

POLITECHNIKA KRAKOWSKA

Python Programming Project

Ana Margarida de Freitas

Helin Salduz

Sude Yilmaz



ERASMUS+

Supervisor: Radosław Kycia

June 12, 2024

Python Programming Project

Ana Margarida de Freitas

**Helin Salduz
Sude Yilmaz**

ERASMUS+

June 12, 2024

Abstract

This project tasks the group with several objectives related to the development of a chat-based decision tree game developed with python, aimed at providing an engaging and interactive experience for the user. The game employs a simple text-based interface where players navigate through a series of decisions, similar to a choose-your-own-adventure story. By interacting with the game through text input, players are introduced to fundamental programming concepts such as conditional statements, user input handling, and basic game logic.

The core of the game lies in its decision tree structure, where each player choice leads to different outcome, creating a branching narrative. Through this structure, players gain hands-on experience with flow control mechanisms, understanding how different decisions influence the progression of the game. Additionally, the game fosters problem-solving skills as players assess the consequences of their choices and adapt their strategy accordingly.

Furthermore, the game can be easily extended and customized to cover a wide range of topics, from basic syntax and control structures to more advanced programming principles.

Keywords: Python, programming, game, chat

Contents

1	Introduction	1
2	Theoretical Part	3
3	Practical part	5
4	Summary	15

Chapter 1

Introduction

Within the context of the Python Programming course, there was a proposal to develop any functional program using the language mentioned above. This report will detail the journey and outcomes of this hands-on experience, accompanied by visual documentation of the results. Throughout the report, we will reference the task made and the obtained results.

The core aim of the project is to showcase the immersive experience of an Erasmus student in Krakow. Through the journey, our Erasmus students will explore the intricacies of the city, the new lifestyle and the discovery process of living abroad.

Our project encompasses the development of a chat-based decision tree game that transports players into the shoes of an Erasmus student newly arrived in Krakow. Within this digital realm, players will navigate through a series of decisions mirroring real-life scenarios encountered by Erasmus students.

As for the methodology used, it's the following:

- **Conceptualization:**

We begin by conceptualizing the game design, crafting a storyline that encapsulates the Erasmus experience in Krakow, and mapping out the decision tree structure.

- **Development:**

Leveraging the versatility of Python, we embark on the development phase, focusing on implementing the chat-based interface, decision tree logic, and gameplay mechanics.

- **Testing and Iteration:**

Rigorous testing ensues, aimed at identifying and rectifying any glitches or inconsistencies. Feedback from testers will inform iterative improvements to refine the game's design and functionality.

- **Documentation:**

We compile comprehensive documentation detailing the project's design, implementation intricacies, and user instructions to facilitate seamless integration into Python programming classes.

- **Presentation:**

The final project is presented in Python programming class, showcasing its immersive gameplay.

Chapter 2

Theorical Part

In order to fulfill the necessities of our project, it was a deal breaker access to the libraries and technologies of Python language. This section describes the two main theoretical aspects of this program.

- Graphical User Interface (GUI) Design

From the research made, it was concluded that it's common in this type of program to rely on events to trigger functions. The *tkinter* library handles these events, like buttons, with callback functions assigned to widgets.

Images are loaded and manipulated for display within the application. While resizing images, it is important to maintain the aspect ratios to avoid distortion.

- Game Design Concepts

A sequence of narrative steps, each represented by a different GUI update, is encountered as the game advances. To help the player go through the narrative, text and graphics are updated.

The player is presented with numerous options throughout the game, and each one can lead to a different path or result. The story is advanced through button clicks, making for an interactive experience.

The game employs timed delays (a technique from *tkinter*) to regulate the story's progression, introducing dramatic pauses to create impact and directing the player's speed.

