README

Workspace Launcher for Visual Studio Code

What is this repository for?

- A utility to list and launch Visual Studio Code workspaces.

Enhancement Roadmap

- Add Linux & MacOS support
- Clean up appearance
- Allow sort by latest used date instead of parent folder > workspace name
- Look for missing workspaces at locations other than the pointers

Why is this a thing?

There are people who are chronically well-organized. They know what they're creating before they start. They create a well-designed directory structure and stick with it...

And then there's me. 😳

I create projects any old place. I mean, I'm gonna throw half of them away anyhow, since they're mostly just experiments. Then, when I decide to keep one, I'll just move it to a more organized folder (sort of). I mean, I'll do that, but then, during a weekend bout of insomnia, I'll decide to completely reorganize my previously organized directories, because I'm certain I totally got them wrong the first (or second... or third...) time. And that's just a taste of what living in my head feels like.

One unfortunate side effect of this is that when Visual Studio Code offers me a selection of recently opened workspaces, some of them aren't really where the software thinks they are.

Moreover, I often want to open one that I haven't visited recently, and the recent workspaces selection doesn't help me there at all.

In the long run, there are a lot of orphans and duplicates in the VS Code pointer files that store workspace locations, even after I've re-visited the moved ones so that the program is aware of their new locations.

I wanted a quick and easy utility that I could use to launch VS code to any past workspace that still exists (where it claims to be), ignoring the missing and orphaned records. Since that isn't an available option in the software itself, I figured I'd just make one myself.

So, here we are. 😁

Setup/Requirements

- Create a virtual environment

    - `py -m venv .venv`
- Activate the virtual environment
    - `.venv\Scripts\activate`
- Install the required Python modules:
    - You can install the requirements all at once using:
        - `py -m pip install -r requirements.txt`
    - Or you can install the individual requirements
        - PySimpleGUI
            - `py -m pip install pysimplegui`
        - sm-utils
            - `py -m pip install sm_utils`
- If you want to compile this to an executable, install the following additional Python modules:
    - PyInstaller
        - `py -m pip install pyinstaller`
    - PyInstaller VersionFile
        - `py -m pip install pyinstaller-versionfile`

How Does Visual Studio Code Handle Workspaces?

Each time a workspace is created or interacted with in VS Code, the software creates or updates a pointer directory containing the workspace information.

In Windows, the default location where these pointers are created is:
`C:\Users\{USERNAME}\AppData\Roaming\Code\User\workspaceStorage`

The workspace pointer folders are created with non-meaningful GUID names, so it is necessary to review the JSON file inside (*workspace.json*) to identify the location of the workspace being referenced.

The content of one of those files is a JSON object containing one key: [folder] with a URL-encoded path as its value, resembling the following:

```
{
    "folder": "file:///c%3A/Path/To/Workspace"
}
```

So, programmatically identifying the active workspaces requires a multi-step approach:

1. Read the workspace pointer files stored by VS Code
2. Decode the URL and parse off leading "file:///" to obtain a local path
3. Verify that the local path exists and is a directory

What's in here?

Program Files

Note: For convenience, all of the files listed below have been merged into a single Python file (***project.py***) for final delivery. However, to facilitate an easy understanding of the components, I have left the documentation below broken down by file. You can still find the separate component files in the */multi-file* subdirectory.

- **workspace.py**: implements the ***Workspace*** class, which defines a VS Code workspace
    - Attributes:
        - **vsc_folder** (*str*): path to the folder that contains the VS Code workspace.json pointer file
        - **workspace** (*str*): path to the workspace folder
        - **name** (*str*): folder name for the workspace
        - **parent** (*str*): parent folder name (that contains the workspace folder)
        - **repo_uri** (*str*): URI to the GIT repository for the workspace (if one exists)
        - **exists** (*bool*): True if the workspace folder is defined and exists
        - **show_repo** (*bool*): When True, show the repository in the display name
        - **show_glyph** (*bool*): When True, prepend the glyph to the repository if applicable
    - Properties:
        - **display_name** (*str*): Display name for the workspace in the select list
    - Methods:
        - **from_vscode_folder**: Class Method to generate a Workspace Instance given the path to a VS Code folder (containing a workspace.json file)
            - Arguments:
                - vsc_folder (*str*): The path to the VS Code folder
                - show_repo (*bool* default=True): When true, include the repository in the display name
                - show_glyph (*bool* default=False): When true, prepend the glyph to the repository in the display name
            - Code Sample:

```
from workspace import Workspace
vsc_folder =
"C:\\Users\\USERNAME\\AppData\\Roaming\\Code\\User\\workspace
Storage\\0d14953ffbc0e69d994e7b502e8cc120"
workspace = Workspace.from_vscode_folder(vsc_folder)
```

