

|  |  |  |
| --- | --- | --- |
| **Operating Systems** | | |
| Lab Manual | | |
| **Department of Computer Science and Engineering**  **The NorthCap University, Gurugram** | | |
|  | |
|  | |
|  | |
|  | |
|  |

**Operating Systems Lab Manual**

**CSL 303**

**Dr. Priyanka Vasisth**

**Dr. Divya Sharma**

**Ms. Poonam Chaudhary**



Department of Computer Science and Engineering

NorthCap University, Gurugram- 122001, India

Session 2019-20

*Published by:*

**School of Engineering and Technology**

**Department of Computer Science & Engineering**

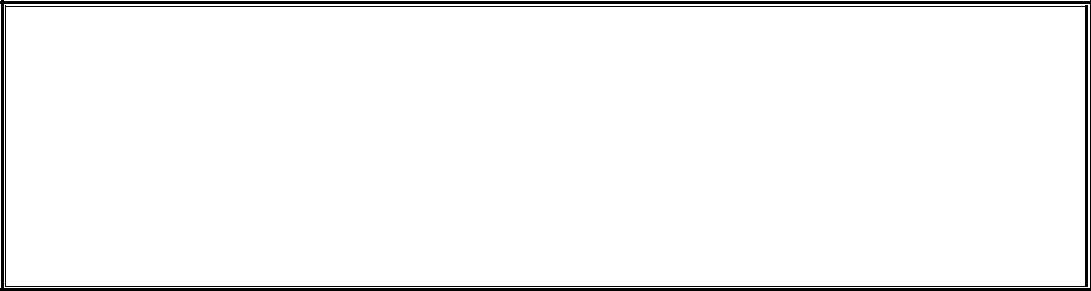
**The NorthCap University Gurugram**

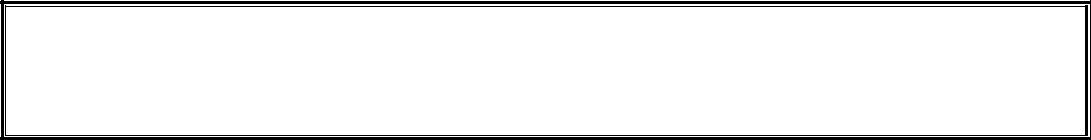
**• Laboratory Manual is for Internal Circulation only**

**© Copyright Reserved**

*No part of this Practical Record Book may be*

*reproduced, used, stored without prior permission of The NorthCap University*



Copying or facilitating copying of lab work comes under cheating and is considered as use of unfair means. Students indulging in copying or facilitating copying shall be awarded zero marks for that particular experiment. Frequent cases of copying may lead to disciplinary action. Attendance in lab classes is mandatory.

Labs are open up to 7 PM upon request. Students are encouraged to make full use of labs beyond normal lab hours.

**PREFACE**

Operating System Lab Manual is designed to meet the course and program requirements of NCU curriculum for B.Tech III year students of CSE branch. The concept of the lab work is to give brief practical experience for basic lab skills to students. It provides the space and scope for self-study so that students can come up with new and creative ideas.

The Lab manual is written on the basis of “teach yourself pattern” and expected that students who come with proper preparation should be able to perform the experiments without any difficulty. Brief introduction to each experiment with information about self-study material is provided. The laboratory exercises will include familiarization with LINUX system calls for process management and inter-process communication; Experiments on process scheduling and other operating system tasks through simulation/implementation. Students would require design process synchronization, CPU scheduling algorithms, memory management and disc management algorithms in high level languages like c, c++, python. Finally, the students would require applying the operating system concepts by experimenting on either xv6/minix operating systems. At the start of each experiment a question bank for preparation and practice is suggested which may be used to test the basic understanding of the students about the experiment. Students are expected to come thoroughly prepared for the lab. General disciplines, safety guidelines and report writing are also discussed.

The lab manual is a part of curriculum for the TheNorthCap University, Gurugram. Teacher’s copy of the experimental results and answer for the questions are available as sample guidelines.

We hope that lab manual would be useful to students of CSE, IT, ECE and BSc branches and author requests the readers to kindly forward their suggestions / constructive criticism for further improvement of the work book.

Author expresses deep gratitude to Members, Governing Body-NCU for encouragement and motivation.

**Authors**

**The NorthCap University**

**Gurugram, India**

|  |  |  |  |
| --- | --- | --- | --- |
| **CONTENTS** | | | |
| S.N. | **Details** | Page No. | |
|  | **Syllabus** |  | |
| 1 | **Introduction** |  | |
| 2 | **Lab Requirement** |  |
| 3 | **General Instructions** |  |
| 4 | **List of Experiments** |  |
| 5 | **List of Flip Assignment** |  |
| 6 | **List of Projects** |  |
| 7 | **Rubrics** |  |
| 8 | **Annexure 1 (Format of Lab Report)** |  |
| 9 | **Annexure 2 (Format of Lab Certificate)** |  |

**SYLLABUS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1. **Department:** | | **Department of CSE** | | |
| 1. **Course Name: Operating Systems** | | 1. **Course Code :** | 1. **L- P** | 1. **Credits** |
| **Code: CSL 303** | 3-2 | 4 |
| 1. **Type of Course (Check one):** | | Programme Core Programme Elective Open Elective  **✓** | | |
| **✓**   1. **Frequency of offering (check one):** Odd Even Either Sem. Every Sem. | | | | |
| 1. **Brief Syllabus:** This is an introductory course which briefs LINUX Operating System Concepts that forms an integral part of computer science engineering in development of software applications in many diverse areas, including Web Development, Windows Applications, Research, Analytics and Processing. It lays the foundation of Process Management & Scheduling, Memory Management, Deadlocks and other Operating system Concepts. | | | | |
| 1. **Total lecture and Practical Hours for this course: 30 Hours**   The class size is maximum 30 learners. | | | | |
| 1. **Course Outcomes (COs)**   Possible usefulness of this course after its completion i.e. how this course will be practically useful to him once it is completed | | | | |
| **CO 1** | The students will be able to understand the basic architecture of Linux. | | | |
| **CO 2** | The students will be able to understand the process management & scheduling of Linux. | | | |
| **CO 3** | The students will be able to understand the memory management of Linux. | | | |
| **CO 4** | The students will be able to understand the inter process communication of Linux. | | | |
| **CO 5** | They will understand the main principles and techniques to handle the deadlocks. | | | |
| **CO6** | They will understand the I/O device management & the VFS of Linux. | | | |
| 1. **UNIT WISE DETAILS No. of Units: -05** | | | | |
| **Unit 1: Introduction to Linux OS Hours: 6**  Introduction & overview: functions of operating systems, Overview of various Operating Systems, Linux architecture, Boot strap loader of Linux, Tasks of the kernel, implementation strategies of kernel, System Calls. | | | | |
| **Unit II: Process Management & Scheduling Hours: 6**  Process priorities, process life cycle of Linux, process representation: process types, process identification numbers, process management system calls, kernel thread, overview of different scheduling algorithms, Linux scheduler: priority and completely fair share scheduling algorithm. | | | | |
| **Unit III: Process Synchronization and Memory Management Hours: 8**  Implementation of Producer- Consumer problem, implementation of semaphores, Page-Replacement Algorithms. | | | | |
| **Unit IV: Deadlocks Hours: 6**  Implementation of Banker’s Algorithm, | | | | |
| **Unit V: Virtual File System Hours: 4**  Disk scheduling algorithms, Introduction to VFS File System types, Common File model, Structure of the VFS | | | | |
| 1. **Guided Project (No. of Hours):** Case Study on Windows OS 2. **Unguided Project (No. of Hours):** Case Study ofLinux, Window, MAC OS | | | | |
| 1. **Brief Description of Self-learning component by students (through books/resource material etc.): Topics:** Linux syntax for shell scripting, revise c/c++/Python and data structure concepts from previous semesters | | | | |
| 1. **Suggested Readings**   GNU/Linux Command−Line Tools Summary [eBook]  <http://www.tldp.org/LDP/Bash-Beginners-Guide/Bash-Beginners-Guide.pdf>  **websites:**   * <https://www.linuxjournal.com/> * <https://www.omgubuntu.co.uk/> | | | | |

1. **INTRODUCTION**

That ‘learning is a continuous process’ cannot be over emphasized. The theoretical knowledge gained during lecture sessions need to be strengthened through practical experimentation. Thus practical makes an integral part of a learning process.

The purpose of conducting experiments can be stated as follows:

* To familiarize the students with the basic concepts, programming skill development and the take home laboratory assignments mainly implementation-oriented which have to be coded in high level language. The lab sessions will be based on exploring the concepts discussed in class.
* Observing basic structure and characteristics of Operating Systems
* Reporting and analyzing the complexities.
* Hands on experience on the experimental setup and software tools

1. **LAB REQUIREMENTS**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Requirements** | **Details** |
| **1** | **Software Requirements** | Linux’s Shell, Python/c/c++ |
| **2** | **Operating System** | Linux Operating System |
| **3** | **Hardware Requirements** | Windows and Linux: Intel 64/32 or AMD Athlon 64/32, or AMD Opteron processor  2 GB RAM  80 GB hard disk space |
| **4** | **Required Bandwidth** | NA |

1. **GENERAL INSTRUCTIONS** 
   1. **General discipline in the lab**
   * Students must turn up in time and contact concerned faculty for the experiment they are supposed to perform.
   * Students will not be allowed to enter late in the lab.
   * Students will not leave the class till the period is over.
   * Students should come prepared for their experiment.
   * Experimental results should be entered in the lab report format and certified/signed by concerned faculty/ lab Instructor.
   * Students must get the connection of the hardware setup verified before switching on the power supply.
   * Students should maintain silence while performing the experiments. If any necessity arises for discussion amongst them, they should discuss with a very low pitch without disturbing the adjacent groups.
   * Violating the above code of conduct may attract disciplinary action.
   * Damaging lab equipment or removing any component from the lab may invite penalties and strict disciplinary action.
   1. **Attendance**

* Attendance in the lab class is compulsory.
* Students should not attend a different lab group/section other than the one assigned at the beginning of the session.
* On account of illness or some family problems, if a student misses his/her lab classes, he/she may be assigned a different group to make up the losses in consultation with the concerned faculty / lab instructor. Or he/she may work in the lab during spare/extra hours to complete the experiment. No attendance will be granted for such case**.**
  1. **Preparation and Performance**
* Students should come to the lab thoroughly prepared on the experiments they are assigned to perform on that day. Brief introduction to each experiment with information about self study reference is provided on LMS.
* Students must bring the lab report during each practical class with written records of the last experiments performed complete in all respect.
* Each student is required to write a complete report of the experiment he has performed and bring to lab class for evaluation in the next working lab. Sufficient space in work book is provided for independent writing of theory, observation, calculation and conclusion.
* Students should follow the Zero tolerance policy for copying / plagiarism. Zero marks will be awarded if found copied. If caught further, it will lead to disciplinary action.
* Refer **Annexure 1** for Lab Report Format

1. **LIST OF EXPERIMENTS**

|  |  |  |
| --- | --- | --- |
| Exp. No. | Division of Experiments | List of Experiments |
| 1 | Basics of Linux | Explain the structure of Linux Operating System |
| 2 | Installation of Ubuntu Operating system |
| 1 | Shell Programs | Write a shell program to find factorial of a number. |
| 2 | Write a shell program to find gross salary of an employee. |
| 3 | Write a shell program to display the menu and execute instructions accordingly  (i)List of file (ii)Process Status (iii) Date (iv) users in program (v) Quit |
| 4 | Write a shell program to find Fibonacci series. |
| 5 | Write a shell program to find largest of three numbers. |
| 6 | Write a shell program to find average of N numbers |
| 7 | CPU Scheduling Algorithms | Write a C program to simulate the following non-preemptive CPU scheduling algorithms to find turnaround time and waiting time.  a) FCFS b) SJF c) Round Robin (pre-emptive) d) Priority |
| 8 | \*Write a C program to simulate multi-level queue scheduling algorithm considering the following scenario. All the processes in the system are divided into two categories – system processes and user processes. System processes are to be given higher priority than user processes. Use FCFS scheduling for the processes in each queue. |
| Implement the following CPU scheduling Algorithms.  i) Round Robin  ii) Priority Based |
| 9 | Deadlock Management  Technique | Write a program to simulate Bankers algorithm for the purpose of deadlock avoidance |
| 10 | Page Replacement  Algorithms | Write a C program to simulate page replacement algorithms  a) FIFO b) LRU c) LFU |
| 11 | Write a C program to simulate page replacement algorithms  a) Optimal |

1. **LIST OF FLIP EXPERIMENTS**
2. Execute the **who** command written in a file to instruct the shell to read input from a file called "myfile1" instead of from the keyboard. Use the **more** command to see the contents of myfile1.
3. Use the date and who commands in sequence (in one line) such that the output of date will display on the screen and the output of who will be redirected to a file called myfile2. Use the more command to check the contents of myfile2
4. Write a sed command that swaps the first and second words in each line in a file.
5. Write a shell script that takes a command –line argument and reports on whether it is directory, a file, or something else.
6. Write a shell script that accepts one or more file name as arguments and converts all of them to uppercase, provided they exist in the current directory.
7. Write a shell script that determines the period for which a specified user is working on the system.
8. Write a shell script that accepts a file name, starting and ending line numbers as arguments and displays all the lines between the given line numbers.
9. Write a shell script that deletes all lines containing a specified word in one or more files supplied as arguments to it.
10. Write a shell script to perform the following string operations:
    * 1. To extract a sub-string from a given string
      2. To find the length of a given string
11. **LIST OF PROJECTS**
    * + 1. Case Study of Window OS
        2. Case Study of Linux OS
        3. Case Study of MAC OS
12. **RUBRICS**

|  |  |
| --- | --- |
| **Marks Distribution** | |
| **Continuous Evaluation(50 Marks)** | **End Semester Exam (20 Marks)** |
| Each experiment shall be evaluated for 10 marks and at the end of the semester proportional marks shall be awarded out of 50. | End semester practical evaluation including Mini project (if any) carries 20 marks. |
| Following is the breakup of 10 marks for each  **4 Marks**: Observation & conduct of experiment. Teacher may ask questions about experiment.  **3 Marks:** For report writing  **3 Marks:** For the 15 minutes quiz to be conducted in every lab. |

**Annexure 1**

**Operating Systems**

**(CSL 303)**

Lab Practical Report



Faculty name: Mrs. Priyanka Student name: Laxmikant Pandey

Roll No.: 18csu1117

Semester: V

Group: DS-B-1

Department of Computer Science and Engineering

NorthCap University, Gurugram- 122001, India

Session 2019-20

**INDEX**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **S.No** | **Experiment** | **Page No.** | **Date of Experiment** | **Date of Submission** | **Marks** | **CO Covered** | **Signature** |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

**Experiment No: #**

Student Name and Roll Number: LAXMIKANT PANDEY 18CSU117

Semester /Section: 5th/B

Link to Code:

Date: 21/11/2020

Faculty Signature:

Remarks:

**Objective**

To familiarize the students to Linux interface.

**Program Outcome**

* The students will understand commands used in Linux.

