Virtual Pet Game “Timmy Tinker”

Project Portfolio

A group project for the
Open Campus SH course “Python - from Beginner to Practitioner”
handed in by

Slavomir Slavov
Abdulgawaad Saboukh
Judit Stas
Janne M. Scheffler
Anna Kaapke

01. August 2024

Contents

Introduction

In this portfolio, we would like to share our ideas behind our Virtual Pet Game project. 'We' stands for five people with very different academic backgrounds who all decided to participate in the Open Campus SH course 'Python - from beginner to practitioner' in order to educate themselves on Python Programming. The course is based on the Udemy Course "100 Days of Code: The Complete Python Pro Bootcamp". We mostly started with little to no previous knowledge in coding and improved our skills a lot in the course of the semester. This project serves as a final project to complete the course. Timmy became the first working name for our Virtual Pet, so we decided to call the project Timmy Tinker. Tinker is a wordplay based on the TKinter library that we used for our graphical user interface.

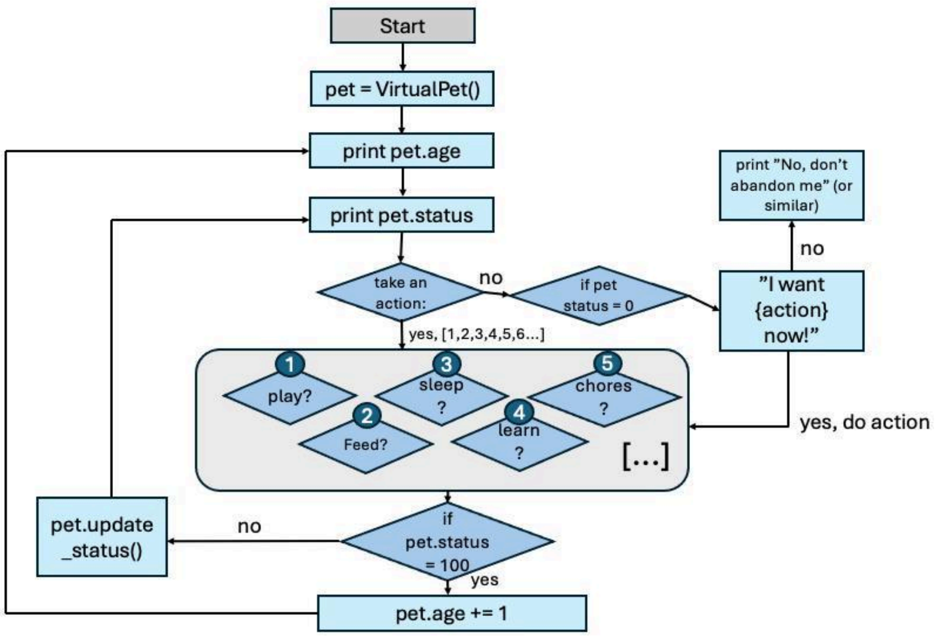
In order to provide the reader with a better overview of what is to be expected, we will shortly portray the goal of our project, and describe our working process on the next pages. Finally, we will also describe challenges we faced as well as improvements we think could be made for this project. The last pages are used for individual reflections of every group member on how they contributed to the project, which individual obstacles they faced and how they experienced working on their first software project. On the very last page, you can find a list of all files handed in at the end.

Goal of our project

The goal of our project was to provide a VirtualPet Programme comparable to a Tamagochi. The user should ideally be able to choose between different pets in the beginning. In order to keep the pet happily alive, the user can choose to perform different actions, just as feeding the pet, taking it to the vet, playing with the pet or letting it sleep. Some of the actions should even open up further possibilities, e.g. different (actual) mini games to play. Each action thereby has different effects on the pet status/data.
Besides the actions having an effect, also time will change the pets data making it grow older. The programme is supposed to show the real time already spent with the pet as well as the virtual in game time.
As the user might not always be able to play as long as he/she would love to - the pet data is supposed to be saved and reloaded if the user wishes to do so.
To make the whole programme more user friendly and enjoyable, a graphical user interface is used to create a playful experience.
On a technical side, the goal was to use the skills acquired in the course of the semester. This includes both actual coding, thinking of the logic by working with flow charts and meaningful use of aid provided by AI, just as e.g. ChatGPT.

