

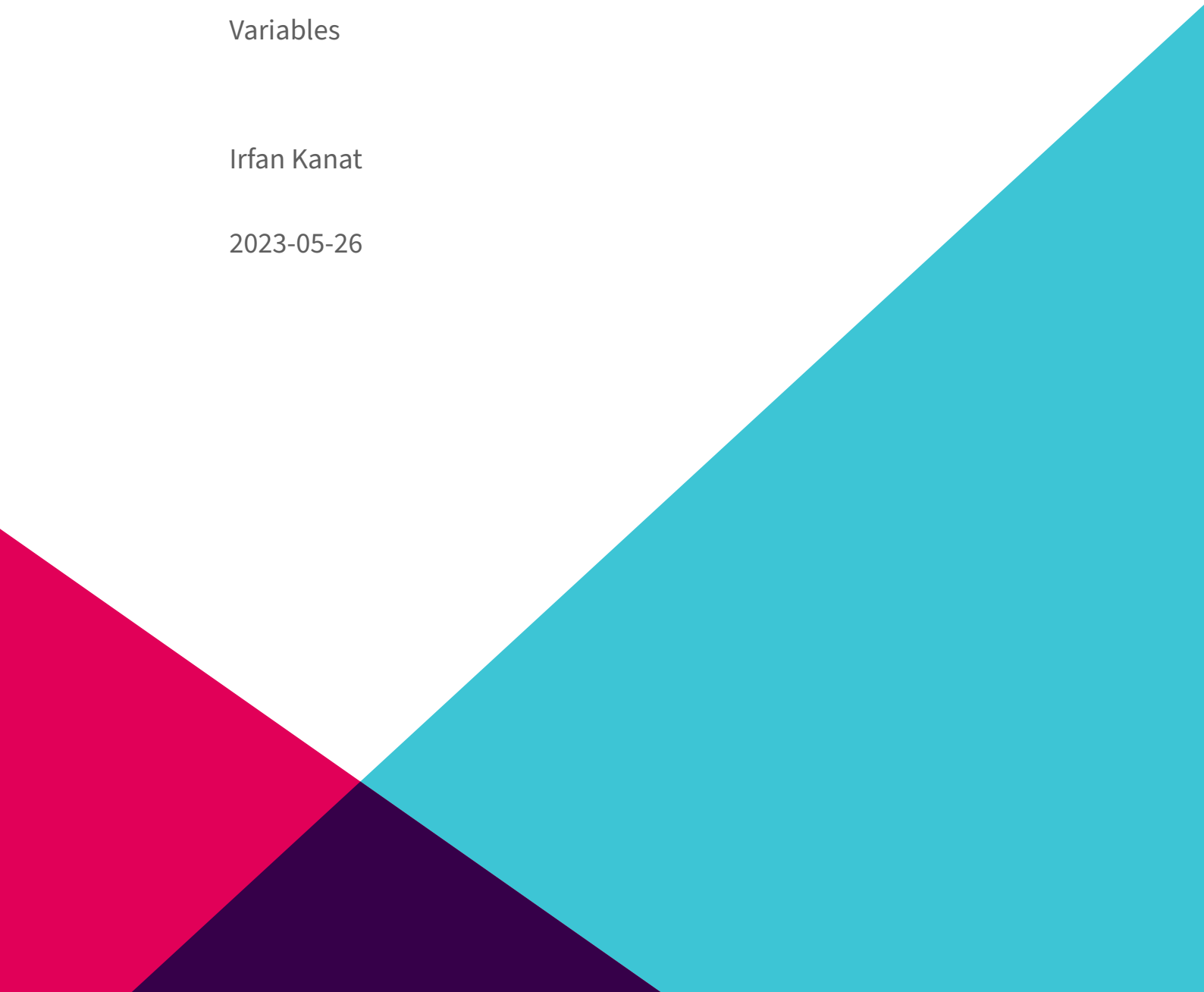
---

# Scripting for Kiddies

#! Where It All Starts

Irfan Kanat

2023-05-26



## **#! Where It All Starts**

The old hash-bang, the pound-bang, hash-pling, or as I like calling it the whole shebang is how scripts get started in Unix like systems.

This is how you tell the operating system the interpreter to use for the rest of the file. Since we are writing bash scripts it will look like this:

```
#! /bin/bash
```

If we were to write a python script it would look like

```
#! /usr/bin/env python3
```

A powershell script would start with

```
#! /usr/bin/pwsh
```

And so on...

## **Writing Your First Script**

You can write your first script in any text editor. Nothing fancy is required.

