

How to create your personal site on GitHub

- A free site without advertisements
- Simple and responsive layout
- Fully customizable with text and images
- Easy to create and easy to update
- Easily create additional web pages
- Display elegantly your Jupyter Notebooks
- No coding experience required
- The only language needed is Markdown
- Can be upgraded to a professional looking site
- Can be used with your personal domain name e.g., myname.com

Contents

- [1. Creating your personal site](#)
- [2. Adding a theme](#)
- [3. Updating your content](#)
- [4. Creating additional project pages](#)
- [5. Upgrading to a professional looking site](#)
- [6. Using your personal Domain Name](#)
- [7. APPENDIX: GitHub Markdown Language](#)

Note: When creating a new repository on GitHub, the default branch is called "main" but in older repositories the default branch was called "master"

<https://hurtlingthrough.space/posts/20200617-master-no-more/>

As I am using some newer and some older repositories in this tutorial, sometimes you may see the default branch as "main" branch and sometimes as "master" branch but other than that, the instructions are the same.

1. Creating your personal site:

TL;DR

1. Create a new GitHub account at <https://github.com> or use your existing GitHub account. (Let's assume your username on GitHub is: `username` Thus, from now on, wherever `username` is mentioned, you will just replace it with your own GitHub username).
2. Create a new repository with this name (exactly this name):
`username.github.io`
3. Give a description to your repository, choose it to be public, and choose to create a README.md file. Leave other options as they are.
4. As soon as your new repository is created, go to the top of the repository page, and click `settings`. Then, under `GitHub Pages`, select `Main Branch` and click `save`.
5. Congratulations, you've done everything needed and your personal site is being prepared. After a few minutes, your personal site will be published at: `https://username.github.io`

2. Adding a theme

From your repository, click `settings`, and then, under the "GitHub Pages" options, select "Choose a theme". Personally, I would suggest the "Cayman" theme, which is responsive.

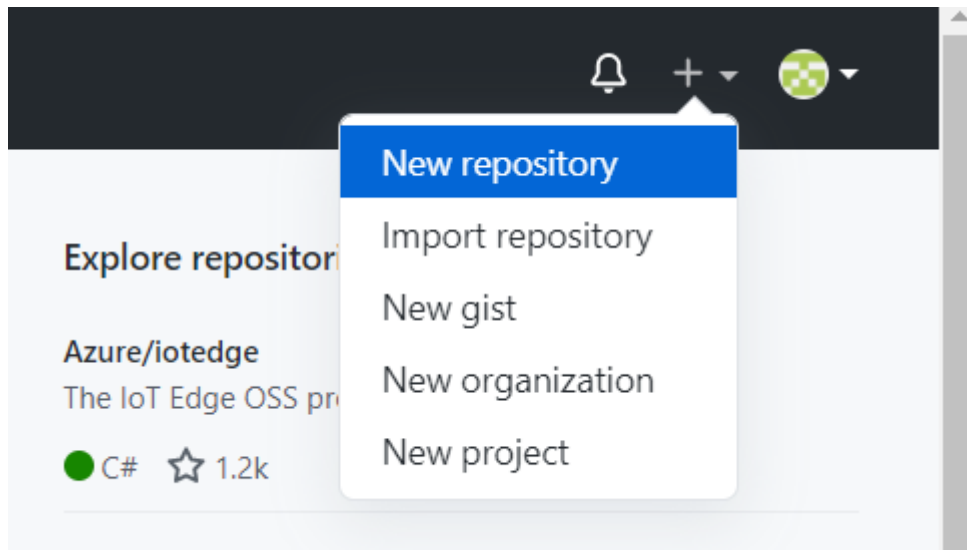
In a few minutes, your site will be updated with the theme you have selected. If you don't see any changes when visiting your site, then just click the reload button on your browser, or press "Ctrl" + "R", so that your browser will reload the page and the updated content will appear.

If at any time you wish to remove the theme from your site, just delete the file "_config.yml" from your repository. To later add a theme, just follow the process of adding a theme to your site.

To customize a theme, e.g., change colors or add/remove buttons, you can read the supporting material for every theme (this though, is more advanced): <https://pages.github.com/themes/>

Detailed process and images on how to create your site and add a theme:

Create a new repository:





Name your repository as **username.github.io**

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner * Repository name *

 antzm / 

Great repository name The repository antzm.github.io already exists on this account. ut didactic-carnival?

Description (optional)

Note: As I have already created in the past the repository named **username.github.io** I cannot use this name in the following example.

Thus, I will use a different repository name **BUT** you should follow the exact steps to create a repository named **username.github.io**

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner * / Repository name *

Great repository names are short and memorable. Need inspiration? [Help](#)

Description (optional)

☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

☐ **Add .gitignore**
Choose which files not to track from a list of templates. [Learn more.](#)

☐ **Choose a license**
A license tells others what they can and can't do with your code. [Learn more.](#)

This will set `main` as the default branch. Change the default name in your [settings](#).

You should name your repository as username.github.io

Write a description

Your repository should be Public

Choose the option "Add a README file"

Create your repository

Your repository has been created:

antzm / github-tutorial

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags Go to file Add file Code

antzm Initial commit d758cb6 now 1 commit

README.md Initial commit now

README.md

github-tutorial

A short tutorial on how to create your personal site on GitHub

Settings

Now click "Settings" and scroll down to the **"GitHub Pages"** options:

➔ GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

Source
GitHub Pages is currently disabled. Select a source below to enable GitHub Pages for this repository. [Learn more.](#)

None ▾ Save

Theme Chooser
Select a theme to publish your site with a Jekyll theme using the gh-pages branch. [Learn more.](#)

Choose a theme

Choose the Source by selecting the **"main"** branch (or the "master" branch in old repositories).

Source
GitHub Pages is currently disabled. Select a source below to enable GitHub Pages for this repository. [Learn more.](#)

None ▾ Save

Select branch

Select branch

main

✓ None

heme using the gh-pages branch. [Learn more.](#)

The folder should be **"/(root)"**, which is the default option.

Source
GitHub Pages is currently disabled. Select a source below to enable GitHub Pages for this repository. [Learn more.](#)

🔗 Branch: main ▾ 📁 /(root) ▾ Save

Theme Chooser
Select a theme to publish

Choose a theme

Select folder

✓ /(root)

/docs

branch. [Learn more.](#)

And now just click "save" and wait a few minutes.

Source
GitHub Pages is currently disabled. Select a source below to enable GitHub Pages for this repository. [Learn more.](#)

🔗 Branch: main ▾ 📁 /(root) ▾ Save

Theme Chooser
Select a theme to publish your site with a Jekyll theme using the gh-pages branch. [Learn more.](#)

Choose a theme

As you can see, there is a message that your site is currently being built and ready to be published on the URL shown.

GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

Your site is ready to be published at <https://antzm.github.io/github-tutorial/>.