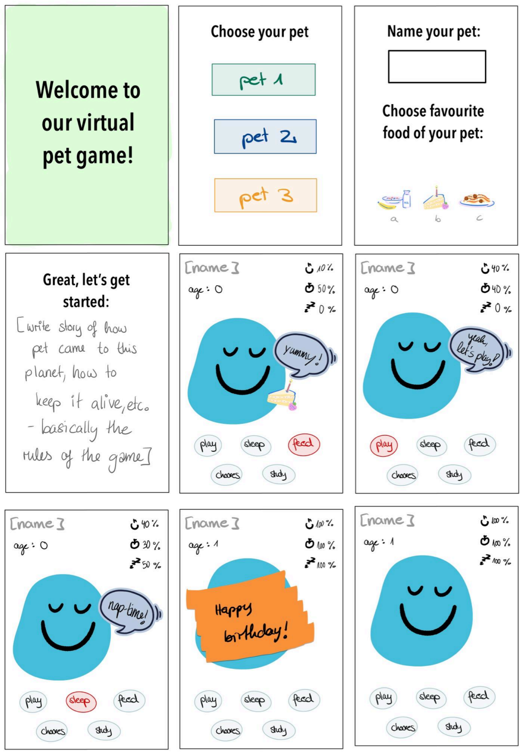
Our working process

In order to get started with our first 'big' project, we met regularly both during the regular course meetings and during other times of the week. We started working as a group in the middle of June and began by sketching our first programme draft as a flow chart diagram:



This flowchart is kept quite simple, but still shows the main structure/foundation we used to implement our project. Based on that, we discussed a lot of ideas and especially in the beginning our meetings were characterized by a lot of brainstorming leading to a huge variety of different ideas, e.g. on how to implement time, which actions the pet could do, etc.

A first challenge was to structure our working process in a meaningful way so that not all the ideas would get lost and also to agree on how we want to implement our ideas, and what actually is supposed to be our outcome. During the whole process we constantly came back to the question, what we wish our end product to look like in order to help us work target oriented. For a first presentation/pitch of our project in front of our course group, we had thought of the basic structure as shown in the diagram above and had first and simple design ideas (right), however we only had very little knowledge on how to implement a graphical user interface by then.



After we had thought of the basic knowledge of our project, we investigated, which libraries could be useful to implement the graphical user interface and after some trying and coding, we decided to go with Tkinter, as it easily allowed us to use buttons, labels and pictures, which could be made even more appealingly designed using the CustomTkinter extension.

We then started with a first programme draft and the library we had agreed on to develop our project further and further. As learned during the course, we decided to implement our code by using classes in Python. As soon as we had agreed on all that, it became easier for us to figure out specific tasks, we could divide among us, however, our group work was characterized by constantly having close contact within the group so that in the end a lot of methods and parts of the project were implemented by not only one person of the group. A more detailed overview of who worked on which code snippets can be found in the reflection chapter at the end of the portfolio. For communication purposes, we connected via WhatsApp and also used the Mattermost Channel to share files. In the beginning, we also used Mattermost to share code, however, we quickly changed to using GitHub, as working on a shared repository turned out to be a lot easier and more convenient. Some of us had no previous knowledge on git, but thanks to the help of one group member, everyone was able to use it efficiently after a short time.
Towards the end, we met several times a week to discuss our progress, give each other hints and to find next tasks. The working atmosphere was very understanding and helpful as well as trusting - so whenever someone had an idea, he/she was free to try and implement it, so that in the end, everyone was able to contribute with individual ideas.

Challenges

As stated above, especially in the beginning, it was challenging to define tasks and to break down our whole project into smaller ‘problems’ to solve, therefore we spent a lot of time discussing ideas but not necessarily with a productive outcome. Especially in the beginning, when we had only very little knowledge on graphical user interface implementation, it was also difficult to define tasks that were possible to implement with the skills we had learned so far. Structuring our process however became better and better the more knowledge we had and the more we discussed and defined our wanted outcome.
Hand in hand with the improvement of the structure we had in the working process itself, we were able to also structure our code more and more in a meaningful way. To avoid losing track of changes we did and to keep everything clear, we agreed on working with one file at first, however this became a long bit in the end, which could maybe be structured more professionally using for example something like the model-view-controller structure.

