**ASSIGNMENT**

**Procedural Programming Term 1**

"Ro-Ro-Ro-Your-Bots™"

The first step of my original plan was to read the brief thoroughly and make notes.

Later I realised the consequences of neglecting to carry this out in enough depth (see below).

Procrastinating diving into actually beginning the assignment proper, I began learning the stipulated referencing system, APA 7th (American Psychological Association, 2025), how the university expected me to format my essay, and how the Python community expected me to format my code. (Van Rossum, Warsaw & Coghlan, 2001)

In actuality, I knew that I didn’t know how to start the coding.

My thinking was that I needed to decide how the program would interact with the user;

Visually, how would I display the status of the robots and workers when all I knew how to output using Python was a single line at a time?

I wanted to learn how to use a Python graphics package. Online, I saw that popular ones appeared to be curses and tkinter, although the pandas data-handling package would also have been sufficient for this project.

I was daunted at the prospect of learning how to use one of these tools in addition to sharpening my basic Python skills enough to work the back-end.

After discussing it with the student mentors and even more influentially my module tutor, I decided to focus on the fundamentals.

I was unsure about how the user would control what was happening in the scenario, but since I wasn't going to use a GUI, it would have to be by textual commands.

While I attempted to get my head round how to handle the various programming challenges, I amused myself with naming the project and sketching a humorous storyline, influenced by the cartoon "Futurama", with its slightly dark irony (Cohen, Groening, Keeler, Katz, 1999 – present).

I wanted to provide as much guidance for the user as possible, rather than make a program mimicking the functioning of one in an actual factory, where the supervisor builds up a familiarity with the software, needing minimal onscreen instruction.

After writing the introduction sequence to my mock-simulation, I was ready to begin coding in earnest. I defined the principal functions, mostly naming them direct from the brief. I added comments to remind me of what they did, and to make the code more readable. the pass keyword was the origin of each function, so that Python would skip over them for now without spawning error messages.

I started reading up on articles about handling real time functioning and on how to use classes, both of which were in the original assignment requirements.

Then, realising the brief had been altered so neither of these were to be used, I had to alter what I'd written to reflect these changes.

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

Some of the difficulties encountered were:

- printing out the loop that displayed the initial workforce, concatenating the string describing each worker to their ID number.

- working out how to use dictionaries, especially setting them up to allow a single function to handle either humans or robots. I studied defining them using a loop and printing out the keys without the values. (Faccioni, 2022), (Schafer, 2017).

- dissecting user input and causing the program to repeat the question for an invalid response, I had the idea to use while True infinite loops, breaking or returning when a suitable answer was given. I refined this technique by using while not loops testing for the presence of a variable created purely as a conditional. I validated and sanitised the input as best I could.

- removing characters from a string, when collecting the input of a list separated by commas. (gsbabil, Potdar, Santilli & johnthagen, 2021)

- writing a time calculator that could format a gross amount of seconds into HH/MM/SS. The calculations kept going way off, giving the wrong number of minutes in a day. In the morning I found it was all down to a typo; I'd inputted a 5 instead of a 6.

- realising I needed the dictionaries of workers and robots to indicate their IDLE / WORKING / FINISHED status, not the actual tasks each were working on. I'd got it to include either IDLE or FINISHED status, as well as 12 status codes to include each task, but it lacked the possibility for the user to change their statuses manually, as that would discard the data about which task they were on (which needs updating manually separately from their status). I had to go through the whole program changing task codes for status codes.

- the print layout I’d been so proud of won’t display on the Command Prompt terminals installed on the university computers; they don’t accept the same format of escape codes as on my own laptop. For my "clear screen" function the impairment is only cosmetic, but when it comes to displaying the task log, printing it without the escape codes to position the cursor renders the data unreadable.

- additionally, my display style will only work on appropriately sized terminal windows. Online advice is to use the Curses package, putting me back where I started. Thankfully my tutor isn’t reliant on the university computers to mark my project, otherwise I'd have to have either abandoned my clear visual layouts, printing it as bare text, or use massively complex f-strings to recreate my tables a line at a time.

- when I updated PyCharm, which involved restarting it, having read that updating apps as soon as an update becomes available keeps them secure against the most recent methods of hacking, it deleted nearly a month of work. When the new version booted up, it used its new feature to link itself to my GitHub repository, but whereas I thought it would update my repository from my current version, it did the opposite and updated editor from the older draft I'd previously stored on GitHub. Thankfully I'd already had a bit of practice with git reset.

Conclusion

Reflecting honestly on my work, I've emerged with several take-aways.

In all truth, I need to read project briefs more thoroughly.

If I'd printed it out and annotated it, I would have had a clearer picture in my mind of how the program was going to have to function.

Some dead ends I went down could perhaps then have been averted.

I committed the rookie error of not backing up regularly. Especially once I (thought I had) learnt how to use Git, I started to become reliant on this, not realising that without a thorough understanding of the complexity of Git, it would probably be easier to just make back-ups as .txt files, notwithstanding the kudos of publishing on GitHub.

Next time I'll make a flow diagram as part of the planning process. My program quickly became labyrinthine in its complexity, and it was hard to keep track mentally of how it would jump between functions, and the order they were all in. After coding 2000+ lines I was aghast to realise my pretty table tracking the progress of specific tasks wouldn't entirely fit the criteria in the brief. It didn't make logical sense: how could it show the percentage each task was towards completion as well as the number of tasks in progress too, unless either it showed each instance of that task individually, or all the tasks had begun at the same time?