Source
Your GitHub Pages site is currently being built from the `main` branch. [Learn more.](#)

Branch: `main`

/ (root)

Save

Theme Chooser
Select a theme to publish your site with a Jekyll theme. [Learn more.](#)

Choose a theme

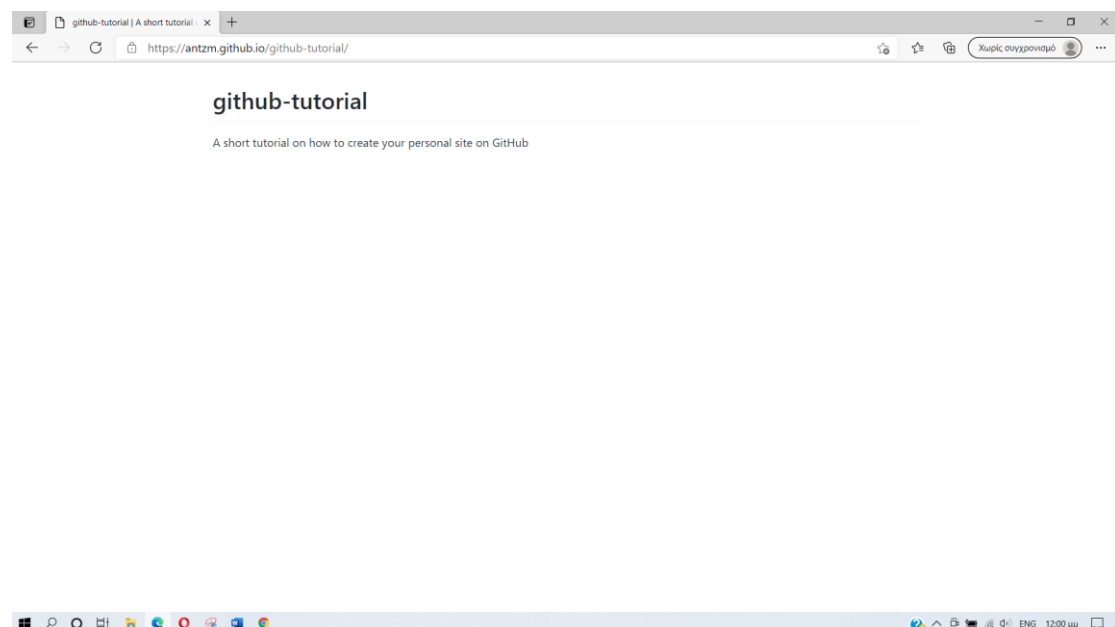
Custom domain
Custom domains allow you to serve your site from a domain other than `antzm.github.io`. [Learn more.](#)

Save

☒ **Enforce HTTPS**
— Required for your site because you are using the default domain (`antzm.github.io`)

HTTPS provides a layer of encryption that prevents others from snooping on or tampering with traffic to your site. When HTTPS is enforced, your site will only be served over HTTPS. [Learn more.](#)

After a few minutes, your site will be up and running:



The content though is just the title of the repository and a short description.

So, let's add a theme to make it more stylish.

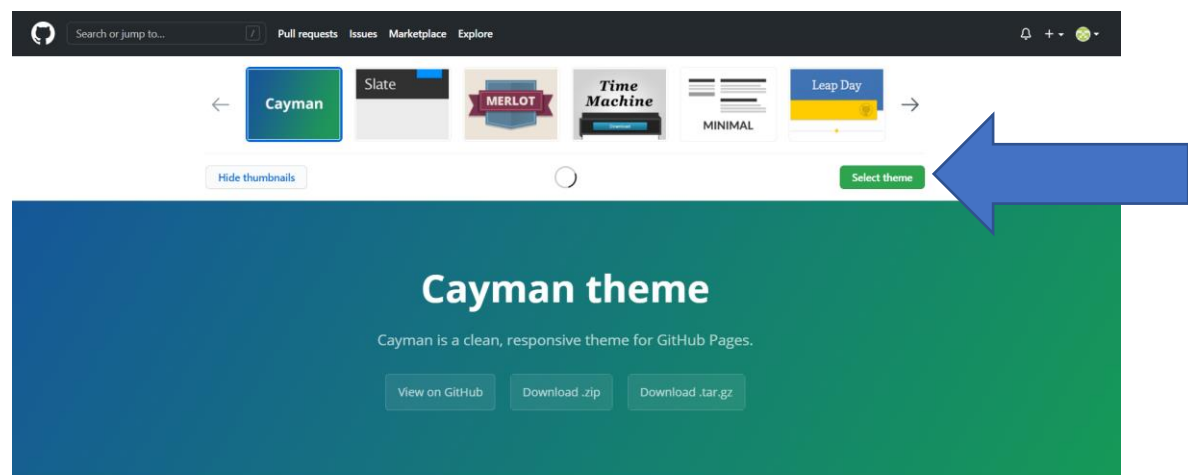
Under the "GitHub Pages" options, click "Choose a theme"

Theme Chooser

Select a theme to publish your site with a Jekyll theme. [Learn more.](#)

Choose a theme

I would suggest the "Cayman" theme which is a responsive theme, adapting nicely to tablets and smartphones.



And then, **Commit** the changes in your repository, as the theme will change the README.md file and will also add a new file (_config.yml) to your repository:



```

30
31 ### Jekyll Themes
32
33 Your Pages site will use the layout and styles from the Jekyll theme you have selected in your [repository settings](https://github.com/antzm/github-tutorial/settings). The name
of this theme is saved in the Jekyll '_config.yml' configuration file.
34
35 ### Support or Contact
36
37 Having trouble with Pages? Check out our [documentation](https://docs.github.com/categories/github-pages-basics/) or [contact support](https://support.github.com/contact) and
we'll help you sort it out.
38

```

Attach files by dragging & dropping, selecting or pasting them.



Commit changes

Create README.md

Add an optional extended description...

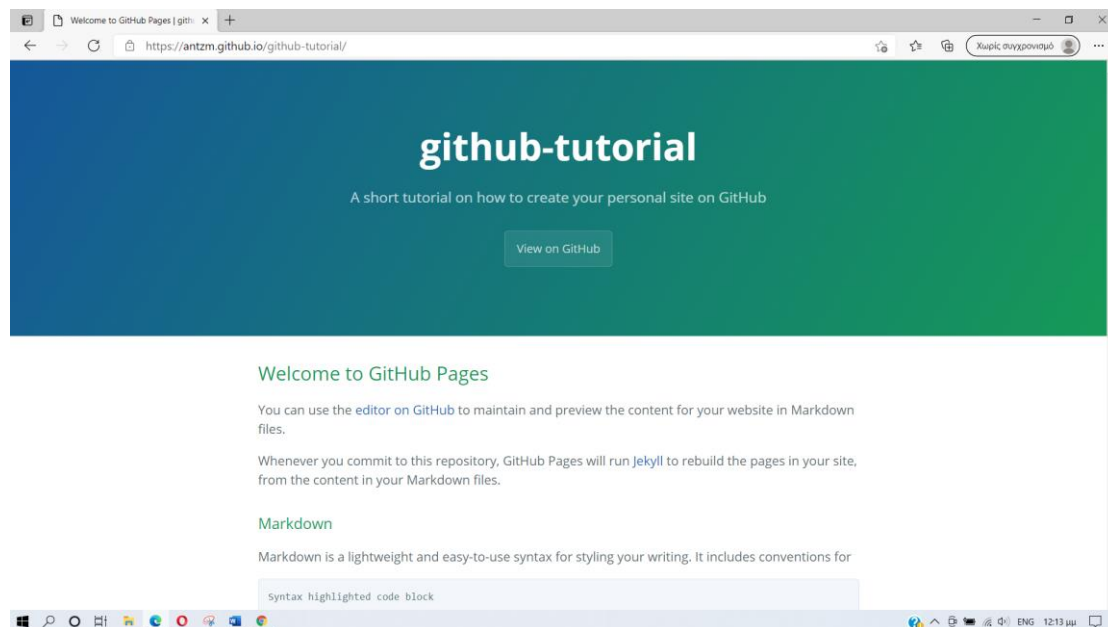
☒ Commit directly to the `main` branch.