If you like something with a GUI, Kali Linux comes with gedit. If you prefer to write your scripts in the command line, you can use nano.

Open a text editor and type in the following:

```
#! /bin/bash
# THIS IS A COMMENT AND WON'T BE EXECUTED
echo "Hello World"
```

The commands you enter after the first line will get executed in order. Anything that starts with a # sign will be treated as a comment and won't be executed. So the second line of this script will also be skipped.

As you can guess, there is quite a bit you can do with this. You can archive files, encrypt them, move them around, record a log of these activities.

Save the file. You can name it anything you like. I recommend first.sh . The convention for shell scripts is to have sh as the extension.

## Executing Your First Script

As you can guess, you will need to use the command line to execute that script. So launch a terminal window and let's get to work.

Unless you want to invoke the interpreter every time you use the script, you need to grant execute privilege to the file.

We will use `chmod` for this purpose.

```
chmod u+x first.sh
```

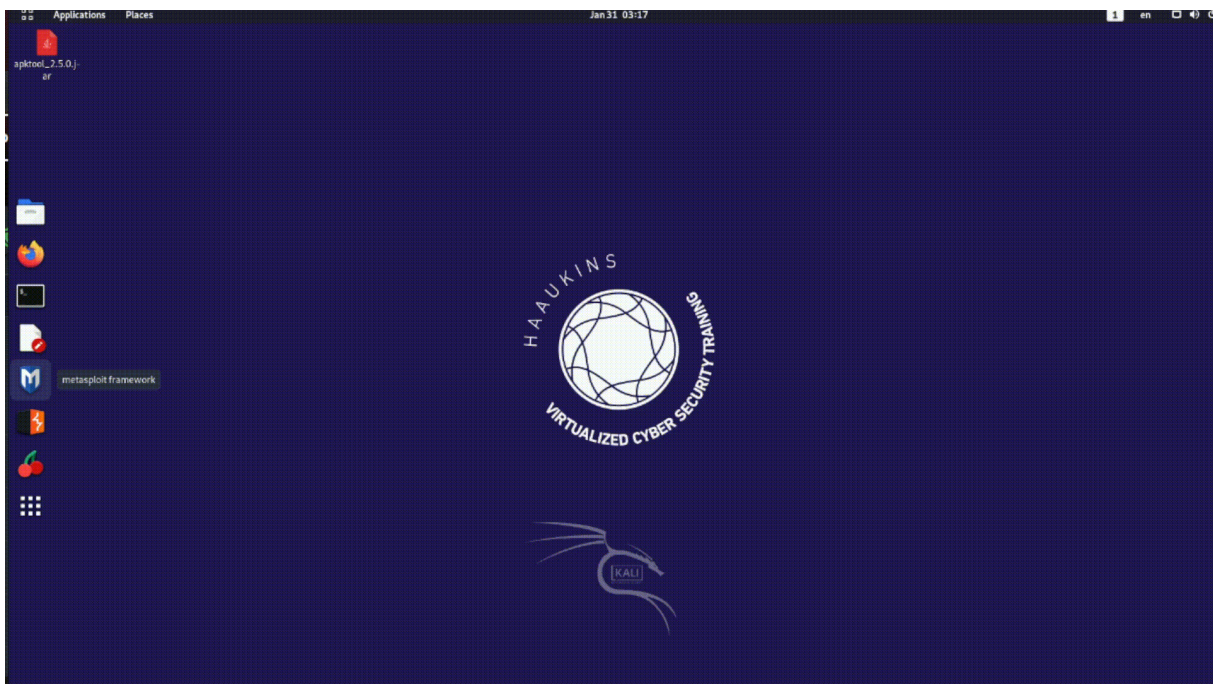
This adds execution privilege to the user of the file.

Then you can execute your script by calling it with either an absolute or a relative path. Here since we are in the same folder, I will use the relative path.

```
./first.sh
```

The dot signifies the current folder the terminal session operates from (present working directory). So the whole command means: run the `first.sh` file in this folder.

If you want to be able to call your scripts like any other file, you may want to put them in a folder that is in your `PATH` variable (`echo $PATH`). That has some security implications, and you may want to hold off on that for the moment.



**Expand on It**

Here is a short exercise for you. Below are the requirements, write a script that accomplishes what is outlined below:

- The script should be named `second.sh`
- When executed the script should display the message “Leave the ransom under the tree in Frederiks Have.”
- The script should append the current date and time into a file named `second.log`.



This work is licensed under a [Creative Commons Attribution 4.0 International License](https://creativecommons.org/licenses/by/4.0/).