RANDOM QUESTION GENERATOR

ABSTRACT

ABSTRACT

The Random Question Generator is a Django-based web application that aims to provide a dynamic and engaging user experience by generating random questions for users to answer. The application leverages the Django framework's capabilities to create a database of diverse questions and deliver them randomly to users.

This project focuses on the development of a user-friendly interface where users can interact with the application and receive random questions tailored to their preferences. The application allows users to select question categories, difficulty levels, or specific topics to customize the question generation process.

KEYWORDS – MCQ’s, Django, MySQL, React JS, CSS, HTML.

INTRODUCTION

INTRODUCTION

The Random Question Generator is a web application designed to provide users with an interactive and engaging experience by generating random questions for them to answer. This project utilizes the Django framework, a powerful Python web framework, to create a dynamic platform that delivers a variety of questions to users.

The primary goal of the Random Question Generator is to create an enjoyable and educational experience for users. By presenting them with random questions from various categories and difficulty levels, the application aims to stimulate learning, critical thinking, and problem-solving skills.

The application allows users to customize their question generation process by selecting specific categories, difficulty levels, or topics of interest. This personalization ensures that users receive questions that align with their preferences and areas of knowledge.

Behind the scenes, the Random Question Generator utilizes Django's database management capabilities. A collection of questions is stored in the application's database, each tagged with relevant metadata such as category, difficulty, and topic. This structured data allows for efficient retrieval and random selection of questions to be presented to users.

The application incorporates features that enhance user engagement and interactivity. Users can receive instant feedback on their answers, track their performance over time, and challenge themselves with time-limited quizzes. Additionally, the ability to bookmark questions, review previous answers, and explore question statistics provides a comprehensive and personalized user experience.

To ensure a visually appealing and intuitive user interface, the Random Question Generator utilizes Django's template system and front-end technologies. The application follows security best practices, implementing user authentication and authorization to protect user data and maintain privacy.

By creating a dynamic platform that generates random questions, the Random Question Generator offers users an opportunity to expand their knowledge, test their understanding, and have fun while doing so. This project highlights the versatility and capabilities of Django in building interactive web applications and emphasizes the importance of engaging users through personalized question generation.

Using Django's models and database management, a collection of questions is stored in the application's database. The questions are tagged with relevant metadata, such as category, difficulty, and topic, to enable efficient retrieval and random selection. The application dynamically generates random questions based on the user's preferences and displays them on the user interface.

To enhance user engagement, the Random Question Generator incorporates features like scoring, time limits, and progress tracking. Users can answer questions, receive instant feedback on their responses, and track their performance over time. Additionally, the application allows users to bookmark questions, review their previous answers, and explore question statistics.

By utilizing Django's template system and front-end technologies, the Random Question Generator ensures an intuitive and visually appealing user interface. The application follows best practices for security, including user authentication and authorization to safeguard user data and ensure privacy.

The Random Question Generator provides an interactive and educational platform for users to test their knowledge, learn new information, and have fun. The project demonstrates the capabilities of Django in creating dynamic web applications and emphasizes the importance of user engagement through personalized question generation.

RELATED WORKS

"Quizzes for Learning: A Controlled Comparison Study of Computer-Based and Traditional Quizzes" by Delgado-Kloos, C., et al. (2010):

This research paper explores the effectiveness of computer-based quizzes in enhancing learning outcomes compared to traditional quizzes. It investigates the benefits of randomized questions and personalized feedback, which are relevant to the development of a random question generator.

"Design and Implementation of a Web-Based Intelligent Tutoring System with a Randomized Question Generation Mechanism" by Vlachos, I., & Pomportsis, A. (2014):

This study presents the design and implementation of a web-based intelligent tutoring system that incorporates a randomized question generation mechanism. It discusses the benefits of generating questions dynamically and personalizing the learning experience for users.

"Generating High-Quality Multiple Choice Questions with Deep Neural Networks" by Dukkipati, A., et al. (2019):

This paper explores the use of deep neural networks to automatically generate high-quality multiple-choice questions. Although not specific to random question generation, it provides insights into question generation techniques and can inform the development of a random question generator.

"A Random Question Generation Algorithm for Adaptive E-Learning Systems" by Alhaddad, R. (2013):

This research paper proposes a random question generation algorithm for adaptive e-learning systems. It discusses the benefits of randomizing questions to create adaptive and personalized learning experiences.

"A Framework for Generating Multiple Choice Questions Automatically" by Kumar, P., et al. (2019):

This study presents a framework for automatically generating multiple-choice questions using natural language processing techniques. While not focused on random question generation, it offers insights into automated question generation processes.

PROBLEM STATEMENT:

The problem at hand is the need to develop a random question generator, which is a web application that generates and presents random questions to users. The goal is to provide an interactive and engaging learning experience by offering diverse questions from various categories and difficulty levels.

The problem statement encompasses the following key aspects:

Question Generation: Designing an algorithm or mechanism to generate random questions dynamically. The generator should have the ability to generate questions from different categories, such as science, history, mathematics, etc., and vary the difficulty levels to cater to users with varying levels of knowledge.

Personalization and Customization: Incorporating features that allow users to customize the question generation process according to their preferences. Users should be able to select specific categories, topics, or difficulty levels, enabling them to focus on areas of interest or challenge themselves with more advanced questions.

User Interface and Experience: Designing an intuitive and user-friendly interface to present the random questions to users. The interface should provide clear instructions, visually appealing question formats, and options for users to provide answers or select choices. Additionally, features such as instant feedback, timers, and progress tracking can enhance the overall user experience.

Database Management: Implementing a database or storage system to store a collection of questions. The database should support efficient retrieval of random questions based on various criteria, such as category, difficulty level, or user preferences.

Security and Data Privacy: Ensuring the security and privacy of user data. Implementing appropriate measures to protect user information, including secure user authentication, data encryption, and adherence to privacy regulations.

FRONT END BLOCK

HTML

HTML (**H**yper**T**ext **M**arkup **L**anguage) is the code that is used to structure a web page and its content. For example, content could be structured within a set of paragraphs, a list of bulleted points, or using images and data tables. As the title suggests, this article will give you a basic understanding of HTML and its functions.

So what is HTML?

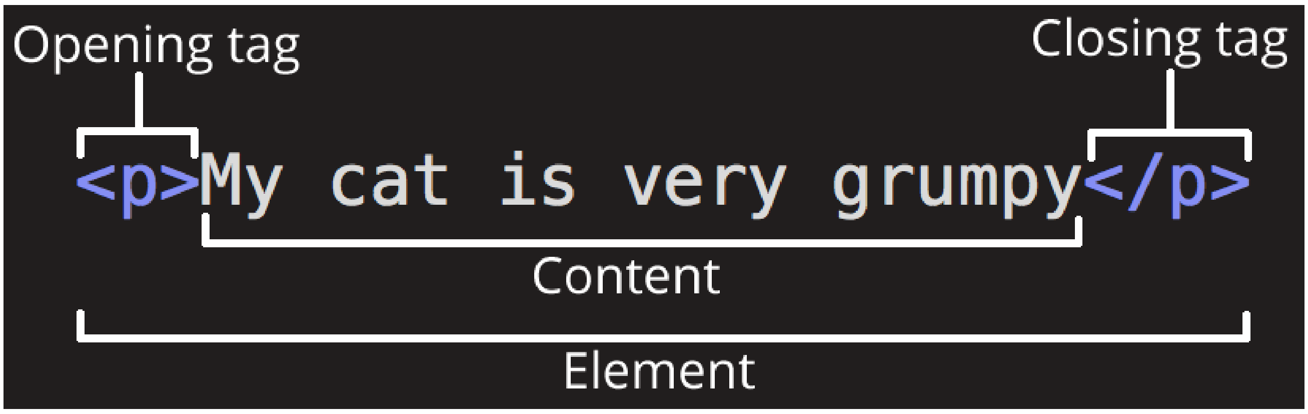
HTML is a *markup language* that defines the structure of your content. HTML consists of a series of elements, which you use to enclose, or wrap, different parts of the content to make it appear a certain way, or act a certain way. The enclosing tags can make a word or image hyperlink to somewhere else, can italicize words, can make the font bigger or smaller, and so on. For example, take the following line of content:

My cat is very grumpy

If we wanted the line to stand by itself, we could specify that it is a paragraph by enclosing it in paragraph tags:

HTMLCopy to Clipboard
<p>My cat is very grumpy</p>
Anatomy of an HTML element

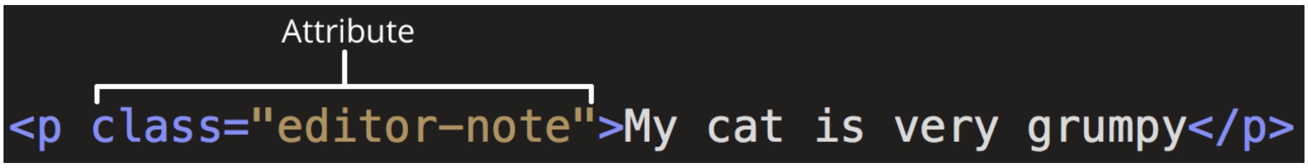
Let's explore this paragraph element a bit further.



The main parts of our element are as follows:

1. **The opening tag:** This consists of the name of the element (in this case, p), wrapped in opening and closing **angle brackets**. This states where the element begins or starts to take effect — in this case where the paragraph begins.
2. **The closing tag:** This is the same as the opening tag, except that it includes a *forward slash* before the element name. This states where the element ends — in this case where the paragraph ends. Failing to add a closing tag is one of the standard beginner errors and can lead to strange results.
3. **The content:** This is the content of the element, which in this case, is just text.
4. **The element:** The opening tag, the closing tag, and the content together comprise the element.

Elements can also have attributes that look like the following:



```
<p class="editor-note">My cat is very grumpy</p>
```

Attributes contain extra information about the element that you don't want to appear in the actual content. Here, class is the attribute *name* and editor-note is the attribute *value*. The class attribute allows you to give the element a non-unique identifier that can be used to target it (and any other elements with the same class value) with style information and other things. Some attributes have no value, such as required.

Attributes that set a value always have:

1. A space between it and the element name (or the previous attribute, if the element already has one or more attributes).
2. The attribute name followed by an equal sign.
3. The attribute value wrapped by opening and closing quotation marks.

Note: Simple attribute values that don't contain ASCII whitespace (or any of the characters " ' ` = < >) can remain unquoted, but it is recommended that you quote all attribute values, as it makes the code more consistent and understandable.

