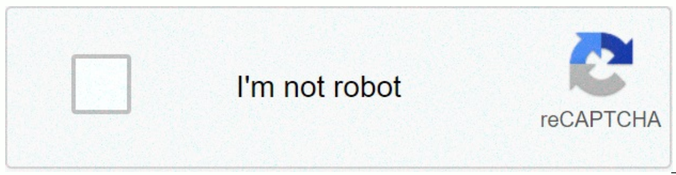

Continue

Python tutorial for beginners pdf with examples

An exclusive assortment of 20+ deep MongoDB tutorial for beginners to learn MongoDB from scratch:MongoDB is an open source platform and document-oriented NoSQL database program for modern apps. MongoDB enables real-time data to be organized and used anywhere. In this tutorial series you will learn more about MongoDB and its features such as installing MongoDB, creating database, aggregation, projection, sharding in MongoDB, etc., at the end of this detailed MongoDB training series. [Scroll down to check the list of all tutorials] List of all MongoDB tutorialsLearn MongoDB in detail with the help of this comprehensive MongoDB training series for beginners and experienced professionals. Overview of tutorials in MongoDB SeriesStart with the first tutorial in the series => FIRST tutorial by Jackie Lohrey Quicken financial management software, with all its possibilities, instructions and options can be overwhelming, especially for a new user. Everything seems equally important, and everything seems like you have to master it now. Before you dive into the process of using accounts and start creating reports and filling in budget amounts, it's important to learn a few basic concepts that can help you optimize Quicken for your individual needs. One of the most important concepts is to use categories and tags to track your account transactions. A Quicken category is a term, such as .B food or household, that defines a broad set of income and expense transactions. You use a single category per transaction to track how you make and spend money. For example, all grocery purchases, regardless of where you buy them, go into the food category, and all income, regardless of source, goes into the Income category. Use a subcategory to provide a larger organization and detailed information within a category, and to help create a budget. This is especially useful in a category like Auto where you cause multiple types of expenses. Creating subcategories for fuel, maintenance, and registration makes it much easier to see where your money is going. Tags are like keywords. They allow you to track transactions within and across category groups. For example, if you want to track the money you spend on ancillary expenses for a child in college, create a tag with the child's name, and then apply it to the appropriate category or subcategory, such as.B food, car: fuel, or phone: mobile phone. Quicken comes with numerous integrated categories that cover many standard items. Not all built-in categories are visible in the Category List window because Most of what you see depends on information you give Quicken during setup. Before you create a new category, access Quicken's built-in categories and see if something is already there. To access built-in categories, select Tools from the main menu, and then select Category List. Click the Add Categories button, select the categories you want to add from the menu on the left side of the and click the Add button. Click OK to exit the category list and return to the list of categories. To add a new category or subcategory, click the New button in the Category List window, create a name for the category or subcategory, provide an optional description, and specify the category type, such as .B income, expenses, or subcategory. If you select Subcategory, use the drop-down box next to the selection to specify the name of the main category. To create a new tag, select Tools from the main menu, and then select Tag List. At the top of the window, click the New button and enter a name and optional description for the tag. Ignore the Copy Number field because it only applies to tracking rental properties. Click OK to get the deal. While it is possible to create categories, subcategories and tags on the fly, this is not always a good option, especially if you are frustrated or just trying to make it fit. If you divide thoughts into categories, subcategories, and tags, you can make Quicken work better and more efficiently. Use categories, subcategories, and tags in trade repositories to identify and organize revenue and expenses. For example, in a checking account register, after entering the recipient and the check, click the drop-down box under the recipient's name to select either the category or subcategory of the transaction. When you create the tables, or click in the next field to select the appropriate tag from the drop-down list. At the end of the month, run a report that shows a breakdown by revenue/expense by category, subcategory, and day. For example, to run a standard Quicken expense report, click Reports on the main menu. Select Expenses, and then Outputby Category to view the report and its pie chart. By default, the expense report is set to an annual list, so adjust the date parameters to suit your needs. Yes, variables, you