**Tutors**

1. Python - Mosh Hamedani
2. JavaScript – Mosh Hamedani
3. HTML and CSS - Brad Traversy
4. Git and GitHub - Colt Steele
5. 100 Days of JavaScript - Ewomazino Akpareva
6. 100 Days of Code: The Complete Python Pro Bootcamp for 2023 - Dr. Angela Yu

**My Software Engineering Journey**

**Git, GitKraken, and GitHub Course at Udemy**

**Day 1, 21st September 2023**

**Git Commands**

* **Pushing:**
* **echo "# softwareEngineering" >> README.md**
* **git init**
* **git add README.md**
* **git commit -m "first commit"**
* **git branch -M main**
* **git remote add origin https://github.com/patrickisnjoroge/softwareEngineering.git**
* **git push -u origin main**
* Make a new directory/folder: mkdir
* Remove a directory: rm -rf
* List contents of a directory: ls
* List including hidden files: ls -a
* Print working directory: pwd
* Open File Explorer of current directory: start .
* git log
* git status
* git init
* git commit -m "added chapter 1"
* git add <file-name>
* git add .
* add and commit: git commit -a -m "message"
* git add outline.txt characters.txt chapter1.txt
* touch outline.txt
* touch characters.txt
* Create a file: touch
* Remove a file: rm
* Open directory in vscode: code .
* Configure editor to use vscode: git config --global core.editor "code --wait"
* Commit formatting
* git log --oneline
* Redoing a previous commit: git commit –amend
* Ignore a file that you don’t want git to track: .gitignore
* Creating a git ignore folder: touch .gitignore
* Get to know what to include in gitignore: gitignore.io

**Branching**

* Viewing branches: git branch
* Quitting: q
* Creating a branch: git branch <branch-name>
* Git switch <branch-name>
* Adding and Committing at the same time: git commit -a -m "commit message"
* Another way of switching: git checkout <branch-name>
* Display branches and get to know current branch: git branch
* Create and immediately switch branches: git switch -c <branch-name>
* Deleting a branch: git branch -d <branch-name>. NB: You have to be outside the branch
* Renaming a branch: git branch -m <new-branch-name>. NB: You have to be inside the branch being renamed.
* Make a branch and switch to it: git checkout -b <file-name>

**Merging**

* Get more info about branches: git branch -v
* Steps:
* Step 1, switch back to master
* Step 2, git merge <branch-name>

**Day 3, 23rd September 2023**

**Continued with Git**

**Git Diff:**

* git diff
* Used to show changes in git
* Comparing two branches: git diff branch1..branch2 or git diff branch1 branch2
* Comparing two commits: git diff commit1..commit2 or git diff commit1 commit2
* Comparing but for a specific file: git diff <branch-name> <branch-name> [filename]
* Compare the current HEAD to the previous commit: git diff HEAD HEAD~1 or HEAD~1
* HEAD~1 means the parent/previous commit of head
* To see unstaged changes: git diff
* To see both staged and unstaged changes since last commit: git diff HEAD [filename]
* To see staged changes for a file: git diff –staged [filename] or git diff –cached [filename]
* To see staged changes: git diff –staged
* To see both staged and unstaged changes: git diff HEAD
* Staging a file: git add [filename]

**HTML**

* A self-closing tag does not need a closing tag.
* <!—{Insert a comment} -->
* Ctr + {?/}: turn a line into a comment or insert an empty comment
* Ctr + F: Search a webpage\
* Ctr + Enter: Go into a new line from anywhere
* CTRL+Z for undo and CTRL+Y for redo
* <strong></strong>: bolds
* <em></em>: italic
* <br> or <br />: line break. Put content into a new line/paragraph. Several inserts several lines.
* <hr>: inserts a line break and includes an actual line.
* <b></b> also bolds but with strong, the text will stand out and CSS will decide how the standing out will be. It can be changed from bold to something else.
* <u></u> for underlining.
* <i></i> for italics
* <s></s>: for strikethrough
* <del></del>: strikethrough
* Copy a line: Shift+Alt+DownArrow
* Ctr + Left/RightArrow: Move word by word
* Home Key: Beginning of line
* End Key: End of line
* Alt + Up/DownArrow: Move a line
* Tag \* number of needed code lines, eg, li\*4+tab to get four list lines
* <div></div>: used to separate labels and inputs. Divisions for separating our mark-up.
* id; used to highlight. Include with the for attribute.
* Ctr+D: select multiple lines
* Value: the actual data that gets sent to the server
* Shift + Tab: Backwards tab

**Day 4, 25th September 2023**

**Continued with HTML**

* a+tab: Creates a link tag
* Inline elements stay within the same line while block elements get pushed down to the next line.
* Style tag: used to add css within HTML
* #: ensures the items go in the same page
* #about = <div id="about"></div>
* id tag: Its purpose is to identify the element when linking (using a fragment identifier), scripting, or styling (with CSS).
* Class: The class is an attribute that specifies one or more class names for an HTML element. The class attribute can be used on any HTML element. The class name can be used by CSS and JavaScript to perform certain tasks for elements with the specified class name.
* NB: with id, they are unique to each section but a class can be repeated across multiple sections. id is when you won’t have the info repeated somewhere else.
* <span></span> used when you want it to be inline while a div would push it to the next line.
* &copy; inserts a copyright symbol.
* &nbsp; pushes a line over.
* &gt; or &#62; greater than
* &lt; or &#60; less than
* &reg; for registered trademark ®
* Currency:
* &cent;
* &pound;
* &yen;
* &euro;
* Card Suits:
* &spades;
* &clubs;
* &hearts;
* &diams;
* <code> &lt;?php echo 'Hello' ?&gt; </code>
* <p>Save the file by pressing <kbd>Ctr + S</kbd></p>



**CSS**



* href points to the location that one wants to load from, e.g., href="css/style.css"
* styling. For id, use #, and for class, use a dot(.) and for tag, just write the tag.
* padding: The CSS padding properties are used to generate space around an element's content, inside of any defined borders. With CSS, you have full control over the padding. There are properties for setting the padding for each side of an element (top, right, bottom, and left).
* Padding vs margin: Padding represents the amount of inner space an element has, while the margin is whitespace available surrounding an element. It's not possible to set padding to auto padding. However, you can use automatic settings for margins. It's not possible to use negative values when defining padding, but you can with margins.
* Multiple selectors select everything (#welcome, #about) but a nested selector will select a specific element within another one (e.g. #welcome p). No comma (,) for the nested.
* body selects the entire webpage.
* Element is tag.
* F12 opens dev tools.
* Ctrl +/- zooms the dev tools in and out.