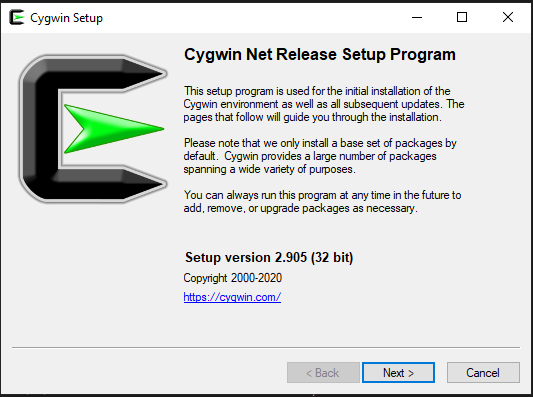
**Problem Statement**

Implement the following things:

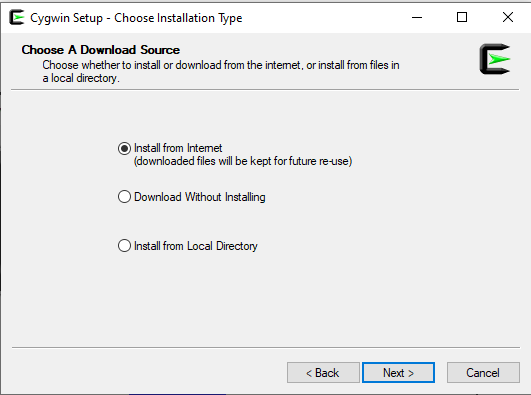
* Cygwin Installation
* Basic Linux commands

**CYGWIN INSTALLATION**

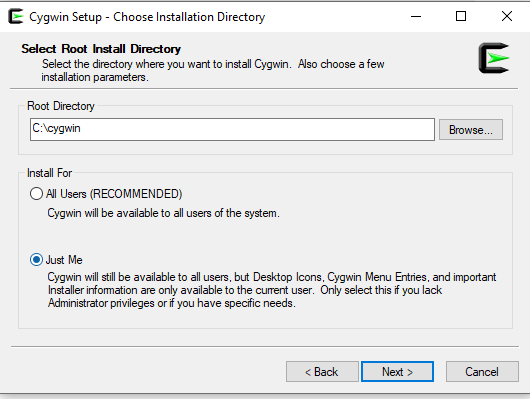
**Step 1**: To install Cygwin in your machine download the Cygwin setup for your respective machine. After downloading double click on setup and press OK. You will see similar window. Click next.



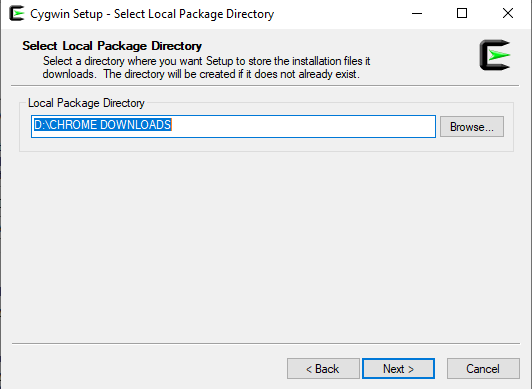
**Step 2**: Now you will see three options. Choose install from internet option and click next to continue.



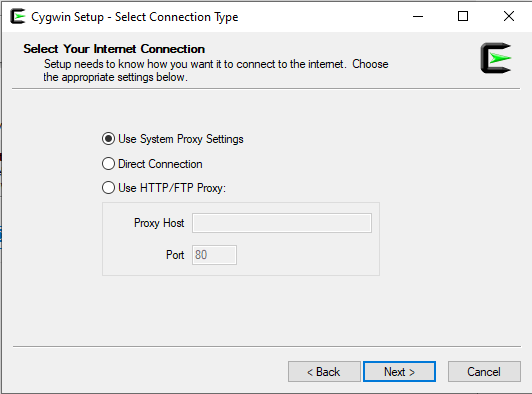
**Step 3**: Now you need to select your root directory. You can change it according to your choice or keep it as default. In the install for option, select just me and hit next.



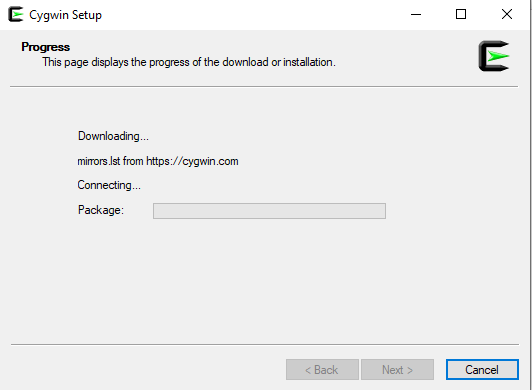
**Step 4:** Now browse to a directory for local packages. After selecting appropriate directory hit next.



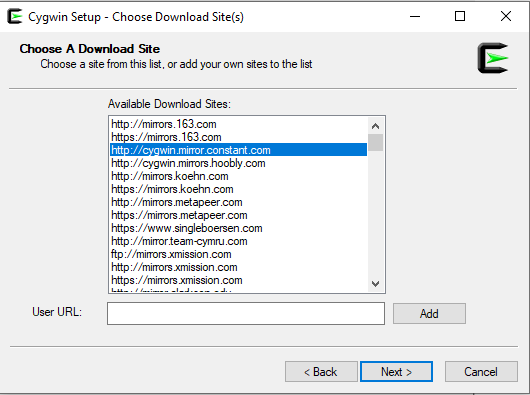
**Step 5**: Now select your connection type. In my case I am keeping it as default.



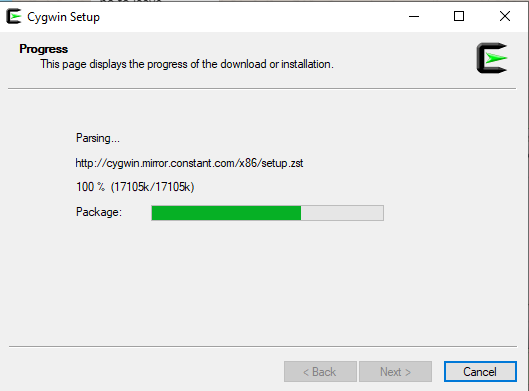
**Step 6**: After selecting connection type, Cygwin will automatically start connecting to the internet.

.

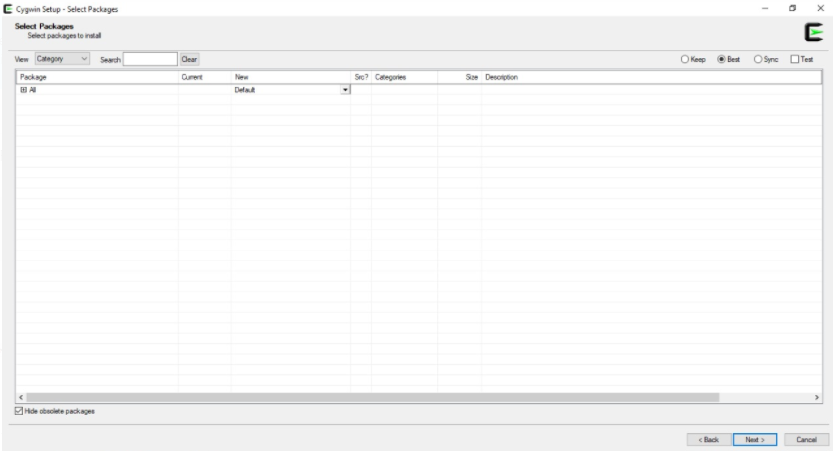
**Step 7**: Now you have to select site for downloading the packages. Select the third site as in image and click next to continue.



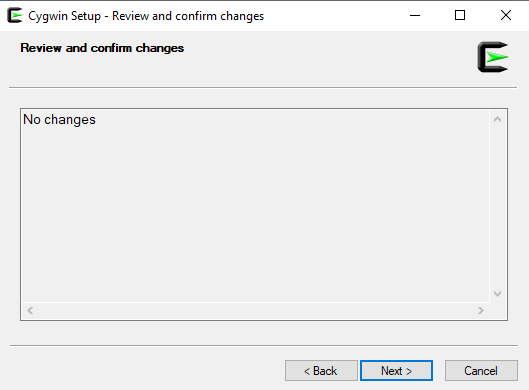
**Step 8**: Now Cygwin will start downloading the default packages from the site.



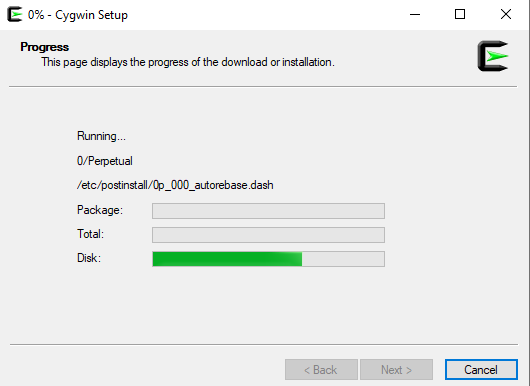
**Step 9**: You will see a screen like the image given below. You can add additional packages you want to download or make no changes and click on next.

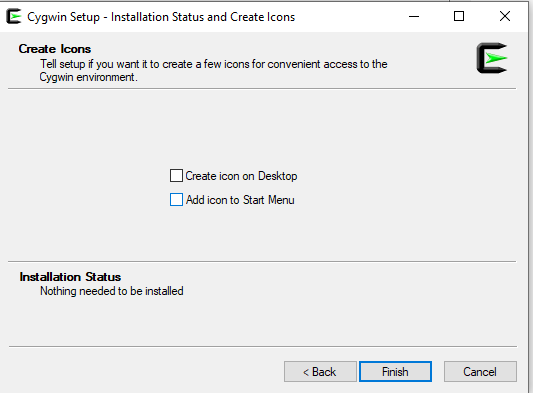


**Step 10**: If you have selected some additional packages, they will show up here else it will show no changes.



**Step 11**: Installation will begin, it will take around 2 minutes. You can see the progress on the top where it is showing currently 0% - Cygwin setup

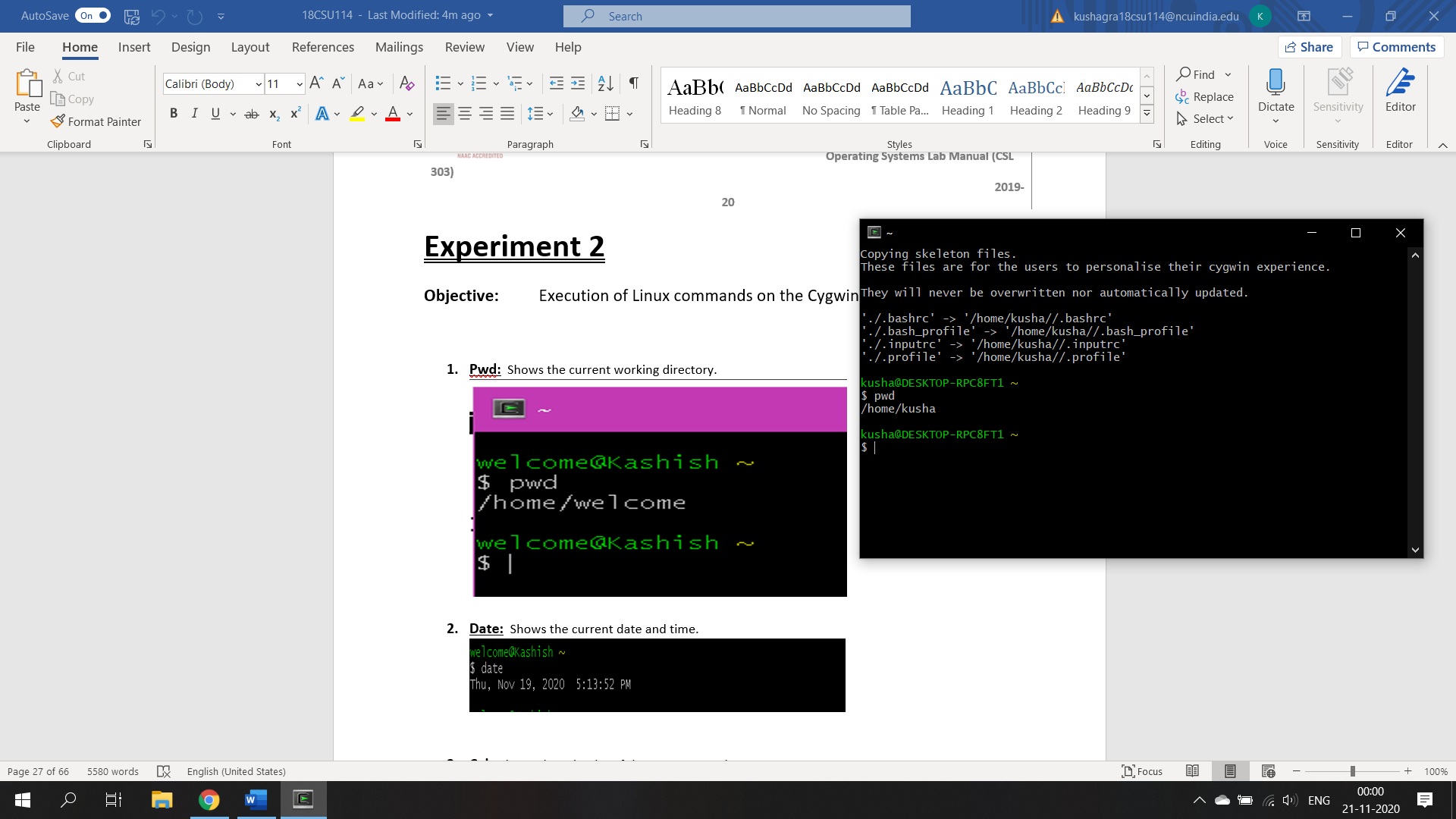


**Step 12**: After the above steps you will see options for icon on desktop and icon to start menu. Its your choice if you want icon on Desktop or start menu or in both places. Just click finish and your installation is complete.

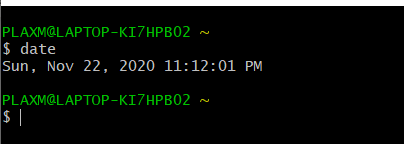
**Experiment 2**

**Objective:**  Execution of Linux commands on the Cygwin terminal.

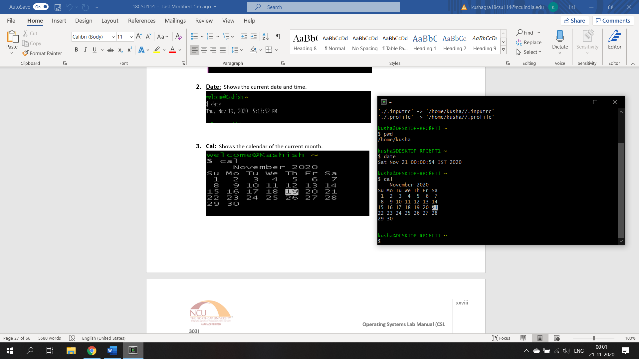
1. **Pwd:** Shows the current working directory.



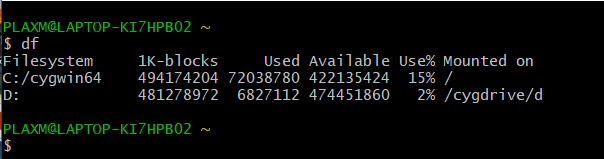
1. **Date:**  Shows the current date and time.



