Fonaments de Sistemes Operatius

Pràctica 1

Programació en Bash i Python d'un generador de configuracions

Laboratori 4

Manuel Ruíz

Tobias Ton Van Den Engel

Xavier Corbacho

Roger Lluch

18/03/2022

Índex

Introducció al problema

En aquesta pràctica es planteja el disseny d'un script, tant en *bash* com en *python*, que crea un fitxer de configuració de cara a la pràctica 2: El joc de la pilota.

Aquest fitxer serà generat a partir dels següents paràmetres d'entrada: les dimensions del camp (posicions en *x* i en *y*), la mida de la porteria, la posició de la paleta i la seva dimensió (posició en *x* i *y* i mida en una sola variable), i les pilotes (posició i velocitat en *x* i *y*). Aquest últim paràmetre es pot rebre fins a 9 cops com a màxim. El fitxer es crearà de 0 o es modificaran els seus camps depenent de si el fitxer es troba creat previ a l'execució de l'*script*.

Decisions de Disseny

El principal objectiu d'aquesta pràctica ha sigut generar un codi que captés les línies de text. Per això s'ha tingut en compte que els paràmetres necessaris han de ser introduïts en ordre, pel que s'ha de comprovar que tots els valors s'han obtingut de manera correcta. Aquesta tasca s'ha realitzat inicialment amb la funció *getops* de *bash* i, a la segona part de la pràctica, s'ha reutilitzat aquesta estructura també a l'executable *python*.

Un cop implementada la captació de variables s'ha aplicat el filtre de l'actual existència del fitxer de sortida: si el fitxer no existeix aquest es crearà. Seguidament s'ha hagut d'aplicar la modificació del fitxer ja existent quan s'executa l'*script* amb paràmetres d'entrada i aquest ja es troba creat.

A l'executable *bash* s'utilitza la funció *validate* que realitza el procés d'emmagatzemar i gravar les dades rebudes al fitxer. Aquesta guarda els valors existents en variables precedides per "load" (*load.fila*) i els valors introduïts, prèviament validats, en variables precedides per "config" (*config.fila*).

L'executable *python* no compta amb la funció *validate* com a tal, però aquesta funcionalitat l'exerceixen una sèrie d'*if*s que comproven si s'ha introduït les variables per teclat. Dins d'aquests condicionals es realitza el mateix procediment que al *validate*

del fitxer *bash*, encara que les variables que utilitza en l'intercanvi d'informació tenen una nomenclatura distinta (s'utilitza el *pars args* com a *getops* i es guarden les variables seguides de *.args*).

Disseny a prova d'errors

A part de les decisions de disseny prèvies i la seva implementació també s'ha hagut de tenir en compte els llindars de cada paràmetre: no es pot tenir una mida de camp més gran de la permesa, la pilota no pot aparèixer fora del camp...

L'execució dels fitxers es realitza de la següent manera:

./config.sh -n config.txt -f files -c cols -p port -0 posPal -m midaPal -1 posPil + velPil

./config.py -n config.txt -f files -c cols -p port -0 posPal -m midaPal -1 posPil + velPil

A continuació es descriuen els paràmetres que fan falta per la creació del fitxer de configuració per facilitar el seu reconeixement al llarg del document.

Fitxer

El primer paràmetre que es rep és el fitxer on hi figurarà la configuració que es generi pels *scripts* (*-n*). Si aquest fitxer ja existeix se sobreescriurà, mentre que en cas contrari es crearà de 0 amb els paràmetres proporcionats.

Camp

Seguidament es capta la que serà la primera fila de text al fitxer de configuració. Aquesta consta dels paràmetres que defineixen el camp de joc. Per a poder crear-lo s'ha tingut en compte els següents llindars:

- Files: mínim de 10 i màxim de 120 (*-f*).
- Columnes: Mínim de 10 i màxim de 36 (*-c*).
- Mida de la porteria: mínim de 8 i màxim de nºfiles-1 (*-p*).

Paleta

A la segona fila de la configuració hi figuraran els paràmetres que defineixen la paleta, que és la part del joc controlable pel jugador a la pràctica 2. La captació dels valors de la paleta es realitza un pèl distinta de la dels valors del camp: la posició de la paleta (tant la fila com la columna separades per una coma) es rep precedida de *-0* i la seva mida seguida de *-m*.

- Files: mínim de 2 i màxim de 118.
- Columnes: Mínim de 2 i màxim de 35.
- Mida de la paleta: mínim de 3 i màxim de nºfiles-1 (*-m*).

Pilotes

Per finalitzar la configuració senzilla de la partida es capta la tercera fila, la qual conté la configuració de la pilota precedida per *-1* amb tots els valors seguits separats per comes. Per a poder crear-la s'ha tingut en compte els següents llindars:

- Fila on sortirà la pilota: mínim de 2 i màxim de 118.
- Columna on sortirà la pilota: Mínim de 2 i màxim de 35.
- Velocitat *x* i *y* de la pilota: mínim de –1.0 i màxim de 1.0.

Aquesta tercera línia del fitxer de configuració pot anar seguida de fins a 8 línies més que defineixin altres pilotes que es trobaran simultàniament al camp de joc. El programa llençarà un missatge d'error en cas que aquest límit se sobrepassi.