* If it is not a Web Safe Font, it will have to be linked in. WSF are available in all web-browsers.
* When a font family isn’t added, the coder will use Times New Roman or the Sans Serif, am not sure. It is set as the default.
* Default font-size: 16px.





* em is a multiplier so 1.2em = 16px \* 1.2. em multiplies by the parent unit.
* rem is set to the default which is usually 16px, so 1 rem = 16px.
* Targeting a span: #welcome p span.
* Colors: rgb(r.g.b), rgb(0,0,0);black to rgb(255,255,255);white.
* rgba = red, blue, green, alpha. background: rgba(0,0,0,0.6); 0.6 for the transparency. Alpha is for transparency of the background color.
* Hexadecimal: #000000; black or #ffffff which is white. Can be cut down to three i.e., #fff or #000.
* ff(red)ff(green)ff(blue).
* #box-1 = <div id="box-1"></div>
* Color changes text color while background-color styles the background
* NB: divs are block level meaning they go all the way to the end. All across.
* ./ represents the current directory.
* background-repeat is for repeating an image.
* background-position: 100px (x-axis) 100px (y-axis position);
* background-size: cover; shows the whole image.
* Image coverage is restricted to content area. An image won’t extend beyond the div.
* background-attachment: fixed; makes an image stay in position even after scrolling.



* The asterisk (\*) selects everything.
* Settings, editor.wordWrap
* Code at lower levels overwrites those at higher levels.
* #box-1.box: # is for id and . is for class. It becomes <div id="box-1" class="box"></div>
* Container/wrap/wrapper.
* With max-width, the content gets wrapped up for easy readability when the width changes.
* NB: padding adds to the width.  box-sizing: border-box; avoids this issue.
* NB: Floating items have to be cleared with the clr function, i.e., .clr {clear: both;} in the CSS and <div class="clr"></div> in the HTML. With CSS grid or flex boxes, this is not done. Floats are almost like a hack to align elements.
* text-decoration: none; removes decorations on links.
* Hover: what happens when one hovers.
* States for links: hover, visited, and active.
* Padding: (top\_and\_botton) then (left\_and\_right), e.g., padding: 10px 20px.
* Border-radius rounds corners, e.g., border-radius: 5px;
* cursor: pointer; brings out the pointer for buttons.
* ul.side-menu = <ul class="side-menu"></ul>
* overflow: auto; brings back background for floating items.
* overflow: hidden; for items that go outside the element and there is a scrollbar and you don’t want the scroll bar there.
* display: inline; puts block items on the same line.
* Inline elements do not go all the way across.
* margin:auto puts elements in the middle however, it only works for block elements so (display:block;) has to be added.
* display: inline; changes block elements to inline and display:block; changes inline elements to block elements.
* display: inline-block; will puts items side to side when they refuse to align.



* z-index brings things closer, e.g., z-index: 1; to reveal a box hidden by a container.
* Instead of line breaks, br, turn the inline elements to block elements. Highly recommended.

**Day 5, 26th September 2023**

**CSS Login Page Task**

* Color changes color of the text while background changes color of the background.
* text-align aligns text.
* Clearing out margin and padding on everything is margin: 0; and padding: 0;
* line-height: 1.8;

**Continued with CSS**

* display: none; does not display the content. visibility: hidden; makes it invisible but the space remains.
* Targeting a class overrides targeting a class element.
* !important; gives precedence.
* With margin, there can be negative but padding no since it is insider the borders.
* Bootstrap themes for design inspirations.

**Hotel Website**

* NB: websites use index.html as the home page.
* Id comes on top of class.
* Clear all margin and padding to remove the defaults of browsers.
* overflow: auto; makes sure the background shows no matter what.
* Links are hidden unlike texts which shows even with background.
* Float moves things
* A comma means both. It is an and.
* (.something) adds a class with a div.
* max-width: (x)px prevents stuff from overflowing to other areas.
* (#something) adds an id with a div.
* <span></span> helps to target specific words/letters.
* Container contains elements within a section.
* Single dot means inside the current file while double dot (..) means we are going outside the current file.
* fa-3x will increase font size by 3, e.g., <i class="fa-solid fa-hotel fa-3x"></i>.
* box-sizing: border-box; prevents the border-box from being added to the padding which would not look great.
* Whatever we have in the “current” page is what we want highlighted.
* By default, an image is displayed as inline.
* py-1 {} is the y-axis. It is padding y-axis(py).
* Opacity: (0 to 1); where 1 is opaque and 0 is completely transparent.
* section#contact-form.py-3 = <section id="contact-form" class="py-3"></section>.
* Shift+right/left arrow to select each letter in a word consecutively.
* outline: none; to remove the default outline of a form then border-color: #f7c08a; to set a desired outline color.

**Day 6, CSS: Responsive Designs**

* Responsive Design: The ability of an app to respond to different screen sizes.



* Normal size is the computer size.
* Inserting a media query: @media(max-width: 500px) {body{}}.
* Common break-point for smartphones is usually 500px and tablets 768px.
* Limiting display width: @media(min-width: 501px) and (max-width: 768px) {body{}}.
* Media types: screen, print, and speech. E.g., @media only screen and (max-width: 500px) {body{}}. Default is all.
* Float to none for smartphones, maybe.
* li{$}\*4 = <li>1</li><li>2</li> <li>3</li><li>4</li>
* the child goes into the parent and then em use the parent to determine size of the child leading top children bigger than parents but for rem, they are based on the root elements so no inner items bigger than outer ones.
* 16px is the standard root element.
* Changing the root element font-size: html {font-size: 10px;}
* 62.5% = 10px.
* NB: rem is affected by browser font-size settings but for em no effect because em uses parent elements which have font sizes. The settings changes the root element font-size hence the effect on re hence rem units makes things more adaptive and responsive. Rem changes with settings.
* Viewport Height (Vh) and Viewport Width (Vw).
* Viewport = browser body. height: 100vh; cover the entire page.
* Each Vh is a slice across and there are 100 slices.
* Background color and image together, background: #333 url();
* When the window is bigger than the actual image, the image repeats.
* A class of box stacks items.



