**18/10/2024**

pyinstaller -n “Scoring Game” 🡪 if wanted to give the name with space, use the name inside “”, else, simply type the name without “”.

pyinstaller -n or --name “Scoring Game” -F or --onefile scoring\_game.py 🡪 To get the .exe file of the python code.

pyinstaller –distpath (full path like - E:\Python\chapters\APPS) 🡪 This will get the .exe file executable under Folder (APPS)

pyinstaller -i (path like - E:\Python\chapters\APPS\pic.jpg) 🡪 This will give the .exe file icon of the provided image but for this **pip install Pillow** needs to be installed.(Pillow needs to be installed to convert the .jpg file to .ico file)

**17/11/2024**

Made the Flow Chart in Google sheets Drawing and then Slept.

**18/11/2024 (updated SECTION and pulled using git pull )**

Learned to update .docx in vs code codespace(GitHub)

1. download the .docx file from GitHub.
2. Make changes to that file.
3. Re-upload it to GitHub.
4. git pull it to the codespace (git pull origin main)
5. View that file using Office viewer(markdown) extension already installed in your created Code space.

**\*\*Git & GitHub Learning\*\***

-

**18/11/2024**--> status|commit|push from terminal itself

git status (to check if any changes is there or not)

git add file\_name (if the file is not added to the repository)

git add . (to add all files once)

git commit -m "you message to show up"

git push

for adding any new file also, do the same.

~~import os from openpyxl import load\_workbook from openpyxl import Workbook(Wrong approach)~~

**if getting error while git push,**

use--> git fetch origin

git status | git merge origin/main

git add <file name and so on..>

git push origin main | git push origin main --force (if you are absolutely certain that you won't lose important changes that might be present on the remote branch. )

**To pull any changes that are made in the directory into the codespace**

git pull origin main

git add <file name>

**19/11/2024 and 20/11/2024 🡪** At my WeekOFF.

When you start the configuration to your **Git locally** and link it with **GitHub**

git config --global user.name “Nikhilgulati841”

git config --global user.email [nikhilgulati868720@gmail.com](mailto:nikhilgulati868720@gmail.com)

git config --list (to list the credentials saved by **Git**)

*when pushing or publishing for the first time🡪 asks for the authentication and takes it to GitHub to authenticate.*

**Clone Repo to the Local VScode**

Git clone <https://github.com/Nikhilgulati841/PyWhileWork.git>

**Changes form Github Vscode to Local VScode🡪**

Changes made in codespace Vscode – git status (***modified file*** -)

git add <file name> or git add . (to add all)

**Changes push Github Vscode-repo-then-local Vscode to pull🡪**

Git add <file\_name> or git add .

Git commit -m “message of commit”

git push (added to the Repo)

* **Download the updates** from the remote repository 🡪 git fetch origin main (**main** is the branch name )
* **To see what’s changed before merging** the real results to you local system, as **fetch** is only used to get the updates, it will not merge the updates 🡪 git log origin/main ()
* *Also, use* ***q*** *to exit from the git log*
* *Remember,* ***not to update any file in local*** *while pulling from remote to local, else it will show error.*
* **specifically pulling from “main” branch or using git pull directly**🡪 Then use git merge origin/main or git pull or git pull origin main, and from one of the **multiple** remote repositories (i.e. **origin** or **upstream**)
* git pull origin main will do two operations, it will **download/fetches updates** and also **merges the changes** into the current branch.

**Adding file & folders from local** to **Remote Github**

First 🡪 Creating a repository of the same name “local folder name” (This should be empty & should not have any Readme.md or any other file, as this Repo will be updated by the local folder directory)

git remote add origin <https line> (instead of **origin** you could have choose **upstream** also, but then everytime you will have to use upstream as your remote repository name, as this can be renamed also.)