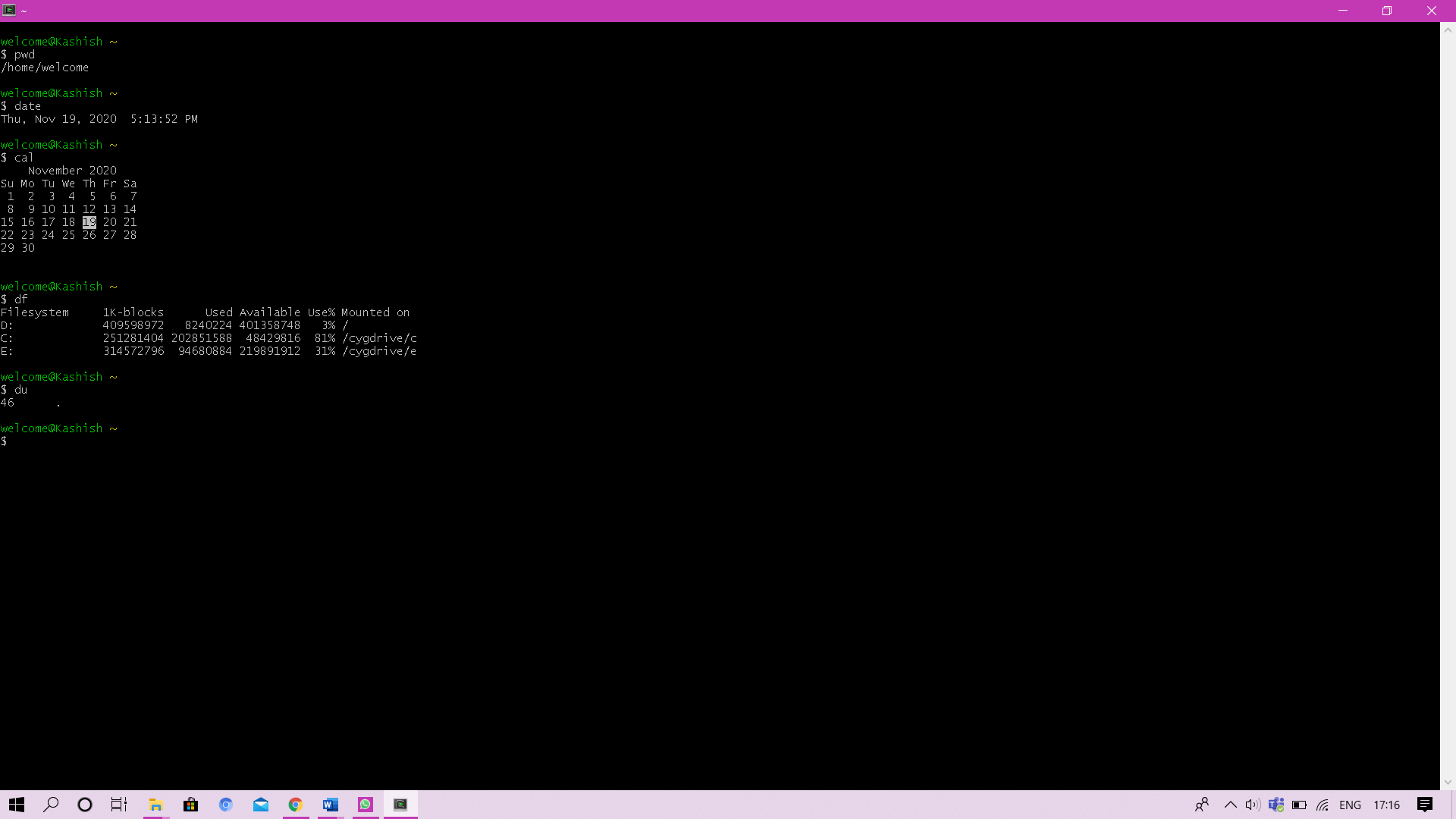
1. **Cal:** Shows the calendar of the current month.



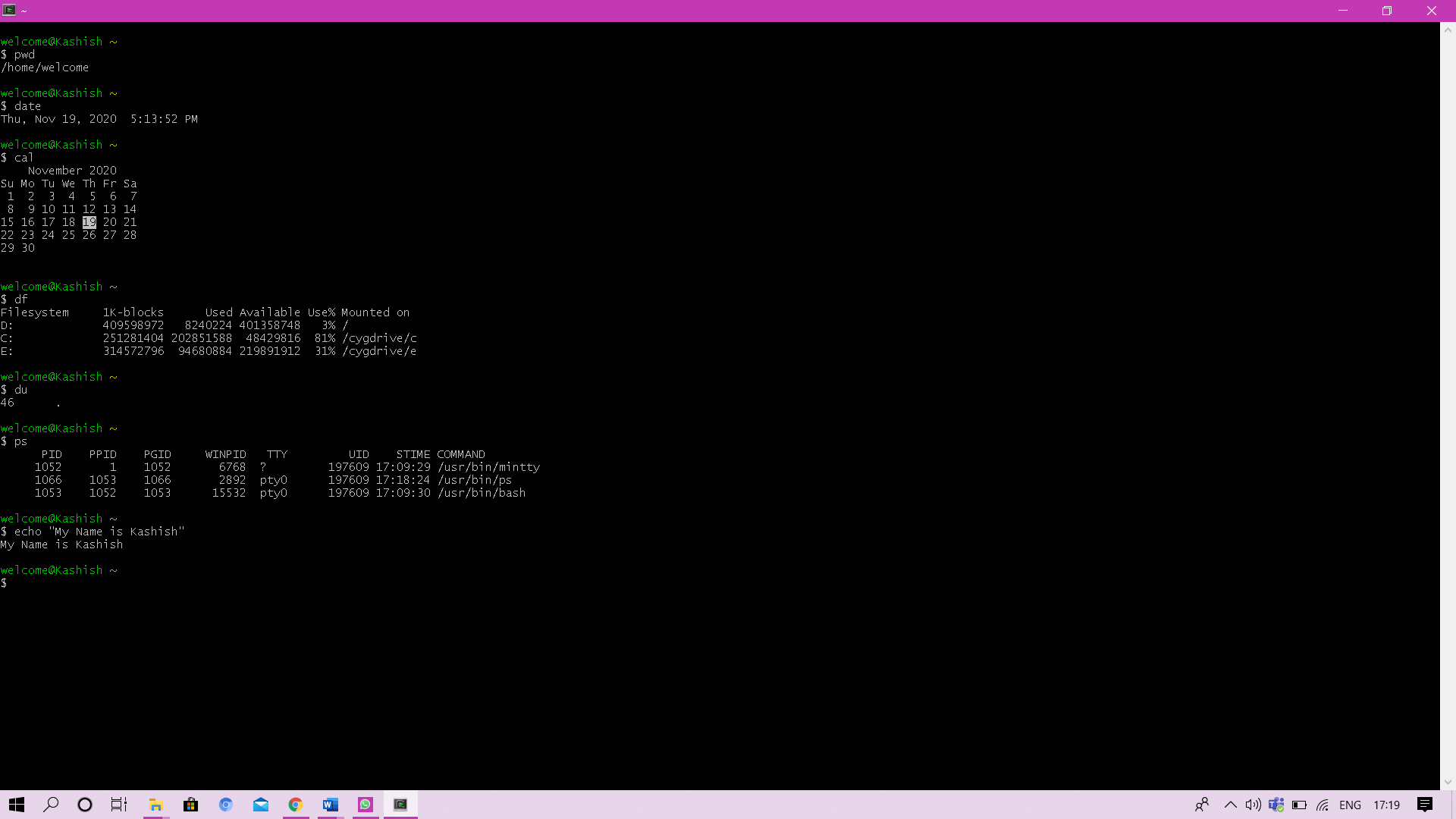
1. **Df:** Shows the disk usage.



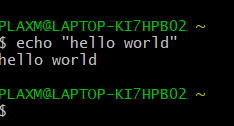
1. **Du:** Shows the directory space usage.



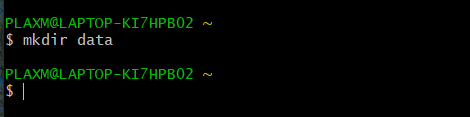
1. **Ps:** Displays the currently working processes.



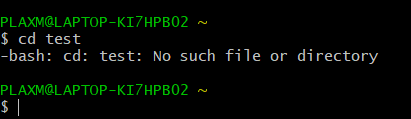
1. **Echo:** Displays the given text.



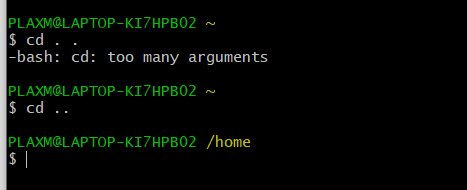
1. **Mkdir:** Creates a new directory.



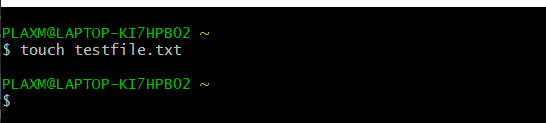
1. **Cd:** Change directory



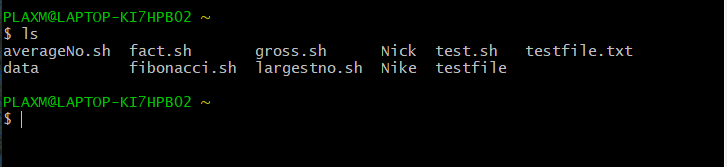
1. **Cd ..** : Go to previous directory.



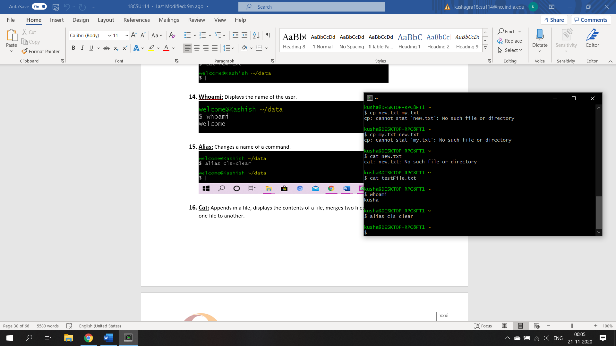
1. **Touch:** Create a new file.



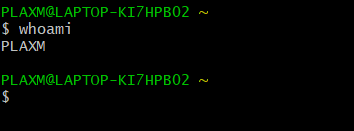
1. **Ls:** Shows the contents in the current directory.



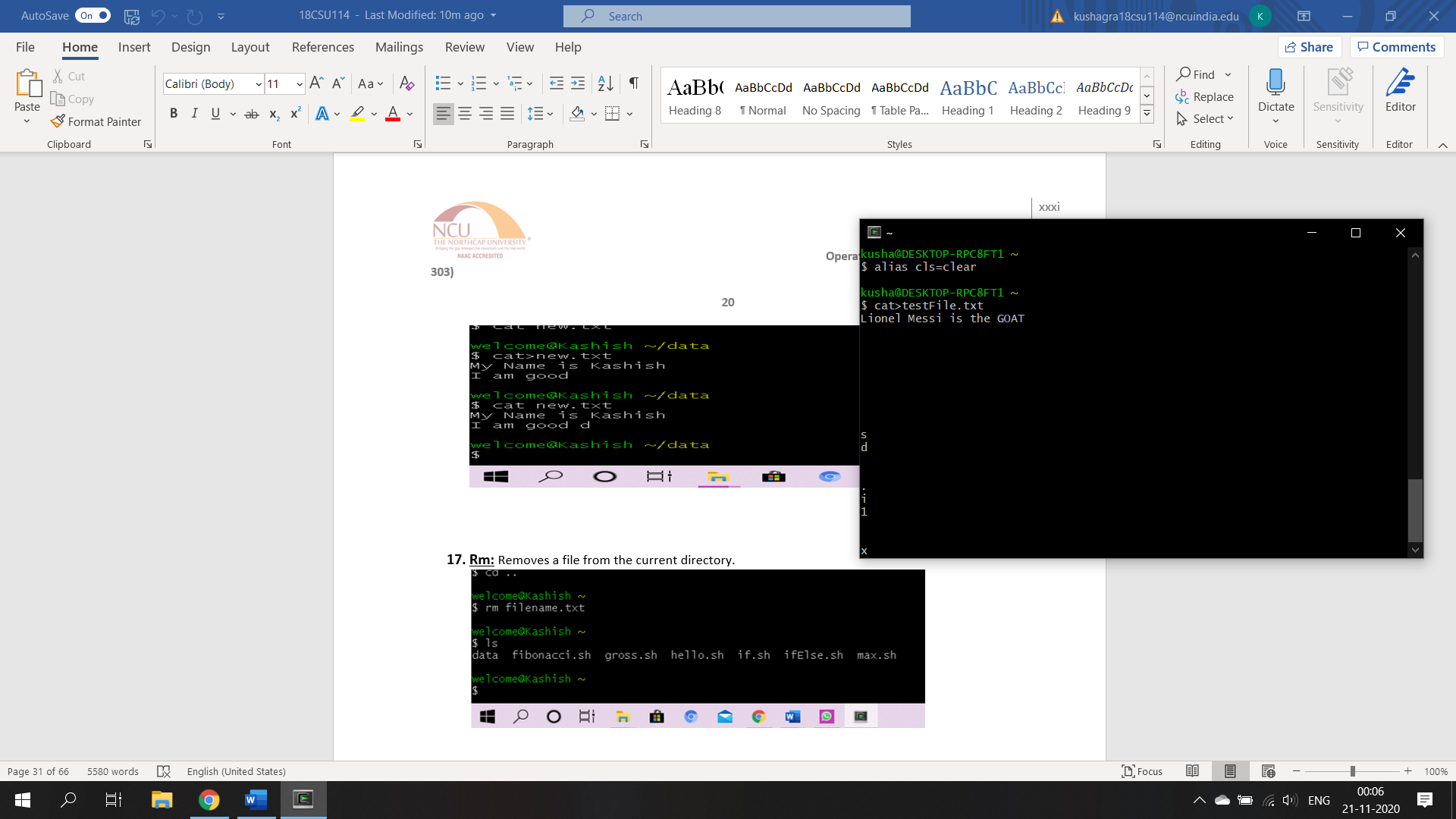
1. **Cp:** Copy a file contents to a new file.



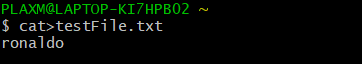
1. **Whoami:** Displays the name of the user.



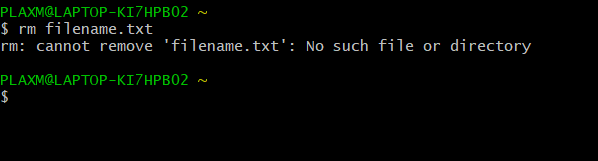
1. **Alias:** Changes a name of a command.



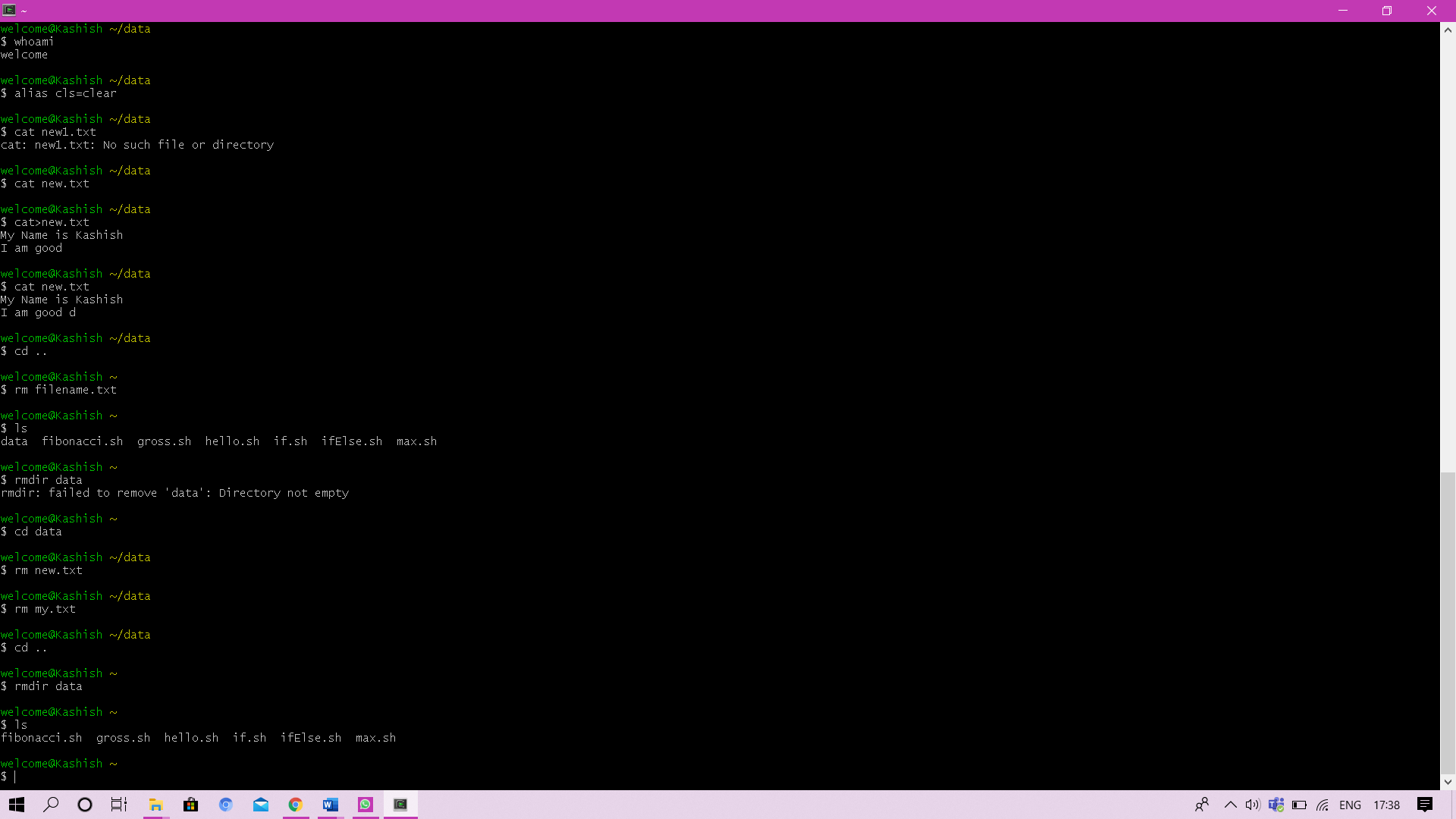
1. **Cat:** Appends in a file, displays the contents of a file, merges two files and copies contents of one file to another.



