getopts

buen tutorial: https://wiki.bash-hackers.org/howto/getopts_tutorial

uso

- la cadena que acompaña a la instrucción getopts indica las opciones disponibles
- `getopts OPTSTRING VARNAME [ARGS…]`
    - OPTSTRING:
        - h: comprueba la opción **-h sin parámetros**. Da error en opciones no soportadas
        - `h:`: comprueba la opción **-h con parámetros**. Da error en opciones no soportadas
        - `abc`: comprueba las opciones **-a**, **-b**, **-c**. Da error en opciones no soportadas
        - `:abc`: comprueba las opciones **-a**, **-b**, **-c**. NO da error en opciones no soportadas
        - resumiendo:
            - `:` al principio de la cadena de OPTSTRING silencia opciones no soportadas
            - `:` detrás de una opción hace que espere parámetro
    - al procesar VARNAME (en un case):
        - `\?` : es usado para las opciones no válidas
        - `:` : es usado para las opciones que requieren parámetros pero no los llevan
        - `a|b|c`: procesa en la misma instrucción las 3 opciones (si así lo necesiamos)
        - `h|*` : el * se usa como comodín para opciones que no se han definido (sin sentido con el uso de **?**)

/via: https://stackoverflow.com/questions/16483119/an-example-of-how-to-use-getopts-in-bash

ejemplos

- https://stackoverflow.com/questions/16483119/an-example-of-how-to-use-getopts-in-bash

primer ejemplo

```
#!/bin/bash

usage() { echo "Usage: $0 [-s <45|90>] [-p <string>]" 1>&2; exit 1; }

while getopts ":s:p:" o; do
    case "${o}" in
        s)
            s=${OPTARG}
            ((s == 45 || s == 90)) || usage
            ;;
        p)
            p=${OPTARG}
            ;;
        *)
            usage
            ;;
    esac
done
shift $((OPTIND-1))
```

```
if [ -z "${s}" ] || [ -z "${p}" ]; then
    usage
fi

echo "s = ${s}"
echo "p = ${p}"
```

```
#!/bin/bash

while getopts ":a:" opt; do
  case $opt in
    a)
      echo "-a was triggered, Parameter: $OPTARG" >&2
      ;;
    \?)
      echo "Invalid option: -$OPTARG" >&2
      exit 1
      ;;
    :)
      echo "Option -$OPTARG requires an argument." >&2
      exit 1
      ;;
  esac
done
```