Chapter 3

Practical part

It starts with the importing and initial setup. It creates the main window, sets the title, the dimensions, and initializes the GUI components.

```
1 import tkinter as tk
2
3 class GameApp:
4     def __init__(self, master):
5         self.master = master
6         master.title("Krakow Chronicles")
7
8         master.geometry("800x600")
9         master.minsize(600, 400)
10        master.resizable(True, True)
11
12        self.title_font = ("Helvetica", 15, "bold")
13        self.custom_font = ("Helvetica", 10, "bold")
14        self.button_font = ("Helvetica", 8)
15
16        master.configure(bg='#f0f0f0')
17
18        self.label = tk.Label(master, text="Welcome to Krakow Chronicles!",
19                               font=self.title_font, bg='#f0f0f0')
20        self.label.pack(pady=10)
21
22        self.image_label = tk.Label(master, bg='#f0f0f0')
23        self.image_label.pack()
24
25        self.button = tk.Button(master, text="Start", command=self.
26                                start_game, font=self.custom_font, bg='#4CAF50',
27                                fg='white', padx=10, pady=5)
```

```

26         self.button.pack(pady=10)
27
28         master.bind('<Return>', self.next_text)

```

Then the goal is to load and resize the images. The images are loaded from the files and resized using the *resize_image* method. However, if it's not possible to load an image the program interprets it as an error. While resizing, the aspect ratios were maintained. The exit button to leave the game is also incorporated, so the user can finish the game at any time.

```

1         try:
2             self.images = {
3                 "welcome": self.resize_image(tk.PhotoImage(file="
welcome_image.png"), 400, 100),
4                 "square": self.resize_image(tk.PhotoImage(file="
square_image.png"), 400, 100),
5                 "basilica": self.resize_image(tk.PhotoImage(file="
basilica_image.png"), 400, 100),
6                 "cloth_hall": self.resize_image(tk.PhotoImage(file="
cloth_hall_image.png"), 400, 100),
7                 "jazz_club": self.resize_image(tk.PhotoImage(file="
jazz_club_image.png"), 400, 100),
8                 "night_market": self.resize_image(tk.PhotoImage(file="
night_market_image.png"), 400, 100),
9                 "performers_image": self.resize_image(tk.PhotoImage(file="
performers_image.png"), 400, 100),
10                "castle": self.resize_image(tk.PhotoImage(file="
castle_image.png"), 400, 100),
11                "museum": self.resize_image(tk.PhotoImage(file="
museum_image.png"), 400, 100),
12                "cafe": self.resize_image(tk.PhotoImage(file="cafe_image.
png"), 400, 100),
13                "nightlife_image": self.resize_image(tk.PhotoImage(file="
nightlife_image.png"), 400, 100),
14                "nightclub": self.resize_image(tk.PhotoImage(file="
nightclub_image.png"), 400, 100),
15            }
16            print("Images loaded successfully.")
17        except Exception as e:
18            print("Error loading images:", e)
19            self.label.config(text=f"Error loading images: {e}")
20            return

```

```

21
22     self.text_queue = []
23     self.next_action = None
24     self.option_buttons = []
25     self.canvas = tk.Canvas(master, width=50, height=50, bg='#f0f0f0',
highlightthickness=0)
26     self.canvas.pack(side=tk.RIGHT, padx=10, pady=10)
27     self.draw_exit_button()
28
29     def draw_exit_button(self):
30         self.canvas.create_oval(5, 5, 45, 45, fill="red", outline="black")
31         self.canvas.create_text(25, 25, text="X", fill="white", font=("
Helvetica", 12, "bold"))
32
33         self.canvas.bind("<Button-1>", self.exit_game)
34
35     def exit_game(self, event):
36         self.master.destroy()
37
38     def resize_image(self, image, width, height):
39         original_width = image.width()
40         original_height = image.height()
41
42         aspect_ratio = original_width / original_height
43         if aspect_ratio > 1:
44             new_width = width
45             new_height = round(new_width / aspect_ratio)
46         else:
47             new_height = height
48             new_width = round(new_height * aspect_ratio)
49
50         return image.subsample(round(original_width / new_width), round(
original_height / new_height))

```

The *start_game* method initializes the game by updating the interface with a welcome image, queuing some introductory text messages, and preparing a button labeled "Explore Krakow" that, when clicked, will trigger further game actions related to exploring Krakow. This method sets up the initial state and user interface for a player beginning their adventure in the game.

```

1     def start_game(self):
2         self.update_image("welcome")

```

```

3     self.queue_texts(["You are an Erasmus student arriving in Krakow,
ready for adventure.",
4                       "Your goal is to explore the city and immerse
yourself in its culture.",
5                       "Let's begin!"])
6     self.next_action = lambda: self.update_button("Explore Krakow",
self.explore_krakow)

