الواجب الثاني

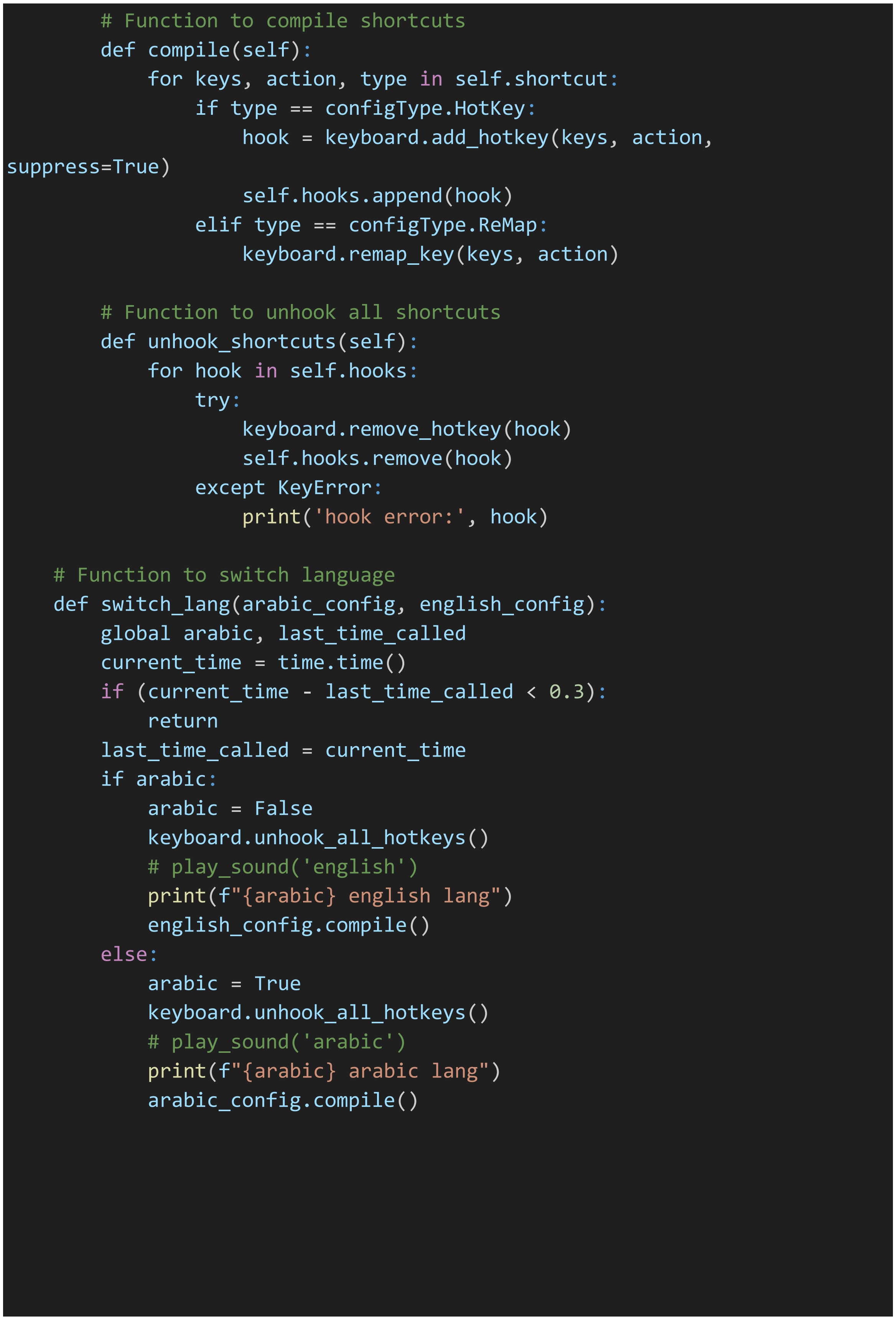
لوحة المفاتيح العربية

الاسم: عبدالرحمن الميمان

الرقم الجامعي:

```
Start
#
# https://github.com/PYTHON01100100/CSC430_87188_1_2024-COMPUTER-ARABIZATION-Course-to-Course-NavigationC_KSU/tree/main/project2
# محتاج اصور المكتب عشان اثبت التجربة
    # Import necessary modules
    Import os, subprocess, time, keyboard, pygame, sys
    Import pdb

    # Function to play audio file
    Function play_audio(file_name)
        audio_folder = "C:\\Users\\d7oom\\Desktop\\Eclipsepro"
        file_path = audio_folder + "\\" + file_name + ".wav"
        pygame.mixer.init()
        sound = pygame.mixer.Sound(file_path)
        sound.play()
        # Wait for the sound to finish playing
        pygame.time.delay(int(sound.get_length() * 800))

    # Variables initialization
    arabic = True
    py_exe = r'C:\Users\d7oom\Desktop\python.exe'
    arabic_script = [py_exe, 'arabic_keyboard.py']
    english_script = [py_exe, 'english_keyboard.py']
    env = os.environ.copy()
    running = True

    # Main loop
    while running:
        # Check current language and execute corresponding script
        if arabic:
            play_audio("arabic")
            proc = subprocess.run(arabic_script, env=env)
            print(proc.stdout)
            arabic = not arabic
        else:
            play_audio("english")
            # Simulate Alt+Shift key press to switch language
            keyboard.press_and_release('alt+shift')
            proc = subprocess.run(english_script, env=env)
            print(proc.stdout)
            arabic = not arabic
    # Function to change keyboard language
    Function change_language()
        current_layout = get_keyboard_layout()

        # Check current layout and switch language if F1 key is pressed
        if current_layout == 'Arabic':
            if keyboard.is_pressed('F1'):
                doSome("english")
        elif current_layout == 'English':
            if keyboard.is_pressed('F1'):
                doSome("arabic")

    # Function to perform language switch and play audio
    Function doSome(str)
        play_audio(str)
        # Send the Alt+Shift key combination to switch language
        keyboard.send("shift+alt")

    # Add hotkey to change language using F1 key
    keyboard.add_hotkey("F1", change_language)

End
```

```
Start

    # Import necessary modules
    Import os, subprocess, time, keyboard, pygame, sys
    From enum import Enum

    # Global variables
    arabic = True
    caps_lock = False
    last_time_called = 0
    end_signal = 'continue'

    # Define an Enum for configuration type
    class configType(Enum):
        HotKey = 1
        ReMap = 2

    # Define a keyboard controller class
    class keyboardController:
        shortcut = []
        hooks = []

        # Function to add shortcut action
        def add_shortcut_action(self, keys, action, type):
            self.shortcut.append((keys, action, type))

        # Function to map English to Arabic characters
        def map_english_to_arabic(self, event_name):
            return self.arabic_to_english.get(event_name)