Certainly a challenge in the beginning was also to figure out a good algorithm on how the pet grows older, thus meaning how to implement the time dimension into our game. We decided to both let actions and actual time have an influence on the time in the end.
Also deciding to go with Tkinter took us a while as there are a lot of libraries which could be used. Experimenting with kinvy and PyQt6 as well as Tkinter (and CustomTkinter) helped to use Tkinter as a library of choice in the end, as both the other ones were quite challenging to use.
Another challenge, which could luckily be resolved quickly by using a JSON-file was how to store a pet's data, when you stop playing, so that the user could continue playing with his/her pet at a later time. Using the pet's name, it can be loaded back into the game.
Sometimes it was difficult to estimate, which technical possibilities we had with our skill background and it took some time to find a way in order to 'solve' the problem, however, a lot of knowledge gained through the Udemy course could be activated and helped us to tackle them.
Besides the technical challenges we faced, it was also not always easy to make as much progress as we wished for, as all of us had full time occupations beside the course, however, we still managed to have a lot of virtual meetings in the evenings to shortly discuss the current state.

Possible Improvements

There are still a few ideas and possibilities on how to improve our game. Some of them were not implemented because it would go beyond the scope of this project both time and skill wise. Here's a list of suggestions, we have to improve our Virtual Pet game:

- implement more buttons
- have pictures on all buttons
- more mini "games" both for playing (idea: simple sketchpad) and feeding the pet
- improve the mini games in order to make them more complex & improve the user experience
- include random events for the vet
- using several pictures to simulate movements of the pet (how about storage/garbage collection? How could we write efficient code?)
- structure the code better in order to have more clearly defined interfaces to implement additional methods
- implement a "finish" of the game (without letting the pet die!) - e.g. pet moves out and new pet moves in.
- create different levels of growing up and adjust functions according to the pet's age
- more functions: e.g. learning, meeting a friend, cleaning up, bathing/showering

- code random reactions of the pet e.g. that it mentions “I’m hungry” when the hunger parameter undergoes a certain value
- different influence on the status if mini games are lost
- keep more structure in our project folder through moving all pictures in a respective subfolder etc.

Especially in the field of efficiency as well as the graphical user interface, improvements could be done. However, we’re quite satisfied with our current version already.

Individual task descriptions and reflections

As stated above, we implemented the project in close collaboration within the group, so a lot of methods and code snippets can actually not be assigned to one person only. To give the course leaders a brief overview, each group member wrote a few lines to outline their focus and contribution to the project.

Abdulgawaad Saboukh (Goda)

Participation: *I looked up a basic code to start with. This code was the foundation upon which we built the program. This was aided by ChatGPT. I brainstormed with the group about which functions/actions could be added to the program. We started with playing, sleeping, and feeding. Later on, the group added other functions to the program.*
I proposed and implemented the initial idea of how the pet should age. The proposal was that every 4 actions, the pet ages one day. Later on the group coined this with a real-time aging. The combination of both ended up to be the method of aging.
I looked up different Graphical User Interface (GUI) libraries to implement graphics into our program. We ended up with using tkinter. I provided the very first code to implement a very primitive tkinter library. Later on the group managed to make it something super decent.
With the help of chatGPT and hours and hours of troubleshooting, I was able to implement pictures that pop up after each action is executed. I used DALL-E to generate pictures for 4 different pets (sheep, cat, shrimp, chicken). Each pet had 5 different pictures that are bound to different actions (eat, sleep, play, watch tv, visit vet). Code was implemented in a way that the picture pops up after the action is takes, stay on for 3 seconds, and then vanishes automatically.

Reflection:

The goal of this project to me was to have a collaborative effort and be able to create a simple game. In my mind, this should not be a complex game because complexity requires better skills, which we do not have yet. I believe we managed to create something good, and I would say, more complicated than what I initially thought we would do.