Canvis i apunts dels apartats anteriors

Per a poder executar correctament les línies de comanda del test de prova proporcionat per l'equip docent s'ha hagut de canviar els paràmetres que s'introduïen com a columnes degut a que aquests valors estaven fora de rang (de 40 a 36).

Si bé és cert que s'ha captat un error potencial en la creació de la paleta relacionat amb la seva mida i la de la porteria (la paleta és més gran que la porteria i, per tant, se solapa

i no es podrà marcar mai gol), aquest romandrà amb les especificacions que hi figuren al guió de la pràctica però amb el que s'hauria de fer per arreglar aquest *bug*.

Per evitar que les pilotes es creessin fora del camp s'ha modificat els llindars superiors de fila i columna (en comptes de 118 i 35 s'utilitzen fila-1 i columna -1).

També s'ha tingut en compte una optimització del codi permetent diferents opcions d'entrada de pilotes: d'una en una (escrivint *-1* a cada entrada de pilota), totes alhora (contingudes per cometes i separades per espais entre sí després del *-1*) i en distints grups de distintes mides.

Pla de proves

Per a comprovar que l'*script* funciona de manera idònia i no hi ha problemes amb els valors conflictius esmentats a l'apartat anterior s'ha dissenyat les següents proves, a part d'utilitzar les proporcionades pel professorat, per avaluar distints aspectes del nostre programa:

1. Avalua crear més pilotes de les permeses.
2. Avalua porteria més gran que el camp.
3. Avalua crear pilotes fora del camp.
4. Avalua paleta mes gran que la porteria (no es pot marcar gol).
5. Avalua paleta mes gran que l'amplada del camp.

Resultats de les proves

Proves professors

Pr1: Crea el fitxer amb el contingut "25 36 20 / 22 30 10 / 21 20 -1.0 0.3"

Pr2: Error per fila fora de rang.

Pr3: Demana fila per teclat i acaba creant el fitxer amb el contingut "?? 36 20 / 22 30 10 / 21 20 -1.0 0.3" (s'ha de tenir en compte que la fila d'entrada ha de ser >=21)

Pr4: Modifica el fitxer i resulta “50 30 20 / 22 30 10 / 21 20 -1.0 0.3”

Pr5: Crea el fitxer amb el contingut “25 36 20 / 22 30 10 / 21 20 -1.0 0.3 / 11 22 -1.0 0.3”

Proves personalitzades

Pr1: No es crea fitxer. Es llença un missatge d’error i no es completa l’execució.

Pr2: Demana mida de porteria més dins de rang (entre 8 i fila-1).

Pr3: Demana posició de creació de les pilotes vàlida.

Pr4: Crea el fitxer amb la paleta més gran que la porteria.

Pr5: Crea el fitxer amb les mides rebudes (paleta més gran del possible).

Codi font

Script en bash

```
#! /bin/bash



err() {

# echo "[$(date +'%Y-%m-%dT%H:%M:%S%z')]: $*" >&2

echo "ERROR: $*" >&2

}


usage() {

echo "USAGE:"

echo "${0}"
```

```
echo "  [-h] Mostra help"

echo "  [-n nomFit] Nom del fitxer on guardar la config."

echo "  [-f numFiles] # Files de la zona (10..120)"

echo "  [-c numColum] # Columnes de la zona (10..36)"

echo "  [-p midaPort] Mida de la porteria (8..numFiles-1)"

echo "  [-0 posFilaPal,posColPal] Fila i columna de la paleta (2..118),(2..35)"

echo "  [-m midaPaleta] Mida de la paleta (3..numFiles-1)"

# TODO Tendria que depender de la mida de la porteria? [3..midaPort-1] (Para evitar el caso que la paleta sea mas grande que la porteria?)

echo "  [-1 posFilaPil,posColPil,velFil,velCol] Fila, columna, velocitat vertical i horitzontal de la pilota en joc (2..118),(2..35),(-1.0..1.0),(-1.0..1.0). Aquesta opt es pot repetir fins a 9 vegades."

}




FILENAME="" # Nom del fitxer on guardar la config

Y=0 # Files

X=0 # Cols

WIDTH_GOAL=0 # Mida porteria

Y_PADDLE=0 # Fila pal

X_PADDLE=0 # Col pal

WIDTH_PADDLE=0 # Mida pal

N=0 # Num. de pilotas
```

