Welcome to Data Science Online Bootcamp

Day 2

dφ

Democratizing Data Science Learning

Learning Objectives

While loop

For loop

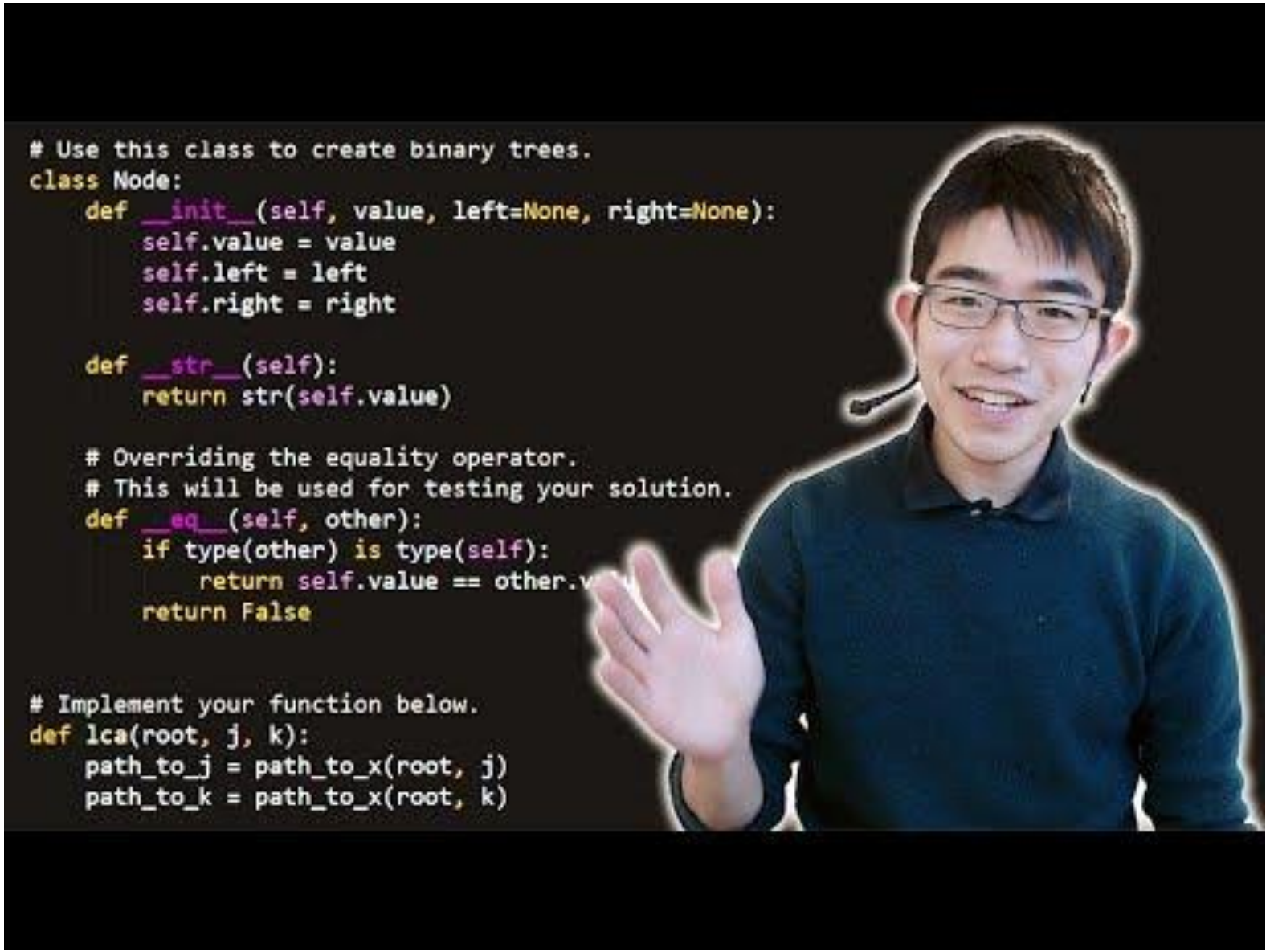
Enumerate

Range

Loops

- A loop is a sequence of instructions that is continually repeated until a certain condition is reached.
- Think about a teacher taking attendance. The teacher calls out a name, and the student responds present.
- The teacher is going through the list of students one by one and calling a name to get “present” if the student is present,
  or “<<silence>>” if they are absent.
- This process will only end when the whole list of students are completed. This will be the condition that will break out of the loop.
- Let’s have a look at the types of loops present in Python!

For Loop


# Use this class to create binary trees.
class Node:
def __init__(self, value, left=None, right=None):
self.value = value
self.left = left
self.right = right
def __str__(self):
return str(self.value)
# Overriding the equality operator.
# This will be used for testing your solution.
def __eq__(self, other):
if type(other) is type(self):
return self.value == other.v
return False
# Implement your function below.
def lca(root, j, k):
path_to_j = path_to_x(root, j)
path_to_k = path_to_x(root, k)

For Loop

- “For loop” is used for **iterating over a sequence** (that can be any data structure - list/tuples or even a string)
- Iteration means performing an action repeatedly
- Syntax:

```
for variable in sequence:
    expression
```

- Which means " for each variable in sequence, execute the expression"

Looping through a List

- Example:
  Let's say we wish to store the heights of our family members in a list and print them one by one.

```
fam_heights = [1.73, 1.68, 1.71, 1.89]
for height in fam_heights:
    print(height)
```

```
1.73
1.68
1.71
1.89
```

- Internal Working:
  - First, we store all the heights in a list named fam_heights
  - Now, we'll go to each element and print it
  - This action will continue until all the elements of the list are printed in order.