Judit Stas

Participation: *I implemented the functions "create_animal_selection_widgets", "select_animal" and "load_images". I also improved others as "create_widgets".*
Mainly I worked on the GUI (using the CustomTkinter library) enhancing it with DALL-E generated picture-buttons (FEED, PLAY, LOAD, START, PLAY, SLEEP, VET, CHECK STATUS, QUIT), arranging them in an appealing manner and improving the player's experience by giving him the option to choose one pet of four given (sheep, shrimp, cat and chicken).

Reflection:
When I look at the very simple flowchart that I prepared for our first group meeting a few weeks ago, I am surprised – because we have clearly exceeded my expectations. Our Virtual Pet Game has become significantly more varied and challenging. Although due to time constraints, we had to focus strongly on what we could realistically implement within the given time. There were many more interesting ideas for further development of the game. I also found a canvas-widget with a simple sketch-pad, which I thought would be nice to implement as an additional mini-game.

The same applied to the development of the GUI. With very modest prior knowledge, I was able to implement a functioning and – in my opinion – visually appealing solution with everything I could gather in terms of input and help (stakeholder flow, course resources, ChatGPT, and a lot of trial-and-error); with self-created picture buttons and the ability for the player to choose their pet. Here too, I started very simply, with just the generated image of a cool chicken that was displayed to the player after entering the desired name. For me, it was quite a challenge to find my way around the libraries – but the fastest way to learn is indeed just by trying things out. And working together with such inspiring and friendly team members in a really collaborative atmosphere was a pleasure, which also helped overcome many difficulties.

Slavomir Slavov

Participation:
I proposed and introduced GitHub and the Git concept to the group, managed the repository, and assisted others in navigating and working with it.
I also introduced and worked with the following libraries in our project: time, json, os, and CustomTkinter.

- *Time: I created functions that represent the real time when starting the game, as well as a virtual pet time used to age the pet, enhancing the user experience.*
- *JSON and OS: I utilized these libraries to implement save and load functions for the pets within our project.*
- *CustomTkinter: I discovered this library while researching Tkinter and used it to create new buttons with widgets, arranged the buttons on different grids, and added a background theme to improve readability. The rest of the team later enhanced our GUI/UX by adding photos to the buttons instead of text.*

Additionally, I worked on debugging functions that encountered issues along the way and merged separate files into our main codebase. I occasionally used ChatGPT to help generate or review my code and spent a considerable amount of time debugging.

Reflection:
I am very happy with what we achieved and proud to be part of my group. We had productive meetings and created a great product that has room for further improvement. My focus was primarily on the functionality of our code, and I wish I could have spent more time enhancing the user experience and game experience. While ChatGPT was helpful, debugging was necessary.

Anna Kaapke

Participation: *In the beginning, I approached the project by drawing a possible version of how it could look in the end to reflect on requirements needed to be fulfilled to realize our virtual pet project and to be able to participate in the drawing of our flow chart diagram and the investigation for a suitable library to implement our graphical user interface.*
In terms of coding, I participated in the project by implementing different features to some functions, just as e.g. the "You won!"/"The pet won!" outputs on the mini game screens and the automatic abortion of the mini games in order to return to the main game (toplevel widget). For that I investigated the Pygame documentation.
I was also involved in placing buttons in our main game, for example in collaboration with Janne I implemented the remove and show feed and play button functions. Especially in the beginning, when everyone started to work in different files, I helped to merge the code into one main file and assisted to maintain the logic for the pet actions.
Furthermore, I did some debugging as well as small tasks, just as e.g. adding the happy birthday picture to our game or a close button for the pop up images triggered by clicking on the action buttons.
A huge focus for me was proper documentation of our code leading to me spending some time writing docstrings to all functions and to add the markdown documentation in our jupyter notebook files. Thereby, the code became more structured and hopefully also people not involved in the project can follow along more easily. In addition I often wrote down key notes during our meetings to help us structure our working process and develop the logic behind our game. To keep our main ideas, thoughts, etc. together and easily accessible, I wrote this accompanying portfolio in close collaboration and evaluation with the other group members.