```

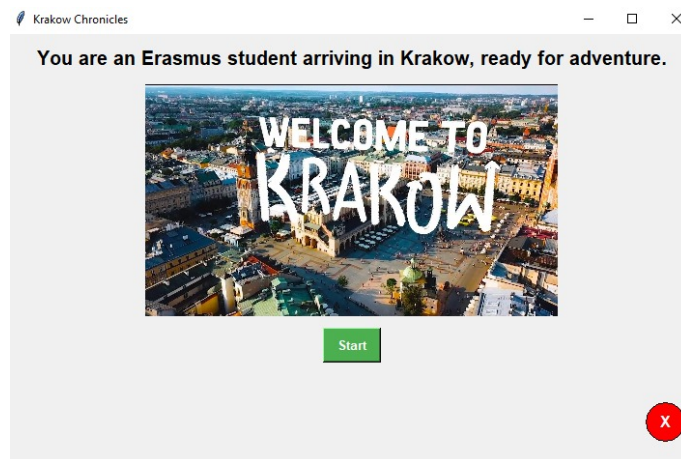


Figure 3.1: Welcome menu

These methods manage displaying text messages, updating button labels and actions, and changing images. The *queue_texts* method initializes a queue of texts and starts displaying them one by one. The *next_text* method displays the next text in the queue or executes a next action if the queue is empty. The *update_button* method changes the button text and action, while the *update_image* method updates the displayed image based on a given key.

```

1     def queue_texts(self, texts):
2         self.text_queue = texts
3         self.next_text(None)
4
5     def next_text(self, event):
6         if self.text_queue:
7             self.label.config(text=self.text_queue.pop(0))
8         elif self.next_action:
9             self.next_action()
10            self.next_action = None
11
12    def update_button(self, text, command):
13        self.button.config(text=text, command=command)

```

```

14
15 def update_image(self, image_key):
16     print(f"Updating image: {image_key}")
17     if image_key in self.images:
18         self.image_label.config(image=self.images[image_key])
19         self.image_label.image = self.images[image_key]
20     else:
21         print(f"Image key {image_key} not found in images.")

```

So it jumps to the actual game, it calls the *clear_buttons* to remove any existing buttons. Then updates the image to the main square of Krakow with the function *update_image*. Queues the text describing the main square and nearby locations and finishes by presenting the options to the player.

```

1 def explore_krakow(self):
2     self.clear_buttons()
3     self.update_image("square")
4     self.queue_texts(["\nYou are in the main square of Krakow.",
5                       "To your left, you see the majestic St. Mary's
6                       Basilica.",
7                       "To your right, there's the historic Cloth Hall."
8                       ,
9                       "Straight ahead, you see bustling streets with
10                      nightlife options.",
11                      "Where would you like to go?"])
12     self.next_action = self.show_day_options

```

Similarly to the previous part, the buttons are cleared and the new options, for the day activities show up.

```

1 def clear_buttons(self):
2     for button in self.option_buttons:
3         button.destroy()
4     self.option_buttons = []
5
6 def show_day_options(self):
7     self.clear_buttons()
8
9     self.button1 = tk.Button(self.master, text="St. Mary's Basilica",
10                             command=self.visit_st_marys, bg='#008CBA',
11                             fg='white', font=self.button_font, padx=5,
12                             pady=3)

