SE Bootcamp

Hyperiondev

Lists & 2D Lists

Lecture – Housekeeping

- ❏ The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly.
- ❏ No question is daft or silly - **ask them!**
- ❏ There are Q/A sessions midway and at the end of the session, should you wish to ask any follow-up questions.
- ❏ You can also submit questions here: http://hyperiondev.com/sbc4-se-questions
- ❏ For all non-academic questions, please submit a query: www.hyperiondev.com/support
- ❏ Report a safeguarding incident: http://hyperiondev.com/safeguardreporting
- ❏ We would love your feedback on lectures: https://hyperionde.wufoo.com/forms/zsgv4m40ui4i0g/

Objectives

- **List methods and indexing**

- **Adding & removing elements**
  - **Append, extend and insert**
  - **Clear and pop**

- **List comprehension**

- **2D Lists**

Lists

★ Lists are used when we need to store a lot of data, or the order in which the data is stored is important.

★ Lists are capable of holding many items in one place as well as keeping the data in order.

★ Python will also provide each piece of data an index that represents its position in the list.

Lists Cont.

★ A list is a specialised format of storing and organising data.

★ A list is basically a group of items / data.

★ Lists are known as sequence data types because they behave like an ordered collection of items.

Indexing Lists

★ Similar to strings, we are able to index and slice lists.

★ However, instead of indexing by character, we index lists by the entire value in that specific position.

Indexing Example

```
names = ["Billy", "Sally", "Cammy"]
print(names[0])

# Result >> "Billy"

print(names[-1])

# Result >> "Cammy"
```

Finding the Length of a List

★ Similar to what we did with strings, we can use the len() function to find the length of a list.

★ Example :

```python
my_list = ["The", "Joy", "of", "Learning"]

print(len(my_list))

# Result >> 4
```

Accessing all values in a List

★ Sometimes we need to access all values / items in a list at the same time; to achieve this we can simply iterate through the list with a for loop

★ This would be especially useful when we cannot exactly see all the data within a list and we must evaluate the data / make adjustments / add to the list / remove from the list.

Hyperiondev

Looping through Lists Example

```python
names = ["Jimmy", "Billy", "Terry", "Kerry", "Joe"]

# Remember that the temp variable in a for loop
#    can be named anything and that it only
#        exists within the loop.
for i in names:
    print(names)


# Result >> Jimmy
#           Billy
#           Terry
#           Kerry
#           Joe
```

In Operator and Lists

```python
names = ["Jimmy", "Billy", "Terry", "Kerry", "Joe"]

name_one = "Lucy"
name_two = "Terry"

if name_one in names:
    print('''
    This if statement will NOT execute, because Lucy
    does not exist in the list
    ''')

if name_two in names:
    print('''
    This if statement will execute, because Terry
    does in fact exist within the list
    ''')
```

Appending data to Lists

★ You can add new items to a list by using the .append() method, keep in mind that append will only add to the end of a list and nowhere else.

★ Example:

```
names = ["Jimmy", "Billy", "Terry", "Kerry", "Joe"]

names.append("Sally")
# The list is now updated with the new item

print(names)

# Result >> ['Jimmy', 'Billy', 'Terry', 'Kerry', 'Joe', 'Sally']
```

Extending Lists

★ Adds multiple values to the end of the list.

★ Similar to append, but adds multiple values instead of only one.

```python
numbers = [1,2,3,4]

numbers.extend([5,6,7,8])

print(numbers)

# Result >> [1,2,3,4,5,6,7,8]
```

Inserting into List

★ Used to add items at a specific position in the list using indexing.

★ Takes two arguments, first is the index, followed by the element to add.

```python
numbers = [1, 2, 3, 4, 5]

numbers.insert(2, 'Hi')

print(numbers)

# Result >> [1, 2, 'Hi', 3, 4, 5]
```

Clearing a List

★ The clear method will remove all elements from a list

★ Keep in mind that .clear() needs no arguments

```
numbers = [1,2,3,4,5]

numbers.clear()

print(numbers)

# Result >> []
```

Popping from a List

★ The pop method will remove an element at an index, then return it.
★ Return means that the removed element can be stored and used in a variable

```python
numbers = [1, 2, 3, 4, 5]

popped_number = numbers.pop()
# If no index is specified, then pop will remove
#    the last element in the list

print(popped_number)

# Result >> 5
```

Hyperiondev

List Comprehension

★ A variant for loop, used to generate new lists. Basically tweaked versions of existing lists.

