Variables

We can declare a variable with `myVariable=x` where `x` can be whatever, including the output of a command with `$(command)`

Metacharacters

1. `*` any sequence of characteres (empty or not), the string can't start with a point.
2. `?` any character (only one)
3. `[charlist]` matchs only one element of the list, this can be a sequence of elements or an intervall of numbers / letters.

Expressions

1. Command-substitution

Bash uses the notation `$(command)` to execute a `command` or a `program` and replace it with its output, variables inside the parentheses must be de-referenced (i.e. use the `$`)

2. Arithmetic calculations

The same as command but with `$((expression))`, but in this case we don't need to use the `$` sign to dereference variables.

3. Number of letters

We can use `${#var}` to get the number of letters of the string stored in `var`

4. Variable name

In case that the boundaries of the name of the variable (or to dereference an array) we can use `${var}` to limit it.

5. Shebang

By definition only `#!` but is known by the expression `#! /bin/bash`, is a magic number that allows the shell to identify it as an script and no as a binary.

Special Variables

1. `$0` : name of the script.
2. `$1, $..., $n` : holds the n-th argument of the script.
3. `$#` holds the number of parameters passed to the script (name not included).
4. `$$` : PID of the current shell.
5. `$!` : PID of the last process started.
6. `$IFS` holds the Internal Field Separator, what bash uses to separate elements into words.

Expansions

1. Braces expansion

Bash allows for bracet expansions and prefixing using `prefix{e1,e2}` also allows for nested braces like `prefix{e1,e2{a,b}}`

2. Series

We can use `{x..y}` to list all the numbers and single letters from `x` to `y` included. If we want to do jumps between them we can specifing `z` like in `{x..y..z}`

Example outputs

```
 $ echo "file_{1,2}"
file_1 file_2

$ echo "file_{1,2{a,b}}
file_1 file_2a file_2b
```

Separators

1. The `;` and the `newline` allows to separate between commands.

Commands

- `xargs` this builts a list of parameters and executes the command that is its parament itself, designed to be used with pipelines.
- `test` used to check a boolean expression, useful in cordination with a control structure.
- `unset` is used to unset a environmental variable.

Conditionals

We can determinate if a file (`-f`) or directory (`-d`) exists using `if`

```
 if [-f ./path/to/file]; then <commands> fi
if [-d ./directory]; then <commands> fi
```