**ASSIGNMENT**

**Procedural Programming Term 1**

**“Ro-Ro-Ro-Your-Bots”**

PLAN:

- Read brief thoroughly and make notes

- learn correct **REFERENCING** - APA 7th

- see [APA 7th Referencing Guide :: Library Services :: University of HuddersfieldAPA 7th Referencing Guide :: Library Services :: University of Huddersfield](https://library.hud.ac.uk/pages/apareferencing/)

- Decide how program is going to interact with the user:

- VISUALS? How do I display the status of the robots and workers when all I know how to display is a single line at a time?

**IDEAS:** Research Python modules for manipulating display:

tkinter allows set-up of a GUI –

panda ???

curses ???

**TOO COMPLICATED**

- Name the program ✓

- storyline - humourously dark sci-fi, a la Futurama

- Main functions

* *learn Python* ***STYLE*** *to format code :*
* *Use PEP8 style guide for Python -* [*https://peps.python.org/pep-0008/*](https://peps.python.org/pep-0008/)
* *learn how to format essays at University of Huddersfield:*
  + *line spacing*
  + *font*
  + *alignment*
  + *page numbering*
* *formatting code*

How is the user/supervisor going to control these robots?

- keyboard commands:

|  |  |  |  |
| --- | --- | --- | --- |
| COMMAND | CODE TO TYPE | FUNCTION CODE | NOTES |
| Remove robot |  |  |  |
| Add human |  |  |  |
| Remove human |  |  |  |
| What next ? | - | what\_next() | Main function to get user input |
| HELP | H | instructions() | Show user instruction manual |
| QUIT | Q | quit\_program() | END ENTIRE PROGRAM |
| ADD | A | add\_robot() |  |
| REMOVE | R | remove\_robot() |  |
| EMPLOY | E | employ\_worker() | Sister-function to ADD, but for a human |
| FIRE | F | fire\_worker() | Sister-function to REMOVE, but for a human |
| CHANGE | C | change\_status() | Switch whether worker is busy, idle or off work, and assign to specific tasks |
| MANAGE | M | monitor\_status() | Display all factory activity |
| LOG | L | log\_tasks() | Display tasks yet to be completed, completed already , or add and remove tasks from the |

WHAT I DID:

Watched YouTube Python tutorial on classes - <https://www.youtube.com/watch?v=ZDa-Z5JzLYM&pp=ygUOcHl0aG9uIGNsYXNzZXM%3D>

Established I was going to use classes to store the information for each robot and human.

Created functions to add or remove robots and humans. Because the functions were as yet empty, I included the pass instruction in them so that the program would skip over them without raising errors.

For each function or class I created I included comments, both to remind me of what the code was for, and also to divide it up so I could read it more easily.

I began to work with a method of keeping a back-up file storing the previous version of the file. Only when I was satisfied that the changes I had made since then were going to work correctly did I update the main version of the file. Then I learnt the basics of Git logging.

I used the scratch files function in PyCharm to test isolated sections of code I was working on without having to run the entire program.

I started reading up on articles about how to allow the program to function in real time and on how to use classes.

Then I found out that the brief had been altered so that real time functionality was no longer required and that no classes were to be used, so I amended what I had written to reflect these changes.

The first code I wrote was the intro, then the function for the user to input how many robots and workers they want. I had to sanitise the input as best as I could, then print a response.

**CURRENT VERSION OF MY PROGRAM:**

# Ro-Ro-Ro-Your-Bots Incorporated

# Program to simulate coordination between n robots and m workers in a cell.

# Import modules

from time import time, ctime, gmtime, struct\_time

# ---- Initialise main variables

# n = number of robots

n = 0

# m = number of workers

m = 0

robot\_dict = {}

human\_dict = {}

""" Set up dictionaries to store current status of robots or humans.

- This function's parameters allow it to be used for both

robots and humans, as it is being passed the variables n or m

and robot\_status or human\_status

"""

def dict\_function(group\_nmbr, dict\_group, label):

for r in range(1, group\_nmbr + 1):

dict\_group[label + str(r)] = "Idle"

return dict\_group

# Function to get user to choose an action

def what\_next():

print()

next\_action = input("Please choose an action (or press H to get Help)").upper()

if len(next\_action) > 8:

print("""Sorry, this user interface has not been designed\n

to cope with such long instructions.""")

what\_next()

# HELP

elif next\_action == "H":

instructions()

# QUIT

elif next\_action == "Q":

quit\_program()

# ADD

elif next\_action == "A":

add\_robot()

# REMOVE

elif next\_action == "R":

remove\_robot()

# CHANGE

elif next\_action == "C":

change\_status()

# EMPLOY

elif next\_action == "E":

employ\_worker()

# FIRE

elif next\_action == "F":

fire\_worker()

# MANAGE

elif next\_action == "M":

monitor\_status()

# LOG

elif next\_action == "L":

log\_tasks()

else:

print("Sorry, I don't think that's a valid option.")

print("Any feedback should be directed to our team.")

what\_next()

# Instructions for Supervisor

# Accessed by pressing H within what\_next() function

def instructions():

print("""\n\n

- - Instructions for how to oversee production

at

Ro-Ro-Ro-Your-Bots Incorporated: - -

\n

When prompted as to what options are available to you,

press a key from these available alternatives

to select from its sub-menu:

\n

[A] = ADD [R] = REMOVE

Add Robot(s) Remove Robot(s)

\n

[E] = EMPLOY [F] = FIRE

Employ Worker(s) Fire Worker(s)

\n

[C] = CHANGE [M] = MANAGE

Change Robot Manage Worker

Status Status

\n

[L] = LOG [H] = HELP !!!

