

```
from abc import ABC, abstractmethod

class Stage:
    def __init__(self, stage_name):
        self.__stage_name = stage_name

    def get_name(self):
        return self.__stage_name

class Character(ABC):
    def __init__(self, character_name, stage):
        self.character_name = character_name
        self.stage = stage

    @abstractmethod
    def get_best_stage(self):
        pass

    @abstractmethod
    def get_reason(self):
        pass

class Steve(Character):
    def __init__(self):
        super().__init__("Steve", Stage("Small Battlefield (SBF)"))

    def get_best_stage(self):
        return self.stage.get_name()

    def get_reason(self):
        return "You get to plank and edgeguard with blocks here. You get stone from mining here, and it's long enough to camp, but not too long."

class Sonic(Character):
    def __init__(self):
        super().__init__("Sonic", Stage("Town & City"))

    def get_best_stage(self):
        return self.stage.get_name()
```

```
def get_reason(self):
    return "The stage's length makes it good for time-outs, the platform layout aids in saving whiffed spin-dashes, and the small side blast zones let edgeguards take stocks earlier."
class Snake(Character):
    def __init__(self):
        super().__init__("Snake", Stage("Hollow Bastion"))

    def get_best_stage(self):
        return self.stage.get_name()

    def get_reason(self):
        return "The single-platform layout is very good for Snake, and Hollow Bastion has less weaknesses than Smashville does."
class Game_and_Watch(Character):
    def __init__(self):
        super().__init__("Mr. Game and Watch", Stage("Hollow Bastion"))

    def get_best_stage(self):
        return self.stage.get_name()

    def get_reason(self):
        return "The size is big enough to where he has room to operate, but small enough to prevent getting camped by opponents. Game and Watch prefers one platform for optimal juggles."
class ROB(Character):
    def __init__(self):
        super().__init__("ROB", Stage("Kalos Pokemon League (Kalos)"))

    def get_best_stage(self):
        return self.stage.get_name()

    def get_reason(self):
        return "The wall allows you to catch gyro offstage a lot easier, while also making it much harder for opponents to avoid ROB's edgeguards. Additionally, the length allows ROB to camp much easier, while also improving his own recovery."
class Pyra_Mythra(Character):
    def __init__(self):
```

```
super().__init__("Pyra and Mythra", Stage("Town & City"))

def get_best_stage(self):
    return self.stage.get_name()

def get_reason(self):
    return "Mythra mostly takes stocks off the sides and since Aegis is so bad when offstage, the long length protects you from being knocked away from the stage. There's so much room on Town for Aegis to play her best."

class Kazuya(Character):
    def __init__(self):
        super().__init__("Kazuya", Stage("Final Destination (FD)"))

    def get_best_stage(self):
        return self.stage.get_name()

    def get_reason(self):
        return "FD has no platforms, making Kazuya's combos practically inescapable. You can't escape to a platform, and Kazuya will never have to guess on platforms. It's close to a free win."

class Diddy_Kong(Character):
    def __init__(self):
        super().__init__("Diddy Kong", Stage("Kalos Pokemon League (Kalos)"))

    def get_best_stage(self):
        return self.stage.get_name()

    def get_reason(self):
        return "Kalos is a long, mostly flat stage, which is what Diddy likes. The platforms are also in very good spots for Diddy Kong, without benefitting other characters all that much. Lastly, Kalos has a wall that Diddy Kong can cling to; a feature that's practically irrelevant on any other stage."

class Minmin(Character):
    def __init__(self):
        super().__init__("Minmin", Stage("Smashville"))

    def get_best_stage(self):
        return self.stage.get_name()
```

```

def get_reason(self):
    return "It's a great length, allowing you to be explosive when needed. It's also easier to two-frame on both the Animal Crossing stages and the big middle platform provides lots of cover from descending aerials."

class Fox(Character):
    def __init__(self):
        super().__init__("Fox", Stage("Pokemon Stadium 2 (PS2)"))

    def get_best_stage(self):
        return self.stage.get_name()

    def get_reason(self):
        return "The stage's length allows Fox to easily maneuver around big hitboxes, while also being quick enough to close that distance easily. The platforms are also very suited for up air juggling and pressure, while also not allowing opponents to platform camp fox."

class CharacterFactory:
    @staticmethod
    def find_character(character_name):
        if character_name.lower() in ["steve"]:
            return Steve()
        elif character_name.lower() in ["sonic"]:
            return Sonic()
        elif character_name.lower() in ["snake"]:
            return Snake()
        elif character_name.lower() in ["g&w", "game and watch", "game & watch", "mr game and watch", "mr. game & watch", "mr game & watch"]:
            return Game_and_Watch()
        elif character_name.lower() in ["rob"]:
            return ROB()
        elif character_name.lower() in ["pyra and mythra", "pyra & mythra", "pyra", "mythra"]:
            return Pyra_Mythra()
        elif character_name.lower() in ["kazuya"]:
            return Kazuya()
        elif character_name.lower() in ["diddy kong"]:
            return Diddy_Kong()
        elif character_name.lower() in ["minmin", "min min"]:
            return MinMin()