☐ Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

Commit changes

Cancel

After a few minutes, your site will look like this:



A few notes:

If you only have a **README.md** file in your repository, your web page will be created from the contents of that file, using the theme you have selected.

If you also add an **index.md** file in your repository, this gets priority and the contents of the README.md will be ignored. Your page will be created based on the contents of your index.md file, using the theme you have already selected.

The highest priority though, goes to an **index.html** file in your repository, as your web page will be created according to that file, ignoring any .md files and any selected theme.

3. Updating your content:

Visiting your new site, the only thing you will see is either the name of your repository and the short description you wrote, or a longer text, in case you had selected a theme.

To include more content in your page, you can edit the `README.md` file in your repository using a language called Markdown.

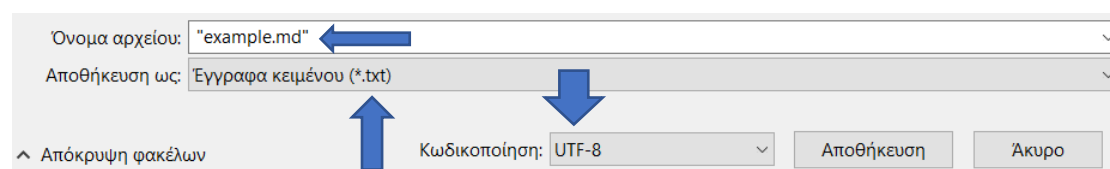
You can either edit the "README.md" file directly on GitHub, but as this is not so convenient, you could create a file on your computer, edit that file and then upload it to GitHub and it will automatically update your old README.md file.

(If you are using Git Bash, I would suggest to follow the above instructions on how to create your site and add a theme. After that, you can clone the repository and continue locally to update your content. Just remember, in case this is a newly created repository, to push your changes as "git push origin main" and not as "git push origin master")

Now, as this is the repository of your personal site, and not a project page, I would rather prefer to keep a README.md file with a short description in the repository. For this reason, I will create a new file named "**index.md**" with the content of the site.

The file can be created locally, on any text editor, using Markdown language and it needs to have the extension .md

Even Windows notepad can be used for this, provided that when saving the file, you will include the full name of the file inside quotes, like "example.md" otherwise your file might be saved as a text file with the name "example.md.txt". Additionally, on Windows notepad, make sure that your file is saved with "UTF-8" encoding.



These things though apply only to Windows notepad, as in most text editors you simply save the file as: example.md without using quotes, and the UTF-8 encoding is used by default.

This is an example of some Markdown written on Windows notepad:

```
example.md - Σημειωματάριο
Αρχείο Επεξεργασία Μορφή Προβολή Βοήθεια
# This is a Markdown example, starting with a header

This is a short thext followed by a bulleted list, {<-- TWO SPACES AT THE END OF THIS LINE !!!}
a table, a link, an image and a quote. {<-- TWO ENTERS AT THE END OF THIS PARAGRAPH}

## And here's an exaple of a bulleted list

* My item
* My item
  * Sub-item
  * Sub-item

## An example of a table

Column 1 | Columnn 2 | Column 3
-----|-----|-----
Text | Text | Text
More Text | More Text | More Text

## An example of a link

[Google Search](https://www.google.com)

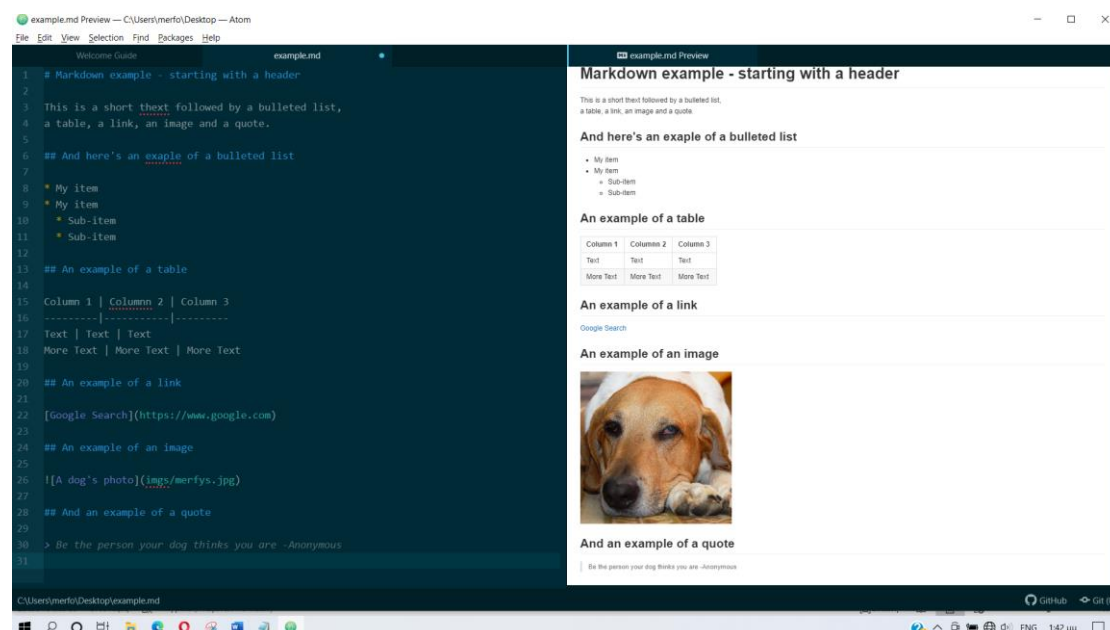
## An example of an image

![A dog's photo](imgs/merfys.jpg)

## And an example of a quote

> Be the person your dog thinks you are -Anonymous|
```

And here's the preview, using the Atom text editor:



