Comprehensive Guide for New Python Programmers

1. Installing Python

- **Official Installer**: Download Python from python.org.
- Be mindful of which versions you install. Some Python packages are only compatible with certain Python versions. This is why creating a virtual environment for every project is so important. See Section 6.

2. Setting Up Visual Studio Code (VSCode)

1. Download Visual Studio Code.
2. For suggestions on how to setup VSCode, see https://www.youtube.com/watch?v=mpk4Q5feWaw
3. Install recommended extensions via the Extensions sidebar (`Ctrl+Shift+X`):
   - Python Extension Pack
   - GitHub Copilot
   - Path Intellisense
   - GitHub Pull Request
   - Better Comments
   - Ruff
   - Material Icon Theme
   - Atom One Dark Theme

3. Configuring VSCode

- **Themes and Icons**: Set via `Settings → Theme` (Atom One Dark) and `File Icon Theme` (Material Icon Theme).
- **Auto Formatting**: Enable formatting on save (`Settings → Format on Save`). Set Ruff as default formatter:

```
"[python]": {
  "editor.formatOnType": true,
  "editor.defaultFormatter": "charliermarsh.ruff"
}
```

4. Git and GitHub Integration

- **Git Installation**: Download and install Git. For more information, see https://www.youtube.com/watch?v=VWbX2B3Q-1g

- **GitHub Account**: Create at GitHub.
- **Configuring Git User Information**:

```
git config --global user.name "Your Name"
git config --global user.email "your.email@example.com"
```

- **Integrating Git with VSCode**:
  1. Open VSCode Command Palette (`Ctrl+Shift+P`), execute `Shell Command: Install 'code' command in PATH` .
  2. Use VSCode Source Control (`Ctrl+Shift+G`) to initialize repositories, commit changes, and sync with GitHub.
  3. For more information, see https://code.visualstudio.com/docs/sourcecontrol/github

5. Project Structure (Using `src/` Layout)

Store all Python projects locally on your hard drive (not Box, Dropbox, Google Drive, etc) and sync with the cloud via Github. Otherwise, you will run into problems with virtual environments between devices.

Example project: Simple inventory management system.

```
project/
├── .venv/
├── .gitignore
├── requirements.txt
├── README.md
├── main.py
└── src/
    └── inventory/
        ├── __init__.py
        ├── models/
        │   ├── __init__.py
        │   └── item.py
        ├── services/
        │   ├── __init__.py
        │   └── inventory_service.py
        └── utils/
            ├── __init__.py
            └── helpers.py
```

Best Practices for Project Organization:

- `.venv` : Directory containing your project's isolated Python environment.
- `.gitignore` : Specifies files and folders Git should ignore.
- `requirements.txt` : Lists Python dependencies.
- `README.md` : Project documentation.

- `src/` : Contains your Python source code.
- Init: https://www.youtube.com/watch?v=VEbuZox5qC4&t=524s

Simple Examples:

- **main.py**:

```
from inventory.services.inventory_service import InventoryService

def main():
    service = InventoryService()
    service.add_item("Laptop", 3)
    print(service.get_inventory())

if __name__ == "__main__":
    main()
```

- **src/inventory/init.py**:

```
from .services.inventory_service import InventoryService
__all__ = ['InventoryService']
```

- **src/inventory/models/item.py**:

```
class Item:
    def __init__(self, name, quantity):
        self.name = name
        self.quantity = quantity

    def __repr__(self):
        return f"{self.name} ({self.quantity})"
```

- **src/inventory/services/inventory_service.py**:

```
from inventory.models.item import Item

class InventoryService:
    def __init__(self):
        self.items = []

    def add_item(self, name, quantity):
        item = Item(name, quantity)
        self.items.append(item)
```

```
    def get_inventory(self):
        return self.items
```

- **src/inventory/utils/helpers.py**:

```
def format_inventory(items):
    return [f"Item: {item.name}, Quantity: {item.quantity}" for item in items]
```

6. Virtual Environments

- Create via VSCode Command Palette (`Ctrl+Shift+P`), type `Python: Select Interpreter` , then select or create environment.
- Activate and install packages using:

```
pip install -r requirements.txt
```

7. Jupyter Interactive Mode

- https://www.youtube.com/watch?v=qFvInA7DKuE&t=4s
- Execute code interactively by selecting it and pressing `Shift + Enter` .

8. Debugging Python in VSCode

- Use VSCode Debugger (`Ctrl+Shift+D`). See https://code.visualstudio.com/docs/debugtest/debugging.
- Set breakpoints and use debugging tools to inspect variables and call stack.

9. GitHub Copilot

- Enable GitHub Copilot in VSCode.
- Receive automatic code suggestions as you type.

10. Object-Oriented Programming (OOP) Basics

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def greet(self):
        print(f"Hello, my name is {self.name} and I'm {self.age} years old.")
```

```
if __name__ == "__main__":
    person = Person("Alice", 30)
    person.greet()
```