need them everywhere when programming! But don't worry, the concept isn't too hard if you understand it. A variable is basically a way to store some information while you are running the program that you can easily access. There are different types of variables, those that store integers, others that store floats, the numbers are with a fraction, there are strings, a sequence of characters and much more! Let's first take a look at the simplest variables out there: integers. Variables always have a name, there may be any combination of letters that you prefer, I would recommend some Names, though. For example, a variable called Counter is pretty clear about what it does, it counts some things, while a variable called lolthegame is nuptit ed it is nuply what it does. So, how do I use this dark magic to store my numbers? Simply with an equal sign! x =5 Here the number 5 is now stored in the variable x, note that the variable is always to the left of the equal sign and the number is on the right! We can also set a sequence of mathematical operators on the right: x =5*2 In this case, x would have the value of 10, since 5*2 5*2 10.Important: The only thing on the left side of the equal sign can be a variable, something like x + 5 = 0 is INVALID, you need to do x = -5 We can also print the contents of variables on the terminal: x =5 * 2print x This will print 10 because x has a value of 10 when printing. In fact, we can use our variables just like normal numbers! This means that we can..... 1.: Use them in mathematical operations: x =7-3print x **2 This will print 16, since x will have 4 and 4 squares 16. 2. : Use it to define other variables x =5 y = x +9print y This is printed 14, as in the definition y the x is replaced by its value, in this case 5, and 5+9 is 14. In fact, we can change x after we have redefined y, and y will not change: x =5 y = x +9 x =10print y As you can see, once a variable is set by changing other variables, you cannot change the contents of the first variable. 3.: Use it to define yourself: x =5 x = x +1print x This will print 6 because x is redefined by itself, in this case incremented by one. So, to summarize, variables, well, integer variables, the only ones we have used so far can be used as numbers wherever you want! When printing and defining variables, even yourself! When Tim Berners-Lee invented the World Wide Web in 1989, there was no JavaScript, no CSS, just HTML. Although HTML has changed a lot over the past 30 years, expanding from its original 18 tags to well over 120, it retains its central importance: it is the basic technology for the web. If your website is based on good HTML, it loads quickly. Browsers render INkrementell HTML, which means that they display a partially downloaded Web page to the user while the browser waits for the remaining files from the server. Modern fashionable development techniques like React require a lot of JavaScript to be sent to the user. When everything is downloaded, the user's device must analyze and run JavaScript before it can even start creating the page. In a slow network or on a cheaper, low-powered device, this can result in a frighteningly slow load and put a heavy load on the battery. Web sites based on good HTML are also robust—even if the styles and scripts cannot be loaded, the content is available. HTML is both backwards compatible and future-proof. Future-Proof HTML To demonstrate the future security of HTML, you can open the very first web browser written by Sir Uncle Timbo himself in 1991. Open a Web page by selecting Document > Open from the full document reference, and type a URL in the box. A well-structured HTML document rendered. It will have no styles (CSS 1 was not specified until 1996, and it was 2000 until IE5 for Mac was shipped with an almost complete implementation) and some of the more exotic punctuation or characters could be replaced by their character codes, but you can still read the content. And for the vast majority of websites, the content is what users come for. Here is .B check the Web Accessibility Checklist for very website rendered in the browser of 1991: Article about Web Accessibility on this website in the WorldWideWeb browser of 1991 To demonstrate the future security of HTML, we look at the very first website in a modern browser – in this case Firefox 77 (Developer Edition): As you can see, it is perfectly rendered – and is completely responsive when you narrow down the window: No-Writes One HTML by Hand Anymore Grandfather! I am sometimes told that no one has to learn HTML anymore, because nowadays no one writes HTML by hand. And that's largely true – 35.8% of the web is powered by WordPress, which assembles pages from templates to which different themes can be applied. Joomla and Drupal work similarly. Other developers use frameworks such as React that glue pre-built components together. But someone still has to write