Reflection: *Doing this project was a lot of fun and a great learning experience! Working in this group was a total joy, as it was possible to learn and to bring in my own ideas within a very safe, helpful and supportive environment. Personally, I learned a lot by investigation and using new libraries (e.g. Tkinter and Pygame), but also by collaboratively going over each other's code in order to improve and debug it. Seeing my skills being efficient enough to help create our project was a confidence boost and a huge motivation to code more.*

Janne Scheffler

Our team started by developing a comprehensive flowchart encapsulating our core ideas. Each member then searched for a foundational code snippet to base our project on. We accumulated multiple code files in our Git repository and consolidated the essential ideas *into a single file. From that point, we worked collaboratively on one codebase to streamline contributions and track changes effectively.*

Contributions and Key Tasks: Function Implementation and Logic Optimization
My primary contributions revolved around developing and optimizing the core functions and complexities of the game logic. This involved defining how different operations affect the pet's parameters (hunger, happiness, health, tiredness, intellect). Each parameter's value was constrained between 0 and 100. Initially, this was handled by multiple lines of code, which I refactored into single lines using min and max functions to enforce these boundaries more efficiently.

I focused on the feeding functionality, integrating food choices via buttons on the Tkinter canvas. This involved ensuring that the GUI accurately reflected the available food options and updated the pet's status accordingly.

For the game function I opted for three mini games that were scripted in separate files to maintain code clarity. The subprocess-module was used to integrate those files into the main application. Those mini-games were initially meant to be based on the turtle-module. Early game ideas included an easy version of snake, a maze-run and also a race similar to the one we coded during the Udemy-course. However, the code quickly turned into its own project and due to time constraints I decided to give pygame a try – which worked well. Even though the complexity of each individual game is limited, they enhance the overall user-experience. The sys-module was recommended by OpenAI to exit the mini-games and thus included in the code. The random-module has been imported in order to introduce random events that happen during playing with the digital pet to add an element of unpredictability. The scores of the parameters adjust accordingly.

Furthermore, I set focus on the TV-function with the idea to have random TV shows that again influence the scores of certain parameters depending on the show type. Initially it was planned to code that function like the random_event function, however it did not work out since one would rather get a hold of events that have to fit to every show than really be able to trigger a random show itself. ChatGPT set the idea to work with a dictionary instead which eventually allowed to implement the intended randomness for the TV show.

I contributed to the GUI design, particularly in structuring the canvas and positioning the buttons. Together with Anna addressed the issue of dynamically showing and hiding buttons (e.g., play and feed buttons) using the forget() method, ensuring a seamless user experience. Additionally, I collaborated with the team to synchronize changes in the VirtualPet and VirtualPetApp classes, ensuring consistent and bug-free code execution. The team collectively brainstormed and resolved the age-related features of the digital pet, which another team-member successfully worked into the code. Our collaborative efforts resulted in a robust digital pet application with well-integrated features and a user-friendly interface.

Reflection: *This group project was very exciting and challenging to me! As a PhD student I usually used python for data visualization. To program an app is a new skill that urged me and my group members to dig into the possibilities of different python libraries and notice the capabilities of the language. During the project we had a relaxed atmosphere in our group where problems were solved together and frequent communication took place. This project enhanced my confidence in coding and motivated me to continue exploring what python has to offer and how I could apply it to my work.*

List of attached files

All our project files can also be accessed online via our GitHub repository: https://github.com/TamagochiSS/Tamagotchi
Pictures used in the project were either created by using DALL-E AI, downloaded from Pixabay or created by ourselves in order to not hurt copyright guidelines.

Code files:

Timmy_tinker_documentation.ipynb
This file holds our main programme logic and a detailed description of all methods used to implement our classes.

hide_and_seek_documentation.ipynb
Documentation of the hide and seek game.

hide_and_seek.py
Hide and seek game triggered when clicking the play hide and seek button.

ball_game_documentation.ipynb
Documentation of the ball game.

ball_game.py
Ball game triggered when clicking the play ball game button.

memory_game.ipynb
Documentation of the memory game.

memory_game.py
Memory game triggered when clicking the play memory button.

pets_data.json
JSON file to save the pet data and to be able to reload it.

simple_sketchpad.py
Another mini game idea, which is not yet added to the main programme.

Besides the code files, there are various pictures used for the buttons and the chosen pet (four different options: sheep, chicken, shrimp, cat).