All about Python Programming

Programming python by Visual Studio Code and Python v3.13.2

Download the latest version of the Python.

Using: Python.exe or Py.exe

- To find the Python path: Where Python
- To display the Python version: Python--version

- Using VS Code
- Create [new project] folder: md new folder
- Create a virtual environment: Python-m venv venv
- Change the current folder path: cd [new folder]
- To run virtual environment and active it: venv\Scripts\activate
- To deactivate the virtual environment: venv\Scripts\deactivate

- To provide run [venv] automatically when the VS Code running:
- create [.vscode] folder and these files:

- File [launch.json] Contains:

```
{
   "version": "0.2.0",
   "configurations": [
      {
         "name": "Python Debugger: Current File",
         "type": "debugpy",
         "request": "launch",
         "program": "${file}",
         "console": "integratedTerminal"
      },
      {
         "name": "Python: Current File",
         "type": "debugpy",
         "request": "launch",
         "program": "${file}",
         "console": "integratedTerminal"
      }
   ]
}
```

- File [settings.json] Contains:

```
{
   "python.defaultInterpreterPath": "./venv/Scripts/python.exe",
   "python.terminal.activateEnvironment": true,
   "terminal.integrated.profiles.windows": {
      "PowerShell": {
         "path": "C:/Windows/System32/WindowsPowerShell/v1.0/powershell.exe",
         "args": [
            "-NoExit",
            "-Command",
            "& { . venv/Scripts/Activate.ps1 }"
```

```
        ]
      },
      "Command Prompt": {
        "path": "C:/Windows/System32/cmd.exe",
        "args": ["/k", "venv/Scripts/activate.bat"]
      }
    },
    "terminal.integrated.defaultProfile.windows": "PowerShell"
}
```

- File [tasks.json] Contains:

```
{
    "version": "2.0.0",
    "tasks": [
     {
       "label": "Activate Virtual Environment",
       "type": "shell",
       "command": ".venv/Scripts/activate",
       "group": "build"
     }
    ]
  }
```

- To install and update Python packages: Pip Install--upgrade--force-reinstall [Package Name, …] or [Package Name==version number, …]
- To get version of pip: Pip –version
- To update pip: Python-m pip install--upgrade pip
- To uninstall package: Pip Uninstall [Package Name]
- To get a list of packages: Pip list
- To make a list of all packages: Pip freeze > requirements.txt
- To install all of requirements.txt file packages: Pip install-r requirements.txt
- To make a list of outdated packages: Pip list--outdated

To create a Django project:
- To install the Django in pc: Pip install django
- To create a Django project: django-admin startproject [myproject]
- To change the current directory to project folder: cd [myproject]
- To run the Django server and test it: Python manage.py runserver or Python manage.py runserver 127.0.0.1:8000 or [another free port number]
- To make an application: Python manage.py startapp [myapp]
- To register application in the project: # myproject/settings.py
- Insert into [INSTALLED_APPS]: [myapp] name

- To generate a migration: Python manage.py [makemigrations]
- To apply a migration: Python manage.py migrate

- To Check for configuration issues: Python manage.py shell
- To Python + Django interactive environment: Python manage.py check

- To create the super admin account: Python manage.py createsuperuser
- To visit the Django website: http://127.0.0.1:8000 or [another free port number]
- To visit the Django website with admin panel: http://127.0.0.1:8000/admin

- To use the SQLite: No need to install, it's internal and simple.
- Example: import sqlite3

```
conn = sqlite3.connect("mydb.sqlite3")
cursor = conn.cursor()
cursor.execute("CREATE TABLE test (id INTEGER PRIMARY KEY, name TEXT)")
conn.commit()
conn.close()
```

- To use SQLite into the Django: Edit the [settings.py] file:

```
DATABASES = {
   'default': {
      'ENGINE': 'django.db.backends.sqlite3',
      'NAME': BASE_DIR / 'db.sqlite3',
   }
}
```

- To install the MongoDB: Pip install pymongo
- Example: from pymongo import MongoClient

```
client = MongoClient("mongodb://localhost:27017/")
db = client["mydatabase"]
collection = db["mycollection"]
collection.insert_one({"name": "Behdad"})
```