Nesting elements

You can put elements inside other elements too — this is called **nesting**. If we wanted to state that our cat is **very** grumpy, we could wrap the word "very" in a <strong> element, which means that the word is to be strongly emphasized:

HTMLCopy to Clipboard
```
<p>My cat is <strong>very</strong> grumpy.</p>
```

You do however need to make sure that your elements are properly nested. In the example above, we opened the <p> element first, then the <strong> element; therefore, we have to close the <strong> element first, then the <p> element. The following is incorrect:

HTMLCopy to Clipboard
```
<p>My cat is <strong>very grumpy.</p></strong>
```

The elements have to open and close correctly so that they are clearly inside or outside one another. If they overlap as shown above, then your web browser will try to make the best guess at what you were trying to say, which can lead to unexpected results. So don't do it!

Void elements

Some elements have no content and are called void elements. Take the <img> element that we already have in our HTML page:

HTMLCopy to Clipboard
```
<img src="images/firefox-icon.png" alt="My test image" />
```

This contains two attributes, but there is no closing </img> tag and no inner content. This is because an image element doesn't wrap content to affect it. Its purpose is to embed an image in the HTML page in the place it appears.

Anatomy of an HTML document

That wraps up the basics of individual HTML elements, but they aren't handy on their own. Now we'll look at how individual elements are combined to form an entire HTML page. Let's revisit the code we put into our index.html example (which we first met in the Dealing with files article):

HTMLCopy to Clipboard
```
<!doctype html>
<html lang="en-US">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width" />
    <title>My test page</title>
  </head>
  <body>
    <img src="images/firefox-icon.png" alt="My test image" />
  </body>
</html>
```

Here, we have the following:

- <!DOCTYPE html> — doctype. It is a required preamble. In the mists of time, when HTML was young (around 1991/92), doctypes were meant to act as links to a set of rules that the HTML page had to follow to be considered good HTML, which could mean automatic error checking and other useful things. However, these days, they don't do much and are basically just needed to make sure your document behaves correctly. That's all you need to know for now.
- <html></html> — the <html> element. This element wraps all the content on the entire page and is sometimes known as the root element. It also includes the lang attribute, setting the primary language of the document.
- <head></head> — the <head> element. This element acts as a container for all the stuff you want to include on the HTML page that *isn't* the content you are showing to your page's viewers. This includes things like keywords and a page description that you want to appear in search results, CSS to style our content, character set declarations, and more.
- <meta charset="utf-8"> — This element sets the character set your document should use to UTF-8 which includes most characters from the vast majority of written languages. Essentially, it can now handle any textual content you might put on it. There is no reason not to set this, and it can help avoid some problems later on.
- <meta name="viewport" content="width=device-width"> — This viewport element ensures the page renders at the width of viewport, preventing mobile browsers from rendering pages wider than the viewport and then shrinking them down.
- <title></title> — the <title> element. This sets the title of your page, which is the title that appears in the browser tab the page is loaded in. It is also used to describe the page when you bookmark/favorite it.
- <body></body> — the <body> element. This contains *all* the content that you want to show to web users when they visit your page, whether that's text, images, videos, games, playable audio tracks, or whatever else.

Images

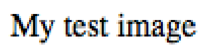
Let's turn our attention to the <img> element again:

HTMLCopy to Clipboard
```
<img src="images/firefox-icon.png" alt="My test image" />
```

As we said before, it embeds an image into our page in the position it appears. It does this via the src (source) attribute, which contains the path to our image file.

We have also included an alt (alternative) attribute. In the alt attribute, you specify descriptive text for users who cannot see the image, possibly because of the following reasons:

1. They are visually impaired. Users with significant visual impairments often use tools called screen readers to read out the alt text to them.
2. Something has gone wrong causing the image not to display. For example, try deliberately changing the path inside your src attribute to make it incorrect. If you save and reload the page, you should see something like this in place of the image:


My test image

The keywords for alt text are "descriptive text". The alt text you write should provide the reader with enough information to have a good idea of what the image conveys. In this example, our current text of "My test image" is no good at all. A much better alternative for our Firefox logo would be "The Firefox logo: a flaming fox surrounding the Earth."

Try coming up with some better alt text for your image now.

Note: Find out more about accessibility in our accessibility learning module.

Marking up text

This section will cover some essential HTML elements you'll use for marking up the text.

Headings

Heading elements allow you to specify that certain parts of your content are headings — or subheadings. In the same way that a book has the main title, chapter titles, and subtitles, an HTML document can too. HTML contains 6 heading levels, <h1> - <h6>, although you'll commonly only use 3 to 4 at most:

HTMLCopy to Clipboard
```
<!-- 4 heading levels: -->
<h1>My main title</h1>
<h2>My top level heading</h2>
<h3>My subheading</h3>
<h4>My sub-subheading</h4>
```
Note: Anything in HTML between <!-- and --> is an **HTML comment**. The browser ignores comments as it renders the code. In other words, they are not visible on the page - just in the code. HTML comments are a way for you to write helpful notes about your code or logic.

Now try adding a suitable title to your HTML page just above your <img> element.

Note: You'll see that your heading level 1 has an implicit style. Don't use heading elements to make text bigger or bold, because they are used for accessibility and other reasons such as SEO. Try to create a meaningful sequence of headings on your pages, without skipping levels.

Paragraphs

As explained above, <p> elements are for containing paragraphs of text; you'll use these frequently when marking up regular text content:

HTMLCopy to Clipboard
```
<p>This is a single paragraph</p>
```

Add your sample text (you should have it from *What will your website look like?*) into one or a few paragraphs, placed directly below your <img> element.

Lists

A lot of the web's content is lists and HTML has special elements for these. Marking up lists always consists of at least 2 elements. The most common list types are ordered and unordered lists:

1. **Unordered lists** are for lists where the order of the items doesn't matter, such as a shopping list. These are wrapped in a <ul> element.
2. **Ordered lists** are for lists where the order of the items does matter, such as a recipe. These are wrapped in an <ol> element.

Each item inside the lists is put inside an <li> (list item) element.

For example, if we wanted to turn the part of the following paragraph fragment into a list

HTMLCopy to Clipboard
```
<p>
  At Mozilla, we're a global community of technologists, thinkers, and builders
  working together…
</p>
```

We could modify the markup to this

HTMLCopy to Clipboard
```
<p>At Mozilla, we're a global community of</p>

<ul>
  <li>technologists</li>
  <li>thinkers</li>
  <li>builders</li>
</ul>

<p>working together…</p>
```

Try adding an ordered or unordered list to your example page.

Links

Links are very important — they are what makes the web a web! To add a link, we need to use a simple element — <a> — "a" being the short form for "anchor". To make text within your paragraph into a link, follow these steps:

1. Choose some text. We chose the text "Mozilla Manifesto".
2. Wrap the text in an <a> element, as shown below:

   HTMLCopy to Clipboard
   ```
   <a>Mozilla Manifesto</a>
   ```

3. Give the <a> element an href attribute, as shown below:

   HTMLCopy to Clipboard

```
<a href="">Mozilla Manifesto</a>
```

4. Fill in the value of this attribute with the web address that you want the link to:

   HTMLCopy to Clipboard
   ```
   <a href="https://www.mozilla.org/en-US/about/manifesto/">
     Mozilla Manifesto
   </a>
   ```

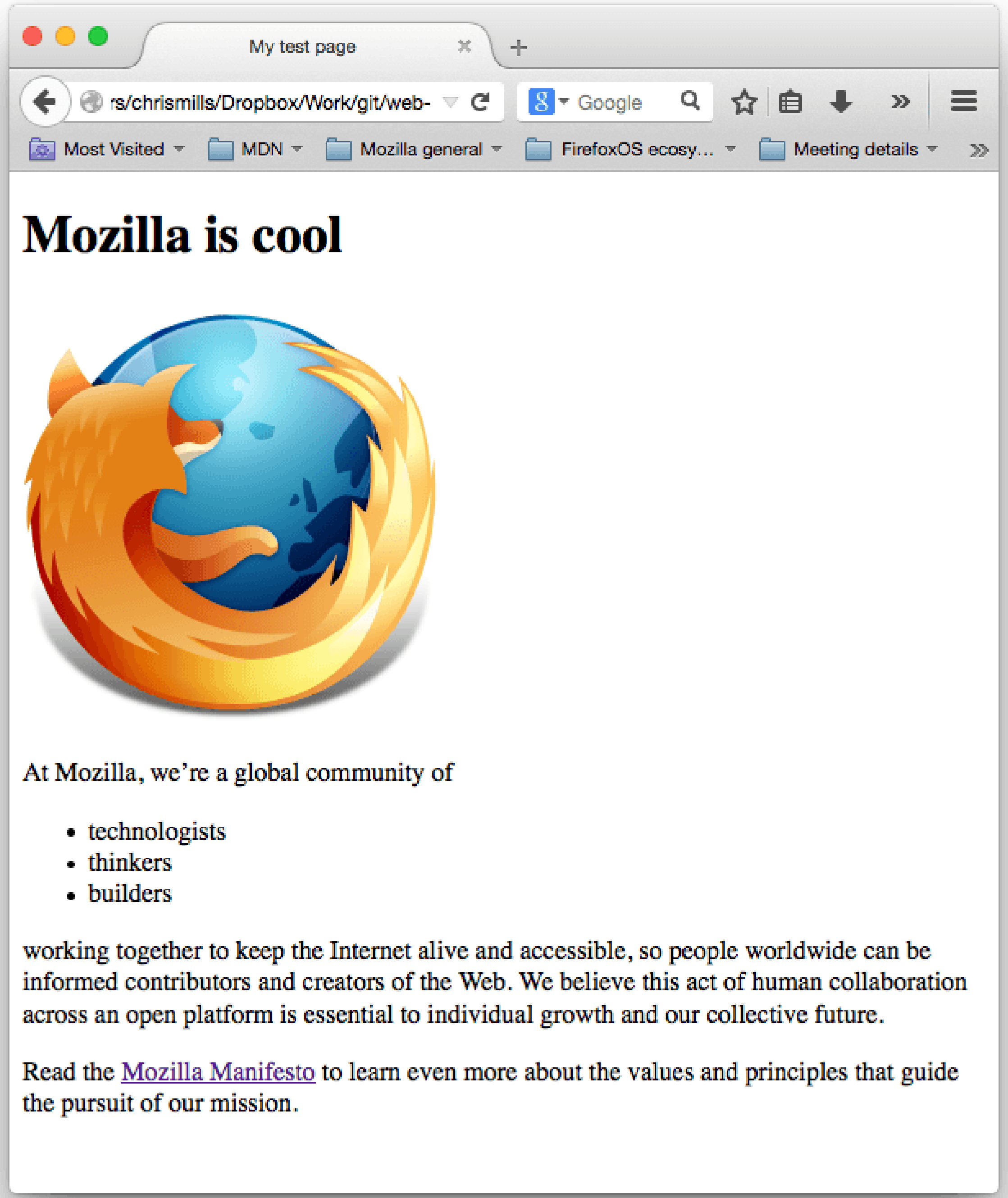
You might get unexpected results if you omit the https:// or http:// part, called the *protocol*, at the beginning of the web address. After making a link, click it to make sure it is sending you where you wanted it to.