```

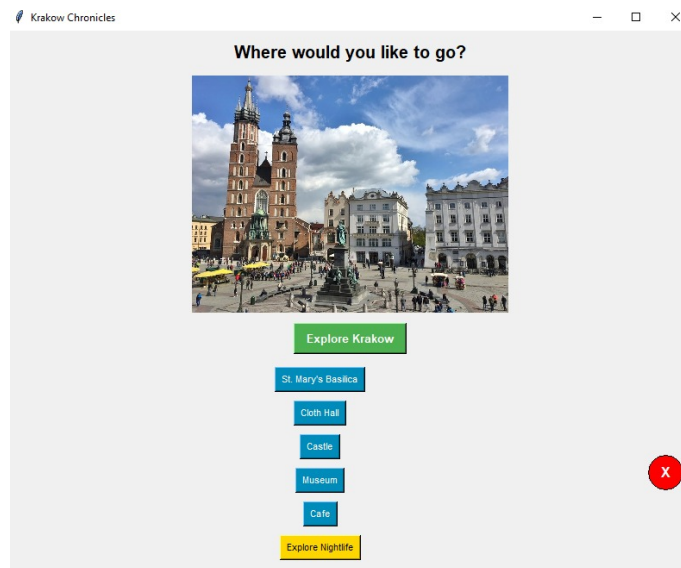


Figure 3.2: Exploring Krakow options

```

11     self.button1.pack(pady=5)
12     self.option_buttons.append(self.button1)
13
14     self.button2 = tk.Button(self.master, text="Cloth Hall", command=
15     self.visit_cloth_hall, bg='#008CBA',
16                               fg='white', font=self.button_font, padx=5,
17                               pady=3)
18     self.button2.pack(pady=5)
19     self.option_buttons.append(self.button2)
20
21     self.button3 = tk.Button(self.master, text="Castle", command=self.
22     visit_castle, bg='#008CBA', fg='white',
23                               font=self.button_font, padx=5, pady=3)
24     self.button3.pack(pady=5)
25     self.option_buttons.append(self.button3)
26
27     self.button4 = tk.Button(self.master, text="Museum", command=self.
28     visit_museum, bg='#008CBA', fg='white',
29                               font=self.button_font, padx=5, pady=3)
30     self.button4.pack(pady=5)
31     self.option_buttons.append(self.button4)
32
33     self.button5 = tk.Button(self.master, text="Cafe", command=self.
34     visit_cafe, bg='#008CBA', fg='white',
35                               font=self.button_font, padx=5, pady=3)
36     self.button5.pack(pady=5)
37     self.option_buttons.append(self.button5)

```



```
33
34     self.button6 = tk.Button(self.master, text="Explore Nightlife",
35                               command=self.explore_nightlife, bg='#FFD700',
36                               fg='black', font=self.button_font, padx=5,
37                               pady=3)
38     self.button6.pack(pady=5)
39     self.option_buttons.append(self.button6)
```

With the same logic as before, it is defined the functions for each activity in the following way:

```
1  def visit_st_marys(self):
2      self.clear_buttons()
3      self.update_image("basilica")
4      self.queue_texts(["\nYou enter St. Mary's Basilica and are
5                          awestruck by its beauty.",
6                          "You spend some time admiring the intricate
7                          details of the altar.",
8                          "As you leave, you notice a group of street
9                          performers outside.",
10                         "Would you like to watch their performance?"])
11     self.next_action = lambda: self.update_button("Watch Performance",
12                                                    self.watch_performance)
```

For the rest of the program it is used the same logic of the other components. It is also defined the night activities.

```
1  def watch_performance(self):
2      self.clear_buttons()
3      self.update_image("performers_image")
4      self.queue_texts(["\nYou join the crowd and watch the street
5                          performers.",
6                          "Their music and acrobatics captivate you, and
7                          you find yourself smiling.",
8                          "After the performance, you feel energized and
9                          ready for more adventures."])
10     self.next_action = lambda: self.update_button("Explore More", self.
11                                                    explore_krakow)
12
13  def visit_cloth_hall(self):
14      self.clear_buttons()
15      self.update_image("cloth_hall")
```

```

12     self.queue_texts(["\nYou enter the Cloth Hall and browse through
various stalls selling souvenirs.",
13                     "You buy a beautiful handmade ornament as a
keepsake.",
14                     "Would you like to explore more of the city?"])
15     self.next_action = lambda: self.update_button("Explore More", self.
explore_krakow)
16
17 def explore_nightlife(self):
18     self.clear_buttons()
19     self.update_image("nightlife_image")
20     self.queue_texts(["\nYou decide to explore Krakow's vibrant
nightlife.",
21                     "You can choose to visit a jazz club, a night
market, or a popular nightclub.",
22                     "Where would you like to go?"])
23     self.next_action = self.show_nightlife_options
24
25 def show_nightlife_options(self):
26     self.clear_buttons()
27
28     self.button1 = tk.Button(self.master, text="Jazz Club", command=
self.visit_jazz_club, bg='#FFD700', fg='black',
29                             font=self.button_font, padx=5, pady=3)
30     self.button1.pack(pady=5)
31     self.option_buttons.append(self.button1)
32
33     self.button2 = tk.Button(self.master, text="Night Market", command=
self.visit_night_market, bg='#FFD700',
34                             fg='black', font=self.button_font, padx=5,
pady=3)
35     self.button2.pack(pady=5)
36     self.option_buttons.append(self.button2)
37
38     self.button3 = tk.Button(self.master, text="Nightclub", command=
self.visit_nightclub, bg='#FFD700', fg='black',
39                             font=self.button_font, padx=5, pady=3)
40     self.button3.pack(pady=5)
41     self.option_buttons.append(self.button3)
42
43     self.button4 = tk.Button(self.master, text="Back to Daylight",
command=self.back_to_daylight, bg='#FFD700', fg='black',
44                             font=self.button_font, padx=5, pady=3)
45     self.button4.pack(pady=5)