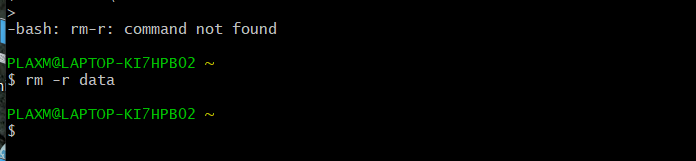
1. **Rm:** Removes a file from the current directory.



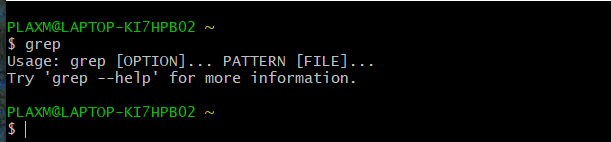
1. **Rmdir:** Removes a directory. The directory should be empty.



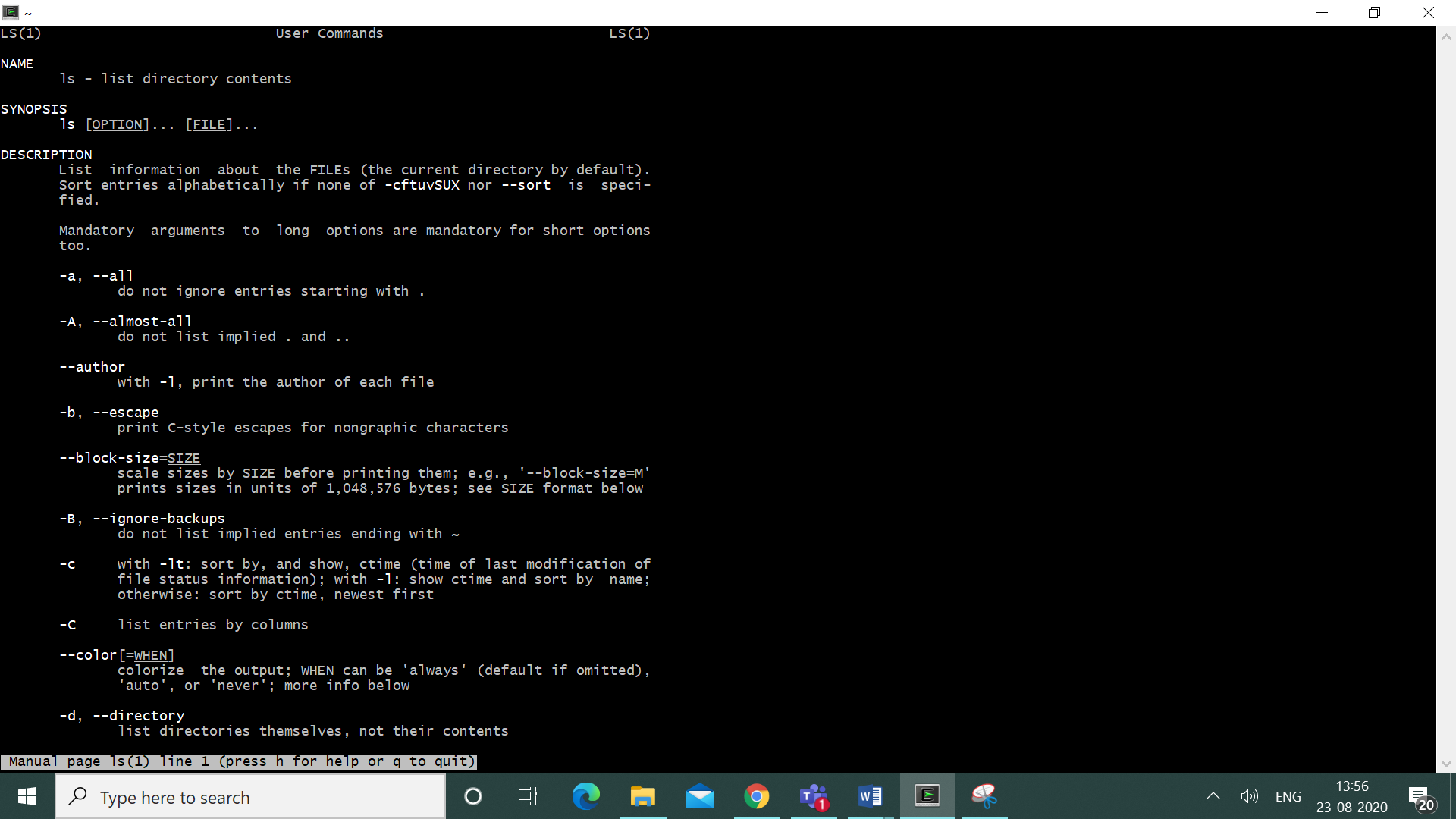
1. **Rm –r:** Removes a non-empty directory by recursively deleting its contents.



1. **Clear:** Clears the terminal screen.
2. **Grep:** Search contents in a file or directory.



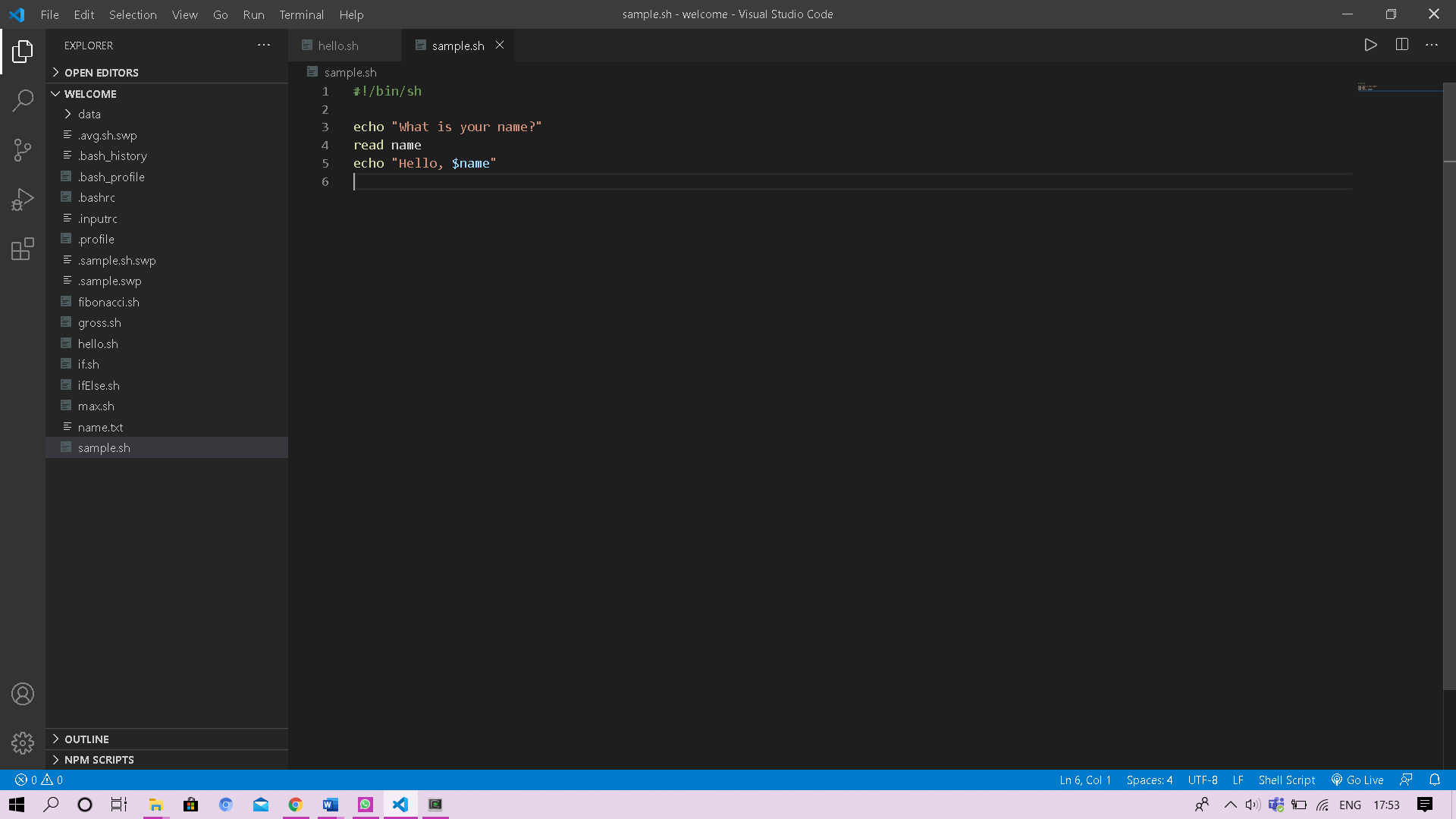
1. **Mv:** move a file from the current working directory to a new existing directory.
2. **Man ls:** Help

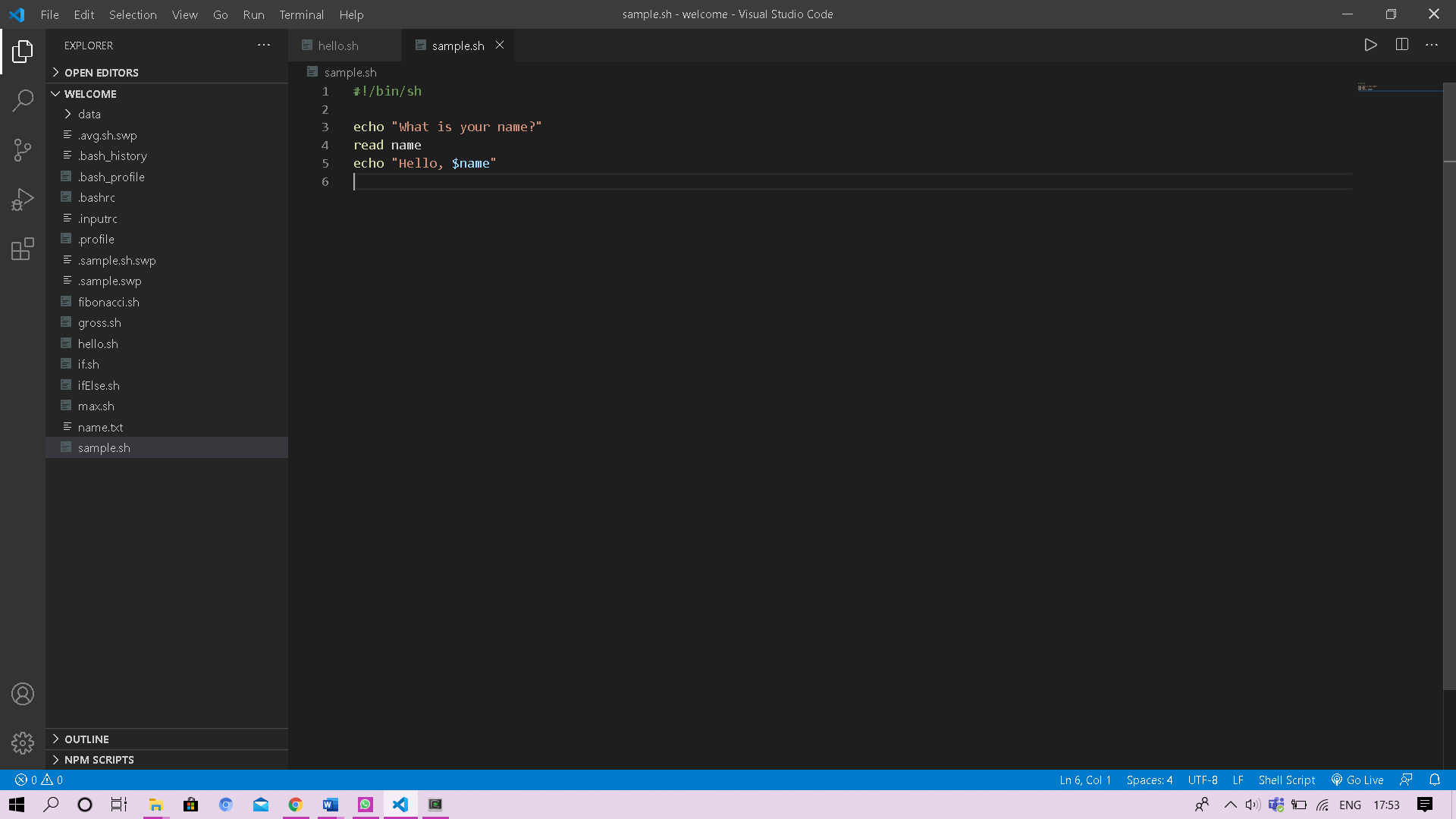


1. **Ping host:** Ping host and output results.
2. **Ctrl+c:** Halts the current command.
3. **Ctrl+z:** Stops the current command, resume with fg in the foreground or bg in the background.
4. **Ctrl+d:** Logout the current session, similar to exit.
5. **Ctrl+w:** Erases one word in the current line.
6. **Ctrl+u:** Erases the whole line.
7. **Ctrl+r:** Type to bring up a recent command.
8. **!!:** Repeats the last command.
9. **Exit:** Logout the current session.

**Experiment 3**

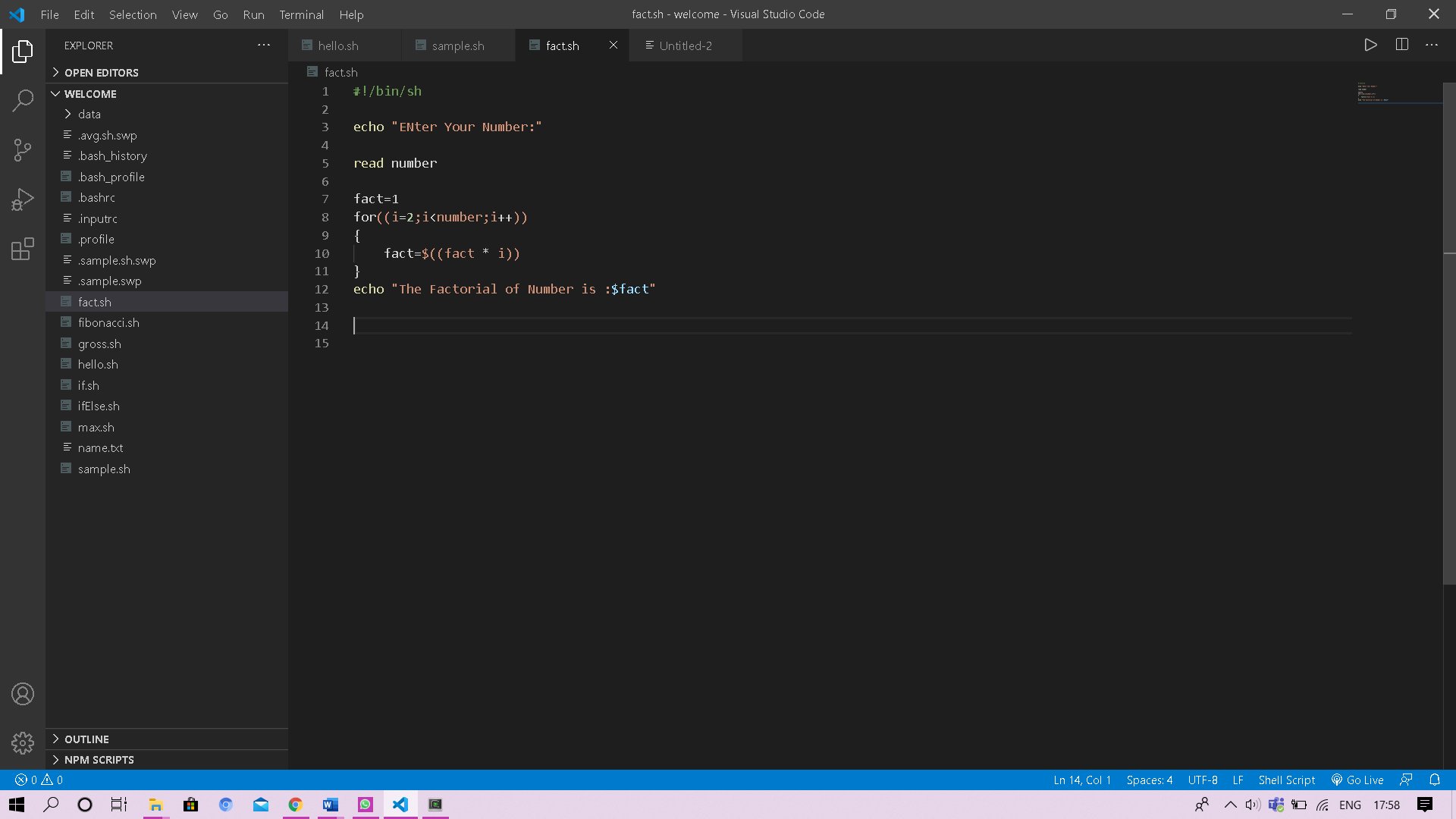
**Objective:** Write a shell program for ‘Hello World’.