SERVES="" # Info posFilaPil posColPil velFil velCol de pilotas

LOAD_Y=0 # Files en fitxer

LOAD_X=0 # Cols en fitxer

LOAD_WIDTH_GOAL=0 # Mida porteria en fitxer

LOAD_Y_PADDLE=0 # Fila pal en fitxer

LOAD_X_PADDLE=0 # Col pal en fitxer

LOAD_WIDTH_PADDLE=0 # Mida pal en fitxer

LOAD_N=0 # Num. de pilotas en fitxer

LOAD_SERVES="" # Info pilotas en fitxer

CONFIG_Y=0 # Files a guardar

CONFIG_X=0 # Cols a guardar

CONFIG_WIDTH_GOAL=0 # Mida porteria a guardar

CONFIG_Y_PADDLE=0 # Fila pal a guardar

CONFIG_X_PADDLE=0 # Col pal a guardar

CONFIG_WIDTH_PADDLE=0 # Mida pal a guardar

```
CONFIG_N=0 # Num. de pilotas a guardar

CONFIG_SERVES="" # Info pilotas a guardar



load_config() {

echo "loading config from \"$FILENAME\" :D"

# Asumir que la config guardada en un fitxer sempre correcta:

LINE_COUNT=0

while read -r LINE; do

case "${LINE_COUNT}" in

0) # La primera: numFiles numColum midaPort

read -r LOAD_Y LOAD_X LOAD_WIDTH_GOAL <<< "${LINE}"

;;


1) # La segona: posFilaPal posColPal midaPaleta

read -r LOAD_Y_PADDLE LOAD_X_PADDLE LOAD_WIDTH_PADDLE <<< "${LINE}"

;;


*) # La tercera -- fins ultima: pilota en joc: posFila posCol velFil velCol

read -r LOAD_BALL_Y LOAD_BALL_X LOAD_BALL_SPEED_Y LOAD_BALL_SPEED_X <<<
"${LINE}"

LOAD_SERVES="${LOAD_SERVES}
${LOAD_BALL_Y},${LOAD_BALL_X},${LOAD_BALL_SPEED_Y},${LOAD_BALL_SPEED_X}"

LOAD_N=$((LOAD_N+1))
```

```
;;

esac
LINE_COUNT=$((LINE_COUNT+1))
echo "LINE=\"${LINE}\""
echo "LINE_COUNT=${LINE_COUNT}"

done < ${FILENAME}
echo :: ${LOAD_Y}=LOAD_Y
echo :: ${LOAD_X}=LOAD_X
echo :: ${LOAD_WIDTH_GOAL}=LOAD_WIDTH_GOAL
echo
echo :: ${LOAD_Y_PADDLE}=LOAD_Y_PADDLE
echo :: ${LOAD_X_PADDLE}=LOAD_X_PADDLE
echo :: ${LOAD_WIDTH_PADDLE}=LOAD_WIDTH_PADDLE
echo
echo :: ${LOAD_N}=LOAD_N
echo :: ${LOAD_SERVES}=LOAD_SERVES
echo
}

save_config() {
```

```
echo "${CONFIG_Y} ${CONFIG_X} ${CONFIG_WIDTH_GOAL}" > ${FILENAME}

echo "${CONFIG_Y_PADDLE} ${CONFIG_X_PADDLE} ${CONFIG_WIDTH_PADDLE}" >> ${FILENAME}

for BALL in ${CONFIG_SERVES}; do

BALL_Y=$(echo $BALL | cut -d, -f1)

BALL_X=$(echo $BALL | cut -d, -f2)

BALL_SPEED_Y=$(echo $BALL | cut -d, -f3)

BALL_SPEED_X=$(echo $BALL | cut -d, -f4)

echo "${BALL_Y} ${BALL_X} ${BALL_SPEED_Y} ${BALL_SPEED_X}" >> ${FILENAME}

echo "firaball=(${BALL_Y}),(${BALL_X}), (${BALL_SPEED_Y}),(${BALL_SPEED_X})"

done

}



# num, min, max

util_is_in_range() {

NUM=$1

MIN=$2

MAX=$3

[ "${NUM}" -ge "${MIN}" ] && [ "${NUM}" -le "${MAX}" ]

}



# num, min, max
```

```
util_float_is_in_range() {
NUM=$1
MIN=$2
MAX=$3
(( $(echo "${NUM} >= ${MIN}" | bc -l) )) && (( $(echo "${MAX} <=  ${MAX}" | bc -l) ))
}

# min, max, prompt
util_ask_for_input_in_range() {
>&2 read -p "[$3] Introdueix int entre $1 i $2: " INPUT
re='^[\-]?[0-9]+$'
if ! [[ "${INPUT}" =~ ${re} ]] ; then
err "Requereix un int (format incorrrecte)."
util_ask_for_input_in_range $1 $2 "$3"
fi
if ! util_is_in_range ${INPUT} $1 $2; then
err "Requereix int entre $1 i $2..."
util_ask_for_input_in_range $1 $2 "$3"
fi
return ${INPUT}
}

# num, min, max
```

```
util_float_is_in_range() {

NUM=$1

MIN=$2

MAX=$3

(( $(echo "${NUM} >= ${MIN}" | bc -l) )) && (( $(echo "${NUM} <=  ${MAX}" | bc -l) ))

}



#variable, MIN, MAX, prompt

# compara si existe VAR o LOAD_VAR,

# valida si esta dentro del rango MIN..MAX,

# y lo guarda en CONFIG_VAR

validate() {

VAR="$1"

LOAD_VAR="LOAD_$1"

CONFIG_VAR="CONFIG_$1"

if [ "0" -eq ${!VAR} ]; then # no hay este VAR por param

if [ "0" -eq ${!LOAD_VAR} ]; then # tampoco estaba en el fitxer (o no hay fitxer)

util_ask_for_input_in_range $2 $3 "$4"

declare -g ${CONFIG_VAR}="$?"

else # darle valor que hay en el fitxer

declare -g ${CONFIG_VAR}=${!LOAD_VAR}

fi
```

```
else # darle valor pasado por param

declare -g ${CONFIG_VAR}=${!VAR}

fi


if ! util_is_in_range ${!CONFIG_VAR} $2 $3; then

util_ask_for_input_in_range $2 $3 "$4"

declare -g $CONFIG_VAR=$?

fi

}



# TODO echo la razón para pedir un param (no se ha pasado, es mayor / menor que el rango, es incompatible, etc)

validate_config() {

validate Y 10 120 "numFiles"

validate X 10 36 "numColum"

validate WIDTH_GOAL 8 $((CONFIG_Y-1)) "Mida porteria"


validate Y_PADDLE 2 118 "posFilaPal"

validate X_PADDLE 2 35 "posColPal"

validate WIDTH_PADDLE 3 $((CONFIG_Y-1)) "midaPaleta"


# get balls

if [ "0" -eq "${N}" ]; then # si no pasa balls by param
```

```
if [ "0" -eq "${LOAD_N}" ]; then # tampoco hay balls en el fitxer (o no hay fitxer)
# echo "bro no me has dado balls ni el config ni por params (con -1) :("
        err "No he rebut pilotes ni per parametres (arg \"-1\"), ni pel fitxer."
        # Tenemos que salir porque no tenemos suficiente info para guardar/crear un config "correcto"
        # (Pedir serves por teclado no tiene sentido)
exit 1
else # no pasa balls por param pero si por fitxer
N=${LOAD_N}