Note: href might appear like a rather obscure choice for an attribute name at first. If you are having trouble remembering it, remember that it stands for **h***ypertext* **ref***erence*.

Add a link to your page now, if you haven't already done so.

Conclusion

If you have followed all the instructions in this article, you should end up with a page that looks like the one below (you can also view it here):



Mozilla is cool

At Mozilla, we're a global community of

- technologists
- thinkers
- builders

working together to keep the Internet alive and accessible, so people worldwide can be informed contributors and creators of the Web. We believe this act of human collaboration across an open platform is essential to individual growth and our collective future.

Read the Mozilla Manifesto to learn even more about the values and principles that guide the pursuit of our mission.

CSS

CSS is a very important part of the modern web. CSS stands for Cascading Style Sheets. Modern web pages use HTML5 in conjunction with CSS to make amazing, beautiful websites. Without CSS, websites would look plain, uninteresting and dated by today's standards.

Brief History of CSS

CSS was first presented as an idea at The Web Conference of 1994 in Chicago. It received some criticism as some argued that to style documents, a full programming language was needed, while CSS was just a simple, declarative format (w3.org). At the next conference in 1995, CSS was again presented, this time with implementations to showcase. Despite the want and need, it wasn't until a year later (in December of 1996) that CSS level 1 became ratified and accepted as a W3C Recommendation.

The current official version is CSS3. This version split CSS into modules, which have been allowed to level independently, allowing some modules to progress to level 4 (A Word About CSS4).

How CSS Works

CSS is interpreted by the browser and applied to the HTML that calls it. Within the stylesheet, developers assign styles to select sections of the HTML document. For example, if we took the first paragraph in a website, likely denoted by the HTML:

```
<p class="firstParagraph">This is the first paragraph.</p>
```

This is the first paragraph.

and if we wanted to apply style using CSS, we could change the syntax to:

```
<p class="firstParagraph" style="font-size: 14px; color: blue;">This is the first paragraph.</p>
```

This is the first paragraph.

Thus changing the font size to 14 pixels and the color to blue instead of black.

What about the cascading? This is how all elements within the select section are assigned to the same style, unless another CSS section overrides it. For example if we took the first section of the HTML tag <body>, which could look like:

```
<div class="first">

  <p class="firstParagraph">This is the first paragraph.</p>

  <p class="secondParagraph">This is the second paragraph.</p>
```

```
</div>
```

This is the first paragraph.

This is the second paragraph.

and added CSS code that refers to “first” as follows:

```
<style>
  .first {
    font-size: 14px;
    color: blue;
  }
</style>
<div class="first">
  <p class="firstParagraph">This is the first paragraph.</p>
  <p class="secondParagraph">This is the second paragraph.</p>
</div>
```

This is the first paragraph.

This is the second paragraph.

In this case, the CSS is applied to both the first and second paragraph, making them both have a font size of 14 pixels and a color of blue.

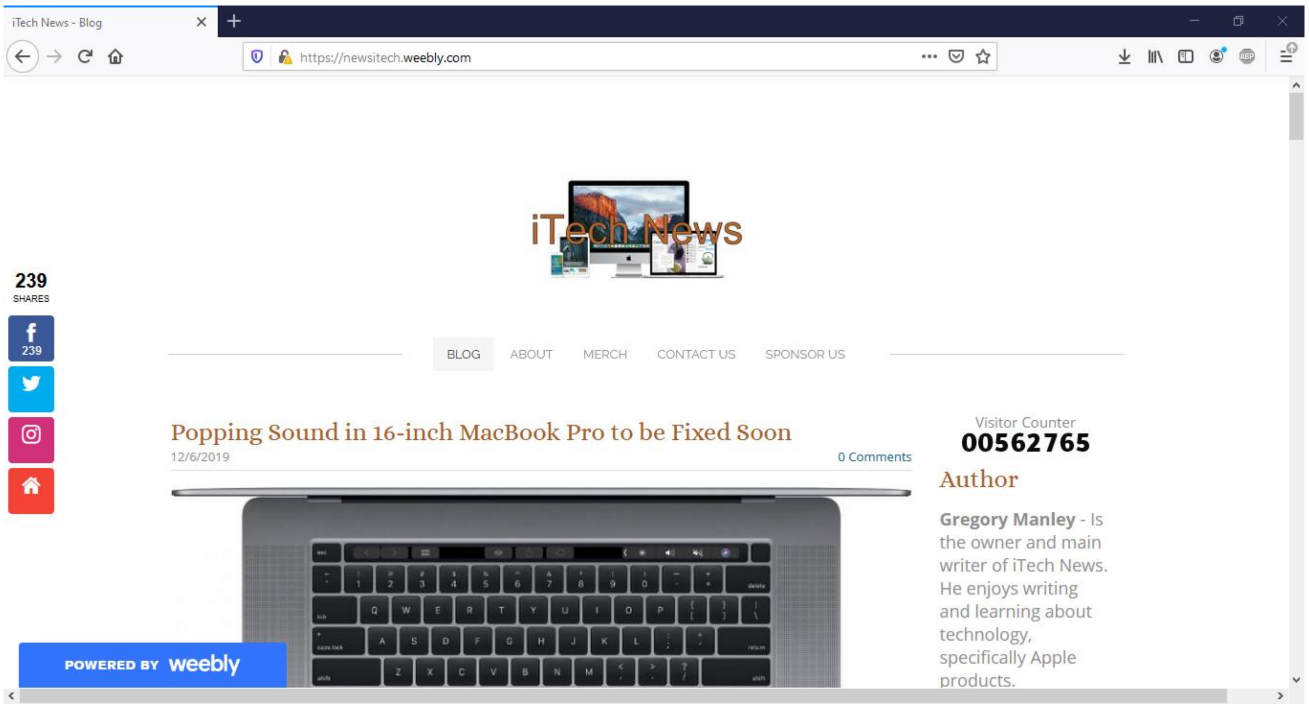
Why Is CSS Important?

According to Learn to Code With Me, “CSS is important because it allows web designers, developer, bloggers, and so forth to make our websites unique and attractive. CSS gives us the opportunity to play with a page layout, adjust colors and fonts, add effects to images, etc… Ultimately, it makes our lives easier. CSS allows us to separate the presentation from the structure (HTML) into different files.”

BigCommerce Essentials says that CSS “is the coding language that gives a website its look and layout. Along with HTML, CSS is fundamental to web design. Without it, website would still be plain text on white backgrounds.”

Let us use the iTech News website as an example. Here it is with

CSS:


iTech News - Blog

https://newsitech.weebly.com

239 SHARES

BLOG ABOUT MERCH CONTACT US SPONSOR US

Popping Sound in 16-inch MacBook Pro to be Fixed Soon

12/6/2019 0 Comments

Visitor Counter 00562765

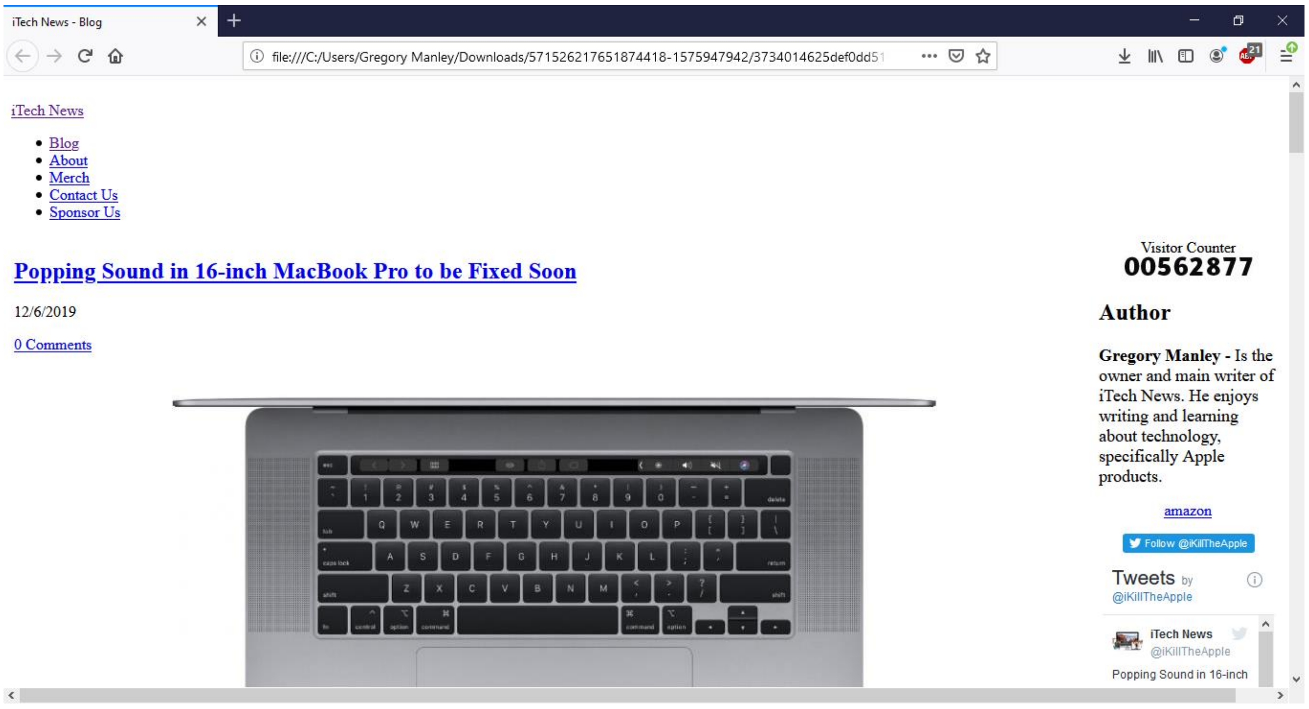
Author

Gregory Manley - Is the owner and main writer of iTech News. He enjoys writing and learning about technology, specifically Apple products.