the templates and components. And developers using WordPress or React need HTML knowledge to evaluate the quality of the templates or components they want to use. An analogy is the decision not to cook and go to a buffet restaurant. You'll get much nicer food that's much more nutritious when each of the dishes has been cooked by professional chefs with the best ingredients than if each is a frozen ready meal served microwaved by a bored kitchen-hand trainee who doesn't care at all. The structure of an HTML element Before we begin to examine what makes good HTML, let's take a look at the structure of an HTML element. (If you already know this, you can skip this section.) Here's an HTML tag that tells the browser: This is a paragraph: <p> Browsers don't care about upper or lowercase letters in HTML: <video> <VIDEO>, and <Video> even <vIdEo> are all the same. But conventionally, we use lower cases because capital snouts look like screams. This tag is an opening day. Most, but not all, tags have a corresponding closing tag. As with the opening tag, the closing tag is enclosed in angle brackets, but here the tag name is preceded by a backslash that tells the browser that this is a closing tag: <p> As you can see, the tag name is surrounded by angle brackets. The opening tag, the content between the opening and closing tag, and the closing tag is an HTML element. Here we have two paragraph elements: <p>I'm a paragraph!</p><p>And so am I!</p> By default, a web browser displays them with an empty line between paragraphs. However, this is only the default setting, the display can be changed with a separate language called Cascading Style Sheets (CSS) without changing the HTML in any way. Tags can also be which give the browser additional information about the tag. Some attributes are allowed for all tags, others only for a subset. Attributes have names and assigned values in the opening tag. This tells the browser that this paragraph is in English: <p lang=en>Some attributes do not seem to take any value. For example, <audio loop=>is an abbreviation for ; either format <audio loop=loop>is</audio> </audio></vIdEo></Video></VIDEO></video> </audio></vIdEo>< Be careful with Boolean attributes; As the specification says, a number of attributes are Boolean attributes. The presence of a Boolean attribute on an element represents the true value, and the absence of the attribute represents the wrong value... To represent an incorrect value, the attribute must be omitted altogether. Tags can have multiple attributes: <p lang=en class=warning> This tells the browser that this paragraph is in the English language, and it belongs to the class called warning (which we can color .B red to differentiate it from the surrounding paragraphs). Some attributes can take multiple values separated by spaces: <p class=discount seasonal> This paragraph is a member of the discount class (perhaps we want to stylize this to draw attention to it) and is also a member of the seasonal class (perhaps we want to show a picture of a Christmas tree next to it). And that's all you need to know about the structure of an HTML tag. Choosing the right HTML element Sometimes I like to think that HTML is like the English language. It's pretty easy to read, but much harder to write – there are so many quirks and irregularities. Although HTML has such a small vocabulary (the average 2.5-year-old toddler speaks more words in his native language than elements in HTML), many people's HTML code is inaccurate. Vague HTML can make it difficult to analyze content for search engines, supporting technologies such as screen readers for blind visitors, and non-traditional devices such as Apple Watch and voice assistants. When I talk about good HTML, I really mean semantic HTML: tags that describe the content as accurately as possible. You have to ask yourself the philosophical question: what is this content? And which day best describes what it is, not what it looks like. Delete your mind from any thought of colors, distances, boundaries, or typography; these are not what the content is, but only its appearance and are defined by CSS. Sometimes it is very easy to describe what some content is. Consider, for example, this list of the three best songs from The Cheeky Girls: Cheeky Song (Touch My Bum) (Hooray, Hooray!) It's a cheeky holiday! Take Your Shoes Off It is easy to see that this is a list and the order of entries counts. The song Touch My Bum is objectively better than Cheeky Holiday (and probably the best song of all time). That's why they're numbered. We would use the HTML <ol> (ordered list) to surround this whole list, and each entry will be in a <li> (list item): Cheeky Song (Touch My Bum)
(Hooray, Hooray!) It's a cheeky holiday!
Take off your shoes
Would you like a to number one? Just use the reverse attribute on:<ol reversed=>
Take off your shoes (Hooray, Hooray!) It's a cheeky holiday!