```

```
46         self.option_buttons.append(self.button4)
47
48     def back_to_daylight(self):
49         self.clear_buttons()
50         self.explore_krakow()
51
52     def visit_jazz_club(self):
53         self.clear_buttons()
54         self.update_image("jazz_club")
55         self.queue_texts(["\nYou enter a cozy jazz club and enjoy a night
56 of smooth jazz music.",
57                             "The atmosphere is relaxed and you feel at ease."
58 ,
59                             "Would you like to explore more of Krakow's
60 nightlife?"])
61         self.next_action = lambda: self.update_button("Explore More", self.
62 explore_nightlife)
63
64     def visit_night_market(self):
65         self.clear_buttons()
66         self.update_image("night_market")
67         self.queue_texts(["\nYou wander through a bustling night market
68 filled with food stalls and handmade crafts.",
69                             "You taste some delicious local delicacies and
70 buy a unique souvenir.",
71                             "Would you like to explore more of Krakow's
72 nightlife?"])
73         self.next_action = lambda: self.update_button("Explore More", self.
74 explore_nightlife)
75
76     def visit_castle(self):
77         self.clear_buttons()
78         self.update_image("castle")
79         self.queue_texts(["\nYou arrive at the historic Wawel Castle and
80 join a guided tour.",
81                             "The guide tells you fascinating stories about
82 the castle's history.",
83                             "Would you like to explore more of the city?"])
84         self.next_action = lambda: self.update_button("Explore More", self.
85 explore_krakow)
86
87     def visit_museum(self):
88         self.clear_buttons()
89         self.update_image("museum")
```

```
79         self.queue_texts(["\nYou visit the National Museum and explore
various exhibits showcasing Polish art and culture.",
80                             "You spend hours admiring the artworks and
learning about the history.",
81                             "Would you like to explore more of the city?"])
82         self.next_action = lambda: self.update_button("Explore More", self.
explore_krakow)
83
84     def visit_cafe(self):
85         self.clear_buttons()
86         self.update_image("cafe")
87         self.queue_texts(["\nYou enter a charming cafe and enjoy a cup of
coffee and a delicious pastry.",
88                             "You relax and soak in the atmosphere, feeling
refreshed.",
89                             "Would you like to explore more of Krakow's
nightlife?"])
90         self.next_action = lambda: self.update_button("Explore More", self.
explore_nightlife)
91
92     def visit_nightclub(self):
93         self.clear_buttons()
94         self.update_image("nightclub")
95         self.queue_texts(["\nYou head to a popular nightclub and dance the
night away.",
96                             "The music is loud and the energy is high, making
it an unforgettable experience.",
97                             "Would you like to explore more of Krakow's
nightlife?"])
98         self.next_action = lambda: self.update_button("Explore More", self.
explore_nightlife)
99
100 root = tk.Tk()
101 app = GameApp(root)
102 root.mainloop()
```

Chapter 4

Summary

This project entails developing a chat-based decision tree game using Python to provide an engaging and interactive experience for users. The game employs a simple text-based interface where players make decisions akin to a choose-your-own-adventure story, aiming to introduce fundamental programming concepts such as conditional statements, user input handling, and basic game logic.

The main objective is to create an immersive experience for an Erasmus student in Krakow, guiding them through various real-life scenarios and decisions they might face. The game structure uses a decision tree, allowing each choice to lead to different outcomes and creating a branching narrative that helps players understand flow control mechanisms and the impact of their decisions.

For this to be possible it was crucial the content learnt in Python Programming classes. The group values this project for the way it was possible to incorporate the reality lived into a fun approach.

For future steps, it would be incorporated a larger network of questions or even implement machine learning, so that the user could provide their own answers. Then a testing period, so that the users could report the relevance of the questions and also provide new ideas to be included.

Therefore, it could have some social impact in organizations like ESN since they could provide this app to their new income students, for them to get to know the city.