POWERED BY weebly

And here it is without any

CSS:


iTech News - Blog

file:///C:/Users/Gregory Manley/Downloads/571526217651874418-1575947942/3734014625def0dd5

iTech News

- Blog
- About
- Merch
- Contact Us
- Sponsor Us

Popping Sound in 16-inch MacBook Pro to be Fixed Soon

12/6/2019

0 Comments

Visitor Counter 00562877

Author

Gregory Manley - Is the owner and main writer of iTech News. He enjoys writing and learning about technology, specifically Apple products.

amazon

Follow @iKillTheApple

Tweets by @iKillTheApple

iTech News @iKillTheApple

Popping Sound in 16-inch

As you can see just the use of CSS can take a webpage from bland and unimportant, to looking well-designed and interesting.

CSS is an important element of modern web design. It takes plain, white background websites and turns them into something amazing, beautiful, and engaging.

DATABASE

MySQL:

MySQL, as a relational database management system (RDBMS), can be utilized effectively in an video gallery website to store and manage various aspects of the application. Here are some key points regarding the use of MySQL for a video gallery website:

video Metadata and Information: MySQL can store essential information about video, such as video titles, descriptions, upload dates, and tags. This data can be structured using tables and columns in the database, allowing for efficient storage and retrieval.

Video Storage: While the actual video files are typically stored on the file system, MySQL can store references or paths to the image files in the database. By storing video URLs or file paths in the database, you can associate the metadata and other information with the corresponding videos.

Relationships and Associations: MySQL can handle relationships and associations between different entities within the video gallery website. For example, you can establish relationships between video s and albums, allowing video s to be grouped and organized into specific albums. These relationships can be defined through foreign key constraints and indexes in the database schema.

Search and Filtering: MySQL offers powerful querying capabilities, enabling efficient search and filtering functionalities for the video gallery. You can construct SQL queries to retrieve specific videos based on criteria like tags, categories, or keywords. This allows users to search for videos within the gallery using custom criteria.

Performance and Scalability: MySQL is known for its performance and scalability in handling large volumes of data. With proper indexing, database optimization, and caching strategies, MySQL can efficiently handle the storage and retrieval of videos and associated metadata in the Random Mcq generator.

Data Integrity and ACID Compliance: MySQL ensures data integrity and compliance with ACID (Atomicity, Consistency, Isolation, Durability) properties, providing reliability and consistency for the video gallery data. ACID compliance ensures that transactions involving video uploads, updates, or deletions are handled reliably and consistently, preventing data corruption or loss.

It's important to note that while MySQL is a popular choice for many web applications, there are other database options available as well, such as PostgreSQL or MongoDB. The choice of database technology depends on factors such as the specific requirements of your video gallery website, expected traffic and scalability needs, and your familiarity with the database technology.

Overall, MySQL provides a robust and reliable solution for storing and managing video metadata and related information in an video gallery website. It offers efficient querying capabilities, data integrity, and scalability, making it a suitable choice for handling the backend database operations of an video gallery.

BACK END BLOCK

DJANGO

Django is a powerful web framework that provides a robust foundation for building web applications, including photo galleries. With its efficient and scalable architecture, Django simplifies the development process and offers numerous features that are particularly beneficial for implementing a Random Mcq generator.

Model-View-Controller (MVC) Architecture: Django follows the MVC architectural pattern, which helps in organizing the codebase of a photo Video application. Models define the structure and relationships of data entities such as videos and albums, while views handle the logic for processing user requests and rendering templates. Templates, in turn, define the visual presentation of the Random Mcq generator.

Models and Database Integration: Django's object-relational mapping (ORM) system allows developers to define models as Python classes, which are then translated into database tables. In the context of a video gallery, models can represent videos, albums, categories, or any other relevant entities. By defining models, developers can easily manage and manipulate data associated with the video gallery, such as videos URLs, descriptions, upload dates, and user interactions.

Videos Uploads and Storage: Django provides built-in features for handling video uploads and storage. The File Field or Video Field fields in Django models allow for convenient management of uploaded video files. Additionally, Django provides settings for configuring the storage location, file naming, and media handling, ensuring secure and efficient storage of the Random Mcq generator.

Admin Interface: Django includes a powerful and customizable admin interface that automatically generates an administration panel for managing the video gallery's data. The admin interface allows authorized users to perform CRUD (Create, Read, Update, Delete) operations on the Random Mcq generator and associated data. It offers a user-friendly interface for uploading, categorizing, and organizing videos without the need to build a separate administration system.

Template System: Django's template system allows for the creation of dynamic and visually appealing user interfaces. Templates can be used to render the video gallery's pages, display videos, handle user interactions, and generate HTML output. By utilizing template tags and filters, developers can efficiently present images, implement pagination, and apply styling to enhance the user experience.

URL Routing and Views: Django's URL routing mechanism, managed through the urls.py file, enables mapping of URLs to specific views and actions. Views handle user requests and implement the logic for fetching and rendering videos, displaying albums, applying filters or sorting, and managing user interactions. By defining URL patterns and associated view functions, developers can create a navigable and user-friendly Random Mcq generator.

The key features of the video gallery website include:

Responsive and dynamic user interface: The frontend will be built using React.js, allowing for the creation of a highly responsive and dynamic user interface. Users will be able to seamlessly navigate through the video gallery, view videos, and interact with the website.

Video organization and categorization: videos will be organized into different albums or categories, making it easier for users to browse and search for specific videos. React.js components will be utilized to create an intuitive and efficient user experience for exploring the video collection.

Video uploading and management: Authenticated users will have the ability to upload video to the gallery. The Django backend will handle video storage and management, while the React.js frontend will provide an intuitive user interface for uploading videos, including features like video preview and progress indicators.

User interactions and social features: Users will be able to like, comment on, and share videos, fostering engagement and community participation. React.js will enable the implementation of interactive components for these user interactions, creating a dynamic and engaging user experience.

User authentication and authorization: Django's authentication system will be used to handle user registration, login, and access control. Users will have personalized accounts, allowing them to manage their uploaded videos and interact with the community.

Seamless integration of Django and React.js: Django's RESTful API capabilities will be leveraged to communicate between the backend and frontend, enabling seamless data exchange and real-time updates. React.js components will interact with the API endpoints to fetch and display videos, handle user interactions, and perform data operations.

By combining the power of Django and React.js, this Random Mcq generator website aims to deliver a modern and user-friendly platform for video sharing and exploration. The Django backend provides robust data management and authentication capabilities, while React.js enhances the user experience with its dynamic and responsive frontend components. The integration between the two frameworks ensures a seamless and efficient development process, resulting in a feature-rich and visually appealing Random Mcq generator.

REACT JS

React.js is a popular JavaScript library that is commonly used for building user interfaces in web applications, including video gallery websites. Here are some key reasons why React.js is beneficial for developing Random Mcq generator.

Component-Based Architecture: React.js is built around the concept of reusable components. In an video gallery website, various components can be created, such as video cards, galleries, pagination, filters, and modal windows. Each component can encapsulate its own logic and UI, making it easier to manage and maintain the codebase. React.js components can be composed together to build complex and interactive interfaces for browsing and displaying video.

Virtual DOM and Efficient Rendering: React.js utilizes a virtual DOM (Document Object Model) to efficiently update and render changes to the user interface. When the state or props of a component change, React.js calculates the minimal set of updates needed and applies those updates to the virtual DOM. This helps in reducing unnecessary re-rendering and optimizing the performance of the video gallery website, especially when there are frequent updates or interactions with videos.

Interactive User Experience: React.js enables the creation of highly interactive and responsive user interfaces. Users can seamlessly interact with the video gallery, such as clicking on video to view them in larger sizes, scrolling through galleries, applying filters, or dynamically loading more video. React.js allows for the implementation of such interactions with ease, providing a smooth and engaging user experience.

Single-Page Application (SPA) Capabilities: React.js is well-suited for building single-page applications, where content is dynamically loaded and updated without requiring full page reloads. In Random Mcq generator, this means that users can navigate between different galleries videos, or views without the need for the entire page to refresh. This enhances the speed and fluidity of the user experience.

Ecosystem and Community: React.js has a vast ecosystem of libraries and tools that can enhance the development process of an video gallery website. Libraries like React Router can be used for managing client-side routing, while state management libraries like Redux or React Context can help in managing video data and application state. The strong community support ensures that developers have access to a wealth of resources, tutorials, and discussions related to building video galleries with React.js.

Overall, React.js provides a robust and efficient framework for building video gallery websites. Its component-based architecture, efficient rendering, interactivity, single-page application capabilities, and supportive ecosystem make it a popular choice for developers aiming to create engaging and responsive user interfaces for video galleries.

TECHNOLOGY:

Introduction to python:

Python:

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural,) object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released in 2000, introduced features like list comprehensions and a garbage collection system capable of collecting reference cycles. Python 3.0, released in 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3.

The Python 2 language, i.e. Python 2.7.x, was officially discontinued on 1 January 2020 (first planned for 2015) after which security patches and other improvements will not be released for it. With Python 2's end-of-life, only Python 3.5.xand later are supported.

Python interpreters are available for many operating systems. A global community of programmers develops and maintains CPython, an open sourcereference implementation. A non-profit organization, the Python Software Foundation, manages and directs resources for Python and CPython development.

Python is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.

Python do?:

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

Why Python?:

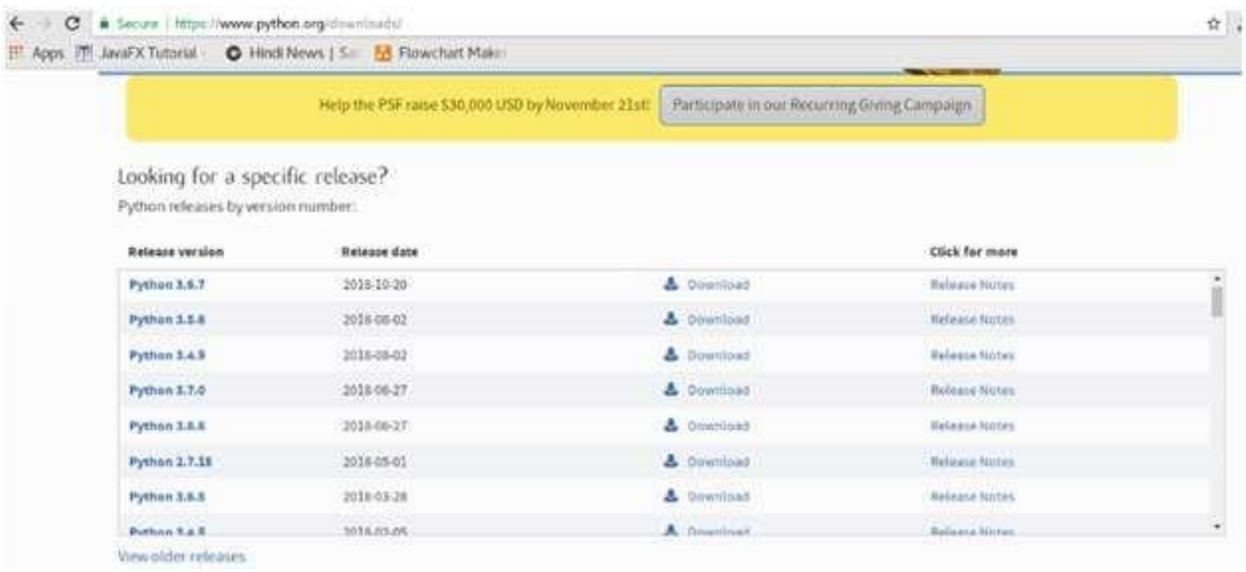
- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-orientated way or a functional way.