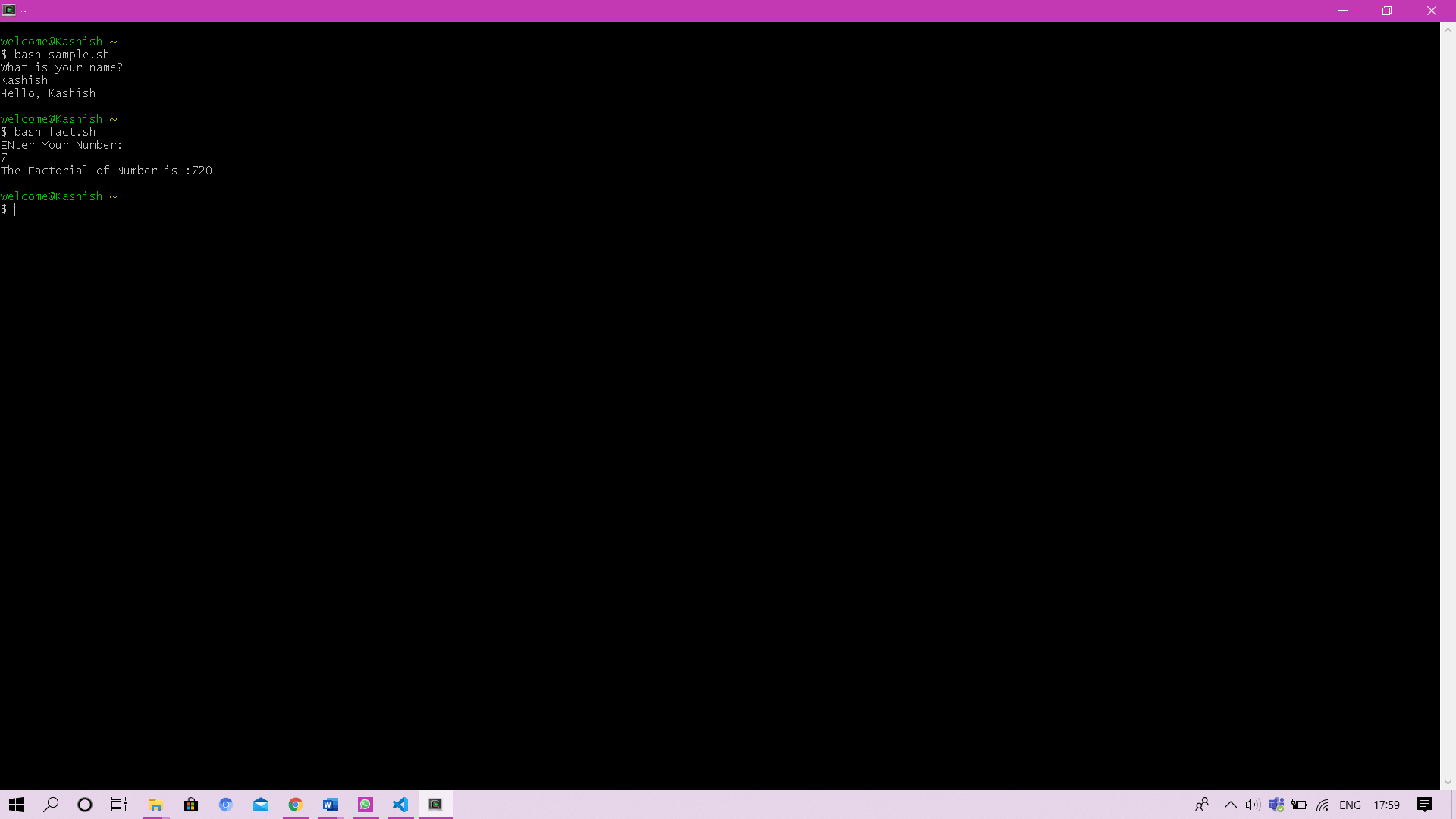




**Experiment 4**

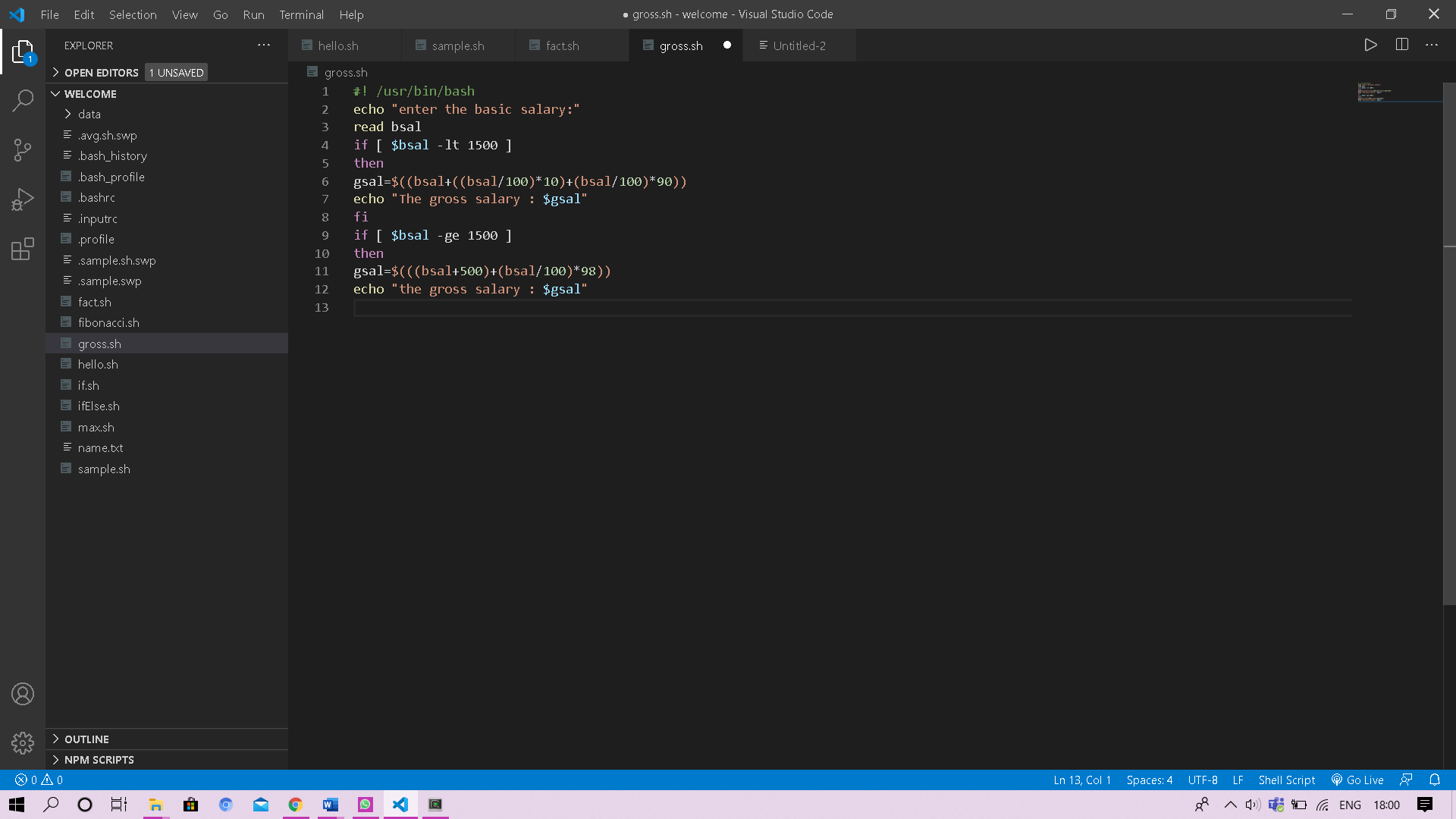
**Objective:** Write a shell program to find factorial of a number.

