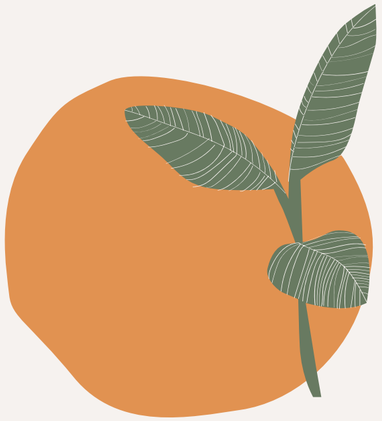
Quizz

-Joshua
-Lukas
-Adam

Jeu où le but est de répondre le plus de manière correcte à un nombre x de questions. Et il est possible d’y jouer avec des microbits, à distance.

Sommaire :



Structure générale

- -Jeu
- -Microbits
- -Base de donnée

Cahier des charges

V1	• Création et peuplement de la base de données. • Interrogation et modification de la base de données en python plus gestion basique du jeu en python. 1. Récupération toutes les questions d'un certain type et d'un niveau et choisir une aléatoirement. 2. Pouvoir crée des lignes pour crée des profils. 3. Pouvoir modifier les lignes des tables. 4. 10 bonnes réponses pour gagner. 5. Tester les réponses. 6. Parti graphique primaire (dans la console) • Gestion du multi-joueurs grâce au micro:bit. 1. Pouvoir géré la sélection des questions à l'aide des boutons A et B de la micro:bit. 2. Pouvoir afficher les différentes réponses sur la micro:bit. 3. Pouvoir jouer en multi-joueurs grâce au micro:bit.
V2	• Création de la partie graphique avec tkinter. 1. Pouvoir afficher les réponses et la question. 2. Pouvoir afficher les bonnes réponses et mauvaise réponse. 3. Pouvoir afficher un chronomètre

Repartition

-Microbits -Jeu -Base de donnée

	Adam H	Joshua	Lukas
V1 :	Gestion du multi-joueurs grâce au micro:bit.	Gestion du jeu plus mise en commun des programmes en python.	Création, Interrogation et peuplement de la base de données.
V2 :	Amélioration du jeu avec de nouvelle fonctionnalité grâce au micro:bit.	Création de la partie graphique avec tkinter.	Ajoute de nouvelle question dans la base de données.

Méthodes de travail

- **Discord**
- **Github**

Parti personelle

- **Class Start :**

```
def choose_player(self)->None:
    """Add a single player to the game"""
    self.microbits.get_player()
    self.player_nb += 1
    self.player_number.config(text=f"{self.player_nb} players")
```

```
def menu_quizz(self)->None:
    """Create the tab above the window"""
    #create menu
    menu_obj = Menu(self.root)

    #Add command to tab
    questions_menu = Menu(menu_obj, tearoff=0)

    for i in range(10):
        questions_menu.add_command(label =str(i+1), command =lambda i=i :self.questions_modifier(i+1))

    #Create each tab
    menu_obj.add_cascade(label ="questions", menu=questions_menu)

    #update
    self.root.config(menu = menu_obj)
```

Parti personelle

- **Class Game:**

```
def check_answer(self, choice, player = None)->None:
    """Check if correct answer"""
# Get the current question from the quiz_data list
    question = self.quiz_data[self.current_question]
    selected_choice = self.choice_btns[choice].cget("text")

# Check if the selected choice matches the correct answer
    if selected_choice == question["answer"]:
    # Update the score and display it
        self.update_score(choice)
        if self.nb_players > 0:
            self.feedback_label.config(text="Correct!", foreground="green")
    else:
        if self.nb_players > 0:
            self.feedback_label.config(text="Incorrect!", foreground="red")

# Disable all choice buttons and enable the next button
    for button in self.choice_btns:
        button.config(state="disabled")
    self.next_btn.config(state="normal")
```

Parti personelle

- **Class Game:**

```
def show_question(self)->None:
    """Display the current question and the answer"""
# Get the current question from the quiz_data list
    question = self.quiz_data[self.current_question]
    self.qs_label.config(text=str(question["question"])) #bug, overload to fix, pain in the ass

# Display the choices on the buttons
    choices = question["choices"]
    for i in range(len(self.quiz_data[self.current_question]['choices'])):
        self.choice_btns[i].config(text=choices[i], state="normal") # Reset button state

# Clear the feedback label and disable the next button
    self.feedback_label.config(text="")
    self.next_btn.config(state="disabled")
def update_score(self, player_answered = None )->None:
    """Update the score"""

    #Will add +1 to the player who answered correctly
    if self.nb_players > 0:
        if len(self.players_list) == 0: #create the list of score if None exist
            for player in self.players_answer:
                self.players_list[str(player)] = 0
        self.players_list[str(player_answered)] += 1

    else:
        #Add 1 and display the score if solo
        self.score += 1
        self.score_label.config(text="Score: {}/{}".format(self.score, len(self.quiz_data)))
```

CONCLUSION

Pour conclure ce projet a été enrichissant, appris pas mal sur la programmation et certains modules, techniques mais aussi le travail en groupe.

Le seul véritable problème étant le manque de communication ralentissant le projet.