SERVES=${LOAD_SERVES}
fi
else # si pasa balls por param, sobre-escribe las q haya en el fitxero
CONFIG_N=${N}
# CONFIG_SERVES=${SERVES}
CONFIG_SERVES=""
fi
# validate balls
N=0 # cuenta otra vez, just in case
for BALL in ${SERVES}; do
if [ "${N}" -lt "9" ]; then # en principio si llega aqui, N es correcta :/
#FIXED ATENCION! esto permite poner mis pelotas dentro Y FUERA de los rangos? numFiles?
BALL_Y=$(echo ${BALL} | cut -d, -f1)
if ! util_is_in_range ${BALL_Y} 2 $((CONFIG_Y-1)); then
```

```
util_ask_for_input_in_range 2 $((CONFIG_Y-1)) "posFilaPil_$N"
fi
BALL_X=$(echo ${BALL} | cut -d, -f2)
if ! util_is_in_range ${BALL_X} 2 $((CONFIG_X-1)); then
util_ask_for_input_in_range 2 $((CONFIG_X-1)) "posColPil_$N"
fi
BALL_SPEED_Y=$(echo ${BALL} | cut -d, -f3)
while ! util_float_is_in_range ${BALL_SPEED_Y} -1 1; do
>&2 read -p "[velFil_$N] Introdueix float entre -1 y 1: " INPUT
re='^[\-]?([0-9]+)|([0-9]*\.[0-9]+)$' # 9 or .9 or 0.9
if [[ "${INPUT}" =~ ${re} ]] ; then
BALL_SPEED_Y=$INPUT
else
                    err "Requereix un float (format incorrecte)."
fi
done
BALL_SPEED_X=$(echo ${BALL} | cut -d, -f4)
while ! util_float_is_in_range ${BALL_SPEED_X} -1 1; do
>&2 read -p "[velCol_$N] Introdueix float entre -1 y 1: " INPUT
re='^[\-]?([0-9]+)|([0-9]*\.[0-9]+)$' # 9 or .9 or 0.9
if [[ "${INPUT}" =~ ${re} ]] ; then
BALL_SPEED_X=$INPUT
else
```

```
                err "Requereix un float (format incorrecte)."
fi
done
N=$((N+1))
CONFIG_SERVES="${CONFIG_SERVES}
${BALL_Y},${BALL_X},${BALL_SPEED_Y},${BALL_SPEED_X}"
#                       echo                    "ADDINGBALLTO_CONFIG_SERVES:
${BALL_Y},${BALL_X},${BALL_SPEED_Y},${BALL_SPEED_X}"
else
        err "He rebut pilotes de sobra... (max=9)"
exit 0
fi
done
}



while getopts ':hn:f:c:p:0:m:1:' arg; do
case "${arg}" in
h)
# echo "        [-h] Mostra help"
usage
exit 0
;;
```

```
n)
# echo "          [-n nomFit] Nom del fitxer on guardar la config."
if [ -z "${FILENAME}" ]; then
FILENAME=${OPTARG}
echo "FILENAME=${FILENAME}" # DEBUG
# [ -f "$FILENAME" ] && load_config
else
err "arg ${arg} doble."
exit 3
fi
;;

f)
echo "  [-f numFiles] # Files de la zona (10..120)"
if [ "0" -eq "${Y}" ]; then
Y=${OPTARG}
else
# err "me has dado -f numFiles-1 2 veces!"
err "arg ${arg} doble."
exit 3
fi
;;
```

```
c)
echo "  [-c numColum] # Columnes de la zona (10..36)"
if [ "0" -eq "${X}" ]; then
X=${OPTARG}
else
# err "me has dado -c numColum 2 veces!"
err "arg ${arg} doble."
exit 3
fi
;;

p)
echo "  [-p midaPort] Mida de la porteria (8..numFiles-1)"
if [ "0" -eq "${WIDTH_GOAL}" ]; then
WIDTH_GOAL=${OPTARG}
else
# err "me has dado -p midaPort 2 veces!"
err "arg ${arg} doble."
exit 3
fi
;;

0)
```

```
echo "  [-0 posFilaPal,posColPal] Fila i columna de la paleta (2..118),(2..35)"

if [ "0" -eq "${Y_PADDLE}" ] && [ "0" -eq "${X_PADDLE}" ]; then

Y_PADDLE=$(echo "${OPTARG}" | cut -d',' -f1)

X_PADDLE=$(echo "${OPTARG}" | cut -d',' -f2)

echo="${Y_PADDLE} ${X_PADDLE}"

else

# err "me has dado -0 (Fila y columna de la paleta) 2 veces!"

err "arg ${arg} doble."

exit 3

fi

;;


m)

echo "  [-m midaPaleta] Mida de la paleta (3..numFiles-1)"

if [ "0" -eq "${WIDTH_PADDLE}" ]; then

WIDTH_PADDLE=${OPTARG}

else

# err "me has dado -m midaPaleta 2 veces!"

err "arg ${arg} doble."

exit 3

fi

;;
```

```
1)

# greater than

for BALL in ${OPTARG}; do

if [ "${N}" -lt "9" ]; then

BALL_Y=$(echo $BALL | cut -d, -f1)

BALL_X=$(echo $BALL | cut -d, -f2)

BALL_SPEED_Y=$(echo $BALL | cut -d, -f3)

BALL_SPEED_X=$(echo $BALL | cut -d, -f4)

# echo "reading ball #$N=(${BALL_Y}),(${BALL_X}),
(${BALL_SPEED_Y}),(${BALL_SPEED_X})"


# rangos al validar (luego)

# if (( $(echo "${BALL_SPEED_Y} >= -1.0" | bc -l) )) &&

#  (( $(echo "${BALL_SPEED_Y} <=  1.0" | bc -l) )); then

#         # echo ok bola speed bien

# else

#         # echo no oka

# fi


N=$((N+1))

SERVES="${SERVES} ${BALL}"

else

                err "He rebut pilotes de sobra... (max=9)"

exit 1
```

```
fi

done

# echo "y > 0"

# echo "        [-1 posFilaPil,posColPil,velFil,velCol] Fila, columna, velocitat vertical i horitzontal de la pilota en joc (2..118),(2..35),(-1.0..1.0),(-1.0..1.0). Aquesta opt es pot repetir fins a 9 vegades."

;;


:)

err "ERROR: Argument \"-${OPTARG}\" necesita parametres."

>&2 usage

exit 1

;;


*)

err "ERROR: Argument \"-${OPTARG}\" no reconegut."

>&2 usage

exit 1

;;


esac

done
```

```
if [ -z "${FILENAME}" ]; then

# err No me has dado FILENAME con -n, adeu

   err "No he rebut nom del fitxer on guardar la config. (arg \"-n (nomFit)\")."

exit 1

fi

if [ -f "${FILENAME}" ]; then

# echo loading config cause it exists

load_config # el config exite, lo cargamos

fi

# echo validate_config

validate_config

# echo save_config

save_config



# echo he guardado el siguiente config

# cat $FILENAME


# echo Adeu
```