* display: flex;



* .item.item-1 = <div class="item item-1"></div>
* .flex-container {display: flex;} turns a container into a flex-box.
* Flex-direction: row (and) column (and) column-reverse (and) row-reverse.
* flex-wrap: wrap; allows wrapping upon changing of screen size.
* flex-shrink: 0; prevents the flex boxes from shrinking.
* flex-grow: 1; allows the flex boxes width’s to changes with screen adjustments.
* flex-basis: is setting the width of the item.
* flex: (grow) (shrink) {and} (basis); e.g. flex: 1 0 200px; but just flex: 1; allows the items to auto-change.
* Default flex-direction (is): row;
* justify-content changes flex-items justification, i.e., justify-content: flex-start/center/flex-end; justify-content: space-around; arranges the flexes around the items - it takes the available space and distributes it around the items including the sides but with justify-content: space-between; there is no space on the sides. So with flex-direction: column; justify-content: space-between; is going to be vertical.
* align-items: also changes alignment but the default is row so it has to be changed to flex-direction: column; for column items.
* NB: Most things are centered. Layouts are mostly centered.
* align-item aligns all the items while align-self is syntax for aligning a single item.
* box-sizing: border-box; so that the padding doesn’t take up the width.
* When you take away properties at the top, you then add them latter in the respective areas.
* Adding img {} in CSS sets the properties of any images.
* Flex allows list items to float.
* The cross axis runs perpendicular to the main axis, therefore if your flex-direction (main axis) is set to row or row-reverse the cross axis runs down the columns.
* .hero::before allows content to be inserted before or after the specific element.
* With positioning, if you have something absolute within a parent, the parent has to be positioned relative.
* Placements, top: ; or left: ;
* .hero \* {} anything/everything that is in hero.
* > means the direct one, e.g., .flex-items > div {} is the direct div child.
* object-fit: cover; prevents stretches.

**Day 7, EdgeLedger Website**

* .callback-form input:focus {outline-color: #28a745;} changes the outline color of a form box.
* fab for social icons, e.g., <a href="#"><i class="fab fa-facebook"></i></a>.
* .footer a {color: #fff;} to be able to see any links.
* This | is a pipe character.
* .flex-columns.flex-reverse .row {flex-direction: row-reverse;} reverses flexes. Two dots used because the class also has a class.
* display: block; will put image in the middle without needing to use flex.
* let scrolled = false; means we haven’t started scrolling yet.
* 'translateY()'; moves things along the y-axis.
* CDN means it will just load from their Content Delivery Network (CDN).

**Continued Learning**

* Shift + F5 = reload.
* div p {} looks for any paragraph that is inside of a div while div > p {} is any paragraph that is a direct child of a div so if the immediate parent is not a div, there will be no effect and div + p {} is the next sibling or directly after the div.
* <a href="#">Page 1</a> the # is where the page will go because <a href="http://google.com" target="\_blank">Google</a> will go to google.com, the (target=”\_blank”) tells the site to open the link in another tab. target=”\_self” remains in the same page.
* a[target] {} is CSS for formatting blank target attributes and a[target=’\_blank’] {} specifies targets with a blank attribute.
* li{Item $}\*n the $ adds incremental numbers, item 1, item 2, item 3, item n.
* list-style: none; removes listing style for list items.
* In coding semi-colon is line the period at the end of a sentence in English.
* In CSS, li:first-child {} is a pseudo-selector for the first child and li:first-child {} is a pseudo-selector for the last child element. So, : is a pseudo-selector. li:nth-child(3) {} selects the third child. li:nth-child(3n+0) {} selects every third child in a list-item.

**Day 8, Continued Learning**

* .is-required:after {content: '\*';} = Name \*
* Box shadow, they come off the box, example is /\* offset-x | offset-y | blur-radius | spread-radius | color \*/ box-shadow: 10px 10px 10px 1px teal;

**Box shadows**

****

**Text Shadows**

****

* Defining a variable, --light-color: #f4f4f4;
* Any variable created in the root scope can be used anywhere in the styles sheet. :root {--light-color: #f4f4f4;} then used like this, background: var(--light-color);
* NB: For animation, a keyframe has to be declared.

**Animation**

****

* The forwards property returns the object to the starting position.
* Background-radius: 50% 0 0 0; each value pertains to a corner.

**Animation**

****

* Transition allows an item to slowly change.
* Hovering: placing an item above another.

**Transition**

****

**Transform**

****

* deg is degrees.
* Skew shifts diagonally.
* Scale increases size.
* The content of a webpage goes into the body tag.
* # always presents id.
* btn btn-dark is button, button-dark.
* overflow: hidden; scrollbar won’t show
* ids are unique to an element and are marked by #.
* Classes are shared by different elements and are marked by a dot (.).
* Default flex is a row so it has to be changed into a column, i.e., .page {display: flex; flex-direction: column;}.
* 1rem by default is 16px.
* Put a comma when you want two elements to have the same properties.



* With a checkbox, you can style the checked state. Class=toggler and type=checkbox for hamburger menus. So, at first, will begin as a checkbox.



* When you use variables in CSS, include the root:.
* Background image formatting, background: url('../img/showcase.jpg') no-repeat center center/cover;
* position: fixed; fixes its position on top of everything else.



* The element+element selector is used to select an element that is directly after another specific element. The adjacent sibling combinator ( + ) separates two selectors and matches the second element only if it immediately follows the first element, and both are children of the same parent element .
* list-style: none; removes bullets of list items.
* color: inherit; The inherit keyword specifies that a property should inherit its value from its parent element.
* ~ is used to target something that precedes another element.
* \* is a reset.
* Overflow-x takes care of horizontal while overflow-y: hidden; there won’t be a vertical scrolling bar.

**Day 9, Continued Learning HTML and CSS**

* Non-responsive means it stays the same even with different screen sizes. The contents do not auto-adjust.
* calc(100vw - 90px)

**CSS Grid**

* An alternate to flexbox.
* Flexbox, having a container, and then having items within that container.

****

****

* Grid supports 2D layouts while flex is only 1 dimensional. Flex is only for a single row or column.

****

****

* Span. The <span> HTML element is a generic inline container for phrasing content, which does not inherently represent anything. It can be used to group elements for styling purposes (using the class or id attributes), or because they share attribute values, such as lang.