Python compared to other programming languages

- Python was designed for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

Python 1.0	January 1994
Python 1.5	December 31, 1997
Python 1.6	September 5, 2000
Python 2.0	October 16, 2000
Python 2.1	April 17, 2001
Python 2.2	December 21, 2001
Python 2.3	July 29, 2003
Python 2.4	November 30, 2004
Python 2.5	September 19, 2006
Python 2.6	October 1, 2008
Python 2.7	July 3, 2010
Python 3.0	December 3, 2008
Python 3.1	June 27, 2009
Python 3.2	February 20, 2011
Python 3.3	September 29, 2012
Python 3.4	March 16, 2014
Python 3.5	September 13, 2015
Python 3.6	December 23, 2016
Python 3.7	June 27, 2018

Python installation procedure:



Help the PSF raise $30,000 USD by November 21st! Participate in our Recurring Giving Campaign

Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.6.7	2018-10-20	Download	Release Notes
Python 3.5.6	2018-08-02	Download	Release Notes
Python 3.4.9	2018-08-02	Download	Release Notes
Python 3.7.0	2018-06-27	Download	Release Notes
Python 3.6.6	2018-06-27	Download	Release Notes
Python 2.7.15	2018-05-01	Download	Release Notes
Python 3.6.5	2018-03-28	Download	Release Notes
Python 3.4.8	2018-02-05	Download	Release Notes

View older releases

Windows Based

It is highly unlikely that your Windows system shipped with Python already installed. Windows systems typically do not. Fortunately, installing does not involve much more than downloading the Python installer from the python.org website and running it. Let's take a look at how to install Python 3 on Windows:

Step 1: Download the Python 3 Installer

1. Open a browser window and navigate to the Download page for Windows at python.org.

2. Underneath the heading at the top that says **Python Releases for Windows**, click on the link for the **Latest Python 3 Release - Python 3.x.x**. (As of this writing, the latest is Python 3.6.5.)

3. Scroll to the bottom and select either **Windows x86-64 executable installer** for 64-bit or **Windows x86 executable installer** for 32-bit. (See below.)

Sidebar: 32-bit or 64-bit Python?

For Windows, you can choose either the 32-bit or 64-bit installer. Here's what the difference between the two comes down to:

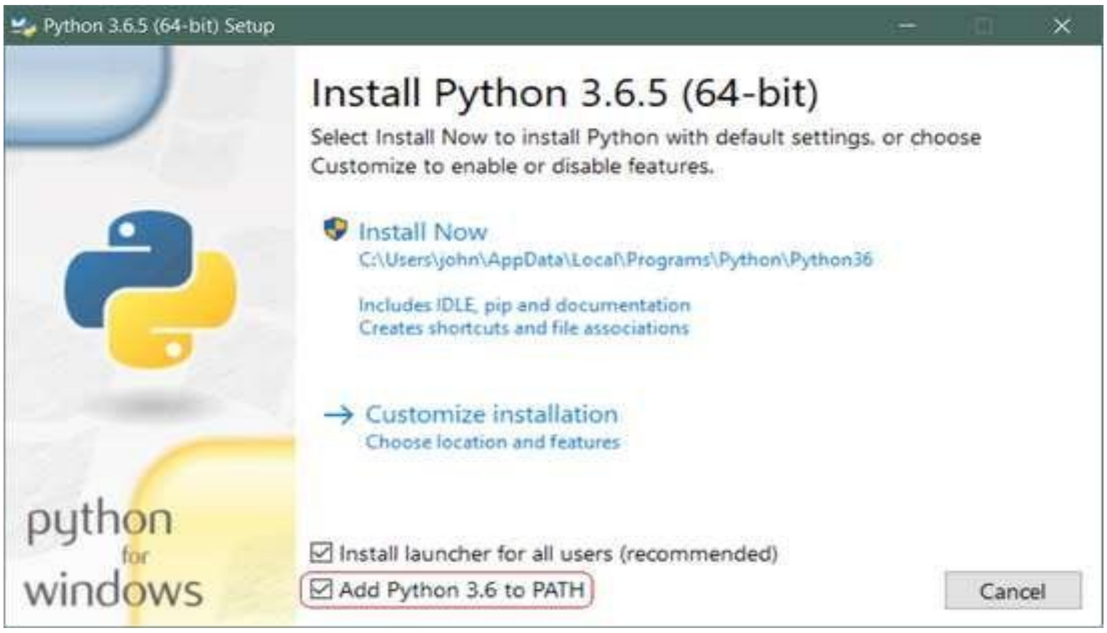
- If your system has a 32-bit processor, then you should choose the 32-bit installer.

- On a 64-bit system, either installer will actually work for most purposes. The 32-bit version will generally use less memory, but the 64-bit version performs better for applications with intensive computation.
- If you're unsure which version to pick, go with the 64-bit version.

Note: Remember that if you get this choice "wrong" and would like to switch to another version of Python, you can just uninstall Python and then re-install it by downloading another installer from python.org.

Step 2: Run the Installer

Once you have chosen and downloaded an installer, simply run it by double-clicking on the downloaded file. A dialog should appear that looks something like this:



Python 3.6.5 (64-bit) Setup

Install Python 3.6.5 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\john\AppData\Local\Programs\Python\Python36

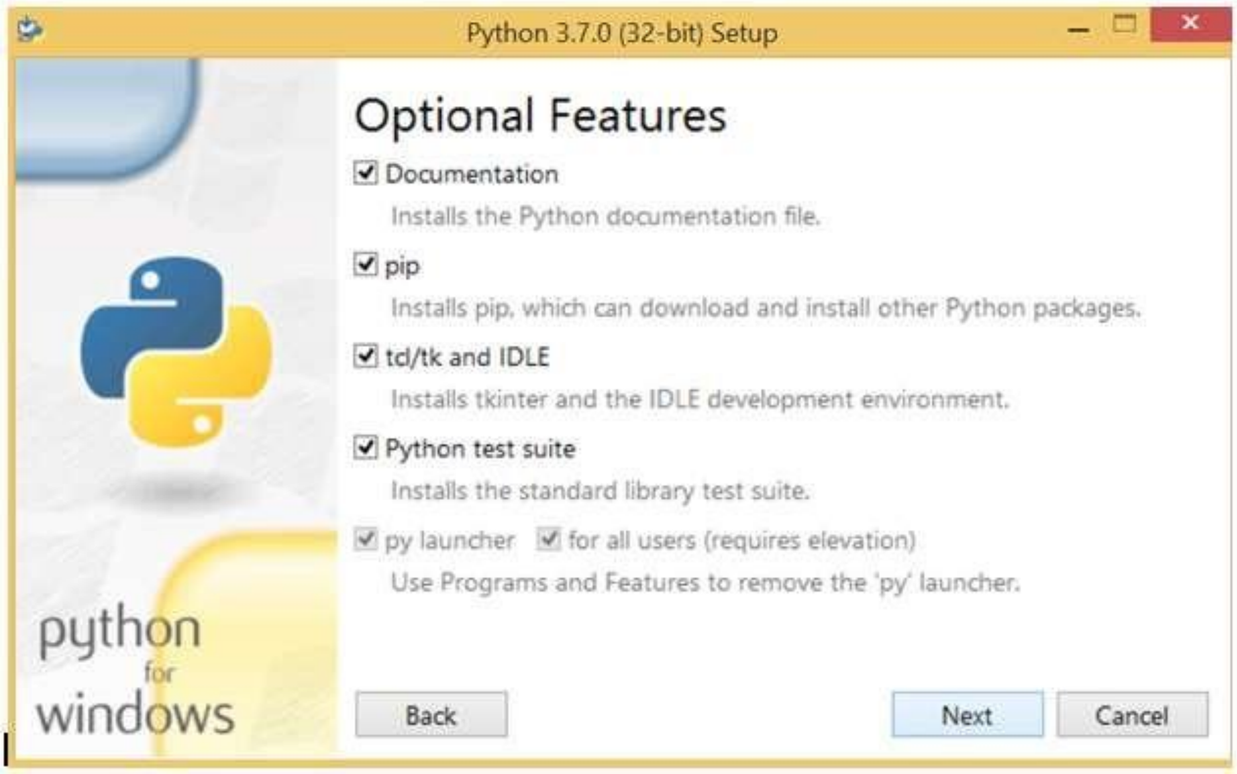
Includes IDLE, pip and documentation
Creates shortcuts and file associations

Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☑ Add Python 3.6 to PATH

Important: You want to be sure to check the box that says **Add Python 3.x to PATH** as shown to ensure that the interpreter will be placed in your execution path.

