1. **Understand folder structure. Perform operations on files in different folders.**

After opening the Visual Studio (an interface to Gitbash), we navigate the existing path via "pwd" and change the directory into a desired location via cd "…". To see the files within the selected folder, we can either use "ls" or "ls -l". The former just shows the folder items in sequential order, while the latter shows them in more detail, e.g., when the file was last updated, who the owner of the file is (as shown below).

```
pwd
cd C:\Users\sarag\OneDrive\Desktop\Assignment
ls -l
```

```
sarag@DESKTOP-D3GN0C0 MINGW64 ~/OneDrive/Desktop/Assignment
$ ls -l
total 18
-rw-r--r-- 1 sarag 197609   162 Mar 30 19:39 '~$Tasks.docx'
-rw-r--r-- 1 sarag 197609    54 Mar 30 20:10  codes.txt
drwxr-xr-x 1 sarag 197609     0 Mar 23 22:48  Data/
-rw-r--r-- 1 sarag 197609 15748 Mar 23 22:49  Tasks.docx
```

To open a subfolder within the selected, we simply use "cd" followed by the subfolder name. In our case, we will open "data subfolder" as shown below.

```
sarag@DESKTOP-D3GN0C0 MINGW64 ~/OneDrive/Desktop/Assignment
$ cd data

sarag@DESKTOP-D3GN0C0 MINGW64 ~/OneDrive/Desktop/Assignment/data
$
```

2. **Automate repeating tasks using Python "for" loops.**

We can automate repeated tasks, such as listing number in the range between 0 and 100. The python will list the numbers as follows. Here, one important part of the coding is, we need to make sure to indent the command "print".

```
>>> for number in range(0, 100):
...     print(number)
...
0
1
2
3
```

3. **Break up work into smaller components using Python functions.**

Breaking up the work into smaller components (e.g., breaking down the codes into functions) is helpful to keep our codes in order and eases the navigation process for the user. As simple example of "date" function is shown as follows:

```
def print_date(year, month, day):
    joined = str(year) + '/' + str(month) + '/' + str(day)
    print(joined)
print_date(2022, 3, 30)
```

```
2022/3/30
```

4. **Use Python "lists" and "dictionaries" appropriately. Demonstrate one of the two.**

One of the advantages of using list is to store as many values as needed in the list, so it saves our time when doing various calculations using these values. An example is shown below:

```
>>> number = [1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> print('number:', number)
number: [1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> print('length:', len(number))
length: 9
```

5. **Automate repeating tasks using Stata "for" loops.**

The logic is similar to what we did in Python: we are automating repeated tasks, especially when there are hundreds of entries in the given data. The example below illustrates how we can generate "race" dummy variables and generating mean values of job interview phone calls that they have received based on their race.

```
. gen white = 1 if (race =="w")
(2,435 missing values generated)

.
. replace white = 0 if(race =="b")
(2,435 real changes made)

. egen LWmean = mean(call) if(white==1) & (l==1)
(3,658 missing values generated)
```

6. **Break up work into smaller components using Stata .do files.**

The logic is also similar to the case in python. A simple example below is showing the processes of creating "mean" of years of experience and generating a new variable EXP by subtracting mean from the actual years of experience and creating a new variable by squaring the EXP variable.

```
egen mexp = mean(yearsexp)
gen EXP = yearsexp - mexp
gen EXP2 = EXP * EXP
```

7. **Read .csv data in in Stata.**

To read csv file in Stata, we can use the following code.

```
. insheet using "gender.csv", comma clear
(30 vars, 295 obs)
```

8. **Fix common data quality errors in Stata (for example, string vs number, missing value).**

To fix a missing value issue, we will replace the empty cells of the variable with zeros. Otherwise, when we e.g., create a dummy variable from a variable with missing values, STATA will generate "1" as opposed to zero, assuming the missing values have infinite values. The example below is

showing this process, we are replacing missing values for the variable “fed” (the employer is federal contractor) with zeros.

```
. replace fed=0 if fed==.
(1,768 real changes made)
```

9. **Aggregate, reshape, and combine data for analysis in Python or Stata. Demonstrate at least one of these data manipulations.**

In STATA, we can show aggregate statistics such as sum and mean using “collapse” command. For example, if we want to see the mean of high- and low-quality resumes by job applicants’ race, we can run the following code and get the results as follows:

```
collapse (mean) new_l=l (mean) new_h=h, by(race)
```

race	new_l	new_h
b	.4977413	.5022587
w	.4977413	.5022587

10. **Prepare a sample for analysis by filtering observations and variables and creating transformations of variables. Demonstrate all three.**