        - **from_workspace_folder**: Class Method to generate a Workspace Instance given the path to a workspace (containing code files)
            - Arguments:
                - workspace_folder (*str*): The path to the workspace folder
                - vsc_folder (*str*): The path to the VS Code folder
                - show_repo (*bool* default=True): When true, include the repository in the display name
                - show_glyph (*bool* default=False): When true, prepend the glyph to the repository in the display name
            - Code Sample:

```
from workspace import Workspace
ws_folder = "C:\\My Python Project"
workspace = Workspace.from_vscode_folder(ws_folder)
```

- **workspace_settings.py**: Implements the ***WorkspaceSettings*** class, which models a settings object to control behaviors throughout the project. This class is also represented one-for-one in the *settings.json* configuration file.
    - Attributes:
        - **exe_path** (*str* default="default"): Path to the VS Code executable
            - Default points to: `C:\\Users\\{USERNAME}\\AppData\\Local\\Programs\\Microsoft VS Code\\Code.exe`
        - **workspace_path** (*str* default="default"): Path to the workspace folder
            - Default points to: `C:\\Users\\{USERNAME}\\AppData\\Roaming\\Code\\User\\workspaceStorage`
        - **username** (*str* default="default"): Username to use to get workspaces
            - Default will be the currently logged-in user
        - **hide_missing** (*bool* default=True): When true, missing workspace folders are omitted from the select list
        - **clean_up_orphans** (*bool* default=False): When true, missing workspace folders have their related VS Code folders removed
        - **show_repos** (*bool* default=True): When true, the repository URL is shown in the select list
        - **font** (*str* default="Consolas"): Name of font to use in the UI
            - Font must already be installed on the system
            - If using the *show_glyphs* option, the font must be a nerd font that includes the Bitbucket and GitHub glyphs
        - **font_size** (*int* default=10): Size of the font to use in the UI
        - **show_glyphs** (*bool* default=False): When true, a glyph is prepended to the repository URL
            - Currently only supports
                - Bitbucket
                - GitHub
        - **x_location** (*int* default=10): Horizontal location of the UI (top-left corner)
            - Measures from the left of the screen
            - Supports a negative number to measure from the right of the screen
        - **y_location** (*int* default=-160): Vertical location of the UI (top-left corner)
            - Measures from the top of the screen
            - Supports a negative number to measure from the bottom of the screen
    - Methods:
        - **post_init**: Obtains username and paths if any of the values are still "default" after initialization.
            - This should only perform work in the instance that a *WorkspaceSettings* object was created without using the provided factory methods.
            - This is the scenario (not recommended) for which this method exists.

```
from workspace_settings import WorkspaceSettings
# Creating a settings object without arguments
# In this instance, only the default values are assigned
settings = WorkspaceSettings()
```

    - See the code samples in *from_file* and *from_dict* below for the preferred approach.
- **from_file**: Class method to generate a *WorkspaceSettings* instance from a JSON file
    - Arguments:
        - filename (*str*): The filename to load
        - folder (*str*): Optional folder name the file is in
    - Code Sample:

```
from workspace_settings import WorkspaceSettings
settings = WorkspaceSettings.from_file("settings.json")
```

- **from_dict**: Class method to generate a *WorkspaceSettings* instance from a dictionary
    - Arguments:
        - settings (*dict[str, any]*): Dictionary containing settings values.
            - Note: Keys must exactly match the class attribute names
    - Code Sample:

```
import json
from workspace_settings import WorkspaceSettings
my_dict = json.load("settings.json")
settings = WorkspaceSettings.from_dict(my_dict)
```

- **_get_user**: Internal class method to get the username attribute for the *WorkspaceSettings* instance.
    - Used by the *from_file* and *from_dict* methods
    - When the value is "default" returns the logged-in user
    - Arguments:
        - **username** (*str*): The currently set username value
- **_get_user_paths**: Internal class method to get the paths to the VS Code executable and VS Code workspace directory
    - Used by the *from_file* and *from_dict* methods
    - Arguments:
        - **username** (*str*): The currently set username value
        - **exe_path** (*str*): The currently set exe_path value
            - Converts "default" to the path to the exe
        - **ws_path** (*str*): The currently set ws_path value
            - Converts "default" to the path to the workspaces folder

- **workspace_locator.py**: Implements the ***WorkspaceLocator*** class, which generates the list of all workspaces in the user's workspace directory for display in the UI select list.

