desktop/CS65/Day11/Lists4.py", line 7, in <module>
    print(beatles[5])
IndexError: list index out of range
```

Lists4.py

Lists; negative index values

But, if you use a negative index, it will count from the back of the list.

```
# But, if you use a negative index, it will count from the back of the list.
beatles = ["John", "Paul", "Ringo", "George"]
print(beatles[-1])
```

Lists4.py

Lists; negative index values

But, if you use a negative index, it will count from the back of the list.

```
# But, if you use a negative index, it will count from the back of the list.
beatles = ["John", "Paul", "Ringo", "George"]
print(beatles[-1])
```

Lists4.py

```
>>> %Run Lists4.py
 George
```

Printing and Updating items in a list

You can print out the entire contents of a list by using `print`

You can also use list notation to change values in a list

```
rainfall_amounts = [0.0, 0.3, 0.71, 0.0, 0.32, 1.1, 0.4]
rainfall_amounts[2] = 0.65
print(rainfall_amounts)
```

Lists5.py

Printing and Updating items in a list

You can print out the entire contents of a list by using `print`

You can also use list notation to change values in a list

```
rainfall_amounts = [0.0, 0.3, 0.71, 0.0, 0.32, 1.1, 0.4]
rainfall_amounts[2] = 0.65
print(rainfall_amounts)
```

Lists5.py

```
>>> %Run Lists5.py
  [0.0, 0.3, 0.65, 0.0, 0.32, 1.1, 0.4]
```

Length of a list

The number of elements within a list is also known as the length of the list

Python provides a built-in function that allows you to easily access the length:

```
len(list_name)
```

Lists6.py

```
glucose_levels = [140, 119, 153, 135, 160]
num_values = len(glucose_levels)
print(num_values)
```

Loops and Lists (this is one way – we'll cover another later)

What will this code do?

Lists6.py

```
glucose_levels = [140, 119, 153, 135, 160]
for i in range(len(glucose_levels)):
    print(glucose_levels[i])
```

Inserting Items in a List

- Appending elements in an empty list with **. append()** method. It adds items to the end of the list
  - A method instructs an object to perform some action
  - It is executed by a "**.**" (dot) symbol followed the method name

```
# building list with append() function
num_list = []

num_list.append(2)
print("num_list: ", num_list)
```

```
Shell
>>> %Run lec11.py

  num_list:  [2]
```

Inserting Items in a List

- Appending elements in an empty list with **.append()** method

```
# building list with append() function
num_list = []

num_list.append(2)
print("num_list: ", num_list)

num_list.append(4)
print("num_list: ", num_list)

num_list.append(6)
print("num_list: ", num_list)
```

```
>>> %Run lec11.py

  num_list:  [2]
  num_list:  [2, 4]
  num_list:  [2, 4, 6]
```

Inserting Items in a List Iteratively

- Appending elements in an empty list with **.append()** method iteratively using **for loop**

```
# building list with append function using loop
num_list = []

for i in range(10, 100, 10):
    num_list.append(i)

print("num_list: ", num_list)
```

```
>>> %Run lec11.py
  num_list:  [10, 20, 30, 40, 50, 60, 70, 80, 90]
>>>
```

Inserting Specific Items in a List Iteratively

- Appending odd numbers (from 1 to 20) in an empty list with **.append()** method iteratively using **for loop**

```
# building list with append function using loop
# insert only odd numbers
num_list = []

for num in range(1, 20):

    if (num % 2 != 0):

        num_list.append(num)

print("num_list: ", num_list)
```

```
>>> %Run lec11.py

  num_list:  [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
```

Exercise #1

1. Create an empty list (no elements in it) called **`my_list`**
2. Use **`append`** to add some numbers into the list (e.g. the last 4 digits of your phone number)
3. Print out the first and last elements in the list
4. Use a loop to print out *all* of the elements in the list
5. (challenging) Use a loop to append some random numbers to the list
6. (very challenging) Can you calculate the average of all of the values in the list?