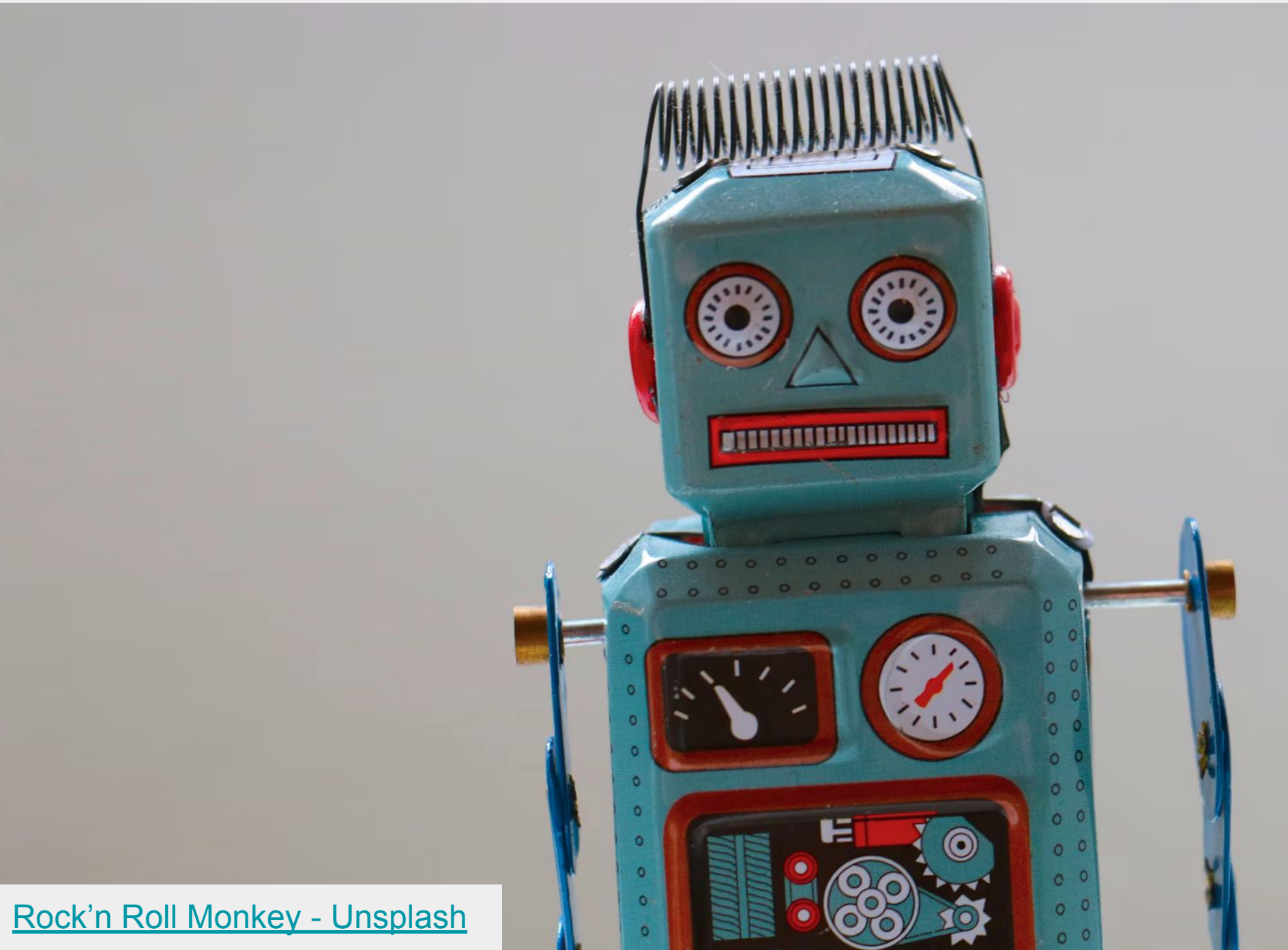
bash scripting – intro

Adapted from materials by Dr. Carrier



Rock’n Roll Monkey - Unsplash

So many questions!

So many questions!

Recall: What is bash?

So many questions!