Script en python

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

```
import getopt, sys



def err(msg):
    print("ERROR: " + msg, file=sys.stderr)



class Config:
    filename=None # Nom del fitxer on guardar la config
    y = 0 # Files
    x = 0 # Cols
    width_goal = 0 # Mida porteria
    y_paddle = 0 # Fila pal
    x_paddle = 0 # Col pal
    width_paddle = 0 # Mida pal
    serves = [] # lista de pilotas, [pilota1, pilota2, ... ]
    # cada pilota es: [posFilaPil, posColPil, velFil, velCol] de pilotas
    def validate():
        return True


def usage(err=False):
    if err:
        err("USAGE: " + sys.argv[0])
```

```
        err(" [-h] Mostra help")
        err(" [-n nomFit] Nom del fitxer on guardar la config.")
        err(" [-f numFiles] # Files de la zona (10..120)")
        err(" [-c numColum] # Columnes de la zona (10..36)")
        err(" [-p midaPort] Mida de la porteria (8..numFiles-1)")
        err(" [-0 posFilaPal,posColPal] Fila i columna de la paleta (2..118),(2..35)")
        err(" [-m midaPaleta] Mida de la paleta (3..numFiles-1)") # TODO Tendria que depender de la mida de la porteria? [3..midaPort-1] (Para evitar el caso que la paleta sea mas grande que la porteria?)
        err(" [-1 posFilaPil,posColPil,velFil,velCol] Fila, columna, velocitat vertical i horitzontal de la pilota en joc (2..118),(2..35),(-1.0..1.0),(-1.0..1.0). Aquesta opt es pot repetir fins a 9 vegades.")
    else:
        print("USAGE: " + sys.argv[0])
        print(" [-h] Mostra help")
        print(" [-n nomFit] Nom del fitxer on guardar la config.")
        print(" [-f numFiles] # Files de la zona (10..120)")
        print(" [-c numColum] # Columnes de la zona (10..36)")
        print(" [-p midaPort] Mida de la porteria (8..numFiles-1)")
        print(" [-0 posFilaPal,posColPal] Fila i columna de la paleta (2..118),(2..35)")
        print(" [-m midaPaleta] Mida de la paleta (3..numFiles-1)") # TODO Tendria que depender de la mida de la porteria? [3..midaPort-1] (Para evitar el caso que la paleta sea mas grande que la porteria?)
        print(" [-1 posFilaPil,posColPil,velFil,velCol] Fila, columna, velocitat vertical i horitzontal de la pilota en joc (2..118),(2..35),(-1.0..1.0),(-1.0..1.0). Aquesta opt es pot repetir fins a 9 vegades.")
```

```
def util_ask_for_input_in_range_float(min, max, prompt=""):
    while True:
        try:
            v = float(input(f"{prompt} Introdueix float entre {min} i {max}: "))
            if min <= v <= max:
                return v
            err(f"Requereix float entre {min} i {max}...")
        except ValueError:
            err("Requereix un float (format incorrecte).")
    return 0


def util_ask_for_input_in_range(min, max, prompt=""):
    while True:
        try:
            v = int(input(f"{prompt} Introdueix int entre {min} i {max}: "))
            if min <= v <= max:
                return v
            err(f"Requereix int entre {min} i {max}...")
        except ValueError:
            err("Requereix un int (format incorrecte).")
    return 0
```

```
def parse_args():
    cfg_args = Config()
    try:
        opts, args = getopt.getopt(sys.argv[1:], ":hn:f:c:p:0:m:1:")
    except getopt.GetoptError as e:
        # print help information and exit:
        print(e)  # will print something like "option -a not recognized"
        usage(err=True)
        sys.exit(2)
    for o, a in opts:
        if o == "-h":
            usage()
            sys.exit()

        elif o == "-n": # nomFit
            if cfg_args.filename is None:
                cfg_args.filename = a
            else:
                err(f"arg {o} doble.")
                sys.exit(3)
```

```
elif o == "-f":
    if cfg_args.y == 0:
        try:
            cfg_args.y = int(a)
        except ValueError:
            err(f"arg \"{o}\" (numFiles) requereix int. (Ha rebut \"{a}\").")
            sys.exit(1)
    else:
        err(f"arg {o} doble.")
        sys.exit(3)

elif o == "-c":
    if cfg_args.x == 0:
        try:
            cfg_args.x = int(a)
        except ValueError:
            err(f"arg \"{o}\" (numColum) requereix int. (Ha rebut \"{a}\").")
            sys.exit(1)
    else:
        err(f"arg {o} doble.")
        sys.exit(3)
```

```
        elif o == "-p":
            if cfg_args.width_goal == 0:
                try:
                    cfg_args.width_goal = int(a)
                except ValueError:
                    err(f"arg \"{o}\" (midaPort) requereix int. (Ha rebut \"{a}\").")
                    sys.exit(1)
            else:
                err(f"arg {o} doble.")
                sys.exit(3)