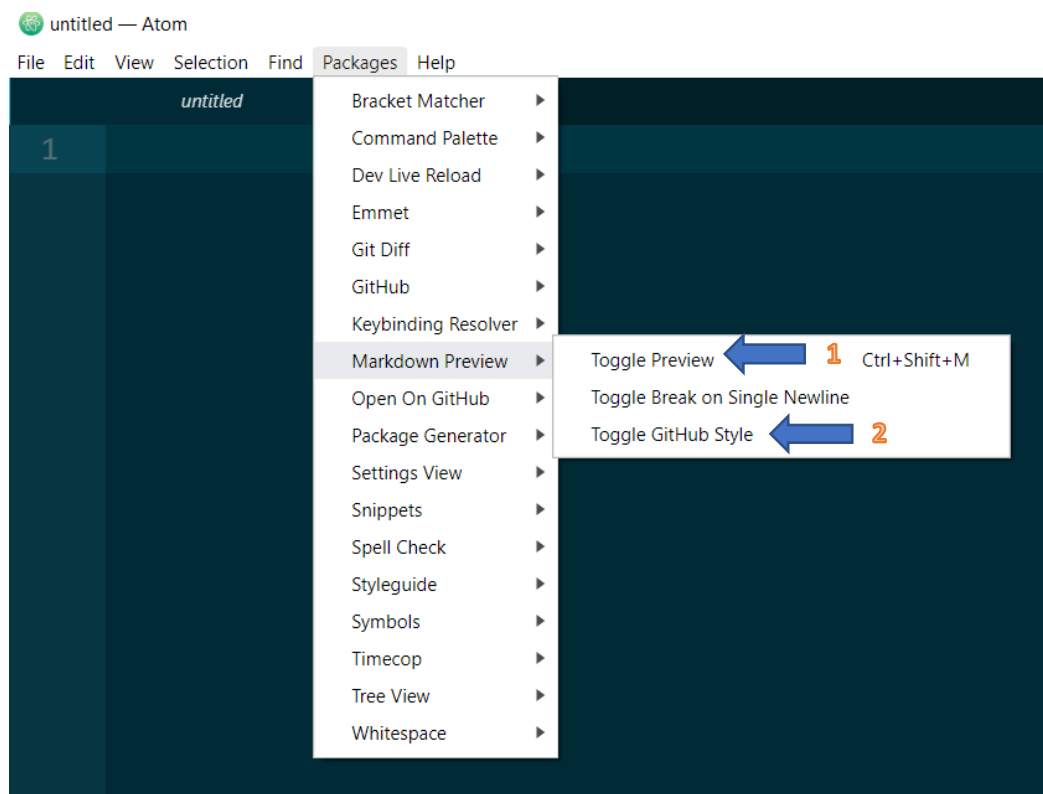
Note that I have used "Ctrl" + "-" on the preview window, so that the preview will appear on one screen.

In my opinion though, Markdown is so simple, that you don't even need to preview it. With a little experience, you will be able to visualize the preview by just looking at your Markdown code and then, you will see the result as soon as you upload your file to your GitHub repository.

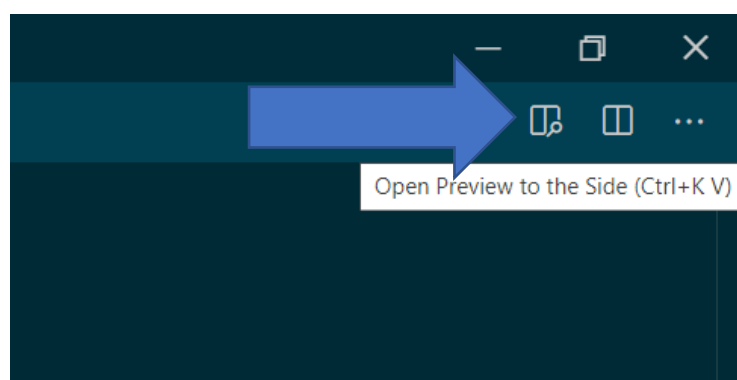
Should you wish though, there are various ways to preview your Markdown while writing.

One approach is to use a Jupyter Notebook to preview your Markdown text. Just create a Markdown cell, type your Markdown text, and then run that cell. Everything that appears in the Appendix (at the end of this document), works also in the Jupyter Notebooks.

The easiest way though to preview a Markdown file, is to use the text editor "Atom" <https://atom.io> which provides a preview window showing how your content will appear on GitHub.



In Visual Studio Code, <https://code.visualstudio.com> whenever you create or open an .md file, a new icon appears at the top right of the window. Clicking that icon, a preview window for the Markdown content appears.



Other text editors, like the Sublime Text editor, can also preview the Markdown files but you should first use the package control to install the proper package.

When writing a text paragraph in Markdown, pressing "Enter" at the end of a line, has no effect on the appearance of the text. That "Enter" is simply ignored, and the text appears in one line. e.g.:

One

Two

Three

Will appear as:

One Two Three

To change a line, you need to **leave two spaces** at the end of a line and then press "Enter".

To change a paragraph, you need to **press "Enter" twice** at the end of a line (in this case, there's no need to also leave two spaces)

The above applies mostly when you write text, but not when you write headers, bulleted lists, tables etc.

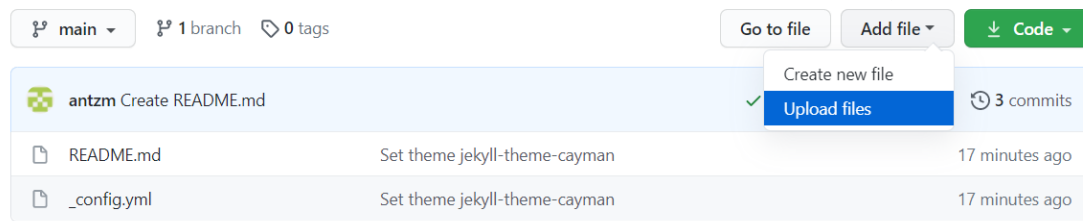
A small tip: If you are having problems with the line changes in a Markdown file, verify that your text editor does not trim the trailing white spaces during save (this could happen only if you changed the default settings). In such a case, the two spaces at the end of a line, will be automatically removed and the text in your .md file and your web page, will appear in one line.

After creating your **index.md** file on your computer, you can upload it to your GitHub repository, together with the folder that contains the images you are using. Those images should always be in a properly named folder (e.g., imgs) and not directly in the root directory.

So, the next step is to upload your index.md, the folder with your images, and then Commit those changes to your repository.

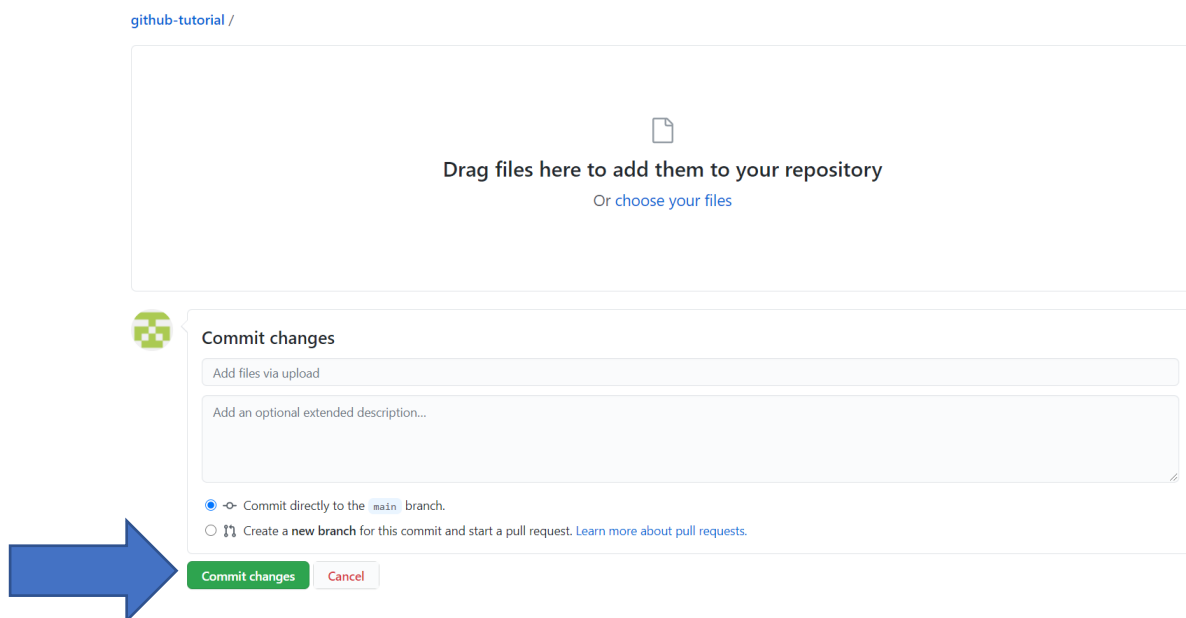
After a few minutes, your personal site will be automatically updated with your new content.