- To install MongoDB and use into Django: Pip install mongoengine

- To install Redis DB (For cache, Celery, queue, or temporary data): Pip install redis
- Example: import redis

```
r = redis.Redis(host='localhost', port=6379, db=0)
r.set('foo', 'bar')
print(r.get('foo'))
```

- To install for use in Django: Pip install django-redis

- To install PostgreSQL (Stable, professional, suitable for real projects): Pip install psycopg2 or pip install psycopg2-binary
- To use into the Django: add in file [settings.py]:

```
DATABASES = {
   'default': {
      'ENGINE': 'django.db.backends.postgresql',
      'NAME': 'mydb',
      'USER': 'myuser',
      'PASSWORD': 'mypassword',
      'HOST': 'localhost',
      'PORT': '5432',
   }
}
```

- Databases and other tools:
- To enter into the shell of MongoDB: mongo
- To enter into the new version of shell of MongoDB: mongosh
- To enter into the shell of Redis DB: redis-cli

- Test and documentation:
- To run the tests: Python-m unittest
- To get the [module_name] document: Python-m pydoc [module_name]

- To install the local packages: Pip install-e .

- Some professional / expert points:
- Python-m pip--help
- Django-admin--help
- Python manage.py--help

- To work with the GitHub, following these instructions:
- To start with the GitHub:
- git init # ایجاد مخزن (repository) محلی جدید
- git clone <url> # کلون گرفتن از مخزن موجود (از GitHub و غیره)

- To Checking the tank status:
- git status # وضعیت فایل‌ها (، آماده، تغییر کرده track/untracked)
- git log # نمایش تاریخچه کامیت‌ها
- git diff # نمایش تفاوت‌ها (قبل از stage)
- git diff--staged # نمایش تفاوت‌ها در فایل‌هایی که stage شدن

- To Add and save changes:
- git add <file> # افزودن یک فایل به stage
- git add . # افزودن همه فایل‌های تغییر یافته
- git commit-m "توضیح کامیت" # ثبت تغییرات با پیام
- git commit-am "تغییر سریع" # افزودن + کامیت فایل‌هایی که track می‌شن

- To use Branches:
- git branch # لیست شاخه‌ها
- git branch new-branch # ساخت شاخه جدید
- git checkout new-branch # سوییچ به شاخه جدید
- git checkout-b feature1 # ساخت و سوییچ همزمان
- git merge other-branch # ادغام شاخه با شاخه جاری
- git branch-d branch-name # حذف شاخه (اگه merge باشه شده)

- To Connecting to a remote repository (Remote):
- git remote add origin <url> # اتصال مخزن راه دور (مثلاً GitHub)
- git push-u origin main # push شاخه اصلی (اولین بار)
- git push # ارسال تغییرات
- git pull # دریافت و ادغام تغییرات
- git fetch # دریافت بدون ادغام

- To Return and recovery:
- git checkout-- <file> # بازگرداندن فایل به آخرین نسخه‌ی commit شده
- git reset HEAD <file> # حذف فایل از stage
- git revert <commit> # برگرداندن یک commit (برگشت امن با commit جدید)
- git reset--hard <commit> # بازگشت کامل به یک commit (خطرناک!)

- To Cleaning and purification:
- git clean-f # حذف فایل‌های غیر track شده
- git stash # ذخیره تغییرات موقت بدون commit
- git stash pop # بازیابی تغییرات stash شده

- To Working with GitHub (simplest version):

```
git clone https://github.com/user/repo.git
cd repo
git checkout-b myfeature
# تغییرات...
git add .
git commit-m "افزودن feature"
git push origin myfeature
```

- To Pro tips for Git:

- git log --oneline --graph نمایش تاریخچه به صورت گرافیکی و خلاصه
- git config --global user.name "Behdad" تنظیم نام کاربر
- git config --global user.email "you@email.com" تنظیم ایمیل کاربر
- git config--global core.editor "code--wait" تعیین ادیتور پیش‌فرض (VS Code)

About:

Behdad Software Developers Group™ Presents

Design, develop and deployment by engineer Behdad Pourtavakoli.

Document written by engineer Behdad Pourtavakoli

1404/01/15- 21:22