**Rename branch🡪** git remote rename origin upstream (means git remote rename “first name” “updated name”)

**To see the branch name and remote link**🡪 git remote -v (once verified the updated name – eg. git push upstream main)

**If the remote “origin”, already exists🡪**

* **To remove origin –** git remote remove origin
* **To rename origin to upstream –** git remote rename origin upstream
* **Without removing setting url of the remote repository coming from –** git remote set-url origin *or* upstream <https link>

**Git init 🡪** git init (to **initialize Git** Repository)

Git remote add origin <https link>

Git add . (add all)

git commit -m “commit message”

git push -u origin main (-u means telling git to push all the changes in **origin main from now.**)

**Remember Checks –**

* If the file in the local system is not saved locally, it will not gather any changes when doing git status and eventually you will not be able to push the changes.
* Check for the branch name, if master and you pushing in the main branch, then git branch -M main (“updated branch name to main”)
* Re-create the GitHub Repo if facing any challenges from local to remote.
* When **new file** (U – **Untracked** file, needs to be saved)
* When **Modified file** (M – Modified file)
* Stages (when used git add – **staged** for commit) (Before add **Red Modified**, after add **Green Modified**)(Now after git commit no new changes but yet to PUSH **No Unstaged**)
* Also, you can add multiple commits eg. 3 different commits and staged all of them but YET have not PUSHED anything. As soon as you git push origin main, this will push all the changes also as well as number of commits also, i.e. 3 different commits (which you can see from **BLAME** option available when you got to your code without editing in the GitHub Repo or **COMMIT**  option showing ***number of commits***)
* In local Vscode use ls -a or -Hidden to get if you get **.git** if not then the remote repo is not created yet.

**Branches –**

* **To check branch in which I am working right now –** git branch (<green colored> branch will show the current branch & <white colored> branches are, in total branches)
* **To rename branch –** git branch -M main
* **To create a new branch –** git branch <name of the branch> or git checkout -b Nikhil (checkout -b will do **two things**, it will **create a new** branch and you will be **switched** to NEW BRANCH “Nikhil”)(but git branch Nikhil, will only create a branch name Nikhil but you will still remain in the main branch)
* **To change the working branch, where you wish to do the changes –** git checkout Nikhil (Nikhil is the branch name that was recently created above using git checkout -b Nikhil)
* **Changes Done or Not Done but Created New branch and want to reflect in GitHub –** git push -u origin Nikhil (this will reflect to the GitHub also) *if with branch made some changes also, then add first and then push with that branch.*
* **Made multiple branches and want to push all –** git push --all origin
* **Want to delete branches then two sub options –**
  + git branch -d Nikhil (will delete the branch **locally**)
  + git push origin --delete Nikhil (will delete the branch **remotely**)
* Workflow🡪 add a file or modified a file 🡪 git branch (check branch)🡪git checkout Nikhil (change the active working branch)🡪 git add <file name>🡪 git commit -m “commit message”🡪 git push origin Nikhil
* ***You can see changes in your local VSCode also when changing tha active branch*** -- git checkout Nikhil or git checkout main,

**Merge –**

* **To revert any merge –** git revert -m 1 <hash code(eg. c1f29ac)> (hash code available in the area you see number of commits)
* **To merge new branch to main –** git merge origin/Nikhil

Work Flow – GitHub Repo🡪 clone🡪 changes (Untracked file|Modified file)🡪 add🡪 commit🡪 push 🡪 then if needed to pull anywhere else, use **specifically pulling from “main” branch or using git pull directly**🡪

-

**VDD**

**Category wise**

* **SUMIF**(category1, *what to search*,prices\numbers)
* **IF**(condition1(met),print this(could be a text | blank space | a value of a cell),Else print this same way)
* For Other category use **SUMPRODUCT**🡪 sumproduct(range(B2:B)<>”category name” |value of a cell)\*(*range*(B2:B)<>”category2 name” | value of a cell)\*…,Price\number range(*form responses*) + sumproduct(range(B2:B)=”Other”, Price range(*form responses*))
* <> means Not Equal to

**Daily wise**

* Already defined a text date in A2, A3,… and add one day to the below cell like that.
* Price column 🡪 **sumif(**range of all the entries of the timestampt(Raw daily sheet’!A:A)**,**A2(daily dates, not the each entries)**,**range of all the price entries (Raw daily sheet’!B:B)**)**
* To sum each Price Entries for the Each Time Entries, and then A2 is each day DATES(daily)

**Raw daily sheet**

* Arrayformula(Range of all the Price Entries | Range of all the Time Entries)
* Use ARRAYFORMULA to automatically prints what comes in the range of cell provided (i.e. form responses range)
* For the dates Arrayformula(DATE(Year(range),Month(range),Day(Range))

**Bill**

* Using arrayformula(B6:B100-B$5), to get the B5 cell locked, so increase B6-B5, B7-B5 and get the total units from cell 5 till the ***Entries***
* **Units for daily** 🡪 if((D7-D6)>0,D7-D6,””), get the latest two subtracted by each other and if greater than 0 then print, else not.