Select the option to upload your files:



Now, make your browser window a little smaller so to be able to drag and drop your file, and your folder with the images, into your repository.

Then, **Commit** your changes.



After a few minutes, your updated personal site will be ready. Just reload the page in your browser to see the changes.

Also, at the right side of your repository, you can see the following "Environments" option, which you can click and see the full history of all the updates you have made to your personal site:

Environments 1

github-pages Active

4. Creating additional project pages

The above process will create your personal site on GitHub, at this address:
<https://username.github.io>

After that, you can create as many **project pages** as you want. They are called project pages, as they are supposed to promote your projects, but you can actually create pages with the content you prefer.

Just go to one of your existing repositories or create a new one (no need to use a special name for your repositories anymore. The special repository name "username.github.io" is only used once, when creating your personal site).

Then, just follow again the steps in this tutorial to **1**. Create your personal site (although now it would be your project page, but the process is exactly the same, as only the name of the repository would be different). **2**. Add a theme, if you would like and **3**. Update the content of your web page.

A few clarifications:

GitHub provides only one site per user at **<https://username.github.io>**

The site can be created using a **README.md**, an **index.md** or an **index.html** file but ONLY ONE web page will be published at this URL address. Any additional .md or HTML files in that repository, cannot be deployed as web pages.

GitHub though, allows to use any of your repositories and create a web page which will be deployed at this URL address:

<https://user-name.github.io/repository-name>

Again, you can deploy ONLY ONE web page in every repository, which will be created from a **README.md**, an **index.md** or an **index.html** file.

When you create your web pages from a README.md or an index.md file, you can also choose a theme. This theme is independent for every repository. i.e., you can use any theme you prefer in a repository, regardless of the choices you have made in other repositories.

To link your personal site and your project pages, use their URL addresses.

Sharing, in an elegant way, your Jupyter Notebooks

The usual approach to share a Jupyter Notebook, is to upload it on a GitHub repository.

Here's though another approach:

Create a repository, which you will use to share your Jupyter Notebook.

On your Jupyter Notebook, include a Markdown cell (either at the beginning or at the end of your Notebook), where you will include a link to your personal site and a link to your repository.

Now, besides saving your Jupyter Notebook as an `.ipynb` file, download also your Jupyter Notebook as an HTML file.

Rename your HTML file as **index.html** and upload in your GitHub repository, both the `.ipynb` file and the `index.html` file.

Then, according to what has already been mentioned in this tutorial, deploy your `index.html` file to a web page.

This way, you will have a web page with your Jupyter Notebook, which loads almost instantly and is also responsive, and there would also be a link if someone would like to go to your repository and download the original Jupyter Notebook.

It's a simple, easy and elegant approach to share your Jupyter Notebooks with the world!

Just remember, in case you had used Markdown code to display images in your Notebook, to also upload in your repository the folder with those images.

If though your Notebook contains output graphs, those graphs are in SVG format and are included inside the HTML code, so there aren't any folders with images in this case.

Limitations on the GitHub pages usage:

Usage limits

GitHub Pages sites are subject to the following usage limits:

- GitHub Pages source repositories have a recommended limit of 1GB. For more information, see ["What is my disk quota?"](#)
- Published GitHub Pages sites may be no larger than 1 GB.
- GitHub Pages sites have a *soft* bandwidth limit of 100GB per month.
- GitHub Pages sites have a *soft* limit of 10 builds per hour.

If your site exceeds these usage quotas, we may not be able to serve your site, or you may receive a polite email from [GitHub Support](#) or [GitHub Premium Support](#) suggesting strategies for reducing your site's impact on our servers, including putting a third-party content distribution network (CDN) in front of your site, making use of other GitHub features such as releases, or moving to a different hosting service that might better fit your needs.

Source: <https://docs.github.com/en/github/working-with-github-pages/about-github-pages>

5. Upgrading to a professional looking site

To create a truly professional site, even without knowing HTML, CSS and JavaScript, you could use a template from <https://templated.co>

Obviously, you will still need to customize the HTML page but this can be easily done by learning only a few basic things on HTML, just enough to customize the titles, text, and links in the sample page.

<https://www.w3schools.com/default.asp>

As a small tip, you could search inside the HTML code for the text you would like to replace, and then type your own text.

Keep in mind though that those templates may have multiple HTML pages in them, but only one HTML file in your repository, named `index.html` can be deployed to a web page. Any additional HTML files in that repository, will NOT be deployed into web pages. Thus, if you would like to use a second HTML page, you will need to create a new repository with that HTML file, named as "`index.html`" and deploy it to a web page. Then use the URL of that page and link it with your landing HTML page.

This is because, each GitHub repository, deployed to a web page, provides you with only one URL address and thus, you can deploy only one file, from each repository, into a web page (always the file named `README.md` or `index.md` or `index.html`). Obviously, many HTML files can be included in a repository, but only the `index.html` file will be deployed to a web page.

The above is the standard behavior when deploying web pages from your root directory but there's also a more advanced option, to deploy your web pages from a folder inside your repository, which is the usual approach when publishing documentation or creating a blog.