```

```

        return Minmin()
    elif character_name.lower() in ["fox"]:
        return Fox()
    else:
        raise ValueError("The character you entered does not exist in Super Smash Brothers
Ultimate or is spelled incorrectly. Check the character list for correct spelling.")

    return None

def menu():
    print("Smash Ultimate Stage Finder")
    print("This program helps users find the best stage for their character.")
    print("1 - See list of all available characters")
    print("2 - Search best stage by character")
    print("3 - Exit")
    return input("Choose an option: ")

CHARACTER_LIST = [
    "Steve",
    "Sonic",
    "Snake",
    "Mr. Game & Watch",
    "ROB",
    "Pyra and Mythra",
    "Kazuya",
    "Diddy Kong",
    "Minmin",
    "Fox"
]

if __name__ == "__main__":
    while True:
        choice = menu()

        if choice == "1":
            print("Available Characters:")
            for c in CHARACTER_LIST:
                print("- " + c)
        elif choice == "2":
            user_input = input("Enter the character's name: ")

```

```

fighter = CharacterFactory.find_character(user_input)
print(f"Best stage for {fighter.character_name}: {fighter.get_best_stage()}")
print(f"Reason:", fighter.get_reason())

elif choice == "3":
    print("Exiting program.")
    break

else:
    print("Invalid choice.")

import tkinter as tk
from tkinter import messagebox
from FinalProject_stageTwo import (
    CHARACTER_LIST, CharacterFactory
)

class SmashGUI:
    def __init__(self, root):
        self.root = root
        root.title("Smash Ultimate Stage Finder")
        root.geometry("400x300")

        title = tk.Label(root, text="Smash Ultimate Stage Finder", font=("Arial", 16))
        title.pack(pady=20)

        btn1 = tk.Button(root, text="1 - See List of Characters", width=30,
                        command=self.show_characters)
        btn1.pack(pady=5)

        btn2 = tk.Button(root, text="2 - Search Best Stage", width=30, command=self.search_stage)
        btn2.pack(pady=5)

        btn3 = tk.Button(root, text="3 - Exit", width=30, command=root.quit)
        btn3.pack(pady=5)

    def show_characters(self):
        char_window = tk.Toplevel(self.root)
        char_window.title("Available Characters")

```

```

char_window.geometry("300x400")

label = tk.Label(char_window, text="Characters:", font=("Arial", 14))
label.pack(pady=10)

for c in CHARACTER_LIST:
    tk.Label(char_window, text=c, font=("Arial", 12)).pack(anchor="w", padx=20)

def search_stage(self):
    search_window = tk.Toplevel(self.root)
    search_window.title("Search Best Stage")
    search_window.geometry("350x200")

    label = tk.Label(search_window, text="Enter Character Name:", font=("Arial", 12))
    label.pack(pady=10)

    entry = tk.Entry(search_window, width=30)
    entry.pack(pady=5)

    def run_search():
        name = entry.get().strip()
        try:
            fighter = CharacterFactory.find_character(name)
            result = f"Best Stage: {fighter.get_best_stage()}\n\nReason:\n{fighter.get_reason()}"
            messagebox.showinfo(f"{fighter.character_name}", result)
        except ValueError:
            messagebox.showerror("Error", "Invalid character name.\nCheck spelling and try again.")

    search_btn = tk.Button(search_window, text="Search", command=run_search)
    search_btn.pack(pady=10)

if __name__ == "__main__":
    root = tk.Tk()
    gui = SmashGUI(root)
    root.mainloop()

```