****

**Grid Columns**



* fr: A fraction of the page. Like flex-1, flex-2, etc.
* Main. The <main> HTML element represents the dominant content of the <body> of a document. The main content area consists of content that is directly related to or expands upon the central topic of a document, or the central functionality of an application.



**Grid Rows**

* grid-template-rows: (1st item, 2nd item, 3rd item, nth item).
* grid-template-columns: repeat(2, 1fr); The repeat is two columns each of size 1fr.
* grid-auto-rows: 3fr; sets the rows of all unspecified rows to size 3fr.
* A pseudo selector, e.g., .item:first-child {} allows us to select an item instead to defining a class for it.



* minmax(300px, 1fr) = a minimum of 300px and a maximum of 1fr.
* Section. A section tag defines the section of documents such as chapters, headers, footers or any other sections. The section tag divides the content into section and subsections. The section tag is used when requirements of two headers or footers or any other section of documents needed.
* Stacked is just 1 column, for instance, in mobile.

**Grid Media Queries**



**Day 10, JavaScript**





**HTML and CSS**

* Clear cache – shift + F5.
* nav#main-nav = <nav id="main-nav"></nav>.
* When you use the before and after, you wanna put in the content, i.e., #showcase:before {content: '';} and when you don’t actually want to insert content (like this example), you just leave it blank.
* Images. Using images in HTML is better when the image has any contextual meaning... if it is a decorative picture without any contextual meaning, then use CSS. CSS is for presentation, HTML is for content.
* text-transform: uppercase;

**Day 11, JavaScript**















**Day 12, JavaScript**







**Day 13**

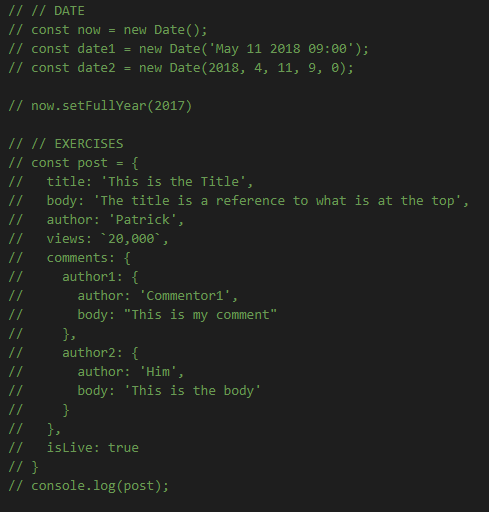
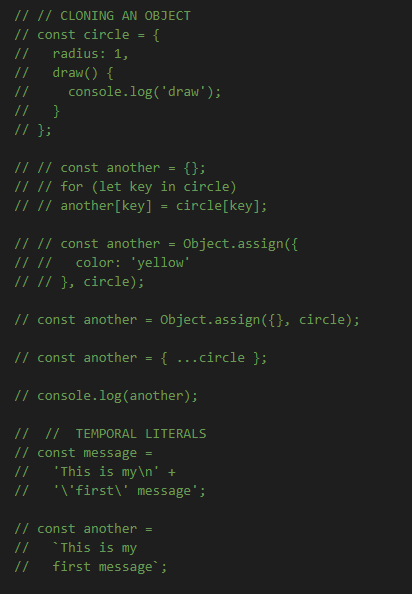
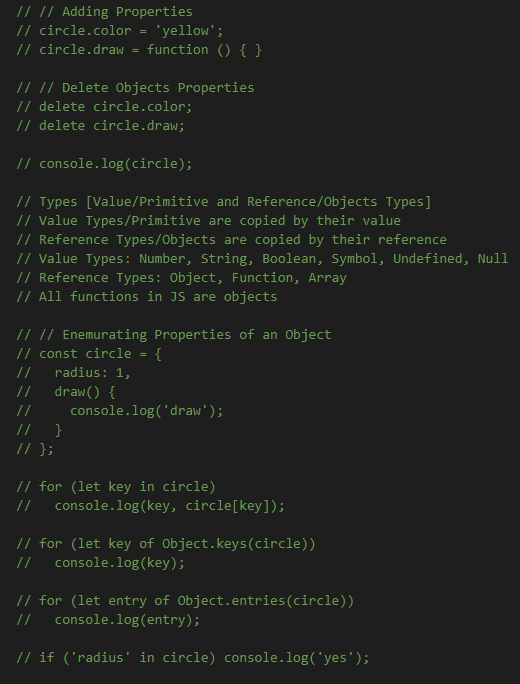
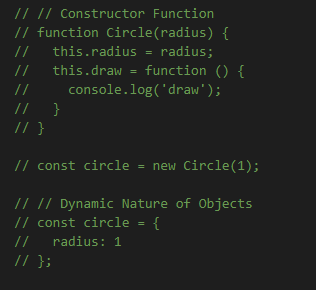
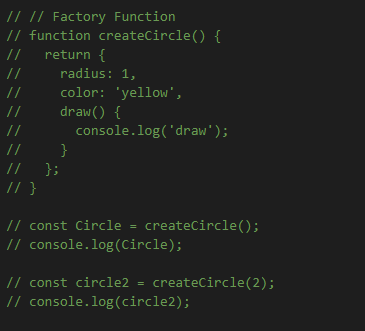
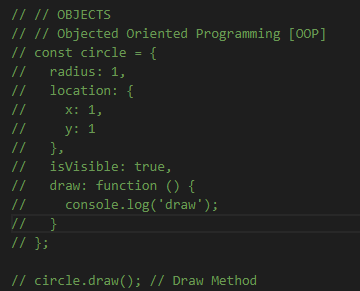
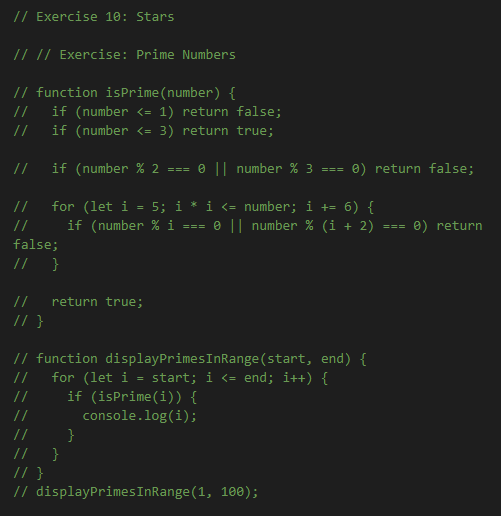
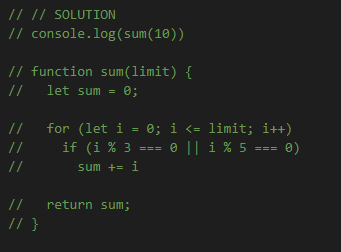
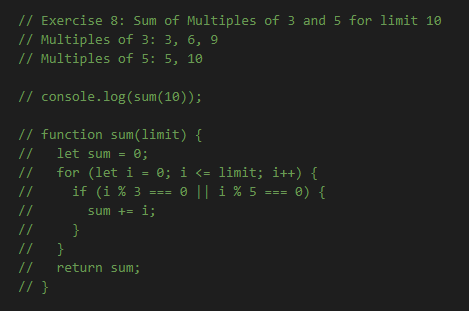
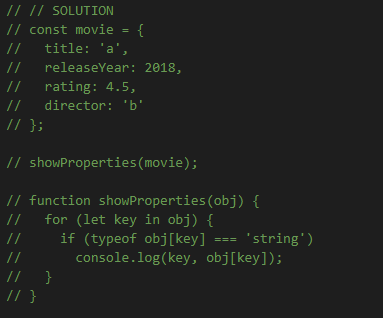
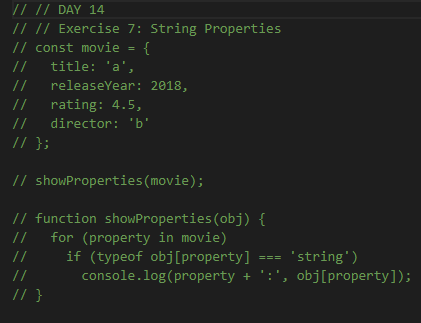
**HTML and CSS: Sass**