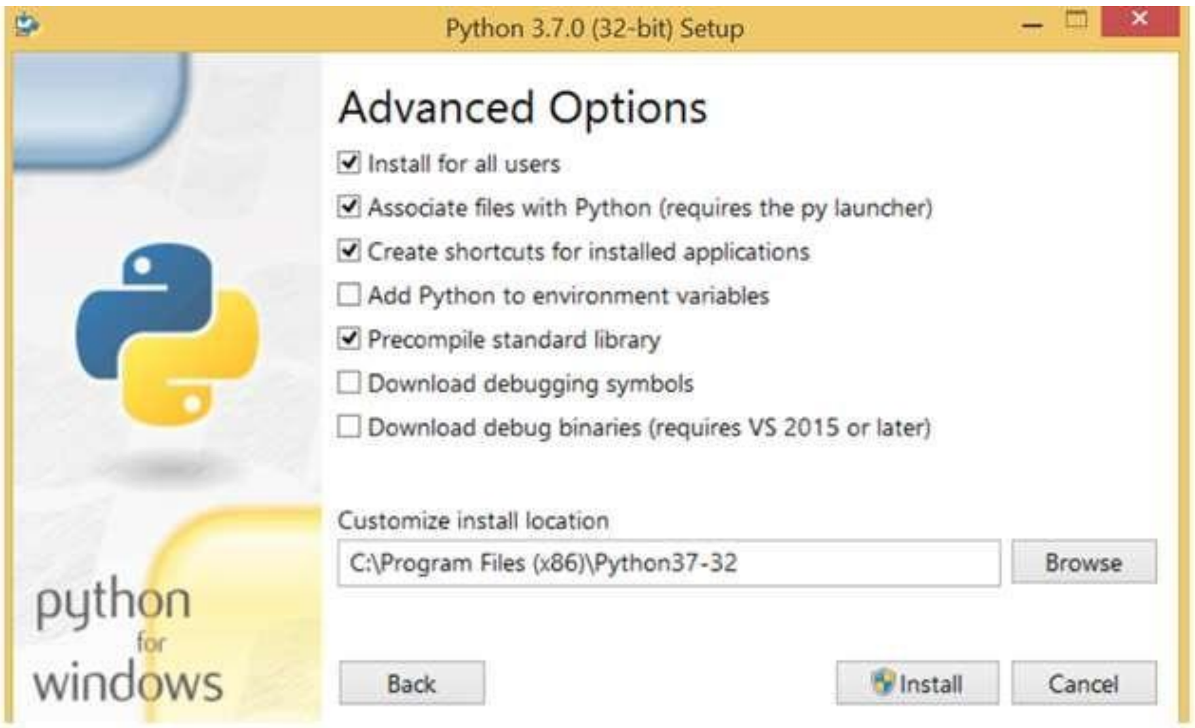
Then just click **Install Now**. That should be all there is to it. A few minutes later you should have a working Python 3 installation on your system.



Python 3.7.0 (32-bit) Setup

Optional Features

- [✓] Documentation
  Installs the Python documentation file.
- [✓] pip
  Installs pip, which can download and install other Python packages.
- [✓] tcl/tk and IDLE
  Installs tkinter and the IDLE development environment.
- [✓] Python test suite
  Installs the standard library test suite.
- [✓] py launcher [✓] for all users (requires elevation)
  Use Programs and Features to remove the 'py' launcher.



Python 3.7.0 (32-bit) Setup

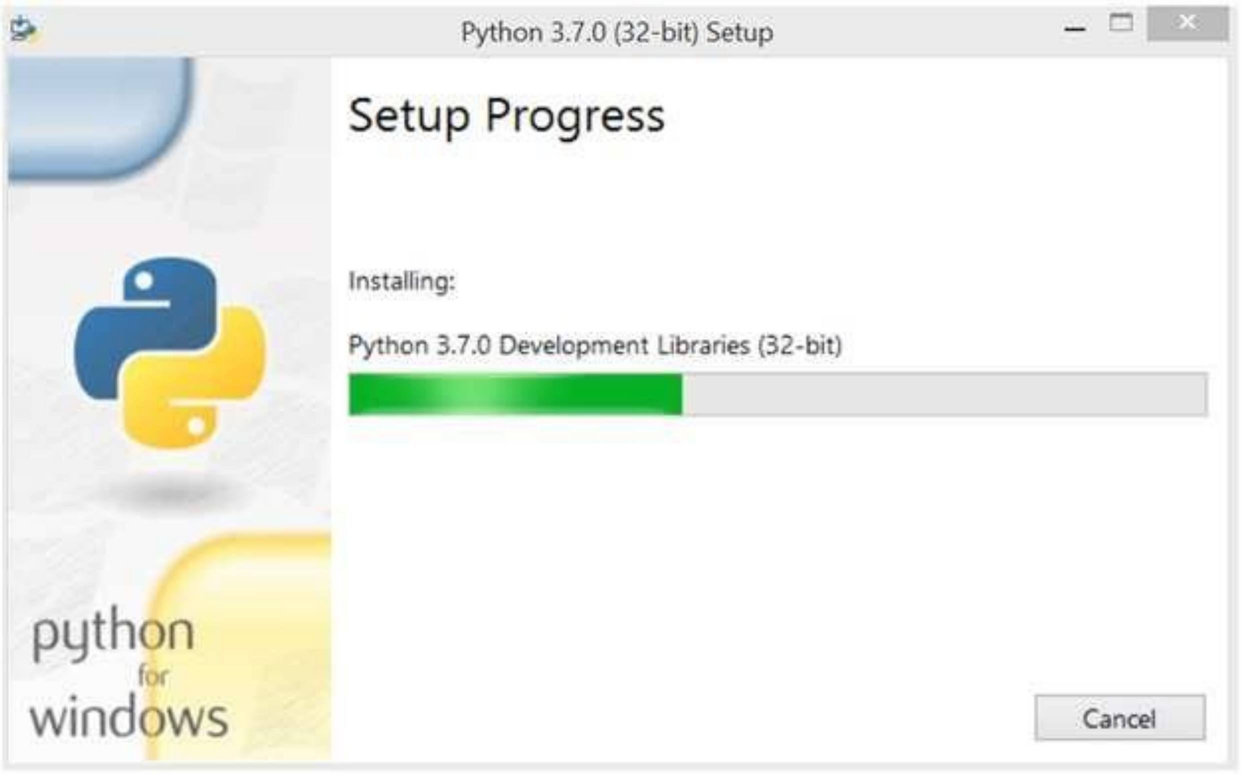
Advanced Options

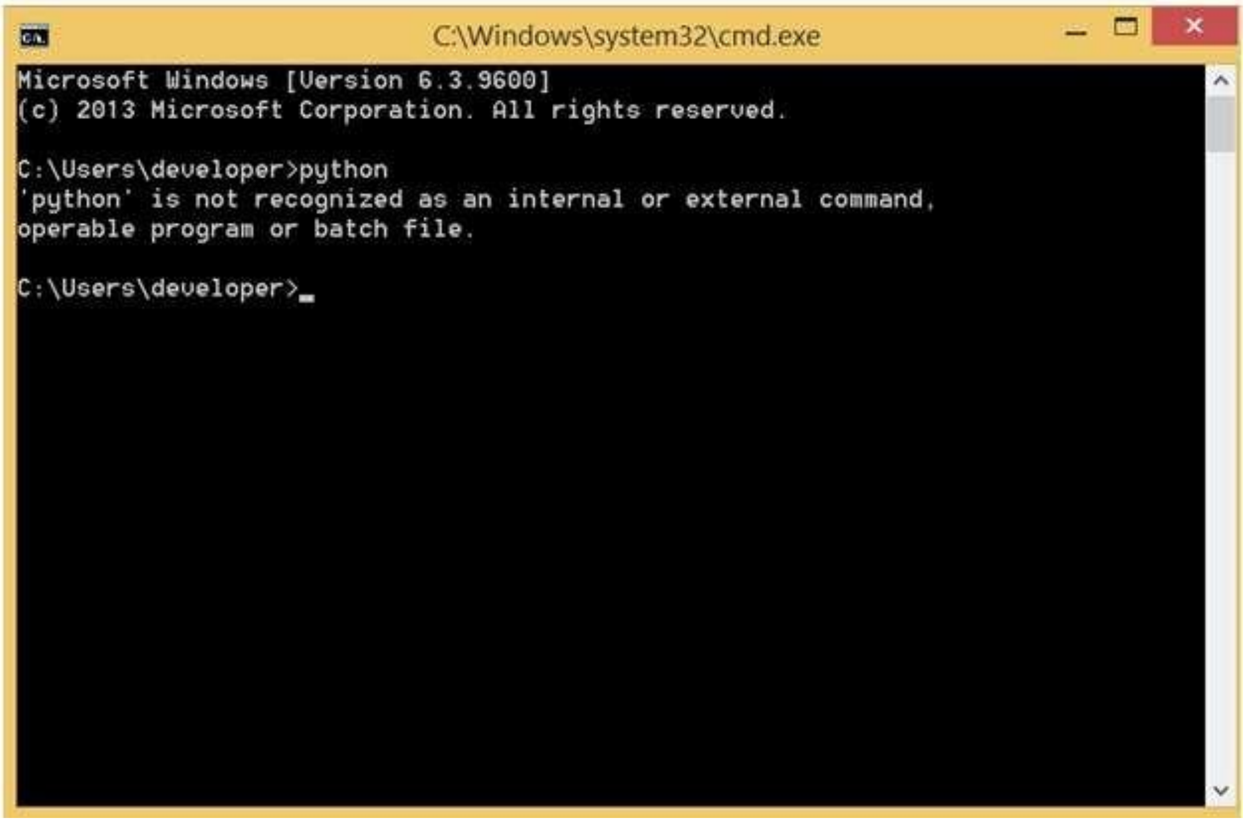
- [✓] Install for all users
- [✓] Associate files with Python (requires the py launcher)
- [✓] Create shortcuts for installed applications
- [ ] Add Python to environment variables
- [✓] Precompile standard library
- [ ] Download debugging symbols
- [ ] Download debug binaries (requires VS 2015 or later)

Customize install location

C:\Program Files (x86)\Python37-32

Now, we are ready to install python-3.6.7. Let's install it.