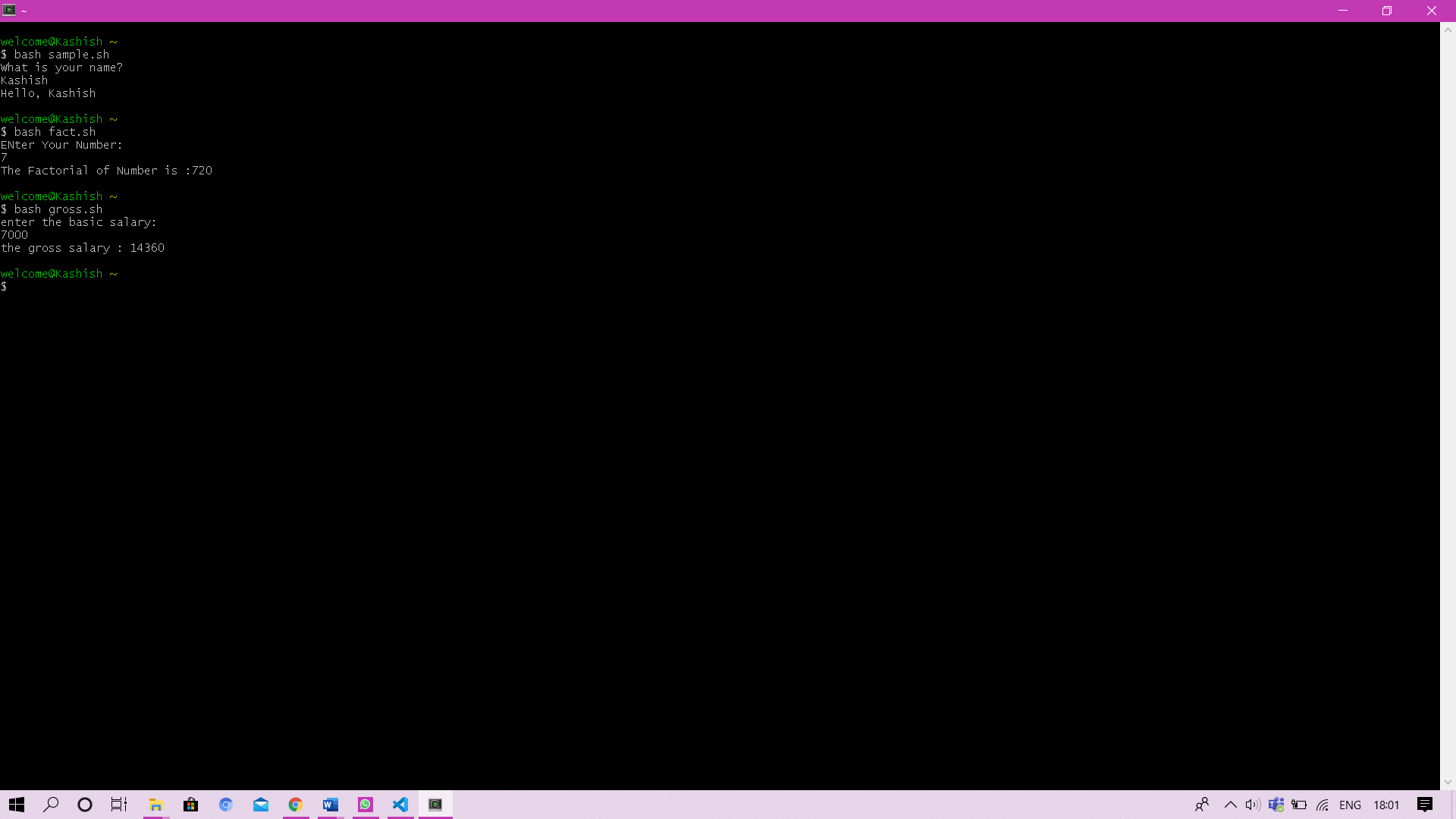




**Experiment 5**

**Objective:** Write a shell program to find gross salary for an employee.

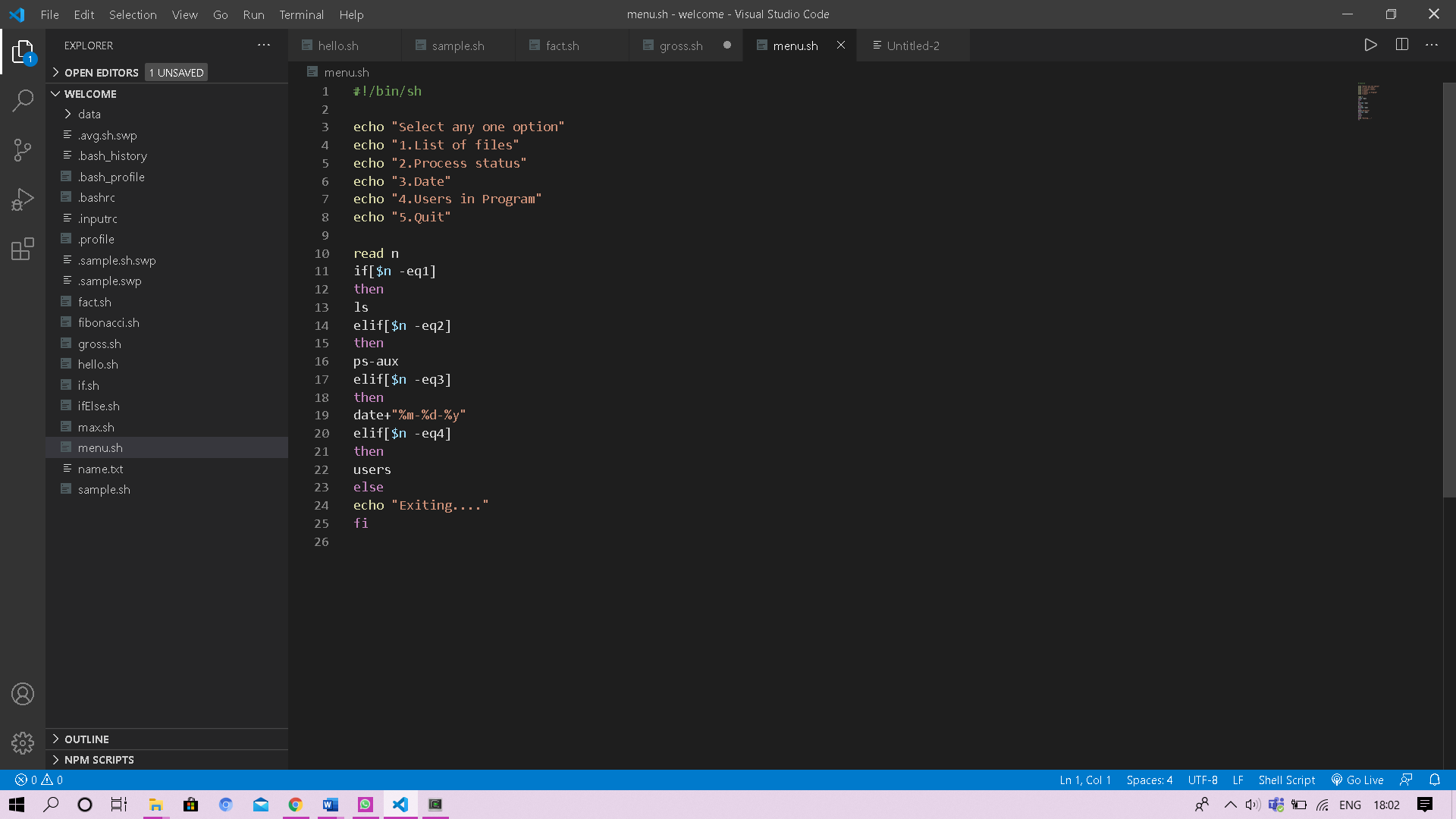


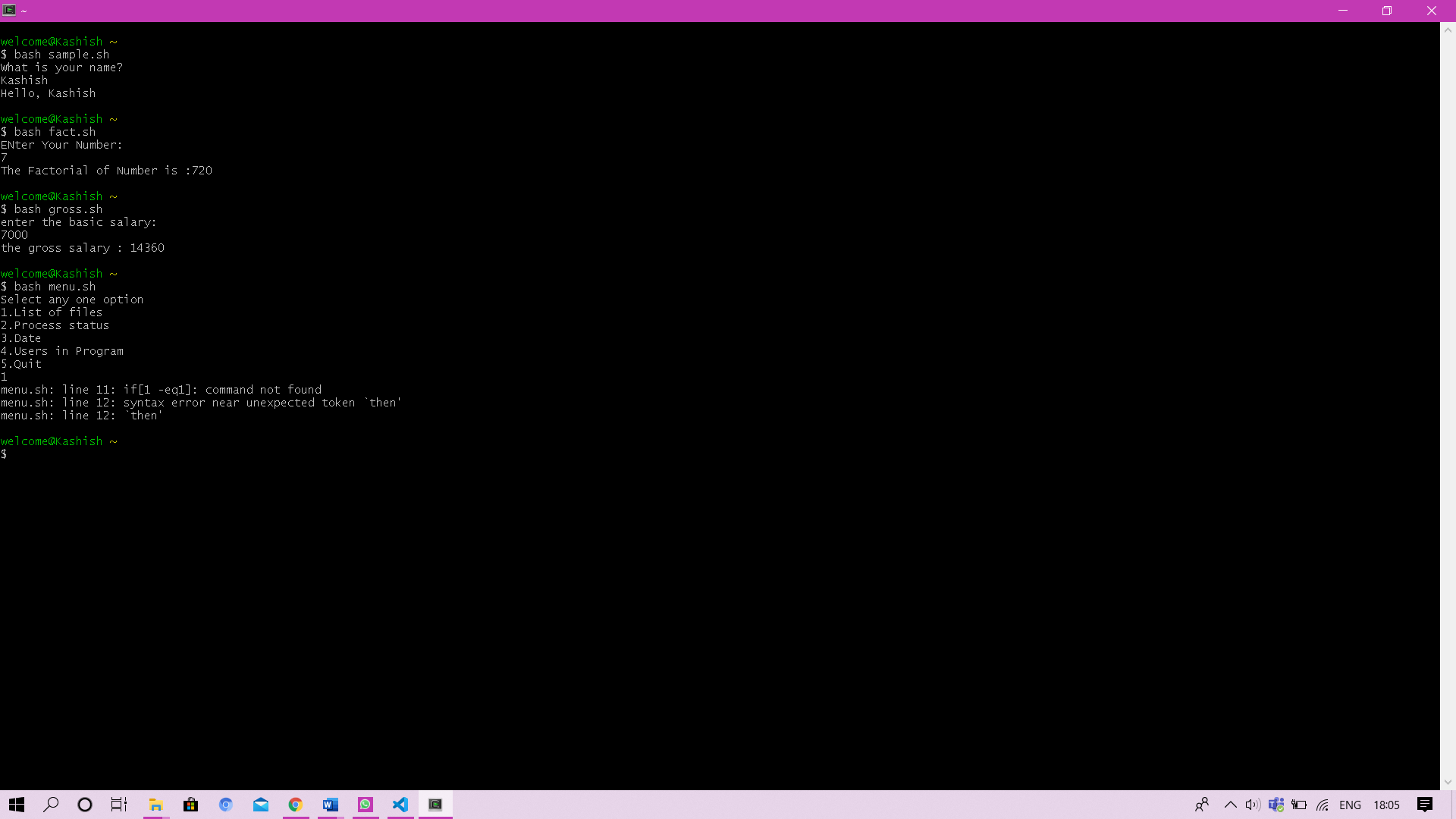


**Experiment 6**

**Objective:** Write a shell program to display the menu and execute instructions accordingly

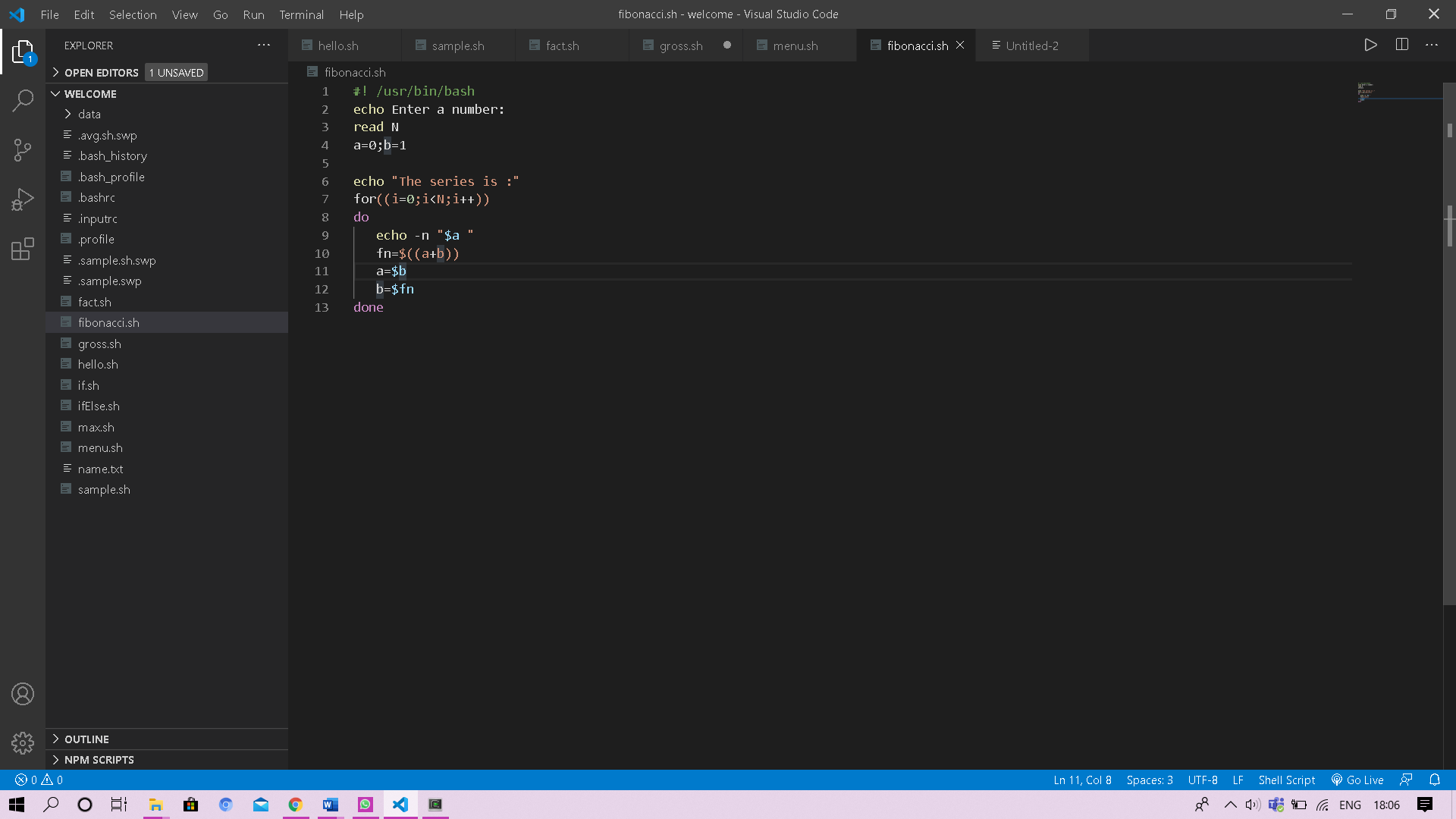
(i)List of file (ii) Process Status (iii) Date (iv) users in program (v) Quit.

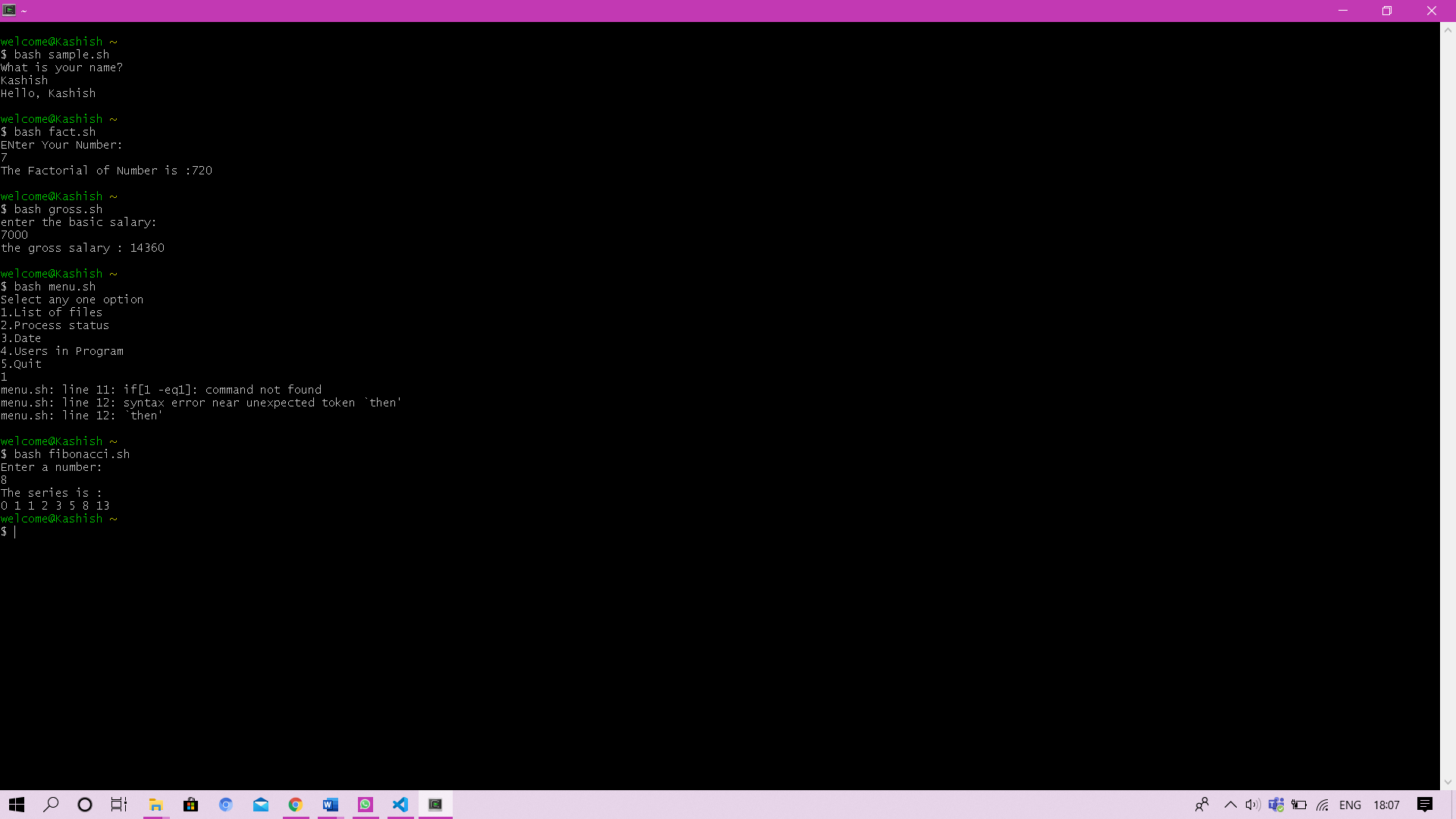




**Experiment 7**

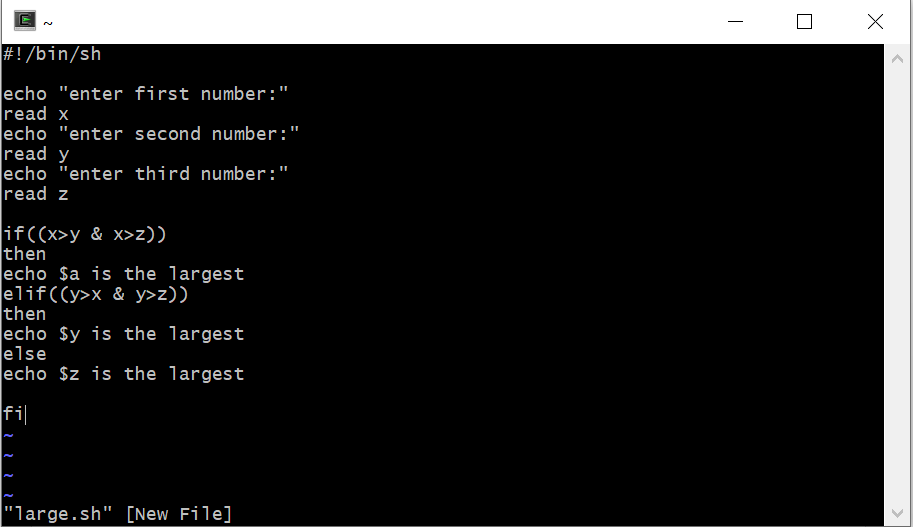
**Objective:** Write a shell program to find Fibonacci series.