* npm init –y will simply generate an empty npm project without going through an interactive process. Create a package.json file since we are installing anything with npm.
* npm install node-sass.
* npm install node-sass -D. The -D makes it dev dependent.



* the compiled scss program will go into dist, a folder.
* scss is the extension for sass files.
* npm run sass is a command line command to constantly watch the scss folder.
* The dist folder is the actual project. The actual product to be deployed.
* $ to define a variable and : to assign a value, e.g., $color: blue;
* “\_” tells the scss not to compile the file into a css file.
* &-a represents whatever is the parent selector.
* A function returns something. You have to use the return keyword.
* webkit and ms for transform.

**Day 14, JavaScript**

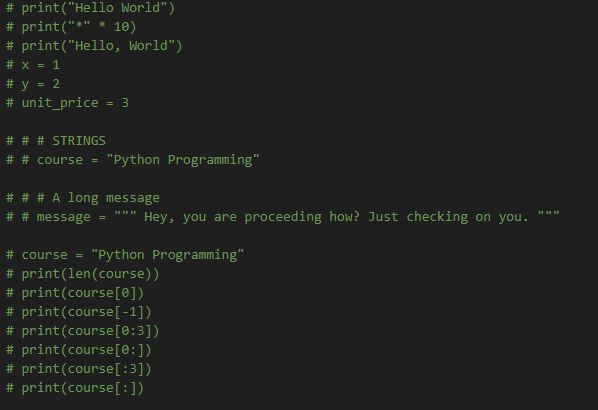


**Day 15, HTML and CSS**

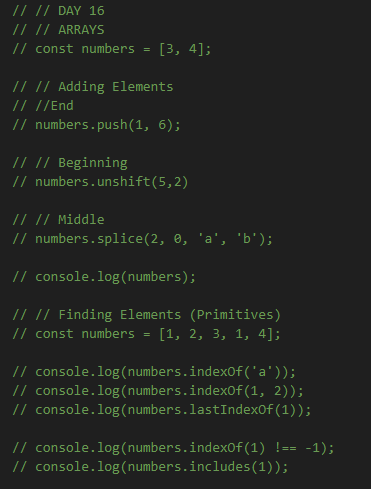
* button[class^='btn-']:hover will select any button with ‘btn-‘.

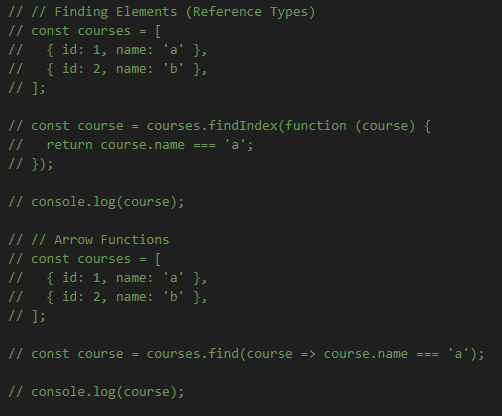
**Day 16, Introduction to Python**

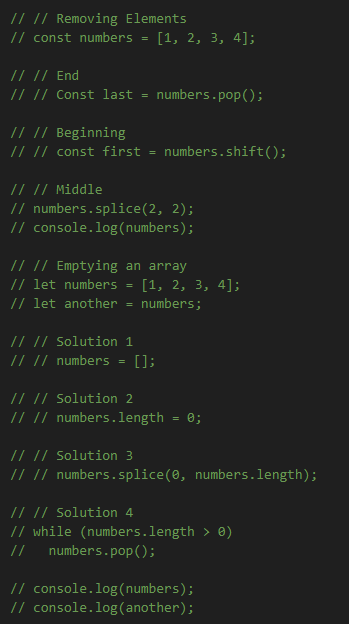
* Introduction to basics of python.