Cheeky Song (Touch My Bum)
Suppose You are in a modern browser, you will see the list that counts down: Take off your shoes Hooray!) It's a cheeky holiday! Cheeky Song (Touch My Bum) Sometimes it is not immediately obvious which tag to use. Consider this horizontal navigation menu on my Cheeky Girls Fan website: Of course, each of the menu items is a link, but what else? It's another list -- a list of the pages on my site. But in this list, the order does not matter; Gabriela could be the first and Monica second. For an unordered list, we use <ul>, with each entry a So: About Monica
<a href=gabriela.html>About Gabriela> The advantage of using a list here is that helpful technology that some disabled people need to use, the web knows what to do with them. For example.B, a screen reader says something like a list of five items and usually gives the user the option to skip the list or jump to the next list. On auxiliary technology: Let's give the screen readers additional information and wrap ours in an element, as follows: <nav>
About Monica
<a href=gabriela.html>About Gabriela> </nav>As you can imagine, the browser tells you that this is the primary navigation. A screen reader user is advised that they are in the and they will probably have a keyboard shortcut to go directly to For example. For most languages, the language tag is fairly easy to determine. The Organization W3C, which creates the standards on which the Web relies, has a guide to selecting a language tag. Line 3 opens the header of the web page – this is a block of information about the page that helps the browser to process it properly. Line 4 tells the browser that we use a character encoding that can handle any human languages and emojis you could ever wish for. If you've ever seen a web page with strange characters like or blank boxes in the middle of the words, you know why we need that line. Line 5 is another magical summon. The browser instructs the browser to display the Web page in the width of the device on which it appears, and helps websites reformat websites through wide desktop displays and narrow mobile screens. Line 6 is the only one you should change. the <title> page is the text that is first read to a screen reader user to make sure they are on the correct page. It also determines the sort order in browser bookmarks so that I use a system that first has the site name (so that all bookmarks in the site are grouped), and then the page title after that. For example, <title>Cheeky Chap — Home</title>, <title>Cheeky Chap — About Monica</title>. There is no real way to use your <title>; The golden rule is to be consistent on all pages of your site. Adding our content Now it's time for us to add the specific code to our website. HTML is a markup language, so we often call the process of converting text content into HTML as marking. The Body Now we have the <head>, we open the <body>. (Strictly speaking, the <head></head> And <body></body> Tags are not necessary, but as you will see them on other people's websites, it is important that you know what they are doing). Type <body> in your code editor. Depending on your editor and its settings, it can automatically become a </body> tag for you. If so, place some blank lines between the opening and closing tag to make sure you don't enter code after closing </body>tag because it won't appear and will be annoying. Remember when I said that when you write your HTML, don't pay attention to what the page will look like? There is one exception to the rule: think about the order in which the content is in. By default, the browser displays all content in the source order (you can change this, but it's more work). So let's look at the content that comes first and write this HTML code. The header The first discrete content on our site is the banner above. This will be at the top of every page on our website. The HTML code for the <header></header> entire banner is enclosed in tags. As the HTML specification says, the header element is a set of introductory or navigational aids. In it we have a picture of the naughty girls, which also serves as a link to the homepage, the main headline for the page (I'm a naughty guy!) and an additional text. Let'</head></title></title> Let'</head></title></title> Look at the image, which is a homepage link. Clickable links (somewhat confusing) do not use the <link> tag; they use the <a> tag, which stands for anchor. This tag has the href attribute that defines the destination of the link. A normal text link is structured as follows: Homepage The text homepage appears on the screen, probably underlined in blue, and when you click it, the browser will delete the screen and load the index.html. However, our link has no text, it is a clickable image. The easiest way to put an image on the web is to use the <img> tag. It is structured as: <img src=dog.jpg alt=a nice dog> The src attribute refers to a file that is dragged into the Web page at this point. The second attribute, old, is for alternate text. This is for those who can not see the image due to visual impairments or because they have turned off images in their browser to make access to the content faster and cheaper. You don't need a picture of .. or picture of... write; a screen reader automatically precedes this. Each <img> must have an alt attribute. Older sites may display blank alt text (alt=). This indicates that the image does not convey any information and has no function, it is purely decorative. In modern Websites, decorative images are applied with CSS and are not in HTML, because it is just presentation and not content. WebAIM provides a useful introduction to writing good alternative text in quiz