# Python Basics & Web Scraping

William Trang

# What is Python?

Python is a versatile and beginner-friendly programming language

It has uses cases within many different kinds of domains and is popular for its readability and ease of use

# Lists

A list is a data structure that can hold other elements. It is mutable and ordered, so you can add/remove elements at any time, and access any element in the list. The elements do not have to be unique. They are fairly similar to arrays in JS

# Dictionaries

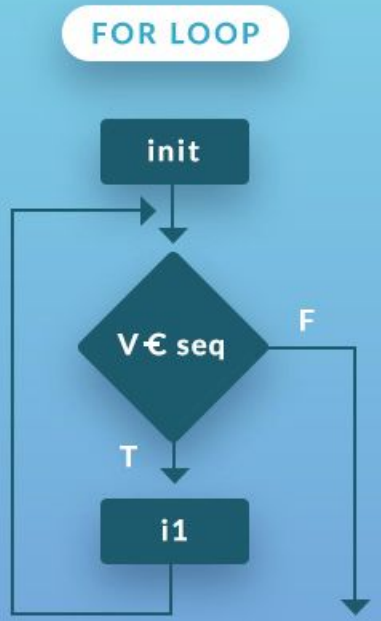
Dictionaries are another data structure in Python. It stores objects in key-value pairs and does not allow for duplicates. They are fairly similar to JSON objects

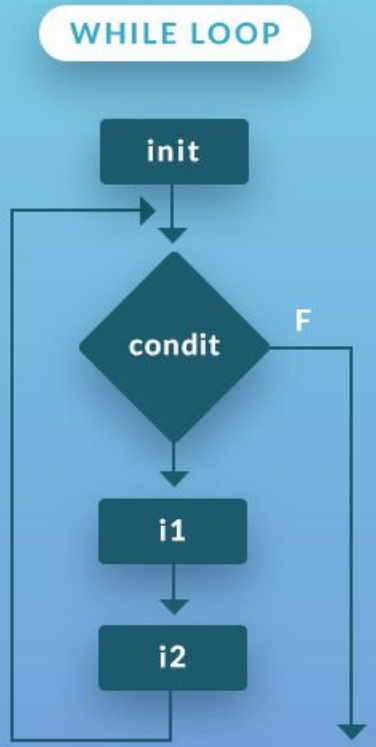
# Loops

Loops are ways to repeatedly run code until a certain condition is met. There are two types of loops in Python: for loops and while loops. The code inside of a loop will run until the loop exits, based on its conditions

For loops are typically used when you have a set number of operations you want to go through. They can also be used to loop through things, such as a list

While loops, on the other hand, keep iterating until a preset condition is met




DataCamp

# Functions

A function is a block of instructions that performs an action and, once defined, can be reused. Functions make code more modular, allowing you to use the same code over and over again

# Libraries for web scraping in Python

There exist two main libraries for web scraping in Python; Selenium and BeautifulSoup. I’m going to talk more about them in the upcoming slides

# Selenium

Selenium is a powerful and popular open-source automation testing framework primarily used for web applications. It provides a way to automate browser actions, interact with web elements, and perform functional testing of web pages. Basically, it can scrape Javascript

# BeautifulSoup

BeautifulSoup is a library specifically designed for web scraping. BeautifulSoup parses HTML or XML documents, allowing you to extract and manipulate data from web pages

# Creating the soup object

A soup object is a representation of the document's HTML that can be interacted with in Python

It can be created by the function

BeautifulSoup(text, parser)

Ex.

soup = BeautifulSoup(r.text, 'html.parser')

# .find() and .find_all()

Returns the first and all instances of the HTML objects with the specifications, respectively. .find_all() returns a list

You can put in things like tag names, attributes, etc

ex.

soup.find_all('p', {'class': 'price_color'})

Finds all p tags with the class ‘price_color’

# .select()

Like .find(), but uses CSS selectors instead of HTML ones

For example, soup.select(‘#nav’) will select the element with the ID ‘nav’

Similarly, soup.select(‘div div p’) will find all p elements within 2 divs

# .get()

Finds the value of an attribute

Ex.

link.get(‘href’) will return the value of the href attribute in the variable link

# Some quick ones

.contents - all contents of a soup

.children - all immediate child elements

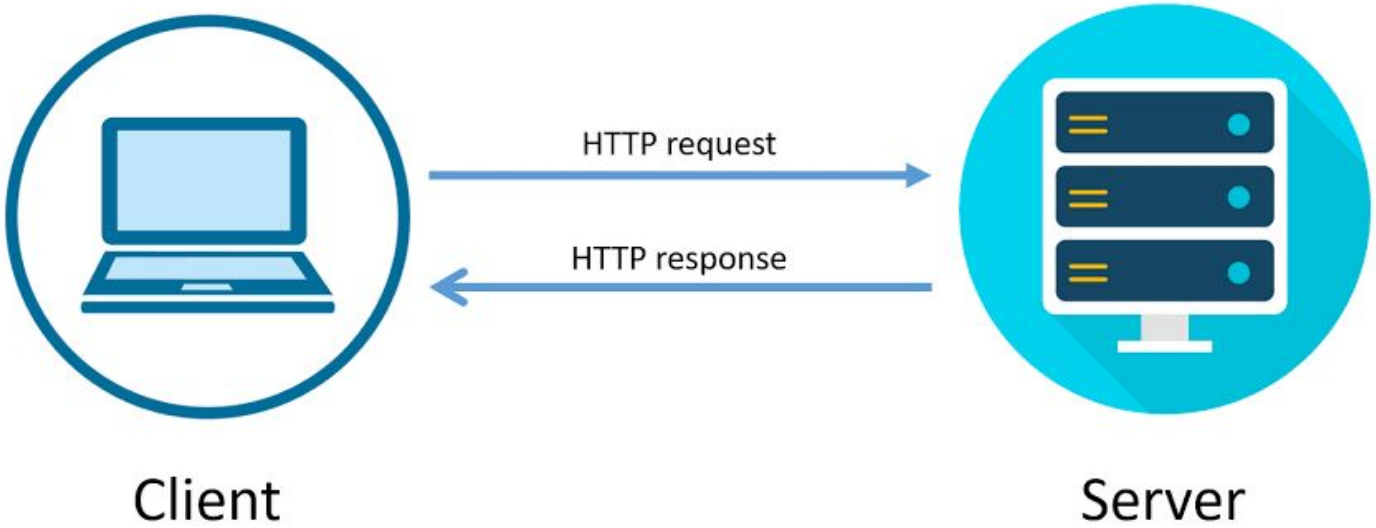
.parent and .parents - all parent elements of the soup

.has_attr - check if a tag has a specific attribute (ie. id, class)

.has_class - check if a tag has a specific class

# HTTP Requests

An HTTP (Hypertext Transfer Protocol) request is a message sent by a client to a server to request some action. In the context of web development, HTTP is the protocol used for communication between a web browser (the client) and a web server



# Requests

The requests library is a powerful tool for making HTTP requests in Python. Basically, it simplifies the process of requesting for information from the client

An example of this done in python is shown below

requests.get(‘https://google.com/’)

# Requests methods

There are 3 main properties in the HTTP response

content - the actual content of the page (the HTML)

headers - basically any other information sent by the client that’s not the HTML

status code - the status of your request (200 = good, 404 = bad, etc.)