To prepare a sample for analysis using the nls data (national longitudinal survey data), we should start preparing the seed for random numbers e.g., 123. This will allow us to reproduce a particular sequence of 'random' numbers. If we have a big data set with many variables, we can filter by dropping the variables that are not going to be used for the analysis and just keep the ones that will be needed. These steps are shown below.

```
. use nls

. set seed 1234

. keep lwage76 age76 black ed76 daded nodaded momed nomomed famed momdad14 sinmom14 step14 nearc4 nea
> rc2 kww south66 smsa66 reg661-reg669 south76 smsa76
```

From the chosen variables above, if we want to for example transform momed (mom’s years of education) variable by taking the log of it.

```
. gen momlog=log(momed)
(15 missing values generated)
```

11. **Save data in Stata.**

To save the filtered and transformed data that we generated in question 10 by replacing the old file, we will use the “save” command as follows.

```
. save "nls.dta", replace
file nls.dta saved
```

12. **Run ordinary least squares regression in Stata.**

If we want to know the effect of age, race, parents’ education, and residential location on salary, we can run the following OLS regression using the relevant variables. The result is shown as follows:

```
. reg lwage76 age76 black ed76 daded momed nearc4 kww south76 smsa76
```

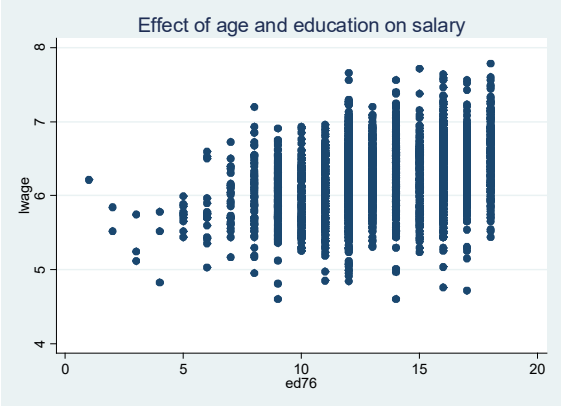
Source	SS	df	MS	Number of obs	=	2,963
				F(9, 2953)	=	133.83
Model	167.196828	9	18.5774254	Prob > F	=	0.0000
Residual	409.910506	2,953	.13881155	R-squared	=	0.2897
				Adj R-squared	=	0.2876
Total	577.107334	2,962	.194837047	Root MSE	=	.37257

lwage76	Coefficient	Std. err.	t	P>|t|	[95% conf.	interval]
age76	.0314024	.0024633	12.75	0.000	.0265724	.0362324
black	-.1321456	.0192283	-6.87	0.000	-.1698479	-.0944433
ed76	.0225928	.0032432	6.97	0.000	.0162337	.028952
daded	-.0033466	.0026331	-1.27	0.204	-.0085095	.0018163
momed	.00503	.0028824	1.75	0.081	-.0006218	.0106818
nearc4	.0159748	.0160013	1.00	0.318	-.0154	.0473496
kww	.0081973	.0011124	7.37	0.000	.0060161	.0103785
south76	-.1191407	.0153823	-7.75	0.000	-.1493019	-.0889796
smsa76	.1456544	.0165186	8.82	0.000	.1132652	.1780436
_cons	4.750391	.0771275	61.59	0.000	4.599162	4.90162

13. Create a graph (of any type) in Stata.

We can generate a scatter plot using “twoway” scatter plot command. This illustrates the effect of e.g., education on wage. The result is shown as follows:

```
. twoway (scatter lwage76 ed76), ytitle(lwage) xtitle(ed76) title(Effect of age and education on sala
> ry)
```



14. Save regression tables and graphs as files. Demonstrate both.

The following outreg2 command saves the regression table into an excel file.

```
. outreg2 using wage, excel replace ctitle("Wage") label
wage.xml
dir : seeout
```

To export a graph from STATA and save it as PDF file, we will use the “graph export” command and get the following result:

```
. graph export wage, as(pdf)
file wage saved as PDF format
```

15. Install a Stata package. (Can be the same as we already did in class.)

To install a package that allows the user to output the analysis result in different file formats (e.g., MS-Word or MS-Excel), we can use the “ssc install” command as follows:

```
. ssc install outreg2
checking outreg2 consistency and verifying not already installed...
all files already exist and are up to date.
```