format. If your image is in SVG format, old versions of Safari (previous version 9.1.1 on the desktop or iOS version 9.3.2) require an additional role attribute (which is ignored by other browsers): Note that there is no <img src=walrus.svg alt=a walrus, pouting coquettishly role=img> </img> closing day. In some older websites, you can see it <img src=dog.jpg alt=a nice= dog=>written as self-closing. Both are okay. To create a clickable image that is a link to your home page, we need to nest one tag in the other. There is a change that we need to make: if the image is a link, the alt-text must describe the destination of the link (home) and not the content of the image. <img src=cheeky-girls.jpg alt=Home> The next part of our header is the name of the page – I'm a Cheeky Chap. Since this is the headline for the entire page, we give it the most important heading level, <h1>I'm a cheeky guy</h1>. HTML has 6 levels of heading, inventively named <h1>, <h2> Etc. There should only be one <h1> on a particular page, and subheadings should not skip a layer. The the headings are very important for users of assistive technologies to navigate a page. If your page has a meaningful heading hierarchy, you can use it to automatically generate a table of contents so visitors can navigate through long texts – Stuart Langridge's generated Toc script can be copied and pasted to do so. Almost all AT has an abbreviation for go to the next </h1></h2></h1> </h1></h2></h1> in the commercial screen reader JAWS and the free screen reader NVDA, the H key means to go to the next header, and the button 2 jumps to the next <h2> (among many other one-button navigation cuts). Now all that remains is the lyrics A homage to the musical genius of the Cheeky Girls. It's not really a headline, it's more like a strapline (a short, easily recalled phrase used by an organization to help people recognize it or its products). Since HTML does not have a dedicated strapline tag, we wrap it in <p>

element, which gives us our full header: <header><img src=cheeky-girls.jpg alt=Home><h1>I am a cheeky chap!</h1><p>A tribute to the musical genius of The Cheeky Girls</p></header>Next on the page comes navigation. We already know how to construct this: <nav>
About Monica
About Gabriela </nav> The main content Now it's time to focus our attention on the main content—the actual flesh of the page. HTML has a special element for this, <main>, which encloses all content in the screenshot above. The headline for the series of three posts is why are the Cheeky Girls so great, so let's give this one headline tag. We already have a <h1> in the header, and it should only be one per page, so we can use it in a <h2>. Next, we have three posts, each within itself, and with its own headline. We use an element for each of these elements. Do not fall into the trap of thinking that is only for news articles or blog posts; think of it as a garment. You can wear another shirt with the pants you are wearing; Your pants are one piece of clothing, your shirt is different. A page full of videos could include any video and summary; a page that is a product list could envelop any product in. With a</h2></h1></main></h2>Restrictions or copyrights. Small print is sometimes used to assign or meet license requirements. Copyright and Attribution is exactly the purpose of this content, so we will mark our footer as follows: <footer> <small>Copyright Bruce Lawson. Header image by Matt Buck.</small> </footer> Quality Assurance And now that we're done, let's check the quality of our code. The W3C has a free HTML validator that can either check a page on the web, upload files, or copy the code and paste it into a form. You should always review your code before you apply styles, write a script, or publish to the Web. If you also select the Outline check box before clicking the Review button, the validator displays all errors, heading hierarchy and logical structure of your page. Do not worry about the warning [nav element without heading]; Sighted users will understand that there will be navigation and supporting technology users will be automatically informed. What if there is no suitable HTML element? Occasionally, you're to find that there is no suitable HTML element for your content. Suppose that in the middle of a paragraph, we would like to indicate that a content snippet is in French. In such cases, there is a generic tag named : <p>When I first heard The Cheeky Girls, thought I, ouh l'l! and bought a few pink hotpants.</p> To wrap larger components, there's another generic tag called &t;for a little bit. Imagine that you want to group your navigation and an ad so that you can format it as a sidebar on the left side of your content. There is no element because HTML deals with the meaning of content, not its display. Our ad will go into an element that defines the specification as represents a section of a page that consists of content that is tangenttoally connected to the content around the page element, and that could be considered separate from that content. Our navigation will go inside</div>to decide how to display information on the Apple Watch. Browser extensions can also use it. In a blog post with some tongue-in-cheek numbers that show how different organizations have increased their search traffic with structured data, Google said: When you add markup to your content, you help search engines understand the different components of a page. Schema has a huge list of different types of vocabulary from geospatial geometry to Hindu meals by airlines and COVID testing facilities. But let's look at the scheme for a blog post. Each article is marked as follows: <article itemscope= itemtype= amp;gt;<h2 