<https://www.smashingmagazine.com/2014/08/build-blog-jekyll-github-pages/>

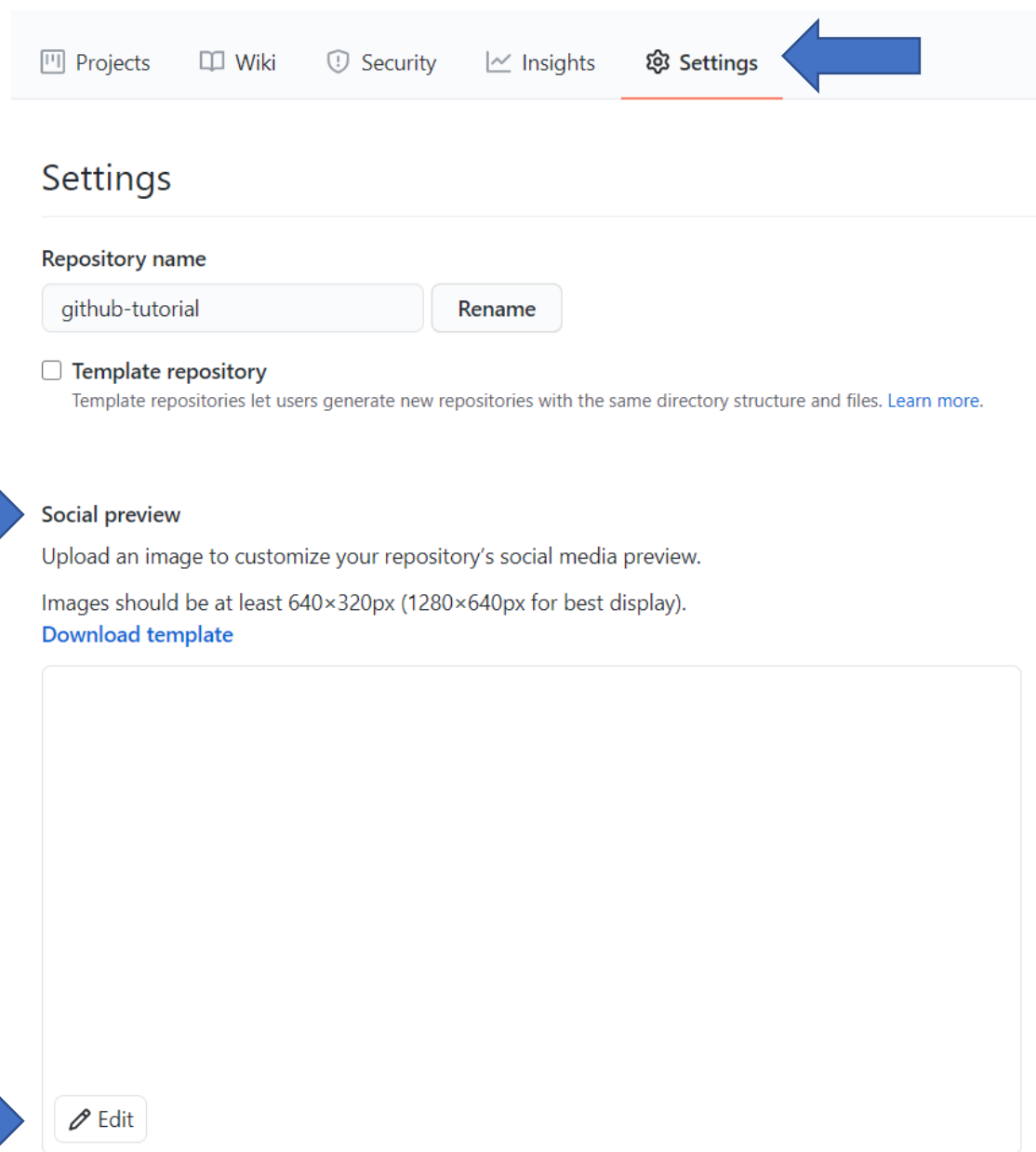
Our approach though, focuses only on creating a web page from the root directory, either using a **`README.md`** or **`index.md`** or **`index.html`** file. With this approach, we can create only one web page for each repository.

Social preview

Another detail, which makes a site look more professional, is to include a carefully created image as your social preview image.

This is the image that will appear when posting a link e.g., to a Slack channel.

To customize your social preview image for one of your web pages, you will need to go to the settings of that repository and choose the option "Social preview".



The screenshot shows the GitHub repository settings page. At the top, there is a navigation bar with links for Projects, Wiki, Security, Insights, and Settings. A blue arrow points to the Settings link. Below the navigation bar, the 'Settings' section is displayed. Under 'Repository name', there is a text input field containing 'github-tutorial' and a 'Rename' button. Below this, there is a checkbox for 'Template repository' which is unchecked, followed by a description and a 'Learn more' link. A large blue arrow points to the 'Social preview' section. This section has the title 'Social preview', a description 'Upload an image to customize your repository's social media preview.', and a note 'Images should be at least 640×320px (1280×640px for best display)'. There is a 'Download template' link. Below this is a large empty rectangular box for the social preview image. At the bottom left of this box, a blue arrow points to an 'Edit' button with a pencil icon.

Projects Wiki Security Insights **Settings**


Settings

Repository name

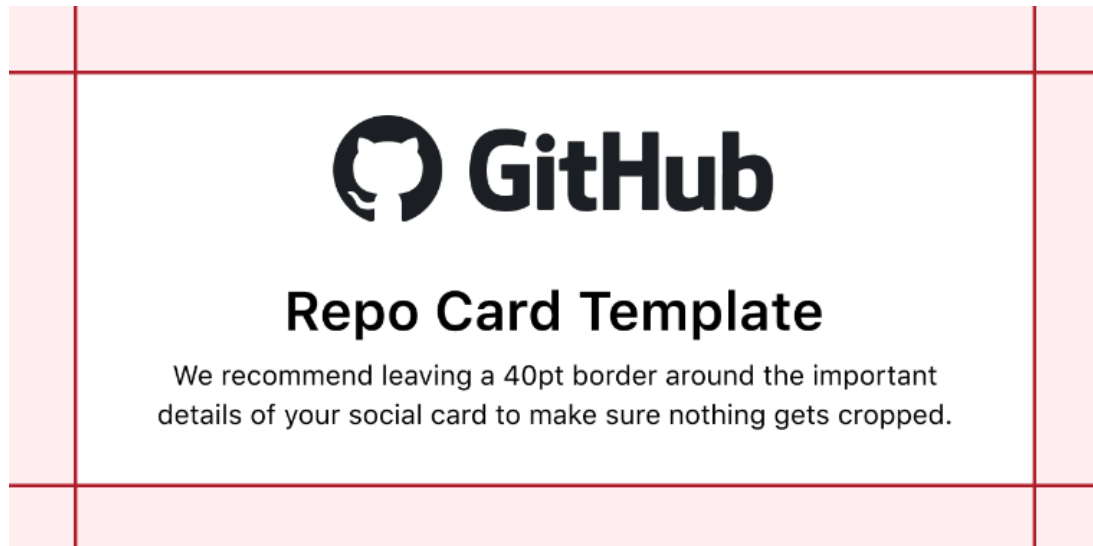
github-tutorial **Rename**

☐ **Template repository**
Template repositories let users generate new repositories with the same directory structure and files. [Learn more.](#)

Social preview
Upload an image to customize your repository's social media preview.
Images should be at least 640×320px (1280×640px for best display).
[Download template](#)



GitHub provides also the following template (Social preview -> Download template) as guidance:



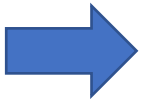
Remember though that if you would like to enable the social preview for one of your web pages, that it would only be enabled for that particular page.

To enable the social preview for another web page, you will need to follow the same steps, as the repositories are independent and the settings in one repository don't affect the other repositories.

6. Using your personal Domain Name

There's an option "Custom domain" in the repository settings to allow you to use your personal domain name, like myname.com, with your GitHub site.

This way, the URL myname.com will be used and appear on the browser but your page will still be hosted and loaded from your personal GitHub site at **username.github.io**



Custom domain

Custom domains allow you to serve your site from a domain other than antzm.github.io. [Learn more.](#)

The following link provides more information on this functionality:

<https://docs.github.com/en/github/working-with-github-pages/configuring-a-custom-domain-for-your-github-pages-site>

As for acquiring a personal domain name, there are various options but if you have an AWS account, you could also use the service named "Route 53":

The screenshot shows the AWS Route 53 console. The top section is the 'Route 53 Dashboard' with four main cards: 'DNS management' (Create hosted zone), 'Traffic management' (Create policy), 'Availability monitoring' (Create health check), and 'Domain registration' (Register domain). Below this is the 'Choose a domain name' page. It features a search bar with 'my-personal-domain' and a dropdown for '.com - \$12.00'. A 'Check' button is next to it. Below the search bar, it shows the availability for 'my-personal-domain.com' as 'Available' for \$12.00 per year. There is also a table of 'Related domain suggestions' including my-personal-domain.io, .mobi, .net, and .org, all available for various prices.