I'll have to swallow my pride and keep a pencil and pad of paper to hand, to keep a record of the principal functions, variables and the alterations in their names as they're passed as arguments to the multiple parameter names of the many functions.

I did well in contemplating the security of the program a great deal, having been reading up on cybersecurity and insecure apps. (Wong, 2024)

Passing a big array like my task log between functions exposes too much data too frequently, too recklessly. Better to section off individual variables or portions of data structures to pass to the functions as needed as arguments instead of the entire log.

Were this a real program enabling management of an industrial workplace, even just the worker IDs would be confidential. Companies have data protection standards they need to observe.

-o0o-

References

American Psychological Association. (2021, 2023, 2025). *Reference Examples*. <https://apastyle.apa.org/style-grammar-guidelines/references/examples>

Library Services. *APA 7th Referencing Guide.* University of Huddersfield. https://library.hud.ac.uk/pages/apareferencing/

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

BowlOfRed. (2022). *Instead of escaping the newline, how about just using explicit quoting?* [Comment on the online forum post *Pythonic way of printing multi-line string*s]. Python. https://discuss.python.org/t/pythonic-way-of-printing-multi-line-strings/19681/3

Cohen, D.X.., Groening, M., Keeler, K., &Katz, C. (Executive Producers). (1999 – present). *Futurama* [TV series].  Fox Broadcasting Company; Comedy Central;  Hulu.

ConnorWill. (2022). *Ansi Escape Sequences cheatsheet.* GitHub. <https://gist.github.com/ConnerWill/d4b6c776b509add763e17f9f113fd25b>

Faccioni, J.L. (2022). *5 Ways to Create a Dictionary in Python*. <https://learnpython.com/blog/create-dictionary-in-python/>

gsbabil, , Potdar, K., Santilli, C. & johnthagen (2010, edited 2021). *Am I missing the point here* [Comment on the online forum post *Remove specific characters from a string in Python*]. Stack Overflow. <https://stackoverflow.com/questions/3939361/remove-specific-characters-from-a-string-in-python>

Howell, A & Burns, S. (2023). *How to roll back Git code to a previous commit.* https://www.techtarget.com/searchitoperations/answer/How-to-roll-back-Git-code-to-a-previous-commit

Javatpoint. *Nested Tuples in Python*. <https://www.javatpoint.com/nested-tuples-in-python>

Library Services. *APA 7th Referencing Guide.* University of Huddersfield. https://library.hud.ac.uk/pages/apareferencing/

ModernMind. (2004). *Python Programming for Beginners Made Easy*. ModernMind Publications.

NKMK. (2023). *Get keys from a dictionary by value in Python*. https://note.nkmk.me/en/python-dict-get-key-from-value/

RealPython. (2017). *Python Function Argument Unpacking Tutorial (\* and \*\* Operators).* <https://www.youtube.com/watch?v=YWY4BZi_o28>

Ramos, LP. (2024). *Dictionaries in Python.* https://realpython.com/python-dicts/

Ronquillo, A (2019) . *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>

Schafer, C. (2017). *Python Tutorial for Beginners 5: Dictionaries - Working with Key-Value Pairs*. Retrieved Jan 31st, 2024 from <https://www.youtube.com/watch?v=daefaLgNkw0>

Stack Overflow community. (2018). *count the number of occurrences of a certain value in a dictionary in python?* Retrieved Jan 6th 2025.Found at https://stackoverflow.com/questions/48371856/count-the-number-of-occurrences-of-a-certain-value-in-a-dictionary-in-python

Stack Overflow community. (2009). *Git replacing LF with CRLF*. Retrieved Dec 29th, 2024 from <https://stackoverflow.com/questions/1967370/git-replacing-lf-with-crlf>

Stack Overflow community. (2008) [*How do I get my program to sleep for 50 milliseconds?*](https://stackoverflow.com/questions/377454/how-do-i-get-my-program-to-sleep-for-50-milliseconds)Retrieved Dec 30th, 2024. Found at <https://stackoverflow.com/questions/377454/how-do-i-get-my-program-to-sleep-for-50-milliseconds>

Stack Overflow community. (2014). *How do you get Python to detect for no input.* Retrieved Dec 31st, 2024. Found at <https://stackoverflow.com/questions/26226489/how-do-you-get-python-to-detect-for-no-input>

Stack Overflow community. (2020*) How do I print these hash symbols using a loop? [duplicate]*. Retrieved Jan 1st, 2025. Found at <https://stackoverflow.com/questions/64541968/how-do-i-print-these-hash-symbols-using-a-loop>

Stack Overflow community. (2011). *Linux Terminal Display and Python.* Retrieved Jan 7th, 2025. Found at

<https://stackoverflow.com/questions/7636984/linux-terminal-display-and-python>

Van Rossum, G., Warsaw, B. & Coghlan, A. (2001). *PEP 8 – Style Guide for Python Code.* Python. <https://peps.python.org/pep-0008/>

W3Schools. (1999 - 2025). *Python Tutorial*. <https://www.w3schools.com/python>

Wong, C. (updated 2024). *Learning the OWASP Top 10*. LinkedIn Learning. Retrieved 11th Aug 2024, from <https://www.linkedin.com/learning/learning-the-owasp-top-10-9364599/a-first-look-at-the-2021-owasp-top-ten-23316459?u=42583876>

Woodring, J. (2023). *Using ANSI Escape Codes to Color Your Terminal Output.* Youtube. <https://www.youtube.com/watch?v=yQ9Ns6Z4Q-s>