Recall: What is bash?

The shell we are using in our terminal, takes input and runs commands.

So many questions!

Recall: What is bash?

The shell we are using in our terminal, takes input and runs commands.

What is a bash script?

So many questions!

Recall: What is bash?

The shell we are using in our terminal, takes input and runs commands.

What is a bash script?

A list of commands to execute with bash

So many questions! (continued)

Why would we use a bash script?

So many questions! (continued)

Why would we use a bash script?

- Automating tasks – makes things simple

So many questions! (continued)

Why would we use a bash script?

- Automating tasks – makes things simple
- We don’t have to remember individual commands

So many questions! (continued)

Why would we use a bash script?

- Automating tasks – makes things simple
- We don’t have to remember individual commands
- Faster than typing in individual commands

So many questions! (continued)

Why would we use a bash script?

- Automating tasks – makes things simple
- We don’t have to remember individual commands
- Faster than typing in individual commands

Use cases?

So many questions! (continued)

Why would we use a bash script?

- Automating tasks – makes things simple
- We don’t have to remember individual commands
- Faster than typing in individual commands

Use cases?

- Sys admin

So many questions! (continued)

Why would we use a bash script?

- Automating tasks – makes things simple
- We don’t have to remember individual commands
- Faster than typing in individual commands

Use cases?

- Sys admin
- Initializing directories / projects

So many questions! (continued)

Why would we use a bash script?

- Automating tasks – makes things simple
- We don’t have to remember individual commands
- Faster than typing in individual commands

Use cases?

- Sys admin
- Initializing directories / projects

Does anyone actually use these?

So many questions! (continued)

Why would we use a bash script?

- Automating tasks – makes things simple
- We don’t have to remember individual commands
- Faster than typing in individual commands

Use cases?

- Sys admin
- Initializing directories / projects

Does anyone actually use these?

Quick note

Bash script != batch script

Batch scripts are a windows thing (similar idea)

Creating and running a bash script

Creating and running a bash script

These are just text files. Generally end in `.sh`

Creating and running a bash script

These are just text files. Generally end in `.sh`

Create, modify, and save your file in a text editor

Creating and running a bash script

These are just text files. Generally end in `.sh`

Create, modify, and save your file in a text editor

- Each line is a command

Creating and running a bash script

These are just text files. Generally end in `.sh`

Create, modify, and save your file in a text editor

- Each line is a command

We need execute permission on this file! How?

Creating and running a bash script

These are just text files. Generally end in `.sh`

Create, modify, and save your file in a text editor

Each line is a command

We need execute permission on this file! How?

```
chmod u+x file.sh
```

Creating and running a bash script

These are just text files. Generally end in `.sh`

Create, modify, and save your file in a text editor

Each line is a command

We need execute permission on this file! How?

```
chmod u+x file.sh
```

To run:

```
./file.sh
```


Shebang!

AKA pound-bang, hashbang

AKA pound-bang, hashbang

```
#! something goes here
```

AKA pound-bang, hashbang

```
#! something goes here
```

This is the first line of our script

Tells system what program to use to run script

AKA pound-bang, hashbang

```
#! something goes here
```

This is the first line of our script

Tells system what program to use to run script

Why would we do this?

AKA pound-bang, hashbang

```
#! something goes here
```

This is the first line of our script

Tells system what program to use to run script

Why would we do this?

- Portability - User just executes file, doesn't need to know what other command to use
- Running bash scripts from other shells

Common examples for bash:

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

```
#! /usr/bin/env bash
```

Common examples for bash:

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

```
#! /usr/bin/env bash
```

What is the difference?

- env uses whatever version of executable comes first in the PATH
- Allows for users to tweak behavior

Common examples for bash:

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

```
#! /usr/bin/env bash
```

What is the difference?

- env uses whatever version of executable comes first in the PATH
- Allows for users to tweak behavior

Not just limited to bash!

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

A note for Mac users

You probably want to update bash

```
brew install bash
```

You'll have a different path from before

Example: `#! /usr/local/bin/bash`

Leaving notes

Even in simple scripts, it can be useful to leave comments

Comments here start at a # and continue to the end of the line

- Shebang is an exception