**FDD**

* Used **iferror**(if condition is met then 🡪 condition, elif some error then in the place of that error print whatever you want)
* Multiple times used, **if**(range🡪fdd[study] =”Yes”,15,-20) with **iferror**
* used mod(value,divisor)=remainder , if this condition of row() is true then print the vale of sum, else not 🡪 if(**mod**(row(2)-row(o$2),7)=0,sum(02:0$2),””) This means to print the value of sum after every 7th cell
* Multiple conditional formatting for different colors to show for 🡪 text starts with | >= | range | etc..
* **=IF(ROW()=9, SUM(O2:O9), IF(MOD(ROW(), 7) = 2, SUM(OFFSET(O$2, (ROW()-8)/7\*7, 0, 7)), ""))** this is what not understandable, but it is for sum of every new week score.
* Then totally the score, giving percentage with Concatenate. Giving a target, then calculating the percentage for the total score as of now.

\*\*NOT DONE\*\*

In \*\***pandas\_dataframe\_timeslots.py**\*\*

-

**\*\*Disclaimer 1-->\*\*** Ask for time slots and ask 7 or till user

did not want to stop using while or for loop, whichever suits you.

\*\*Disclaimer 2-->\*\* Make this data printed as a schedule.csv or xlsx as schedule 1, Schedule 2 or Date 1, Date 2 (Learn Read and write to the file using python)

\*\*Disclaimer 3-->\*\* Use the append method to store N number of time slots and also to update and delete use the list.

\*\*Disclaimer 3-->\*\* As soon as you have typed all your Slots, Display it also. So, 1<sup>st</sup> Table for Time Slot display & 2<sup>nd</sup> for Full Schedule Display.

\*\*Disclaimer 4-->\*\* Make def functions for \*\*ias={tslots,full\_schedule,update:(tslots,dt(defined tasks)),delete-a slot (dt\_data.drop(tslotslist[user\_input\_S.No-1]),rename}\*\* and then the name of the def function in the dictionary and ask the user, if the user wanted to add the timeslots, defined tasks, reminders|Help. Then call the action taken by the user as ias[update], ias[full\_schedule]

\*\*1-->\*\* df = df.rename(columns={'A': 'X', 'B': 'Y'}) or| df.rename(columns={'A': 'X'}, index={'X': 'W'}, inplace=True) or| df.index.name = 'Index\_Name'

-

1--> Use User input to ask for \*\*Time Slots\*\* as t1,t2,t3... t7 or user input until user want's to stop

2--> Same as above, ask user to enter the \*\*Defined Tasks\*\* as d1,d2,d3... same number as \*\*Time Slots\*\*

3--> Same goes with \*\*Reminders\*\* as r1,r2... same number as \*\*Time Slots\*\*

4--> Use data\_dict={"Time Slots":[t1,t2,t3,...],"Defined Tasks":[d1,d2,d3....],"Reminders":[r1,r2,r3....]}

\*\*Second Normal approach\*\* --> Typing in the schedule simply to the dictionary

------------------------------------------------------------------------------------------------------------------------------------------------------------------

\*\*NOT DONE\*\*

In \*\*A New File of Python\*\*

use the CSV or xlsx file to retrieve the data from it and use it in python using the pandas \*\*DataFrame\*\* or any other library. maybe this is in the \*\*Python TuteDude course\*\*

------------------------------------------------------------------------------------------------------------------------------------------------------------------

\*\*NOT DONE\*\*

Get the wifi names and the passwords for all the wifi connected to the laptop using python

------------------------------------------------------------------------------------------------------------------------------------------------------------------

\*\*NOT DONE\*\*

get your \*\*ideas\_while\_work.md\*\* file copy paste in your \*\*JFoldknewme.docx\*\* file saved in Drive and see what happens.

------------------------------------------------------------------------------------------------------------------------------------------------------------------

------------------------------------------------------------------------------------------------------------------------------------------------------------------