        elif o == "-0":
            if cfg_args.y_paddle == 0 or cfg_args.x_paddle == 0 :
                #a = "posFilaPal,posColPal"
                s = a.split(',')
                if len(s) == 2:
                    cfg_args.y_paddle, cfg_args.x_paddle = s
                    try:
                        cfg_args.y_paddle = int(cfg_args.y_paddle)
                        cfg_args.x_paddle = int(cfg_args.x_paddle)
                    except ValueError:
                        err(f"arg \"{o}\" (posFilaPal,posColPal) requereix int,int. (Ha rebut \"{a}\").")
                        sys.exit(1)
```

```
        else:
            err(f"arg \"{o}\" (posFilaPal,posColPal) requereix int,int. (Ha rebut \"{a}\").")
            sys.exit(1)
    else:
        err(f"arg {o} doble.")
        sys.exit(3)

elif o == "-m":
    if cfg_args.width_paddle == 0:
        try:
            cfg_args.width_paddle = int(a)
        except ValueError:
            err(f"arg \"{o}\" (midaPaleta) requereix int. (Ha rebut \"{a}\").")
            sys.exit(1)
    else:
        err(f"arg {o} doble.")
        sys.exit(3)

elif o == "-1":
    for serve_str in a.split(" "):
        # print(f">read serve #{len(cfg_args.serves)} = {serve_str}")
        # FIXED comprov que de verdad hay 4 valores separados por ','
        s = serve_str.split(',')
```

```
            if len(s) == 4:
                ball_y, ball_x, ball_speed_y, ball_speed_x = serve_str.split(',')
                try:
                    ball_y = int(ball_y)
                    ball_x = int(ball_x)
                    ball_speed_y = float(ball_speed_y)
                    ball_speed_x = float(ball_speed_x)
                    if len(cfg_args.serves) < 9:
                        cfg_args.serves.append([ball_y, ball_x, ball_speed_y, ball_speed_x])
                        # print(f">add serve from args: {serve_str}")
                    else:
                        err("He rebut pilotes de sobra... (max=9)")
                        sys.exit(1)
                except ValueError:
                    err(f"arg \"{o}\" (posFilaPil,posColPil,velFil,velCol) requereix int,int,float,float. (Ha rebut \"{serve_str}\").")
                    sys.exit(1)
            else:
                err(f"arg \"{o}\" (posFilaPil,posColPil,velFil,velCol) requereix int,int,float,float. (Ha rebut \"{serve_str}\").")
                sys.exit(1)

        else:
            err(f"Unable to handle arg \"{o}\"?")
```

```
            sys.exit(2)


    return cfg_args



def load_config(filename):
    cfg_load = Config()
    try:
        with open(filename) as f:
            for n,line in enumerate(f):
                if n == 0:
                    cfg_load.y, cfg_load.x, cfg_load.width_goal = line.split()
                    cfg_load.y, cfg_load.x, cfg_load.width_goal = int(cfg_load.y), int(cfg_load.x), int(cfg_load.width_goal)
                elif n == 1:
                    cfg_load.y_paddle, cfg_load.x_paddle, cfg_load.width_paddle = line.split()
                    cfg_load.y_paddle, cfg_load.x_paddle, cfg_load.width_paddle = int(cfg_load.y_paddle), int(cfg_load.x_paddle), int(cfg_load.width_paddle)
                    pass
                else:
                    ball_y, ball_x, ball_speed_y, ball_speed_x = line.split()
                    cfg_load.serves.append([int(ball_y), int(ball_x), float(ball_speed_y), float(ball_speed_x)])
```

```
    except FileNotFoundError as e:
        print("Fitxer \"" + filename + "\" no existeix, el crearem.")
    cfg_load.filename = filename
    return cfg_load


def validate_config(cfg_args, cfg_load):
    cfg = Config()

    if cfg_args.y == 0:
        if cfg_load.y == 0:
            err("No he rebut numFiles ni per parametres (arg \"-f\"), ni pel fitxer.")
        else:
            cfg.y = cfg_load.y
    else:
        cfg.y = cfg_args.y

    if cfg_args.x == 0:
        if cfg_load.x == 0:
            err("No he rebut numColum ni per parametres (arg \"-c\"), ni pel fitxer.")
        else:
            cfg.x = cfg_load.x
    else:
```

```
    cfg.x = cfg_args.x


if cfg_args.width_goal == 0:
    if cfg_load.width_goal == 0:
        err("No he rebut midaPort ni per parametres (arg \"-p\"), ni pel fitxer.")
    else:
        cfg.width_goal = cfg_load.width_goal
else:
    cfg.width_goal = cfg_args.width_goal


if cfg_args.y_paddle == 0:
    if cfg_load.y_paddle == 0:
        err("No he rebut posFilaPal ni per parametres (arg \"-0\"), ni pel fitxer.")
    else:
        cfg.y_paddle = cfg_load.y_paddle
else:
    cfg.y_paddle = cfg_args.y_paddle


if cfg_args.x_paddle == 0:
    if cfg_load.x_paddle == 0:
        err("No he rebut posColPal ni per parametres (arg \"-0\"), ni pel fitxer.")
    else:
        cfg.x_paddle = cfg_load.x_paddle
```

```
    else:
        cfg.x_paddle = cfg_args.x_paddle



    if cfg_args.width_paddle == 0:
        if cfg_load.width_paddle == 0:
            err("No he rebut midaPaleta ni per parametres (arg \"-m\"), ni pel fitxer.")
        else:
            cfg.width_paddle = cfg_load.width_paddle
    else:
        cfg.width_paddle = cfg_args.width_paddle


    if not cfg_args.serves:
        if not cfg_load.serves:
            err("No he rebut pilotes ni per parametres (arg \"-1\"), ni pel fitxer.")
            sys.exit(1) # Tenemos que salir porque no tenemos suficiente info para guardar/crear un config "correcto"
            # (Pedir serves por teclado no tiene sentido)
        else:
            cfg.serves = cfg_load.serves
    else:
        cfg.serves = cfg_args.serves


    # validate Y 10 120 "numFiles"
```

```
if not 10 <= cfg.y <= 120:
    cfg.y = util_ask_for_input_in_range(10, 120, "[numFiles]")

# validate X 10 36 "numColum"
if not 10 <= cfg.x <= 36:
    cfg.x = util_ask_for_input_in_range(10, 36, "[numColum]")

# validate WIDTH_GOAL 8 $((CONFIG_Y-1)) "Mida porteria"
if not 8 <= cfg.width_goal <= cfg.y - 1:
    cfg.width_goal = util_ask_for_input_in_range(8, cfg.y - 1, "[midaPort]")

# validate Y_PADDLE 2 118 "posFilaPal"
if not 2 <= cfg.y_paddle <= 118:
    cfg.y_paddle = util_ask_for_input_in_range(8, 118, "[posFilaPal]")

# validate X_PADDLE 2 35 "posColPal"
if not 2 <= cfg.x_paddle <= 35:
    cfg.x_paddle = util_ask_for_input_in_range(2, 35, "[posColPal]")

# validate WIDTH_PADDLE 3 $((CONFIG_Y-1)) "midaPaleta"
if not 3 <= cfg.width_paddle <= cfg.y - 1:
    cfg.x_paddle = util_ask_for_input_in_range(3, cfg.y - 1, "[midaPaleta]")
```

```
    # if not 3 <= cfg.width_paddle <= cfg.width_goal - 1: # TODO Tendria que depender de la mida de la porteria? [3..midaPort-1] (Para evitar el caso que la paleta sea mas grande que la porteria?)
    #     cfg.x_paddle = util_ask_for_input_in_range(3, cfg.width_goal - 1, "[midaPaleta]")

    tmp_serves = [] # var aux para no modificar la lista sobre la que iteramos
    for n, serve in enumerate(cfg.serves):
        ball_y, ball_x, ball_speed_y, ball_speed_x = serve  # se puede hacer ya que asumimos que la config guardada esta SIEMPRE bien
        # print(f"loading serve#{n}: {serve})

        # if not 2 <= ball_y <= 118: ##FIXED ATENCION! esto permite poner mis pelotas dentro Y FUERA de los rangos? numFiles? (permite pelotas fuera del tablero?)
        #     ball_y = util_ask_for_input_in_range(2, 118, f"[posFilaPil#{n}]")
        # if not 2 <= ball_x <= 35: ##FIXED ATENCION! esto permite poner mis pelotas dentro Y FUERA de los rangos? numFiles?
        #     ball_x = util_ask_for_input_in_range(2, 35, f"[posColPil#{n}]")

        if not 2 <= ball_y <= cfg.y - 1:
            ball_y = util_ask_for_input_in_range(2, cfg.y - 1, f"[posFilaPil#{n}]")
        if not 2 <= ball_x <= cfg.x - 1:
            ball_x = util_ask_for_input_in_range(2, cfg.x - 1, f"[posColPil#{n}]")
        if not -1.0 <= ball_speed_y <= 1.0:
            ball_speed_y = util_ask_for_input_in_range(-1.0, 1.0, f"[velFil#{n}]")
        if not -1.0 <= ball_speed_x <= 1.0:
```

```
            ball_speed_x = util_ask_for_input_in_range(-1.0, 1.0, f"[velCol#{n}]")
        tmp_serves.append([ball_y, ball_x, ball_speed_y, ball_speed_x])

    cfg.serves = tmp_serves
    cfg.filename = cfg_load.filename
    return cfg



def save_config(cfg):
    try:
        with open(cfg.filename, "w") as f:
            f.write(f"{cfg.y} {cfg.x} {cfg.width_goal}\n")
            f.write(f"{cfg.y_paddle} {cfg.x_paddle} {cfg.width_paddle}\n")
            for serve in cfg.serves:
                ball_y, ball_x, ball_speed_y, ball_speed_x = serve
                f.write(f"{ball_x} {ball_x} {ball_speed_y} {ball_speed_x}\n")
    except OSError as e:
        err(str(e))
        err(f"No puc guardar al fitxer: \"{cfg.filename}\" (OSError)")



def main():
    # print("Hola desde main")
```

```
    # print("parsing args")
    cfg_args = parse_args()
    if cfg_args.filename is None:
        err("No he rebut nom del fitxer on guardar la config. (arg \"-n (nomFit)\").")
        sys.exit(1)

    # print("loading config")
    cfg_load = load_config(cfg_args.filename) # Info que ya hay en el fitxer

    # print("validating config")
    cfg = validate_config(cfg_args, cfg_load)

    # print("saving config")
    save_config(cfg)

    # print("Adeu desde main")

if __name__ == "__main__":
    main()
```