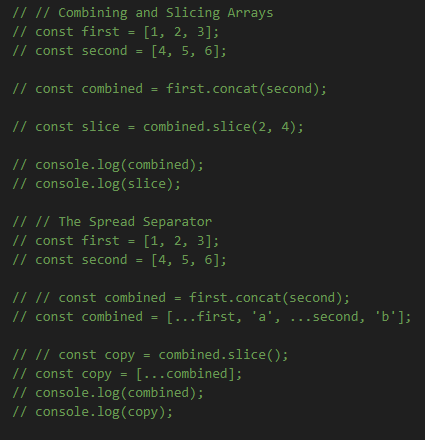
****

**JavaScript**

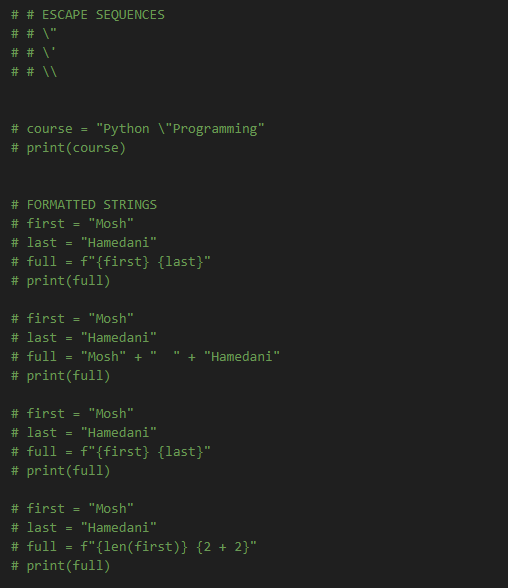


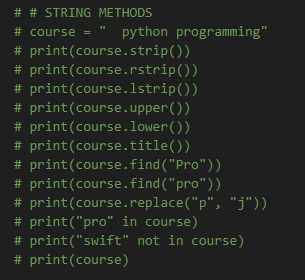


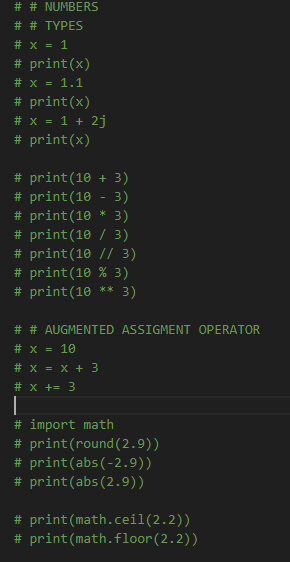


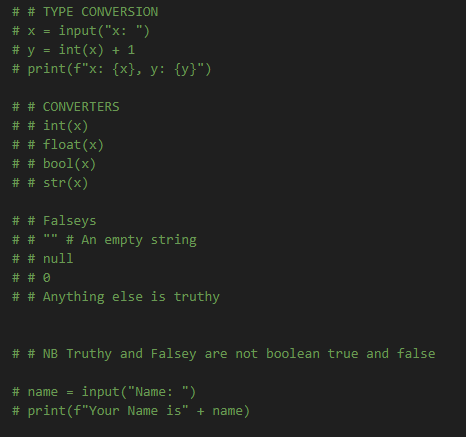


**Day 17, JavaScript and Python, and HTML and CSS**

****

****

****

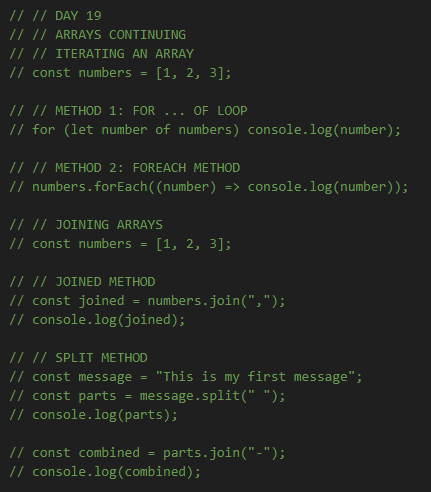
****

**Day 18, HTML and CSS**

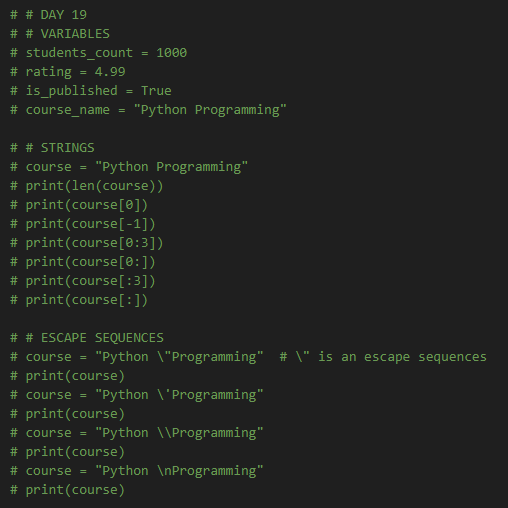
* Finished the portfolio website.
* Finished the HTML and CSS course by Brad Traversy.

**Day 19**

**JavaScript**

****

**Python**



**Day 20**

**JavaScript**

**Python**

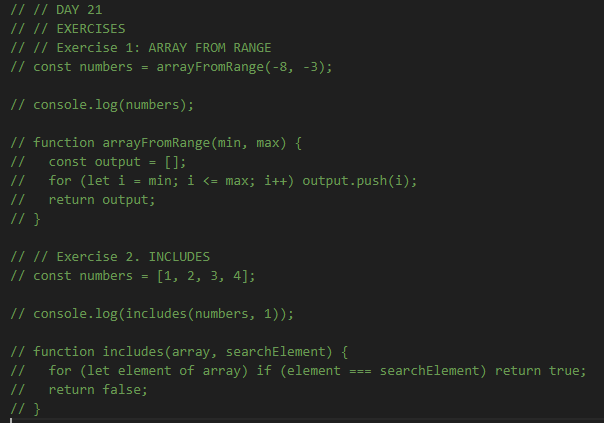
**Git and GitHub**

**Stashing**

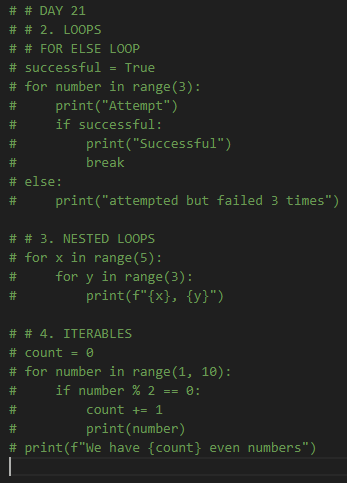
* Allows switching branches even without committing.
* Allows changes to be saved even without committing them.
* The command is: git stash or git stash changes.
* git stash pop removes the most recently stashed changes in your stash and re-applies them to your working copy.
* git stash apply the changes remain in the stash while in pop, the changes are removed from the stash.’
* git stash list: view what is in the stash.
* git stash apply stash@{2} applies changes in a specific stash or $ git stash drop stash@{0} to remove a specific stash from a list.
* Git stash clear to completely empty out a stash.