7. APPENDIX: GitHub Markdown Language

Markdown is not a standardized language, which means that various sites use their own version of Markdown. Most things are similar, but there are also some differences. GitHub uses the version called GitHub Markdown which can be used in the README.md and the index.md files, but when these files are deployed to web pages, there are some certain features of the GitHub Markdown language that won't work on a web page.

For this reason, what is mentioned below refers mainly to the GitHub Markdown features that have been tested and work on a web page, (except the part "Navigating inside a Markdown file").

Also, all the Markdown features in this appendix, have been tested and work fine in the Markdown cells of a **Jupyter Notebook** (either locally, or on the Google Collab, or on the Amazon SageMaker)

In the following pages you will find all the important things you need to know about the Markdown language.

(The markdown syntax is shown first, and just below there is a preview)

Blue letters: Descriptions and Explanations

Red letters: Markdown code

Black letters: Preview of the Markdown code

Headers

```
# This is a level 1 heading
## This is a level 2 heading
### This is a level 3 heading
#### this is a level 4 heading
##### this is a level 5 heading
##### this is a level 6 heading
```

This is a level 1 heading

This is a level 2 heading

This is a level 3 heading

this is a level 4 heading

this is a level 5 heading

this is a level 6 heading

Font styles

```
**this thext is bold**
_ this text is italized_
*this text is also italized*
```

this thext is bold

this text is italized

this text is also italized

```
**combination of _italized text_ inside bold text**
_combination of **bold text** inside italized text_
```

combination of *italized text* inside bold text

combination of ***bold text*** inside italized text

Paragraphs and Text

To start a new paragraph, leave an empty line between the previous and the next paragraph (i.e. press "Enter" twice).

First paragraph

Second paragraph

First paragraph

Second paragraph

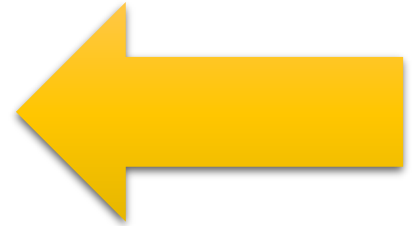
To start a new line, leave two spaces at the end of a line.

First line <- two spaces at the end of the line

Second line

First line

Second line



Note:

One enter at the end of a line, does not change the line. Thus, if you write text and simply press enter at the end of each line, that text will appear as a single line.

To place the text on a new line, leave two spaces at the end of a line and then press enter.

To start a new paragraph, just press enter twice.

Some special characters (e.g. a *) may not appear correctly in your README.md or in your web page.

In such situations, you should place a \ character before your symbol, like *

Tables:

Note: There’s no need to align the | on the first and second row, other than the visual appearance of the code. The functionality would be the same, even if you write ---|---|--- in the second row.

```
header one | header two | header three
-----|-----|-----
first item in #1 | first item in #2 | first item in # 3
second item in # 1 | second item in # 2 | second item in # 3
third item in # 1 | third item in # 2 | third item in # 3
```

header one	header two	header three
first item in #1	first item in #2	first item in # 3
second item in # 1	second item in # 2	second item in # 3
third item in # 1	third item in # 2	third item in # 3

```
items align left | items align center | items align right
:--- | :---: | ---:
item 1a | item 2a | item 3a
item 1b | item 2b | item 3b
item 1c | item 2c | item 3c
```

items align left	items align center	items align right
item 1a	item 2a	item 3a
item 1b	item 2b	item 3b
item 1c	item 2c	item 3c



Unordered Lists:

```
* first item  
* second item  
* third item
```

- first item
- second item
- third item

```
* first item  
* second item  
  * second item a  
  * second item b  
  * second item c  
* third item  
  * third item a  
  * third item b  
  * third item c
```

- first item
 - second item
 - second item a
 - second item b
 - second item c
 - third item
 - third item a
 - third item b
 - third item c
-

Ordered Lists:

1. first item
2. second item
3. third item

1. first item
2. second item
3. third item

1. first item
2. second item
 1. second item a
 2. second item b
 3. second item c
3. third item
 1. third item a
 2. third item b
 3. third item c

1. first item
 2. second item
 - i. second item a
 - ii. second item b
 - iii. second item c
 3. third item
 - i. third item a
 - ii. third item b
 - iii. third item c
-

Task Lists:

* [] unchecked item 1
* [x] checked item 2
* [x] checked item 3

- ☐ unchecked item 1
- ☒ checked item 2
- ☒ checked item 3

* [] unchecked item 1
 * [] subitem 1a
 * [] subitem 1b
* [x] checked item 2
* [x] checked item 3
 * [x] subitem 3a

- ☐ unchecked item 1
 - ☐ subitem 1a
 - ☐ subitem 1b
- ☒ checked item 2
- ☒ checked item 3
 - ☒ subitem 3a

Links:

[GitHub](http://github.com)

[GitHub](http://github.com)

Code:

```
this is an `inline code` example
```

```
this is an inline code example
```

```
...
```

```
This is a longer
```

```
code example
```

```
...
```

```
This is a longer
```

```
code example
```

Code highlighting:

JavaScript:

```
```javascript
for (i=0; i<10; i++) {
 console.log(i);
}
```

for (i=0; i<10; i++) {
  console.log(i);
}
```

HTML:

```
```html
<section class="intro">
 <h1 class="intro-heading">intro heading</h1>
 <p class="intro-text">intro text</p>
</section>
```

<section class="intro">
  <h1 class="intro-heading">intro heading</h1>
  <p class="intro-text">intro text</p>
</section>
```

CSS:

```
```css
.intro {
 margin: 0;
 padding: 0;
 color: #1f3d7a;
 font: Verdana, Arial, sans-serif;
}

.intro {
 margin: 0;
 padding: 0;
 color: #1f3d7a;
 font: Verdana, Arial, sans-serif;
}
```

## Python *(using the Cayman theme)*:

```
```python
# transpose a 2D array
array = [['a', 'b'], ['c', 'd'], ['e', 'f']]
transposed = zip(*array)
print(transposed) # [('a', 'c', 'e'), ('b', 'd', 'f')]
```
```

```
transpose a 2D array
array = [['a', 'b'], ['c', 'd'], ['e', 'f']]
transposed = zip(*array)
print(transposed) # [('a', 'c', 'e'), ('b', 'd', 'f')]
```

## Quote:

>“We change the world not by what we say or do, but as a consequence of what we have become.”

– David R. Hawkins

“We change the world not by what we say or do, but as a consequence of what we have become.” — David R. Hawkins

---

## Images:

![Merfys, a brown and white dog looking at the camera](imgs/merfys.jpg)



**IMPORTANT:** In the path, use only the folder and the name of the image. i.e., **![My Photo](imgs/myphoto.jpg)** BUT don't include the name of the repository. If needed, in the web pages that will be created, the name of the repository is added automatically to the path, so that the page will be able to locate the images.

Also, when using images, place them in a folder named e.g., images or imgs and not directly in your root directory.

If your images don't appear on your web page, right click on the original images and check if their extension is written as .jpg .png .gif or maybe as .JPG .PNG .GIF and then use small or capital letters for the extension, depending on your image.

---

## Clickable Images

First, write separately the code for the link and the code for the image:

Link:

```
[Google's PacMan Game](https://www.google.com/logos/2010/pacman10-i.html)
```

Image:

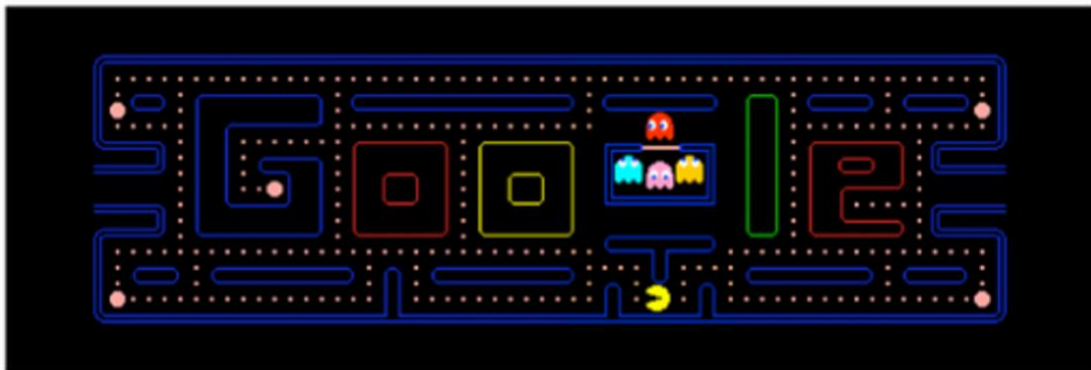
```
![Google PacManLogo](imgs/logo.jpg)
```

Now, use the code for the link BUT replace whatever is inside the [...] with the full code of the image:

i.e., It's like we are replacing the text that describes the link, with an image that describes the link.

Clickable Image:

```
![Google PacManLogo](imgs/logo.jpg) (https://www.google.com/logos/2010/pacman10-i.html)
```



---

## Insert a line

---

---

## Navigating inside a Markdown file or inside a Jupyter Notebook.

In long Markdown files and long Jupyter Notebooks, you can use the following approach to create a table of contents and navigate the document.

### Important:

This approach can be used on a README.md but the links will only function on the .md file. **The links will NOT function if you deploy the file into a web page.** (In such a case, you could create an index.md without the links, and deploy your page from the index.md, while keeping also your README.md with the links).

This approach can also be used inside a Jupyter Notebook to help you navigate. The nice thing though, is that **when you save your Jupyter Notebook as an HTML file, these links will still function** and they will help you navigate the HTML file.



Let's assume that we have a README.md with the following headers, at various parts of the document:

```
Probability and Statistics - Main Concepts
```

```
Introduction
```

```
...
```

```
Probability distribution
```

```
...
```

```
Permutations
```

```
...
```

```
Combinations
```

```
...
```

```
Regression
```

```
...
```

```
Linear regression
```

```
...
```

```
Multiple linear regression
```

Let's create now a table of contents to help us navigate inside our Markdown file:

```
Contents {two stars, without spaces, for bold, and
two spaces and "Enter" at the end of the line, for
changing line}

* [Introduction](#Introduction)
* [Probability distribution](#Probability-distribution)
* [Permutations](#Permutations)
* [Combinations](#Combinations)
* [Regression](#Regression)
* [Linear regression](#Linear-regression)
* [Multiple linear regression](#Multiple-linear-regression)
```

Steps:

The \* (and a space) in front of each line, is for creating a bulleted list

Inside the [...] we write the text what will appear on the screen

Inside the (...) we write the link to a certain header in the Markdown file. We place only one # in front of each link (regardless of if it's a header 1 or header 2 etc.) and then, we write the header exactly as it appears, but we replace all the spaces with - as there shouldn't be any spaces inside these links.

We follow the exact same approach if we would like to place a link, at certain parts of the Markdown file, for navigating back to the top of the page, i.e.:

```
[Top of Page](#Probability-and-Statistics---Main-Concepts)
```

## A few tips on the Markdown language

Below, are a few tips on how to use more efficiently the Markdown language.

### Creating Markdown Tables with Text

In the first line, we write the headers separated by |

In the second line we write at least three - - - or more, separated by a | and this is done so many times, as the number of columns in our table.

Then, in every row we write our text and we separate the columns using the | symbol.

Also, there should be one empty line above, and one empty line below the table.

```
Column 1	Column 2	Column 3
Lorem Ipsum | Lorem Ipsum | Lorem Ipsum
Lorem Ipsum | Lorem Ipsum | Lorem Ipsum
Lorem Ipsum | Lorem Ipsum | Lorem Ipsum
```

| Column 1    | Column 2    | Column 3    |
|-------------|-------------|-------------|
| Lorem Ipsum | Lorem Ipsum | Lorem Ipsum |
| Lorem Ipsum | Lorem Ipsum | Lorem Ipsum |
| Lorem Ipsum | Lorem Ipsum | Lorem Ipsum |

## Creating Markdown Tables with Links

To create a table with links, we simply place the links inside the table like this:

```
Column 1 | Column 2
```

```
---|---
```

```
[Google Search](https://www.google.com) | [Google Search](https://www.google.com)
```

```
[Google Search](https://www.google.com) | [Google Search](https://www.google.com)
```

| Column 1                                           | Column 2                                           |
|----------------------------------------------------|----------------------------------------------------|
| <a href="https://www.google.com">Google Search</a> | <a href="https://www.google.com">Google Search</a> |
| <a href="https://www.google.com">Google Search</a> | <a href="https://www.google.com">Google Search</a> |

# Creating a table with images

This approach can be used when we would like to place images next to each other and we simply place the code for the images inside the table cells.

```
Column 1	Column 2
![Our story](imgs/story.jpg) | ![Our story](imgs/story.jpg)
![Our story](imgs/story.jpg) | ![Our story](imgs/story.jpg)
```

| Column 1                                                                            | Column 2                                                                             |
|-------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|
|   |   |
|  |  |

## Creating a table with clickable images

This approach can be used when we would like to link, for example, to our project pages using images of our projects.

In this case, we place the code for the clickable images inside the cells.

Column 1 | Column 2

---|---

[![Our story](imgs/story.jpg)](https://www.google.com) | [![Our story](imgs/story.jpg)](https://www.google.com)

[![Our story](imgs/story.jpg)](https://www.google.com) | [![Our story](imgs/story.jpg)](https://www.google.com)

| Column 1                                                                            | Column 2                                                                             |
|-------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|
|   |   |
|  |  |

There are various mistakes in this tutorial, as I wasn't able to review it thoroughly, but hopefully, at a later time, you will probably find an updated version on my Google Drive: <https://drive.google.com/drive/folders/11nJnp99N2iStfBb8IWuSVu5heBfkOpyS?usp=sharing>

Best wishes,  
Antonis

PS: If you have noticed a mistake, please let me know by using this anonymous form: [https://docs.google.com/forms/d/e/1FAIpQLSfH6lTIq1Tclt9wCIfOu4M6SXZlwWxi5eNBWZlcmFM4rKEfxA/viewform?usp=sf\\_link](https://docs.google.com/forms/d/e/1FAIpQLSfH6lTIq1Tclt9wCIfOu4M6SXZlwWxi5eNBWZlcmFM4rKEfxA/viewform?usp=sf_link)