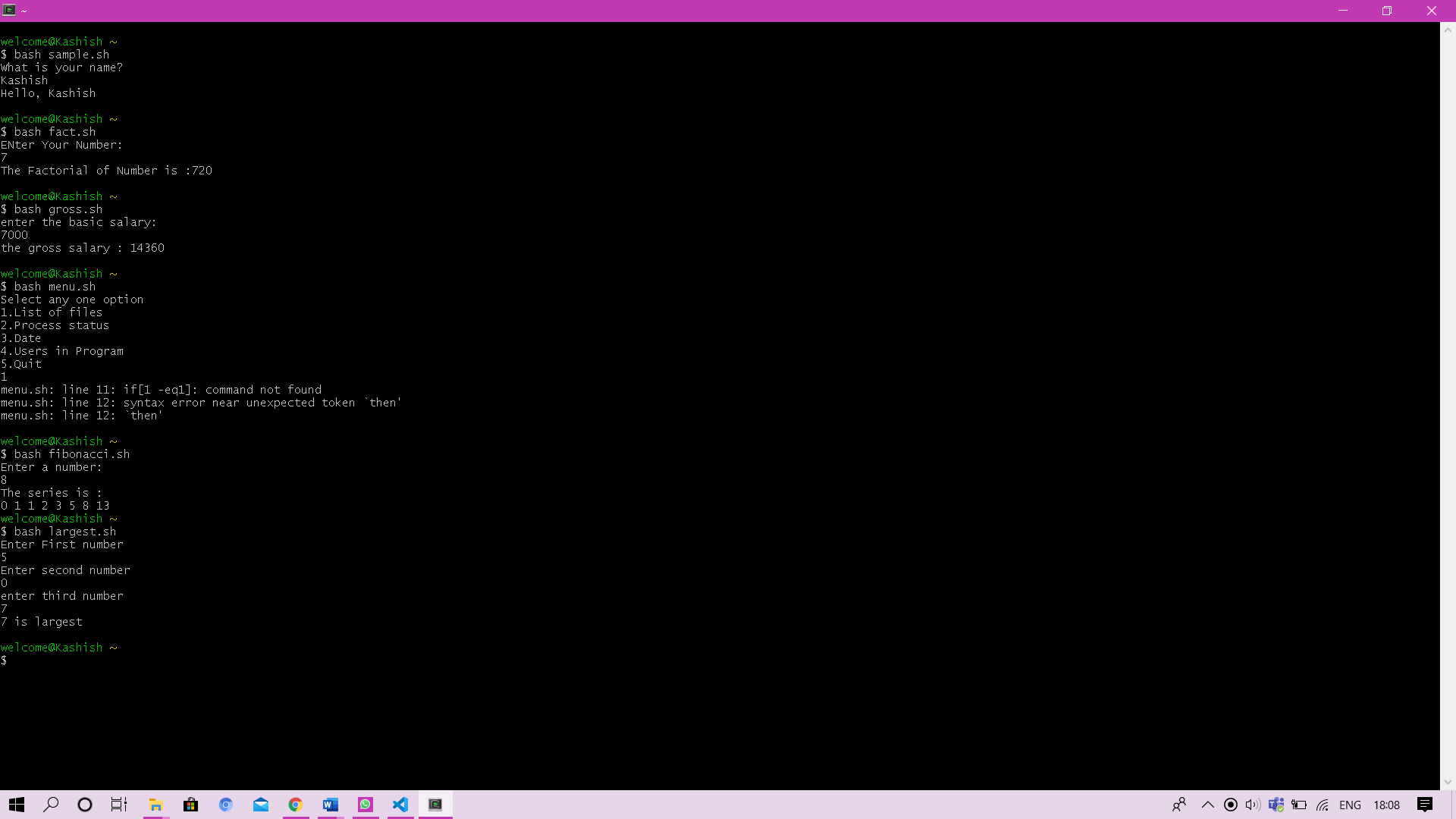




**Experiment 8**

**Objective:** Write a shell program to find largest of 3 numbers



.

**Experiment No: #**

Student Name and Roll Number: Laxmikant Pandey

Semester /Section:5th/B

Link to Code:

Date: 21/11/2020

Faculty Signature:

Remarks:

**Objective**

To familiarize the students about shell scripting.

**Program Outcome**

* The students will understand scripting language.

**Problem Statement**

Implement the following questions.

1. Write a shell program to find factorial of a number.
2. Write a shell program to find gross salary of an employee.
3. Write a shell program to display the menu and execute instructions accordingly

(i)List of file (ii)Process Status (iii) Date (iv) users in program (v) Quit

1. Write a shell program to find Fibonacci series.
2. Write a shell program to find largest of three numbers.
3. Write a shell program to find average of N numbers

**Experiment No: #**

Student Name and Roll Number: : Laxmikant Pandey

Semester /Section: 5th/B

Link to Code:

Date: : 21/11/2020

Faculty Signature:

Remarks:

**Objective**

To familiarize the students about CPU scheduling Algorithms.

**Program Outcome**

* The students will understand the First-cum-first-serve and shortest job first algorithm.

**Problem Statement**

Implement the following CPU scheduling Algorithms.

1. FCFS
2. Shortest Job First

**Background Study:**

**FCFS**

* The simplest CPU-scheduling algorithm is the first-come, first-served (FCFS) scheduling algorithm. With this algorithm, processes are assigned the CPU in the order they request it.
* There is a single queue of ready processes.
* The implementation of the FCFS policy is easily managed with a FIFO queue. When a process enters the ready queue, its PCB is linked onto the tail of the queue.
* The average waiting time under the FCFS policy, however, is often quite long.

**SHORTEST JOB FIRST**

* This algorithm associates with each process the length of the process's next CPU burst.
* When the CPU is available, it is assigned to the process that has the smallest next CPU burst. If the next CPU bursts of two processes are the same, FCFS scheduling is used.
* The SJF scheduling algorithm gives the minimum average waiting time for a given set of processes
* The real difficulty with the SJF algorithm knows the length of the next CPU request.
* Shortest Job first has the advantage of having minimum average waiting time among all scheduling algorithms.
* It is a Greedy Algorithm.
* It may cause starvation if shorter processes keep coming. This problem can be solved using the concept of aging.

**Algorithm/ flowchart:**

**FCFS:**

1. Input the processes along with their burst time (bt).
2. Find waiting time (wt) for all processes.
3. As first process that comes need not to wait so waiting time for process 1 will be 0 i.e. wt[0] = 0.
4. Find **waiting time** for all other processes i.e. for

process i ->

wt[i] = bt[i-1] + wt[i-1] .

1. Find **turnaround time** = waiting\_time + burst\_time for all processes.
2. Find **average waiting time** = total\_waiting\_time / no\_of\_processes.
3. Similarly, find **average turnaround time** =total\_turn\_around\_time / no\_of\_processes.

**FLOWCHART:**

**Diagram

Description automatically generated**

**SJF:**

1. Traverse until all process gets completely executed.
   1. Find process with minimum remaining time at every single time lap.
   2. Reduce its time by 1.
   3. Check if its remaining time becomes 0
   4. Increment the counter of process completion.
   5. Completion time of current process = current\_time +1;
   6. Calculate waiting time for each completed process.
2. wt[i]= Completion time - arrival\_time-burst\_time
3. Increment time lap by one.
4. Find turnaround time (waiting\_time+burst\_time).