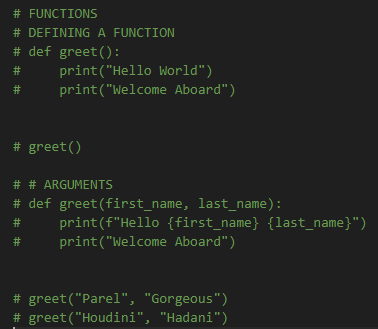
**Day 21**

**JavaScript**

****

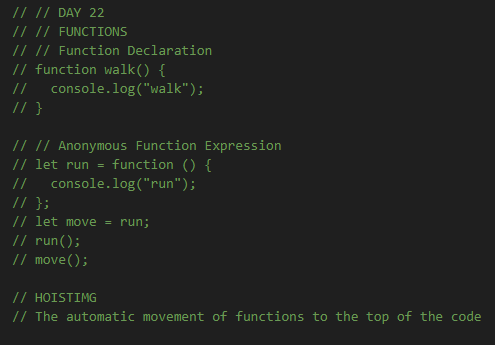
**Python**

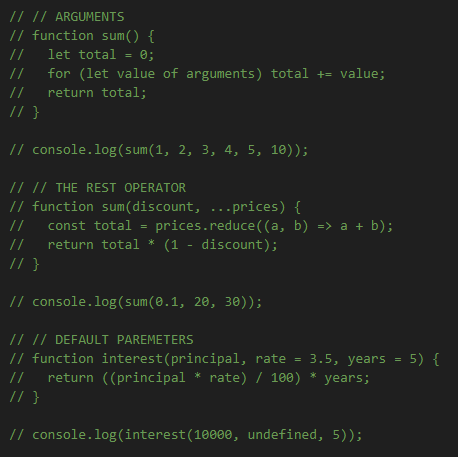
****

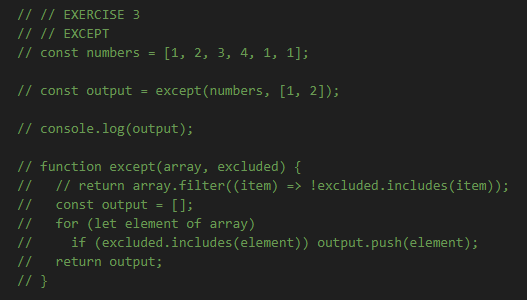
****

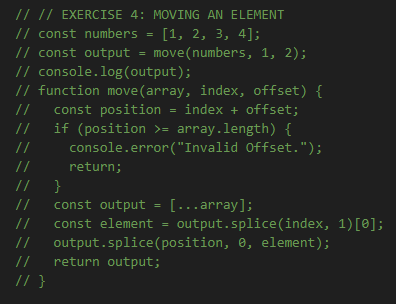
**DAY 22**

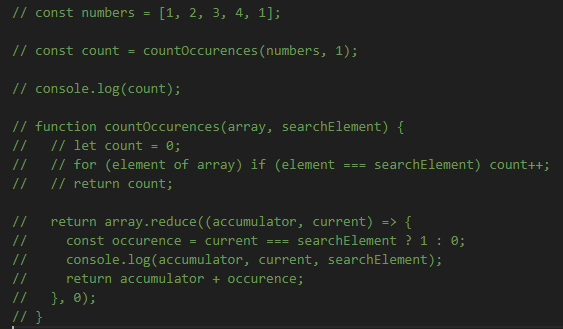
**JAVASCRIPT**



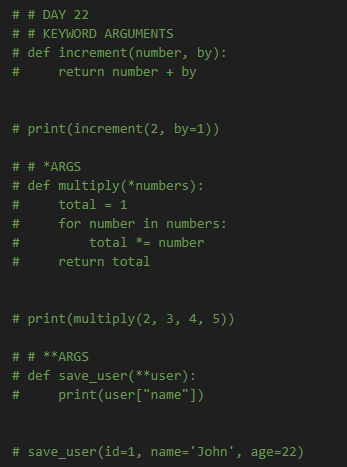


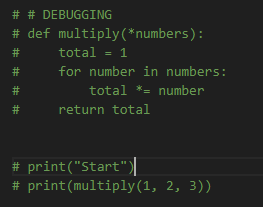






**PYTHON**



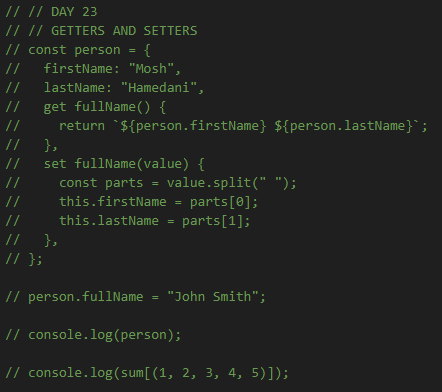


* Ctrl + Home: move to start of document.
* Ctrl + End: move to end of document.

**Debugging**

* F9: select breakpoint when cursor is on the desired line.
* F5: launch debugger.
* F10: move along lines when debugging.
* F11: move to the start when you have reached end of code.

**DAY 23, JavaScript**



**Day 24, Git and GitHub**

* Detaching head: git checkout <commit-hash>. Time travelling back to a commit.
* Reattaching head: git switch master
* git checkout HEAD~1 will be what comes 1 commit before head (it is the last commit. One commit before head) and HEAD~2 is 2 commits before head.
* (git switch –) takes you back to where you left of.
* Discarding changes: git checkout HEAD <file> or git checkout -- <file>
* git restore <file-name> restores files to the contents in the HEAD provided you have not committed since head points to master which points to the last commit. However, it is irreversible, any changes made will be lost.
* git restore –source HEAD~1 app.js will restore the contents of app.js to its state from the commit prior to HEAD and then git restore <file-name> to go back to the last commit.
* git restore –staged <file-name> removes a file from staging that has already been added but you don’t want to include it in the next commit.
* Regular reset: git reset <commit-hash> will reset the repo back to a specific commit. It retains file changes but resets HEAD to the specified commit. Useful for when commits were made on the wrong branch but you want to retain the work. The work can now be moved to another branch.
* Hard reset: git reset –hard <commit> will remove both the commits and the changes in the working directory.
* git revert <commit-hash> creates a brand-new commit which reverses/undoes the changes from a commit. It undoes the changes made in a commit resulting in a commit with same content as the previous commit. Because it results in a new commit, you will be prompted to enter a commit message. It keeps the commit history but changes are reverted.