itemprop=headline>The Birth of Naughty Girls</h2><time itemprop=dateCreated pubdatePublished datetime=1982-10-31>31 October 1982</time><gt; </p><p>.... </p></article>Here, the search engine itemscope attribute tells you that the extent of this blog post is contained in this item <article>. The different itemprops are specified in the schema vocabulary, except for the Pubdate value on itemprop, which is not schemad but is required by Apple for WatchOS. You'll notice that I have a<time>element in this example. The datetime attribute specifies a date (and possibly a time) in the unique YYYY-MM DD format, while the element's visible text content can be anything: The Cheeky Girls' birthday, Tuesday, 31 Oct 82, The Feast of Saint Karen and the Immaculate Surfboard. Forms Forms make the Web interactive, not just a read-only media. Nobody likes to fill out forms, and even fewer people like to program them. The golden rule for forms is that each input field must have a corresponding label. Here's a demo I made of an unlabeled form field compared to an labeled form field. They look identical, but the top one does not have a proper label, while the second. Click the text label at the bottom and you'll see that it focuses on the corresponding input. This makes it much easier for someone with engine control difficulties to focus an input – or perhaps for you trying to check a tiny check box on a small screen in a bumpy move. It is also important for screen language users who cross fields in a form (by default, only links and form fields are focused by tab stops). When they are inserted into an input field, the screen reader specifies the contents of the associated label, field type, and any other information, such as .B whether it is a required field. The code for this is simple. The input field a unique ID, and the label is associated with the attribute for: <label for=food>What is your favorite food? </label><input id=food> Taken together, a simple one-field form would look like this: <form action=/form-handling-page.html method=post><p><label for=food>What is your favorite food?</label><input id=food name=food></p><button type=submit>Send</button></form> The action attribute of the </time></article> </time></article> is the URL of the page that will receive the form to do any required work, and probably say something like thank you! Your information has been received. The method attribute tells the browser how to send this information to the server. The name attribute on the <input> allows the server to access the information submitted in each field, and should therefore be unique in this form, except in the case of radio buttons where all mutually exclusive options have the same name (so the browser knows that it is a group). Hide labels Occasionally, you may not want a visible label. Here's an example if a label labeled Search feels like overkill before typing. We can link the input field to the text Search, which is the contents of the Send button with aria-labelledby: <input type=text aria-labelledby=searchbutton><button id=searchbutton type=submit>Search</button> We could have used Dasarier label:<input type=text aria-label=Search>But it's always better to prefer visible text on a page because it translates when the page is run by a translation tool, while text isn't hidden in HTML attributes. Grouping form fields groups a cluster of related fields. For example, the field pattern with three dates of birth date, or any cluster of radio buttons or related check boxes. as the first child of provides an overarching label for the entire group that gives them context. Then the individual fields and their labels let you know what choices/options etc. are available. OptionalUse the type attribute to tell the browser what type of input is expected: <input type=tel> tells the browser that a phone number is expected. <input type=email> expects a valid email address. <input type=number> the browser is indicated to reject non-numeric inputs. Note that type=number should only be used for quantities — phone numbers, credit card numbers, passport numbers are numeric, but not quantities. On mobile devices, most browsers use the input type to customize the offered virtual keyboard: type=email displays an alphanumeric keyboard with an A symbol and a dot, as it is included in all email addresses. type=number displays only digits and a decimal separator that matches the user's regions. type=date displays a kind of system date selection. There are many types of inputs—too many to list here—but the underlying markup approach remains the same: think about what your content means, not what it will look like. Auto-fill forms To automatically fill out forms, visitors need to do less so they are more likely to fill out a form and sign up/buy. Autofill on browsers: A Deep Dive is a great eBay article about it. Obsolete elements There are more HTML tags—the developer-oriented specification lists them all. It is important that you avoid using items that are marked as deprecated because these are outdated parts of HTML that still exist in the specification because they are used in some old (or poorly written) Web sites, but are now deprecated. Conclusion In the course of this article, you have learned some of the most commonly used HTML tags. Most importantly, you've learned the philosophy behind writing good semantic HTML: choose the HTML tag that most accurately represents the meaning of the content without thinking about how it's displayed. So now go and build fantastic websites that are lightning fast and have excellent SEO, and that work across all kinds of devices, for people with all different skills and disabilities! Disabilities!