        # Function to suppress a specific shortcut
        def suppress_shortcut(self, keys):
            self.add_shortcut_action(keys, lambda: None,
configType.HotKey)
```



```python
        # Function to compile shortcuts
        def compile(self):
            for keys, action, type in self.shortcut:
                if type == configType.HotKey:
                    hook = keyboard.add_hotkey(keys, action,
suppress=True)
                    self.hooks.append(hook)
                elif type == configType.ReMap:
                    keyboard.remap_key(keys, action)

        # Function to unhook all shortcuts
        def unhook_shortcuts(self):
            for hook in self.hooks:
                try:
                    keyboard.remove_hotkey(hook)
                    self.hooks.remove(hook)
                except KeyError:
                    print('hook error:', hook)

    # Function to switch language
    def switch_lang(arabic_config, english_config):
        global arabic, last_time_called
        current_time = time.time()
        if (current_time - last_time_called < 0.3):
            return
        last_time_called = current_time
        if arabic:
            arabic = False
            keyboard.unhook_all_hotkeys()
            # play_sound('english')
            print(f"{arabic} english lang")
            english_config.compile()
        else:
            arabic = True
            keyboard.unhook_all_hotkeys()
            # play_sound('arabic')
            print(f"{arabic} arabic lang")
            arabic_config.compile()
    # Function to change signal
    def change_signal():
        global end_signal
        end_signal = 'end'

    # Create an instance of keyboard controller for English configuration
    english_config = keyboardController()

    # Suppress default shortcuts for English configuration
    english_config.suppress_shortcut('caps lock')
    english_config.suppress_shortcut('shift + backslash')
    english_config.suppress_shortcut('backslash')
    english_config.add_shortcut_action('f8', lambda: print('end'),
configType.HotKey)

    # Compile English configuration
    english_config.compile()

    # Wait for F1 key press to start
    keyboard.wait('f1', suppress=True)

End
```

```
Start

    # Import necessary modules
    Import os, subprocess, time, keyboard, pygame, sys

    # Print the running Python executable and environment variables
    Print("Running python exe:", sys.executable)
    Print(os.environ.copy())

    # Import keyboard module and Enum class
    Import keyboard
    From enum import Enum

    # Global variables initialization
    arabic = True
    caps_lock = False
    last_time_called = 0
    end_signal = 'continue'

    # Define Arabic to English key mapping dictionary
    arabic_key_mapping = {
        'q': 'ض', 'w': 'ص', 'e': 'ث', 'r': 'ق', 't': 'ف', 'y': 'غ', 'u':
'ع', 'i': 'ه',
        'o': 'خ', 'p': 'ح', '[': 'ج', ']': 'د',
        'a': 'ش', 's': 'س', 'd': 'ي', 'f': 'ب', 'g': 'ل', 'h': 'ا', 'j':
'ت', 'k': 'ن',
        'l': 'م', ';': 'ك', '\'': 'ط',
        'z': 'ئ', 'x': 'ء', 'c': 'ؤ', 'v': 'ر', 'b': 'لا', 'n': 'ى', 'm':
'ة', ',': 'و',