Looping through a String

- Even strings are iterable objects, they contain a sequence of characters:

```
for i in "apple":
    print(i)
```

```
a
p
p
l
e
```

- As you can see, each character of the string is printed in a separate line.
- We can even apply string methods in the for loop.

```
for c in "family":
    print(c.capitalize())
```

```
F
A
M
I
L
Y
```

Enumerate

- With the for loop, you were able to print the heights of your family members.
- But what if you want to access the index of each element of the list as well? Here is where the enumerate function comes into play.

- The enumerate function iterates over the elements of a list and associates an index with them. You need to use 2 variables (index and height in this case) to store the values given by enumerate.

```
for index,height in enumerate(fam_heights):
    print("index " + str(index) + " : " + str(height) )
```

```
index 0 : 1.73
index 1 : 1.68
index 2 : 1.71
index 3 : 1.89
```

While Loop


jupyter
While Loops and The Break Statement
Logout
File
Edit
View
Insert
10
In [6]:
given_list = [5,
total3 = 0
i = 0
while i < len(g
total3 += g
i += 1
print(total3)
17
In [ ]:
given_list2 = [5, 4, 4, 3, 1, -2, -3, -5]
Yes, we can.

While Loop

- With the while loop we can execute a set of statements repeatedly **as long as a condition is true.**

- Syntax:

```
while condition:
    statement(s)
```

- All the statements indented by the same number of character spaces after a while condition are considered to be part of a single block of code. **Python uses indentation as its method of grouping statements.**

While Loop

- For example:

```
x = 1
while x < 4 :
    print(x)
    x = x + 1
```

```
1
2
3
```

- Let us understand what is happening in the above program.
  - Value of x is assigned as 1
  - The while loop starts with a condition that x must be less than 4
  - The next 2 statements have the same indentation and will be considered a part of while loop
  - The value of x is printed. Initially, x=1
  - The value of x is then increased by 1. So now, x=2
  - Control goes back to the condition line, x is less than 4 (2<4). This means that the following statements will be executed again
  - This continues until x is assigned a value of 4. Then, the condition fails as x is not less than 4
  - The next 2 statements will not be executed and the while loop will end.

Range

- range() allows user to generate a series of numbers within a given range.

- The user can decide where that series of numbers will begin and end as well as how big the difference will be between one number and the next.

- range() takes mainly three arguments:
  - A **start** argument is a starting number of the sequence. i.e., lower limit. By default, it starts with 0 if not specified.
  - A **stop** argument is an upper limit. i.e., generate numbers up to this number, The **range() doesn't include this number in the result.**
  - The **step** is a difference between each number in the result. The default value of the step is 1 if not specified.

- range() only works with the integers. You can not use float number or any other type in a start, stop and step argument of a range().

Range

- Example 1: Using only one argument

```
# Print first 5 numbers using range function
for i in range(5):
    print(i, end=', ')
```

By default, print statement prints in different lines. If you want to print output in the same line, you can use the end argument.
',' specifies that each output will be separated by a comma.

Output:

```
0, 1, 2, 3, 4,
```

- Only a stop argument is passed to range() . So by default, it takes start = 0 and step = 1.

Range

- Example 2: Using two arguments (i.e., start and stop)

```
# Print integers within given start and stop number using range()
for i in range(5, 10):
    print(i, end=', ')
```

Output:

```
5, 6, 7, 8, 9,
```

- By default, it took step value as 1.

Range

- Example 3: Using all three arguments

```
# using start, stop, and step arguments in range()
print("Printing All even numbers between 2 and 10 using range()")
for i in range(2, 10, 2):
    print(i, end=', ')
```

Output:

```
Printing All even numbers between  and 10 using range()

2, 4, 6, 8,
```

- All three arguments are specified i.e., start = 2, stop = 10, step = 2. The step value is 2 so the difference between each number is 2.

More concepts

- A few more concepts like break, continue and nested loops are commonly used in Python.
- **Must learn:** Learn about important these concepts from the below cheatsheet:

  https://www.codecademy.com/learn/learn-python-3/modules/learn-python3-loops/cheatsheet

  Tip: *If you are unable to follow, run the code and make out the difference*

- Find out how the range function works with strings and list.
  Hint: *You'll use their lengths instead*

Let's Practice!

1. Print First 10 natural numbers using while loop.

2. Iterate over the following list and print the elements:

```
list1 = [12, 15, 32, 42, 55, 75, 122, 132, 150, 180, 200]
```

3. Accept a number n from user and print its multiplication table

4. Use the enumerate function to print the elements of this list along with the indices:

```
grocery = ['bread', 'milk', 'butter']
```

5. Write a program to input number from user and find sum of all numbers between 1 to n.

Let's Practice!

6. Create a sequence of numbers from 3 to 5, and print each item in the sequence.

7. Create a sequence of numbers from 3 to 19, but increment by 2 instead of 1.

8. Print the letters of the string "Python" in the same line:
   a. Using a simple for loop
   b. Using the range function

The above questions are for self-practice and ungraded, you don't need to upload them on Learning Platform.

You can attempt the quiz after this!

That’s it for the day. Thank you!

Feel free to post any queries in the #help channel on Slack