    - Attributes:
        - **_settings** (*WorkspaceSettings*): The settings for the current execution
        - **_workspaces** (*list[Workspace]*): The list of workspaces found for the user
            - This should be accessed from within the class only, as it does not obey settings filters like *hide_missing* (see property below)
    - Properties:
        - **workspaces** (*list[Workspace]*): The list of workspaces to display in the UI
            - Filters out missing workspaces if the *hide_missing* attribute is true in the *_settings* object
    - Methods:
        - **init**: Initializes the *WorkspaceLocator* object
            - Arguments:
                - **settings** (*WorkspaceSettings* default=None):
                    - Preferred option
                    - The *WorkspaceSettings* object to be used
                - **settings_json** (*dict[str, any]* default=None):
                    - A dictionary containing all of the settings needed for a *WorkspaceSettings* object
                    - Used in conjunction with *WorkspaceSettings.from_dict()* when *settings* is not provided.
                - **settings_file** (*str* default=None):
                    - The path to the settings JSON file.
                    - Used in conjunction with *WorkspaceSettings.from_file()* when neither *settings* nor *settings_json* is provided.
        - **load_workspaces**: Traverses the VS Code workspace directories and generates the list of workspaces, sorted by *display_name*
            - Arguments: (none)
        - **clean_up_orphans**: Traverses the list of workspaces and deletes the VS Code reference folders for any that are missing (Workspace.exists == False).
            - Does not execute unless _settings.clean_up_orphans == True
            - Arguments: (none)

- **workspace_launcher.py**: Implements the ***WorkspaceLauncher*** class, which uses PySimpleGUI to create and display the UI for the user to select and launch workspaces.

    - Attributes:
        - **_settings** (*WorkspaceSettings*): The settings object for the application
        - **_workspace_locator** (*WorkspaceLocator*): Used to generate the list of workspaces to display in the select list
        - **workspace_filter** (*PySimpleGUI.InputText*): Text box to allow the user to filter the results displayed in the select list
        - **vsc_toggle** (*PySimpleGUI.Checkbox*): When checked, selecting a workspace launches Visual Studio Code to the workspace
        - **url_toggle** (*PySimpleGUI.Checkbox*): When checked, selecting a workspace launches the default web browser to the repository URL (if one exists)
        - **workspace_selector** (*PySimpleGUI.Combo*): Displays the (filtered) list of workspaces to select.

        - **window** (*PySimpleGUI.Window*): Main UI window containing all of the PySimpleGUI controls.
    - Methods:
        - **create_ui**: Generates the UI window and launches it for user interaction. Raises events when any of the GUI controls are changed.
            - Arguments: (none)
            - Code Sample:

```
from workspace_launcher import WorkspaceLauncher
launcher = WorkspaceLauncher("settings.json")
launcher.create_ui()
```

        - **_get_ui_position**: Compares the screen size to the X and Y location settings and returns the computed (x, y) position for the upper left corner of the UI as a tuple
            - Arguments
                - **window** (*PySimpleGUI.Window*): The Window instance for the UI (used to obtain screen dimensions)
        - **_launch_workspace**: Launches an instance of Visual Studio code at the workspace location
            - Arguments
                - **selected_workspace** (*Workspace*): The workspace selected by the user in the UI
        - **_launch_repository**: Launches the repository URL (if one exists) in the default browser
            - Arguments
                - **selected_workspace** (*Workspace*): The workspace selected by the user in the UI
        - **resource_path**: This function exists only to work around an issue with PyInstaller where the icon does not display on the compiled application (not part of the main program).
            - Arguments:
                - **file_name** (*str*): The file to provide a resource path for
    - Event Handlers:
        - **on_filter_change**: Called after the event when the user types in the filter field and resets the list to include only workspaces that include the filter text in the display name.
            - Arguments:
                - **filter_text** (*str*): The text value currently in the filter field
        - **on_workspace_select**: Called after the event when the user selects a workspace from the list. Calls the following functions:
            - If the *vsc_toggle* box is checked, calls *_launch_workspace()*
            - If the *url_toggle* box is checked, calls *_launch_repository()*
            - Arguments:
                - **selected_workspace** (*Workspace*): The workspace selected by the user
        - **on_vsc_toggle_change**: Called after the event when the user checks or unchecks the *vsc_toggle* checkbox. If a workspace is selected and the user checked the box, calls *_launch_workspace()*. Otherwise does nothing.
            - Arguments:
                - **selected_workspace** (*Workspace*): The workspace selected by the user