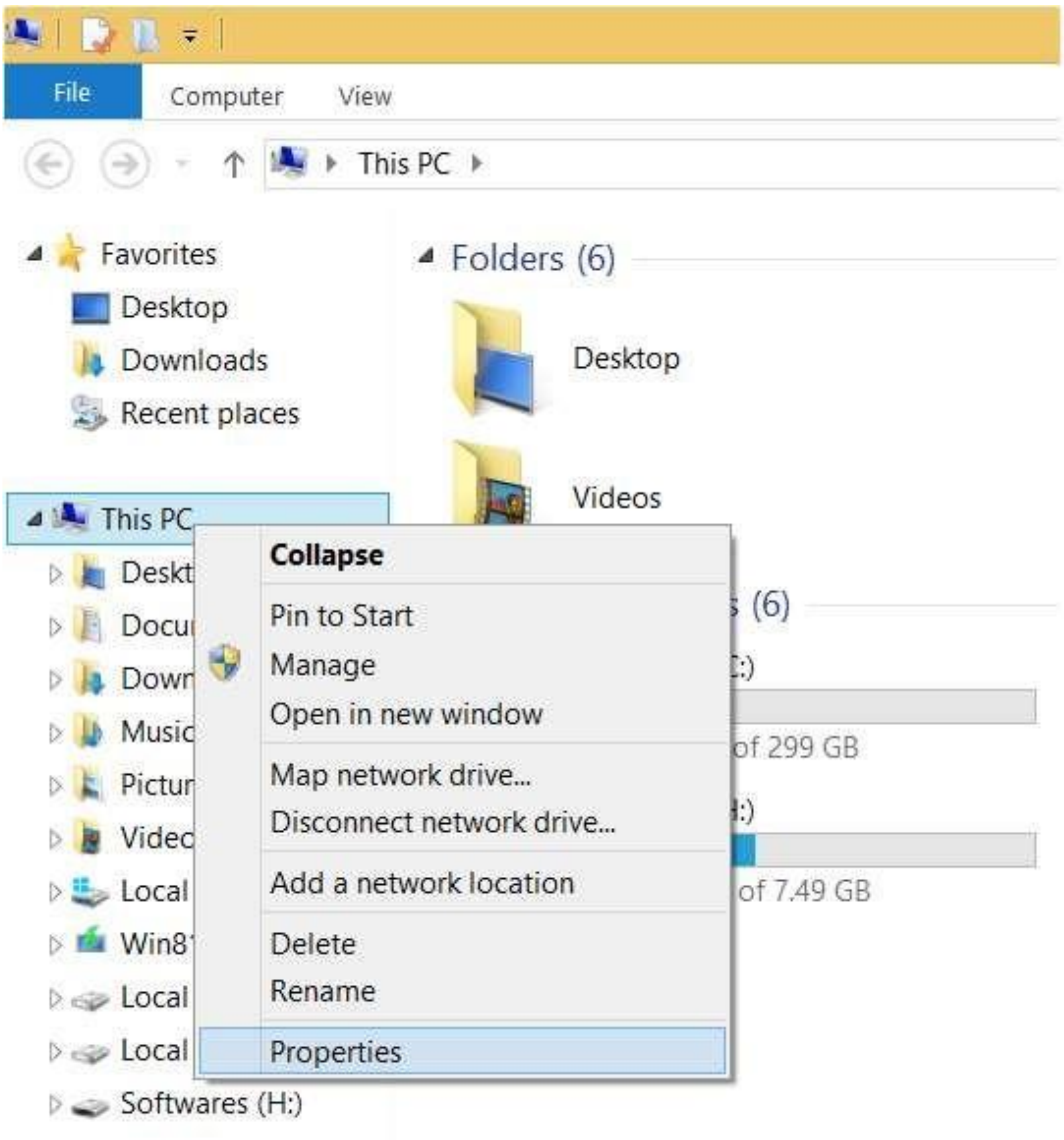

Python 3.7.0 (32-bit) Setup
Setup Progress
Installing:
Python 3.7.0 Development Libraries (32-bit)
python for windows
Cancel



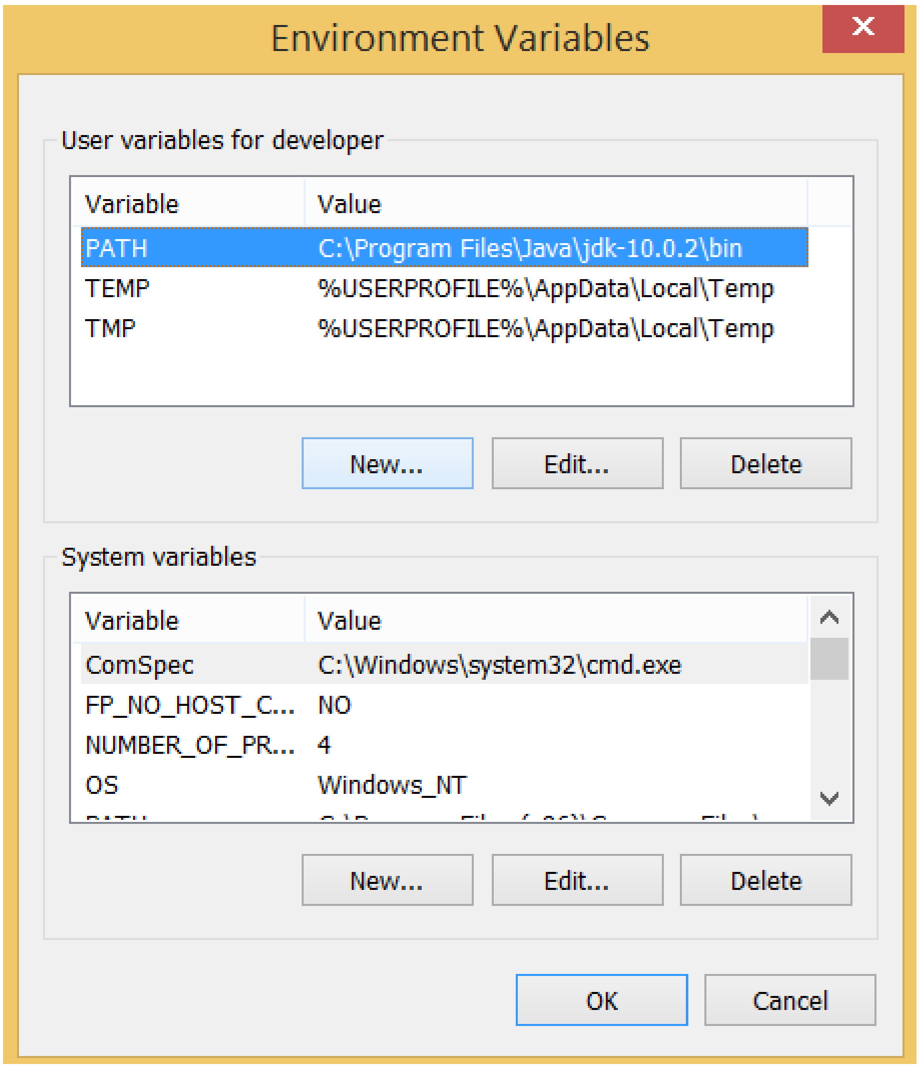
```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\developer>python
'python' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\developer>_
```



To set the path of python, we need to the right click on "my computer" and go to Properties → Advanced → Environment Variables.


Environment Variables

User variables for developer

Variable	Value
PATH	C:\Program Files\Java\jdk-10.0.2\bin
TEMP	%USERPROFILE%\AppData\Local\Temp
TMP	%USERPROFILE%\AppData\Local\Temp

System variables

Variable	Value
ComSpec	C:\Windows\system32\cmd.exe
FP_NO_HOST_C...	NO
NUMBER_OF_PR...	4
OS	Windows_NT

Type **PATH** as the variable name and set the path to the installation directory of the python shown in the below image.

Now, the path is set, we are ready to run python on our local system. Restart CMD, and type **python** again. It will open the python interpreter shell where we can execute the python statements.

Mac OS based

While current versions of macOS (previously known as “Mac OS X”) include a version of Python 2, it is likely out of date by a few months. Also, this tutorial series uses Python 3, so let’s get you upgraded to that.

The best way we found to install Python 3 on macOS is through the Homebrew package manager. This approach is also recommended by community guides like The Hitchhiker’s Guide to Python.

Step 1: Install Homebrew (Part 1)

To get started, you first want to install Homebrew:

1. Open a browser and navigate to http://brew.sh/. After the page has finished loading, **select the Homebrew bootstrap code under "Install Homebrew"**. Then hit cmd+c to copy it to the clipboard. Make sure you've captured the text of the complete command because otherwise the installation will fail.

2. Now you need to **open a Terminal app window, paste the Homebrew bootstrap code, and then hit** Enter. This will begin the Homebrew installation.

3. If you're doing this on a fresh install of macOS, you may get a pop up alert **asking you to install Apple's "command line developer tools"**. You'll need those to continue with the installation, so please **confirm the dialog box by clicking on "Install"**.

At this point, you're likely waiting for the command line developer tools to finish installing, and that's going to take a few minutes. Time to grab a coffee or tea!

Step 2: Install Homebrew (Part 2)

You can continue installing Homebrew and then Python after the command line developer tools installation is complete:

1. Confirm the "The software was installed" dialog from the developer tools installer.

2. Back in the terminal, hit Enter to continue with the Homebrew installation.

3. Homebrew asks you to enter your password so it can finalize the installation. **Enter your user account password and hit** Enter to continue.

4. Depending on your internet connection, Homebrew will take a few minutes to download its required files. Once the installation is complete, you'll end up back at the command prompt in your terminal window.

Whew! Now that the Homebrew package manager is set up, let's continue on with installing Python 3 on your system.

Step 3: Install Python

Once Homebrew has finished installing, **return to your terminal and run the following command**:

```
$ brew install python3
```

Note: When you copy this command, be sure you don't include the $ character at the beginning. That's just an indicator that this is a console command.

This will download and install the latest version of Python. After the Homebrew brew install command finishes, Python 3 should be installed on your system.

You can make sure everything went correctly by testing if Python can be accessed from the terminal:

1. Open the terminal by launching **Terminal app**.
2. Type pip3 and hit Enter.
3. You should see the help text from Python's "Pip" package manager. If you get an error message running pip3, go through the Python install steps again to make sure you have a working Python installation.

Assuming everything went well and you saw the output from Pip in your command prompt window…congratulations! You just installed Python on your system, and you're all set to continue with the next section in this tutorial.

Packages need for python based programming:

- **Numpy**
  NumPy is a Python package which stands for 'Numerical Python'. It is the core library for scientific computing, which contains a powerful n-dimensional array object, provide tools for integrating C, C++ etc. It is also useful in linear algebra, random number capability etc.
- **Pandas**
  Pandas is a high-level data manipulation tool developed by Wes McKinney. It is built on the Numpy package and its key data structure is called the DataFrame. DataFrames allow you to store and manipulate tabular data in rows of observations and columns of variables.
- **Keras**
  Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. Use Keras if you need a deep learning library that: Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
- **Sklearn**

Scikit-learn is a free machine learning library for Python. It features various algorithms like support vector machine, random forests, and k-neighbours, and it also supports Python numerical and scientific libraries like NumPy and SciPy.

- **Scipy**
  SciPy is an open-source Python library which is used to solve scientific and mathematical problems. It is built on the NumPy extension and allows the user to manipulate and visualize data with a wide range of high-level commands.
- **Tensorflow**
  TensorFlow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow.
- **Django**
  Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.
- **Pyodbc**
  pyodbc is an open source Python module that makes accessing ODBC databases simple. It implements the DB API 2.0 specification but is packed with even more Pythonic convenience. Precompiled binary wheels are provided for most Python versions on Windows and macOS. On other operating systems this will build from source.
- **Matplotlib**
  Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002.
- **Opencv**
  OpenCV-Python is a library of Python bindings designed to solve computer vision problems. Python is a general purpose programming language started by Guido van Rossum that became very popular very quickly, mainly because of its simplicity and code readability.
- **Nltk**
  Natural Language Processing with Python NLTK is one of the leading platforms for working with human language data and Python, the module NLTK is used for natural language processing. NLTK is literally an acronym for Natural Language Toolkit. In this article you will learn how to tokenize data (by words and sentences).
- **SQLAIchemy**

SQLAlchemy is a library that facilitates the communication between Python programs and databases. Most of the times, this library is used as an Object Relational Mapper (ORM) tool that translates Python classes to tables on relational databases and automatically converts function calls to SQL statements.