★ Written as : _(1)_ for _(2)_ in _(3)_

★ First position contains the action to be taken on each element of a list

★ Second position contains the index variable, and the third position is the list we iterate over.

★ Reads like : "for _(2)_ in _(3)_, do _(1)_"

Hyperiondev

Example

```python
numbers = [1, 2, 3, 4, 5]

double_numbers = [num * 2 for num in numbers]

print(double_numbers)

# Result >> [2, 4, 6, 8, 10]
```

2D Lists

- How does a list become two-dimensional?
- Create a list, where each element is a list.
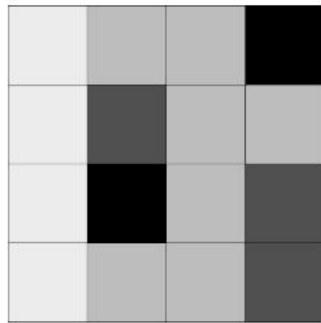
236	189	189	0
236	80	189	189
236	0	189	80
236	189	189	80

- **grayscale_image = [[236, 189, 189, 0], [236, 80, 189, 189], [236, 0, 189, 80], [236, 189, 189, 80]]**

Hyperiondev

How to Create a 2D List

number_of_rows = 3
number_of_columns = 2
empty_grid = [[None] * number_of_columns for _ in range(number_of_rows)]

- Creates a 2 x 3 list, all elements are the value None.

Output: [[None, None], [None, None], [None, None]]

How to Iterate over a 2D List

```python
for row in my_2d_list:  # outer loop for rows
    for col in row:  # inner loop for columns
        # Do something.


my_2d_list = [1, 2, 3, 4, 5, 6]


for row in my_2d_list:
    for col in row:
        print(col)
```

Indexing a 2D List

```
grayscale_image = [[236, 189, 189,   0],
                   [236,  80, 189, 189],
                   [236,   0, 189,  80],
                   [236, 189, 189,  80]]

last_pixel = grayscale_image[3][3]
```
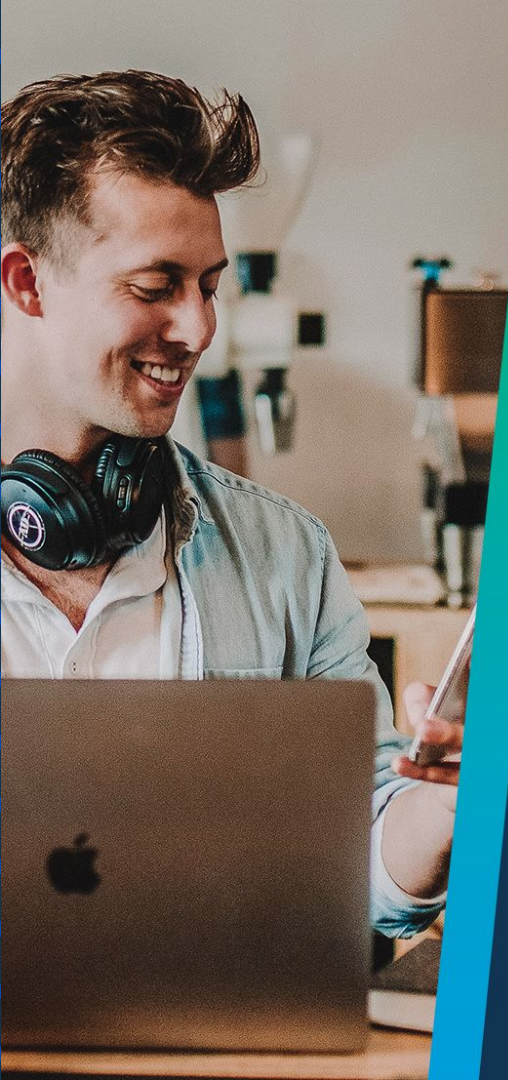
Hyperiondev

Hyperiondev

Q & A Section

Please use this time to ask any questions relating to the topic, should you have any.

Hyperiondev

Thank You for Joining Us