            - **on_url_toggle_change**: Called after the event when the user checks or unchecks the *url_toggle* checkbox. If a workspace is selected, a repository exists for the selected workspace, and the user checked the box, calls *_launch_repository()*. Otherwise does nothing.
                - Arguments:
                    - **selected_workspace** (*Workspace*): The workspace selected by the user

- **workspace_program.py**: Contains the main() function to execute the overall program
    - Functions:
        - **main**: The main function to execute
            - Calls function to verify the program is running on Windows
                - Displays an alert and exits otherwise
            - Obtains the path to the *settings.json* file
                - Checks first for command-line argument
                - If none is provided, defaults to "settings.json"
            - Creates a full path to the settings file and obtains a *WorkspaceSettings* object from the *get_settings* function
            - Creates an instance of *WorkspaceLauncher*
            - Calls *WorkspaceLauncher.create_ui()*
            - Arguments: (none)
        - **verify_windows**: Returns true if the current OS is Windows, otherwise false.
            - Arguments: (none)
        - **unsupported_os_alert**: Displays a PySimpleGUI pop-up window informing the user that the current OS is unsupported.
            - Arguments:
                - **suppress_alert** (*bool*):
                    - If true, the popup window will not be shown, and only the alert in the terminal will appear.
        - **get_settings**: Obtains the settings object to use for the *WorkspaceLauncher*

- **.\one-file\vscode_workspace_launcher.py**:
    - Places all of the above program components in a single Python file
    - Duplicate of *project.py*
    - Used with PyInstaller to simplify generating an executable version

Supporting Files

- **settings.json**: Contains the user-configurable settings as a JSON object
    - Defaults:

```
{
    "exe_path": "default",
    "workspace_path": "default",
    "username": "default",
    "hide_missing": true,
```

```
    "clean_up_orphans": false,
    "show_repos": true,
    "font": "CaskaydiaCove Nerd Font",
    "font_size": 10,
    "show_glyphs": true,
    "x_location": 10,
    "y_location": -160
}
```

- **rocket.ico**: Icon file for the PySimpleGUI window
  Note: This image comes from pngtree.com and is not usable for commercial purposes
  

- **requirements.txt**: Python requirements file for installed modules
  - Content:

```
PySimpleGUI
sm_utils
```

- **.\one-file\version.yaml**: YAML file to generate PyInstaller version file
- **.\one-file\version.txt**: PyInstaller version file converted from YAML using pyinstaller-versionfile

Test Files

Note: For convenience, all of the files listed below have been merged into a single Python file (***test_project.py***) for final delivery. However, to facilitate an easy understanding of the components, I have left the documentation below broken down by file.

- **test_workspace_program.py**: PyTest tests for the main executable file
  - Fixtures:
    - **default_settings_file**: Relative path to the settings file on the current workstation.
  - Test Functions:
    - **test_verify_windows**: Tests the function to verify that the OS it's running on is Windows
    - **test_get_settings**: Test to verify that a proper *WorkspaceSettings* object is returned with a valid JSON file and an invalid file path returns nothing.
    - **test_unsupported_os_alert**: Test to validate that the program exits with a meaningful message when run on an unsupported OS.

- **test_workspace.py**: PyTest tests for the *Workspace* class
  - Fixtures:
    Note: Fixtures must be set to values existing on the current workstation

        - **default_vsc_path**: Path to a VS Code pointer folder for a workspace on the current workstation
        - **default_ws_path**: Path to a workspace on the current workstation
        - **default_ws_name**: Name of the default workspace on the current workstation
        - **default_parent**: Parent folder of the default workspace on the current workstation
        - **default_repo_url**: URL of the default workspace repository
        - **defaults**: Default workspace for testing/comparison
            - Uses the above fixtures as its values
    - Test Functions:
        - **test_init**: Test initializer without any values
            - Arguments: (none)
        - **test_init_defaults**: Test initializer with values but no factory functions
            - Arguments
                - **defaults** (*Workspace* fixture)
        - **test_from_vsc_folder**: Test Workspace factory using VSC folder
            - Arguments
                - **defaults** (*Workspace* fixture)
        - **test_from_invalid_vsc_folder**: Test Workspace factory using invalid VSC folder
            - Arguments: (none)
        - **test_from_ws_folder**: Test Workspace factory using WS folder
            - Arguments
                - **defaults** (*Workspace* fixture)
        - **test_from_ws_folder_no_vsc**: Test Workspace factory using WS folder (without VSC folder)
            - Arguments
                - **defaults** (*Workspace* fixture)
        - **test_from_invalid_ws_folder**: Test Workspace factory using invalid WS folder
            - Arguments: (none)
        - **test_display_name**: Test display name
            - Arguments
                - **defaults** (*Workspace* fixture)