- **Urllib**
  urllib is a Python module that can be used for opening URLs. It defines functions and classes to help in URL actions. With Python you can also access and retrieve data from the internet like XML, HTML, JSON, etc. You can also use Python to work with this data directly.

Installation of packages:

Syntax for installation of packages via cmd terminal using the basic

Step:1- First check pip cmd

First check pip cmd

If ok then

Step:2- pip list

Check the list of packages installed and then install required by following cmds

Step:3- pip install package name

The package name should as requirement.

DJANGO:

Django is a web application framework written in Python programming language. It is based on MVT (Model View Template) design pattern. The Django is very demanding due to its rapid development feature. It takes less time to build application after collecting client requirement.

Django was design and developed by Lawrence journal world in 2003 and publicly released under BSD license in July 2005. Currently, DSF (Django Software Foundation) maintains its development and release cycle.

Django was released on 21, July 2005. Its current stable version is 2.0.3 which was released on 6 March, 2018.

Popularity

Django is widely accepted and used by various well-known sites such as:

- Instagram
- Mozilla
- Disqus
- Pinterest
- Bitbucket
- The Washington Times

Features of Django

- Rapid Development
- Secure
- Scalable
- Fully loaded
- Versatile
- Open Source
- Vast and Supported Community

Rapid Development

Django was designed with the intention to make a framework which takes less time to build web application. The project implementation phase is a very time taken but Django creates it rapidly.

Secure

Django takes security seriously and helps developers to avoid many common security mistakes, such as SQL injection, cross-site scripting, cross-site request forgery etc. Its user authentication system provides a secure way to manage user accounts and passwords.

Scalable

Django is scalable in nature and has ability to quickly and flexibly switch from small to large scale application project.

Fully loaded

Django includes various helping task modules and libraries which can be used to handle common Web development tasks. Django takes care of user authentication, content administration, site maps, RSS feeds etc.

Versatile

Django is versatile in nature which allows it to build applications for different-different domains. Now a days, Companies are using Django to build various types of applications like: content management systems, social networks sites or scientific computing platforms etc.

Open Source

Django is an open source web application framework. It is publicly available without cost. It can be downloaded with source code from the public repository. Open source reduces the total cost of the application development.

Vast and Supported Community

Django is an one of the most popular web framework. It has widely supportive community and channels to share and connect.

A Django project contains the following packages and files. The outer directory is just a container for the application. We can rename it further.

- **manage.py:** It is a command-line utility which allows us to interact with the project in various ways and also used to manage an application that we will see later on in this tutorial.
- A directory (djangpapp) located inside, is the actual application package name. Its name is the Python package name which we'll need to use to import module inside the application.
- **__init_.py:** It is an empty file that tells to the Python that this directory should be considered as a Python package.
- **settings.py:** This file is used to configure application settings such as database connection, static files linking etc.
- **urls.py:** This file contains the listed URLs of the application. In this file, we can mention the URLs and corresponding actions to perform the task and display the view.
- **wsgi.py:** It is an entry-point for WSGI-compatible web servers to serve Django project.

Running the Django Project

Django project has a built-in development server which is used to run application instantly without any external web server. It means we don't need of Apache or another web server to run the application in development mode.

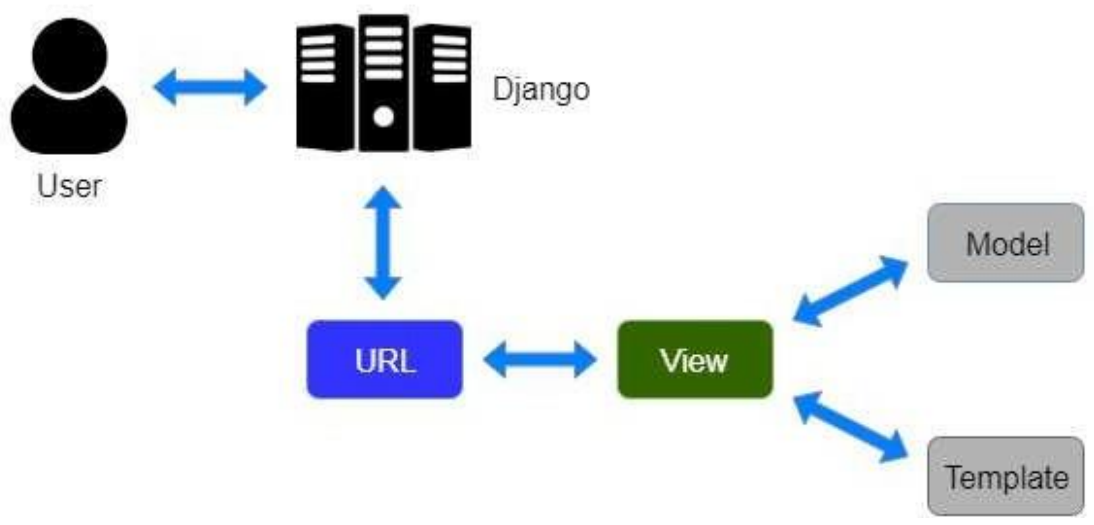
Django MVT

The MVT (Model View Template) is a software design pattern. It is a collection of three important components Model View and Template. The Model helps to handle database. It is a data access layer which handles the data.

The Template is a presentation layer which handles User Interface part completely. The View is used to execute the business logic and interact with a model to carry data and renders a template.

Although Django follows MVC pattern but maintains it?s own conventions. So, control is handled by the framework itself.

There is no separate controller and complete application is based on Model View and Template. That?s why it is called MVT application



Django Model

In Django, a model is a class which is used to contain essential fields and methods. Each model class maps to a single table in the database.

Django Model is a subclass of **django.db.models.Model** and each field of the model class represents a database field (column).

Django provides us a database-abstraction API which allows us to create, retrieve, update and delete a record from the mapped table.

Django Views

A view is a place where we put our business logic of the application. The view is a python function which is used to perform some business logic and return a response to the user. This response can be the HTML contents of a Web page, or a redirect, or a 404 error.

All the view function are created inside the **views.py** file of the Django app.

Django Templates

Django provides a convenient way to generate dynamic HTML pages by using its template system.

A template consists of static parts of the desired HTML output as well as some special syntax describing how dynamic content will be inserted.

Why Django Template?

In HTML file, we can't write python code because the code is only interpreted by python interpreter not the browser. We know that HTML is a static markup language, while Python is a dynamic programming language.

Django template engine is used to separate the design from the python code and allows us to build dynamic web pages.

Django URL Mapping

Well, till here, we have learned to create a model, view, and template. Now, we will learn about the routing of application.

Since Django is a web application framework, it gets user requests by URL locater and responds back. To handle URL, **django.urls** module is used by the framework.

Django Static Files Handling

In a web application, apart from business logic and data handling, we also need to handle and manage static resources like CSS, JavaScript, images etc.

It is important to manage these resources so that it does not affect our application performance.

Django deals with it very efficiently and provides a convenient manner to use resources.

Django Model Form

It is a class which is used to create an HTML form by using the Model. It is an efficient way to create a form without writing HTML code.

Django automatically does it for us to reduce the application development time. For example, suppose we have a model containing various fields, we don't need to repeat the fields in the form file.

For this reason, Django provides a helper class which allows us to create a Form class from a Django model.

Django Database Connectivity

The **settings.py** file contains all the project settings along with database connection details. By default, Django works with **SQLite,** database and allows configuring for other databases as well.

Database connectivity requires all the connection details such as database name, user credentials, hostname drive name etc.

To connect with MySQL, **django.db.backends.mysql** driver is used to establishing a connection between application and database. Let's see an example.

We need to provide all connection details in the settings file. The settings.py file of our project contains the following code for the database.

TIME LINE

TIME LINE

PROPOSED SOLUTION:

The proposed solution is to develop a random question generator web application using Django, a Python web framework. This solution leverages Django's powerful features and modular architecture to create an efficient and customizable platform for generating and presenting random questions to users.

Database Design and Management:

Design a database schema to store a collection of questions. Each question should have attributes such as category, difficulty level, content, and possible answer choices.

Utilize Django's Object-Relational Mapping (ORM) to define models representing the question and its attributes. Configure the database connection and migrations to create the necessary tables.

Question Generation Algorithm:

Develop an algorithm that selects random questions from the database based on user preferences and criteria such as category, difficulty level, or topic.

Use Django's query capabilities to retrieve random questions, applying filters based on user-selected preferences.

Consider implementing a weighting mechanism to ensure a balanced distribution of questions across categories and difficulty levels.

User Interface and Experience:

Create a user-friendly interface using Django's template system and HTML/CSS to present the randomly generated questions.

Design intuitive forms or question formats that allow users to provide answers or select choices.

Include features like instant feedback on user responses, timers for timed quizzes, and progress tracking to enhance the user experience.

User Authentication and Customization:

Implement Django's user authentication system to allow users to create accounts, log in, and save their preferences.

Provide options for users to customize the question generation process, such as selecting specific categories, difficulty levels, or topics of interest.

Security and Privacy:

Follow Django's security best practices, including protecting against common vulnerabilities such as Cross-Site Scripting (XSS) and Cross-Site Request Forgery (CSRF).

Secure user data by implementing appropriate measures, including password hashing and encryption, and adhering to privacy regulations.

Testing and Deployment:cq

Develop unit tests to ensure the functionality and reliability of the random question generator application.

Deploy the application on a production server, considering scalability, performance, and hosting requirements.

By implementing this proposed solution using Django, the random question generator will provide users with a dynamic and engaging learning experience. Users can access a wide range of random questions based on their preferences, enhancing their knowledge and fostering critical thinking skills. The customizable nature of the application allows users to tailor their learning experience, making it suitable for various domains and skill levels.

IMPLEMENTATION

SOFTWARE REQUIREMENT:

PROGRAMMING LANGUAGE: Python

BACKEND: Django

FORNTEND: React JS

DATABASE: MySQL

HARDWARE REQUIREMENT:

Processor: Pentium IV or higher

RAM: 256 MB

Space on Hard Disk: minimum 512MB

CODE LINK

https://github.com/irfanvali/Dudekula-irfan-vali/blob/main/MCQ.zip

APPLICATION URL LINK

https://github.com/irfanvali/Dudekula-irfan-vali/blob/main/MCQ.zip