        '.': 'ز', '/': 'ظ'
    }

    # Define an Enum for configuration type
    class configType(Enum):
        HotKey = 1
        ReMap = 2

    # Define a keyboard controller class
    class keyboardController:
        shortcut = []
        hooks = []
        # Function to add shortcut action
        def add_shortcut_action(self, keys, action, type):
            self.shortcut.append((keys, action, type))

        # Function to map English to Arabic characters
        def map_english_to_arabic(self, event_name):
            return self.arabic_to_english.get(event_name)

        # Function to suppress a specific shortcut
        def suppress_shortcut(self, keys):
            self.add_shortcut_action(keys, lambda: None,
configType.HotKey)

        # Function to compile shortcuts
        def compile(self):
            for keys, action, type in self.shortcut:
                if type == configType.HotKey:
                    hook = keyboard.add_hotkey(keys, action,
suppress=True)
                    self.hooks.append(hook)
                elif type == configType.ReMap:
                    keyboard.remap_key(keys, action)

    # Function to toggle Caps Lock
    def toggle_caps_lock():
        global caps_lock
        caps_lock = not caps_lock
        print("Caps Lock is now", "on" if caps_lock else "off")

    # Function to handle Arabic typing
    def arabic_caps(input):
        global caps_lock
        if not caps_lock:
            keyboard.write(english_to_arabic(input))
        else:
            keyboard.press_and_release('shift+' + input)

    # Function to convert English to Arabic characters
    def english_to_arabic(english_char):
        global arabic_key_mapping
        return arabic_key_mapping.get(english_char)
    # Function to change signal
    def change_signal():
        global end_signal
        end_signal = 'end'

    # Create an instance of keyboard controller for Arabic configuration
    arabic_config = keyboardController()

    # Define Arabic shortcuts
    arabic_config.add_shortcut_action('b', lambda: keyboard.write('ال'),
configType.HotKey)
    arabic_config.add_shortcut_action('shift+g', lambda:
keyboard.write('أل'), configType.HotKey)
    arabic_config.add_shortcut_action('shift+b', lambda:
keyboard.write('آل'), configType.HotKey)
    arabic_config.add_shortcut_action('shift+t', lambda:
keyboard.write('إل'), configType.HotKey)
    arabic_config.add_shortcut_action('caps lock', toggle_caps_lock,
configType.HotKey)
    arabic_config.suppress_shortcut('shift+backslash')
    arabic_config.suppress_shortcut('backslash')
    arabic_config.add_shortcut_action('f8', lambda: print('end'),
configType.HotKey)

    # Define Arabic keys
    arabic_keys = [
        'a', 's', 'd', 'f', 'g', 'h',
        'j', 'k', 'l', 'm', 'n', 'q',
        'r', 't', 'w', 'x', 'c', 'v',
        'y', 'u', 'i', 'o', 'p', 'z'
    ]
    for key in arabic_keys:
        arabic_config.add_shortcut_action(key, lambda key=key:
arabic_caps(key), configType.HotKey)

    # Compile Arabic configuration
    arabic_config.compile()

    # Wait for F1 key press to start
    keyboard.wait('f1', suppress=True)

End
```