**Day 25**

**JavaScript, prototypes**

**Day 26, Git and GitHub**

**GitHub**

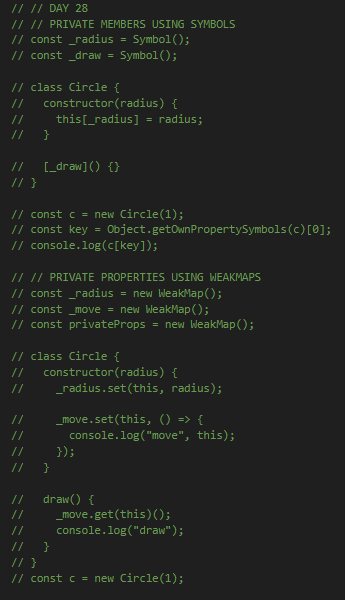
* git clone <url>. It will clone any hosted repo.
* git remote -v lists the current remotes in a repo.
* git remote add <name> <url>
* git remote rename <old> <new>
* git remote remove <name>
* git push <remote> <branch>

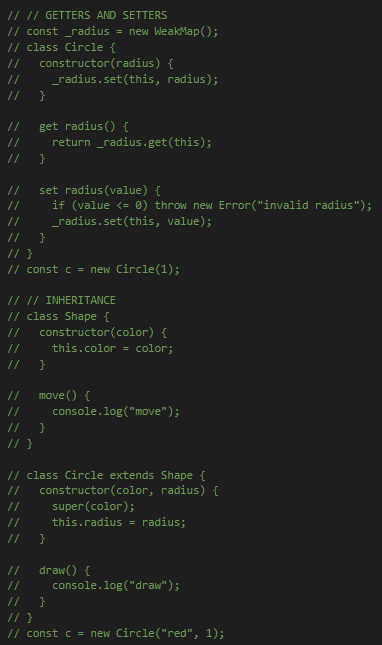
**Day 27, Git and GitHub**

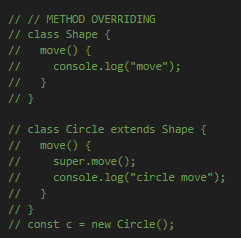
* git push <remote> <local-branch>:<remote-branch>, for example, git push origin pancake:waffle.
* git push -u origin master will allow us to set the upstream of the branch we are pushing. Allows to just use git push for consequent pushes.
* git branch -M main remains the branch to main.

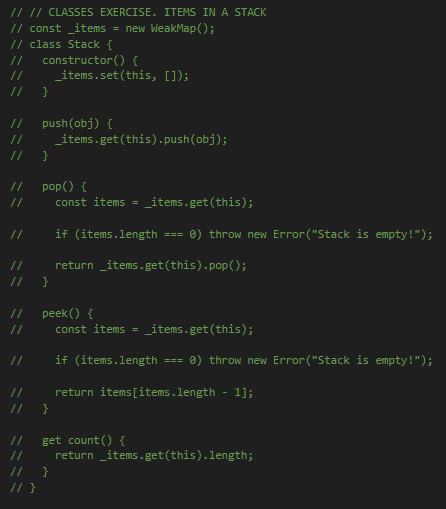
**Day 28**

**JavaScript**

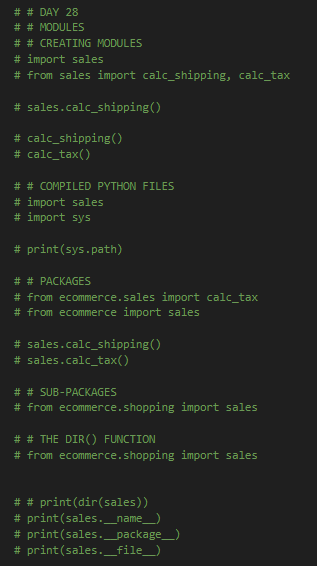


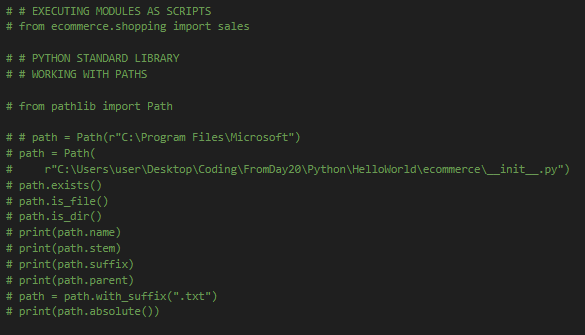






**Python**





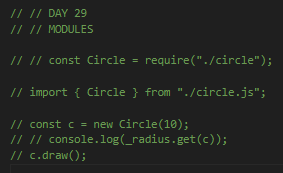
**Git and GitHub**

* git branch -r will show the branches our local repository knows about.
* git fetch origin master will update origin/master.
* git fetch <remote> fetches branches and history from a specific remote repository.
* Fetch gets changes from GitHub and brings them to the machine.

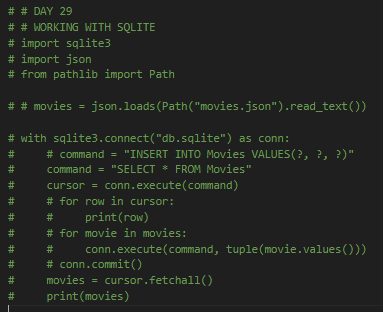
**Day 29**

* Finished JavaScript Advanced Tutorial by Mosh Hamedani

**JavaScript**



**Python**



**Git and GitHub**

* git pull <remote> <branch> will merge changes into a branch. Changes will be merged into the branch you are currently on, for example, git pull origin master.

**GitHub Grab Bag: Odds & Ends**

* cat <filename> to view the contents of a file on the CLI.

**Day 29**

**Git and GitHub**

**GitHub Grab Bag: Odds & Ends**

**Day 32**

**JavaScript**

* Started 100 Days of JavaScript by Ewomazino Akpareva.
* Created Counter, Random Number Generator, Random Hex Color Generator apps.

**Git and GitHub**

**Day 35**

* Started 100 Days of Python by Dr. Angela Yu.’

**Day 36**

**JavaScript**

**Git and GitHub**

**Git Collaboration Workflows**

**Day 37 to Day 43**

* Worked on personal projects.

**Day 44**

* Resumed to 100 Days of Python.

**Day 45**

* Continued with 100 Days of JavaScript.