- **test_workspace_locator.py**: PyTest tests for the *WorkspaceLocator* class
    - Fixtures:
        - **settings_path** (*str*): Path to settings.json file
    - Test Functions:
        - **test_init_no_settings**: Test initializing without passing settings
            - Arguments: (none)
        - **test_init_with_settings**: Test initializing passing settings
            - Arguments: (none)
        - **test_init_with_settings_file**: Test initializing passing settings JSON file
            - Arguments
                - **settings_path** (*str* fixture)
        - **test_workspaces**: Test obtaining workspaces
            - Arguments
                - **settings_path** (*str* fixture)

Usage

- To run the application without creating an executable:
    - If *settings.json* is in the same directory as the Python files, run the following command from the terminal:
      `python.exe project.py`
      or
      `python.exe one-file\project.py`
    - To point to your JSON file with a different name or at a different location, add its relative path as an argument:
      `python.exe project.py settings\my_settings.json`
      or
      `python.exe one-file\project.py settings\my_settings.json`
- To generate a stand-alone executable, run the following command in the one-file directory:
    - `pyinstaller --onefile project.py --windowed --add-data "rocket.ico:." --icon=rocket.ico --version-file=version.txt`
- To update the properties (version number, etc.) of the executable, do the following before generating the .exe:
    - Edit "version.yaml" with the values you want for the properties, then run the following command:
    - `create-version-file version.yaml --outfile version.txt`
- Make sure a copy of settings.json is in the same directory as the compiled executable
- To run with the default VSCode paths, leave path items in settings.json set to `default`
- To run with VSCode paths other than the defaults, edit the settings.json file:
    - exe_path:
        - This is the path to the Visual Studio Code executable
        - Default (Windows):
          `C:\\Users\\{USERNAME}\\AppData\\Local\\Programs\\Microsoft VS Code\\Code.exe`
    - workspace_path:
        - This is the path to the folder containingVisual Studio Code workspaces
        - Default (Windows):
          `C:\\Users\\{USERNAME}\\AppData\\Roaming\\Code\\User\\workspaceStorage`
    - username:
        - The user whose workspaces should be displayed
        - Default: Current Windows User

- To change the position where the window appears when launched, modify the `x_location` and/or `y_location` values in settings.json
    - x_location: (`"x_location": n`) where n is an integer:
        - n > 0: places the top, left corner n pixels from the left of the screen.
        - n < 0: places the top, left corner |n| pixels from the right of the screen.
        - n = 0: will be treated as a 10-pixel offset from the left
    - y_location: (`"y_location": n`) where n is an integer:
        - n > 0: places the top, left corner n pixels from the top of the screen.
        - n < 0: places the top, left corner |n| pixels from the bottom of the screen.
        - n = 0: will be treated as a 10-pixel offset from the top
- The following additional parameters can be set in the settings.json file:
    - `hide_missing`
        - When `true`, workspaces whose folders are missing will not be included in the select list
    - `clean_up_orphans`
        - When `true`, workspaces whose folders are missing will have their VS Code workspace folders removed
        - This is obviously an aggressive step, so leave this `false` unless you're absolutely sure you don't want them around
    - `show_repos`
        - When `true`, names in the select list will include their repository URLs
    - `show_glyphs`
        - When `true`, repository URLs (if displayed) will be prepended with the nerd-font glyphs for their sites.
        - Currently only supports the following glyphs:
            - Bitbucket
            - GitHub
        - Note: This feature requires that the `font` setting (below) be an installed nerd font with the glyphs available
            - I use the "CaskaydiaCove Nerd Font" for example
                - A copy is included as fonts\CaskaydiaCove.zip
    - `font`
        - Name of the (installed) font to be used in the UI
    - `font_size`
        - Display size of the font in the UI as `"font_size": <int>`
        - I recommend not setting this lower than 10 or higher than 20

Known Conflicts/Compatibility Notes

- Default paths only support Windows

Documentation

- See "Usage" (above)
  or
- Review the PDF version of this file (README.pdf)

Version History

- v1.0 - 2/25/2024 - Initial release (CS50P Final Project)