Log Tasks Display these instructions

\n

[Q] = QUIT

Resign your position

& END PROGRAM !\n """)

# Function to add robot

# Accessed by pressing A within what\_next() function

def add\_robot():

n += 1

# Function to remove robot.

# Accessed by pressing R within what\_next() function

def remove\_robot():

n -= 1

# Function to change status of a robot:

# Accessed by pressing C within what\_next() function

# statuses available = idle / working / finished task

def change\_status():

pass

# Function to employ new worker.

# Accessed by pressing E within what\_next() function

def employ\_worker():

m += 1

# Function to fire a worker.

# Accessed by pressing F within what\_next() function

def fire\_worker():

m -= 1

# Function to track worker

# Accessed by pressing T within what\_next() function

# statuses available = idle / working / finished task

def track\_worker():

pass

# Function to order available robots and workers to carry out tasks.

# Accessed by pressing O within what\_next() function

def order\_to\_task():

pass

# Log tasks

# Accessed by pressing L within what\_next() function

# Classify tasks as not started / in progress / completed

def log\_tasks():

pass

# Function to monitor progress of robots, workers and tasks

# Accessed by pressing M within what\_next() function

# Should regularly update status and identify problems

# eg. Idle robots, idle workers, incomplete tasks.

def monitor\_status():

pass

# Function to quit program

def quit\_program():

quit = input("Are you sure you want to quit? (Y/N)").upper()

if quit == "Y":

exit()

elif quit == "N":

what\_next()

################################# --- BODY OF PROGRAM --- ################################

# ---- Introduction

print("Program to simulate coordination between n robots and m workers in a robotic cell")

input("\nPress Enter to continue...")

print("\n Scenario:")

print("""\n You are the supervisor of Ro-Ro-Ro-Your-Bots, where robots work alongside

human workers in a robotic cell, producing industry-standard robot workers which are

to be shipped out to a more substantial plant, wherein they will forge components

that can be boxed and sold in flat-packs, and assembled in situ into factories that

produce robots, built by robotic cells of robot workers labouring alongside humans...

""")

input("\nPress Enter to continue...")

# ----Ask user how many robots they intend to employ (n):

# ---- & check the value entered for n is valid :

n = int(input("\n How many robot workers do you want for your factory?\n Please choose a number between 1 and 100. "))

while n < 1 or n > 100:

print("\n Sorry, but that isn't a valid response.")

if n < 0:

print("\n How can we have a negative amount? We don't owe nobody our robots !!")

elif n == 0:

print("\n We can't build robots with no robots to build them.")

elif n > 100:

print("\n We can't risk having that many droids onsite.")

print("\n HAVE YOU NEVER SEEN BLADE RUNNER?")

n = int(input("\n How many robot workers do you intend to employ?\n Please choose a number between 1 and 100. "))

# Call function to set up a dictionary storing the status of each robot:

robot\_dict = dict\_function(n, robot\_dict, "Droid ")

# ----Output initial number of droids (n)

print(f"\nOK, Here are your {n} droids!\n")

for counter in range(1, n + 1):

if (counter) % 10 == 0:

print("\n")

else:

pass

if counter < 10:

print(f"[Droid {counter} : {robot\_dict["Droid " + str(counter)]}]", end=" ")

else:

print(f"[Droid {counter} : {robot\_dict["Droid " + str(counter)]}]", end=" ")

# ----Ask user how many humans they intend to employ (m):

# ---- & check the value entered for n is valid :

m = int(input("\n\n How many human workers do you intend to employ?\n Please choose a number between 1 and 100. "))

while m < 1 or m > 100:

print("\n Sorry, but that isn't a valid response.")

if m < 0:

print("\n DOES NOT COMPUTE !!!")

elif m == 0:

print("\n We need at least one human worker to prevent a droid revolt.")

elif m > 100:

print("\n You can't afford to pay out wages for over 100 workers.")

m = int(input("\n\n How many human workers do you intend to employ?\n Please choose a number between 1 and 100. "))

# Call function to set up a dictionary storing the status of each human:

human\_dict = dict\_function(m, human\_dict, "Human ")

# ----Output initial number of human workers (m)

print(f"\nOK, Here are your {m} slaves!\n")

for counter in range(1, m + 1):

if (counter) % 10 == 0:

print("\n")

else:

pass

if counter < 10:

print(f"[Human {counter} : {human\_dict["Human " + str(counter)]}]", end=" ")

else:

print(f"[Human {counter} : {human\_dict["Human " + str(counter)]}]", end=" ")

# Display Instructions initially

instructions()

# Call function to ask for user action

what\_next()

References

Bell, A. (2018). *Get programming : learn to code with Python.* Manning Publications.

Schafer, C. (2016). *Python OOP Tutorial 1: Classes and Instances.* Retrieved November 2nd, 2024 from <https://www.youtube.com/watch?v=ZDa-Z5JzLYM&pp=ygUOcHl0aG9uIGNsYXNzZXM%3D>

Ronquillo, A. *A Beginner’s Guide to the Python time Module.* Retrieved November 9th, 2024 from <https://realpython.com/python-time-module/>

Schafer, C. (2016). *Python OOP Tutorial 1: Classes and Instances*. Retrieved November 9th, 2024 from <https://www.youtube.com/watch?v=ZDa-Z5JzLYM&pp=ygUOcHl0aG9uIGNsYXNzZXM%3D>

W3Schools. <https://www.w3schools.com/python>

Watched YouTube Python tutorial on classes -