Conclusions

Degut a que aquest treball pràctic demanda un *script* en *bash* i un en *python* amb la mateixa funció, la segona part del treball no ha sigut tan pesada ja que la seva implementació s'ha basat en una traducció gairebé literal del codi de la primera part, el qual s'ha emportat el temps de disseny i desenvolupament del codi així com el de recerca.

Encara que creiem que la implementació del *getops* és útil hem vist que aquesta pràctica ha implicat un desenvolupament de codi bastant llarg i pesat. Tot i això serà satisfactori veure com el nostre generador de configuracions fa possible l'execució de la segona pràctica.

Autoavaluació

Els criteris que s'han tingut en compte són els següents:

1. Decisions de disseny, ús de funcions amb paràmetres i documentació (1/1)
2. Disseny del Joc de Proves i documentació (1/1)
3. Creació d'un fitxer amb opcions vàlides passades per línia d'ordes amb una pilota (0,5/0,5)
4. Demanar opcions concretes a l'usuari si no s'ha entrat per línia d'ordres (0,5/0,5)
5. Demanar opcions concretes a l'usuari si estan fora del rang (0,5/0,5)
6. Passar més d'una pilota per línia d'ordres, controlant màxim 9 pilotes (0,5/0,5)
7. Què es fa si s'afegeixen pilotes i el fitxer existeix? (0,5/0,5)
8. Tractament d'errors/excepcions (0,5/0,5)
9. Validació dinàmica dels rangs de les variables (0,5/0,5)
10. Implementació i ús correcte de les comandes de la bash (2/2)
11. Implementació i ús correcte de les comandes python (2/2)

Pel que l'autoavaluació d'aquest grup, objectivament, és de 10/10 ja que s'ha aconseguit cobrir tots els requisits. Aquesta nota segurament sigui menor però no tenim els medis per avaluar exactament el percentatge assolit en cada punt.