**FLOWCHART:**

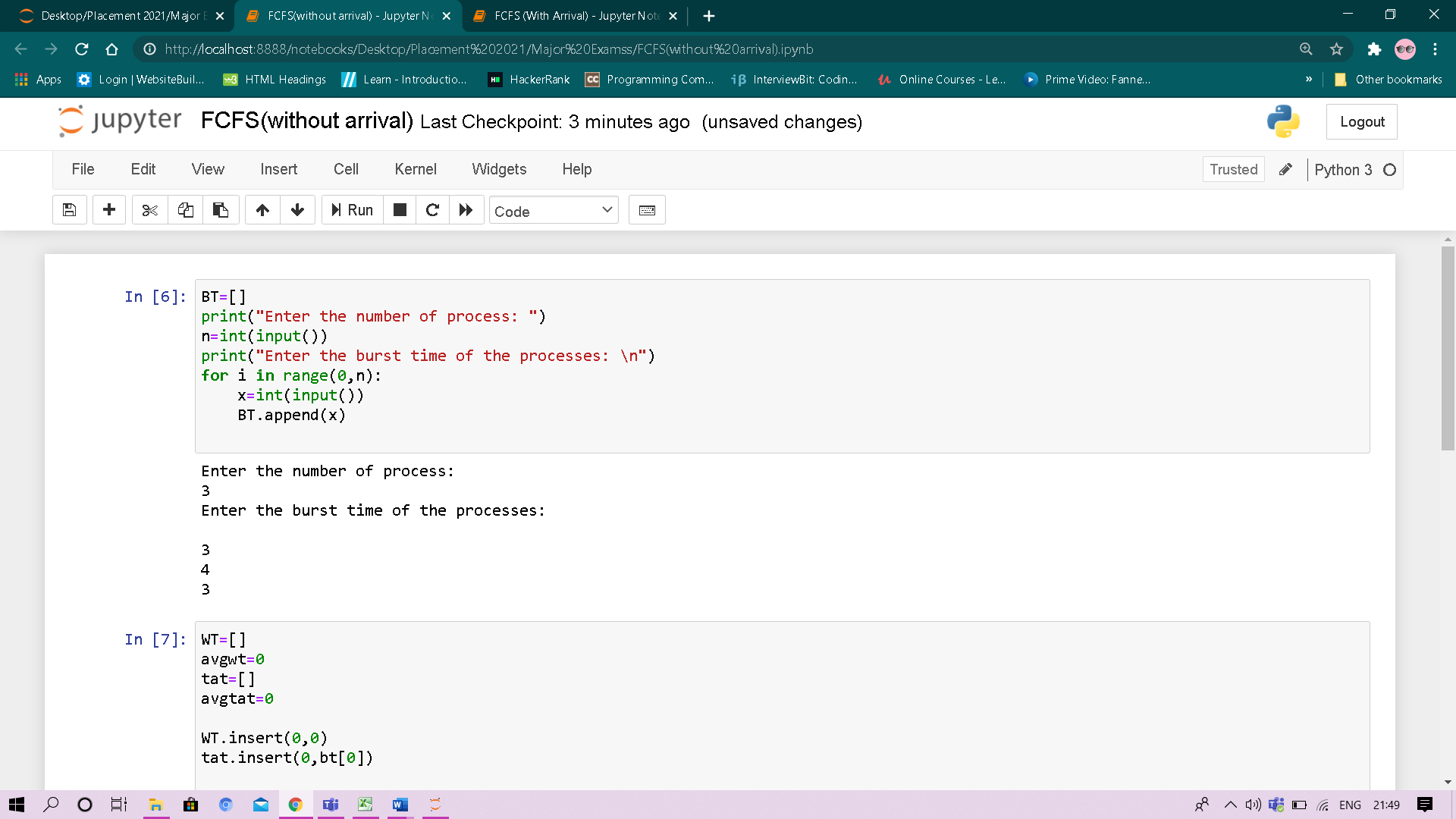
**Diagram

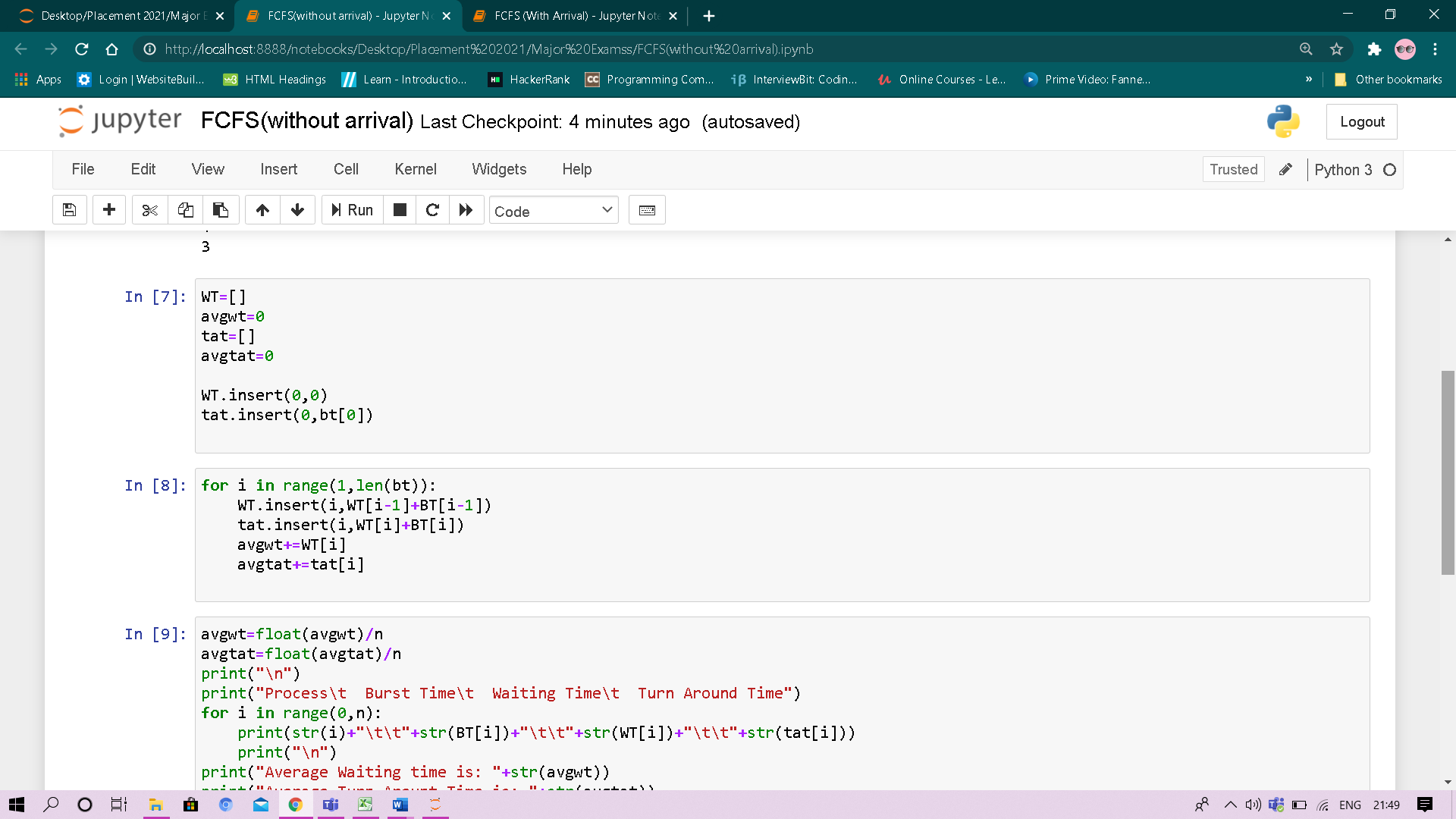
Description automatically generated**

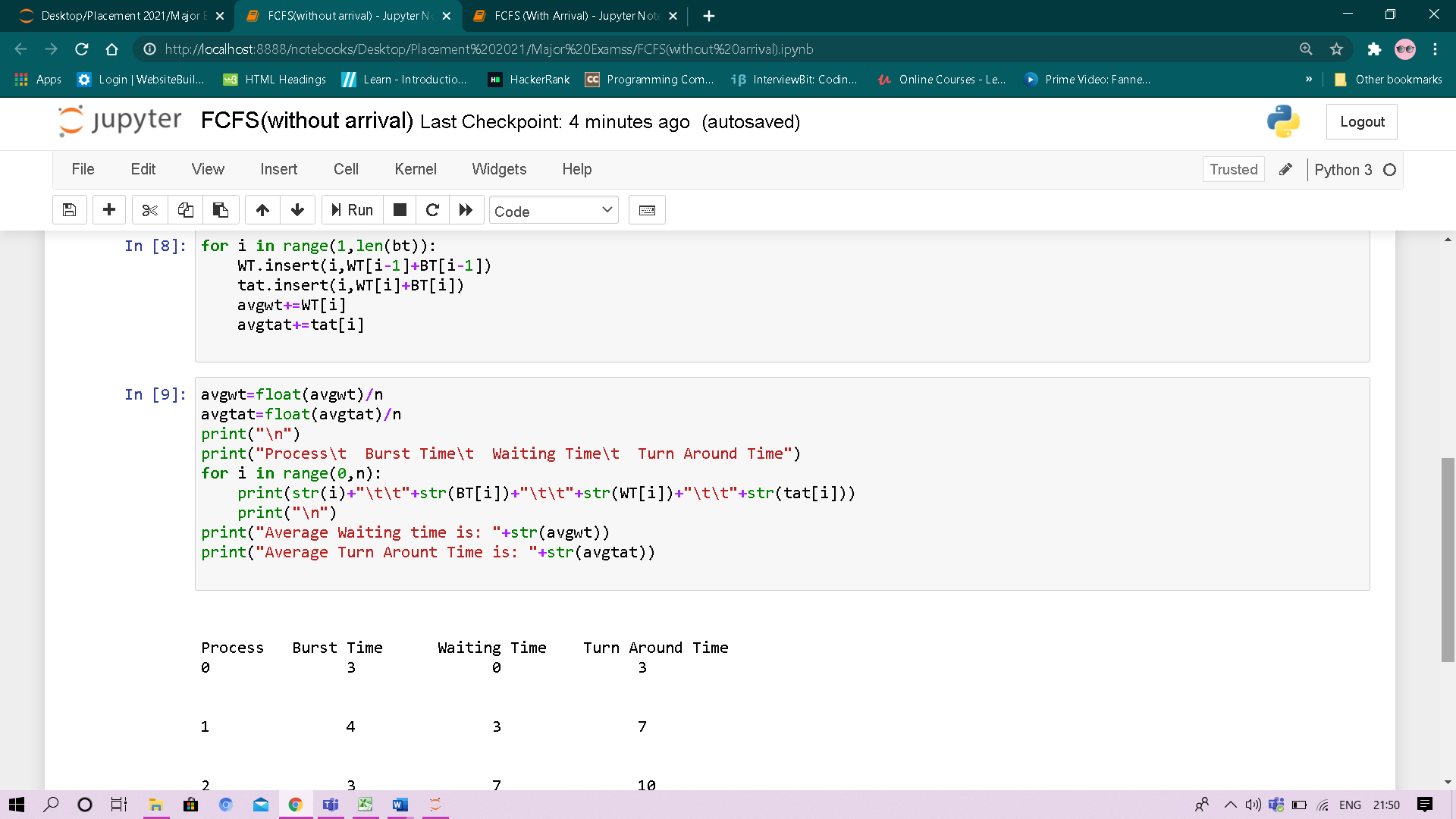
**Code**

**FCFS**

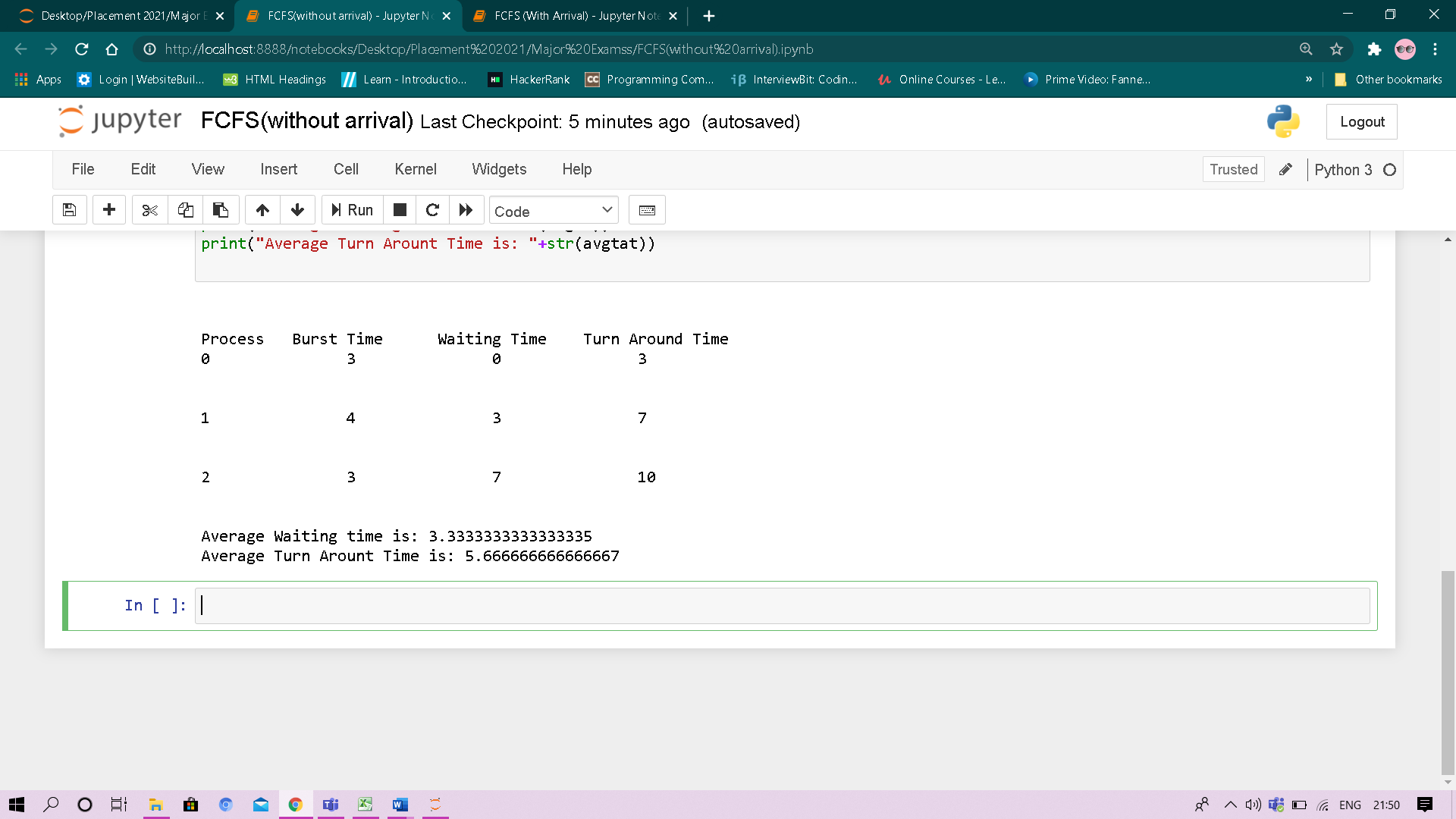
Without Arrival Time:



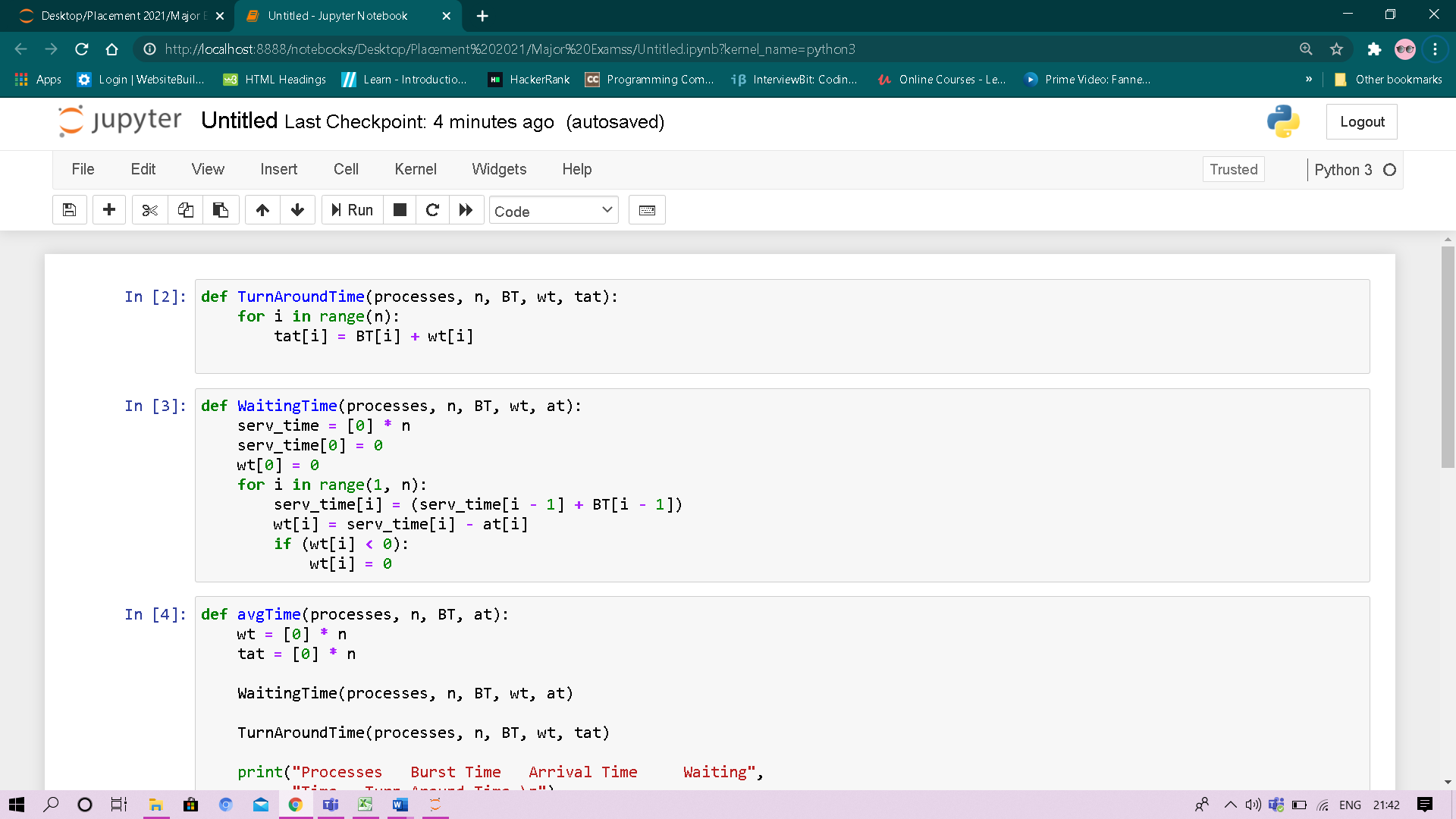


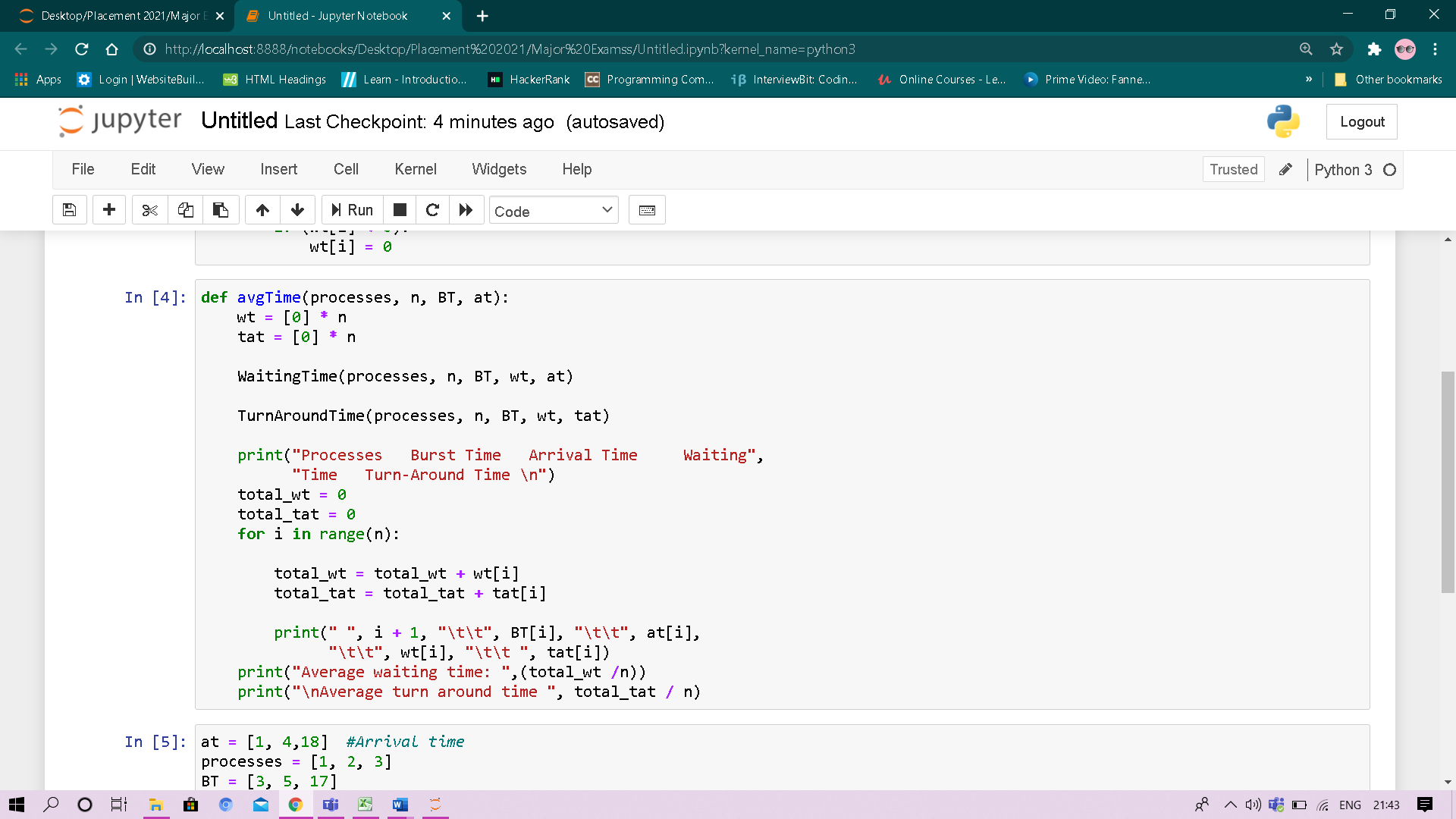


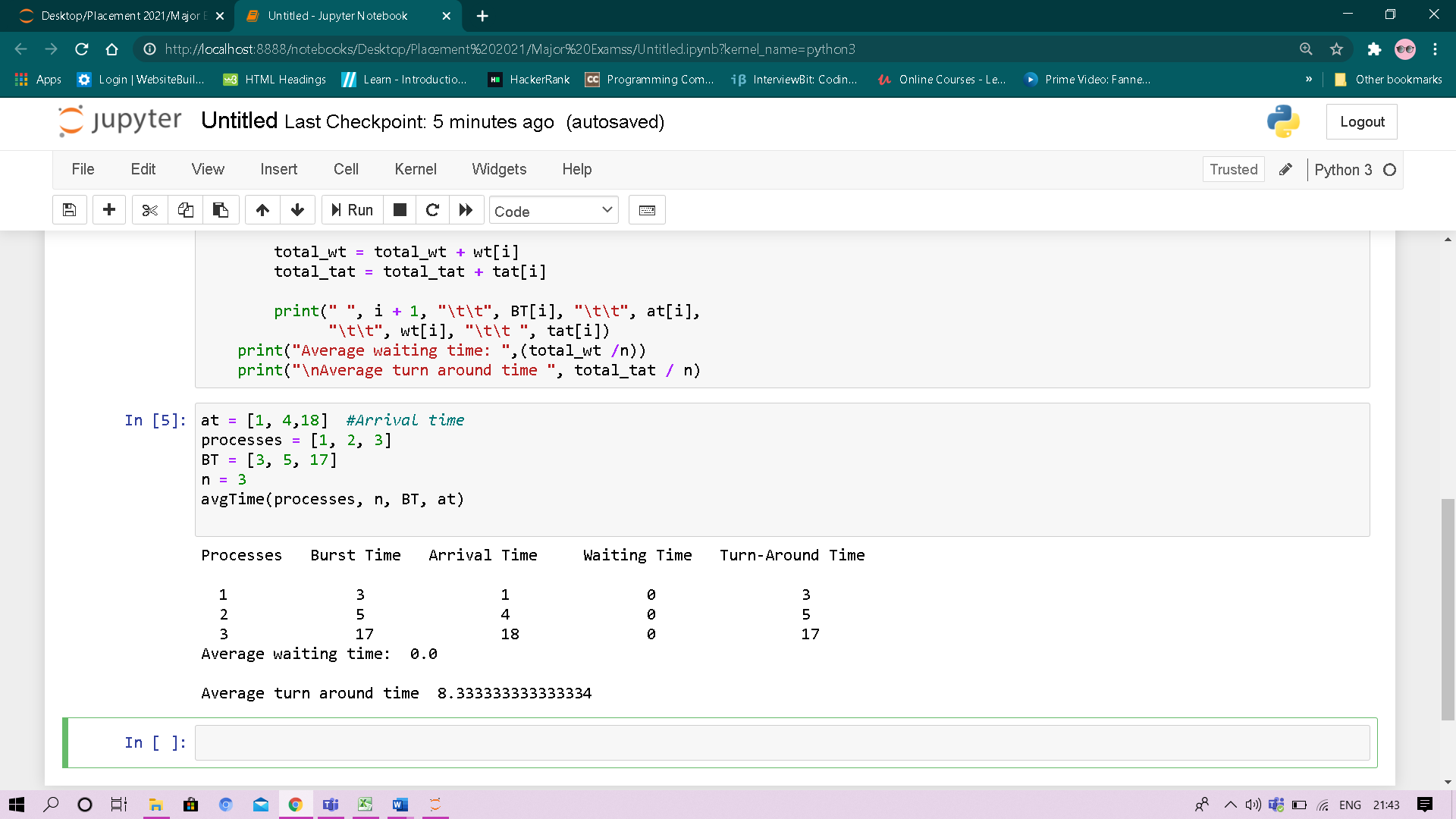
Output:



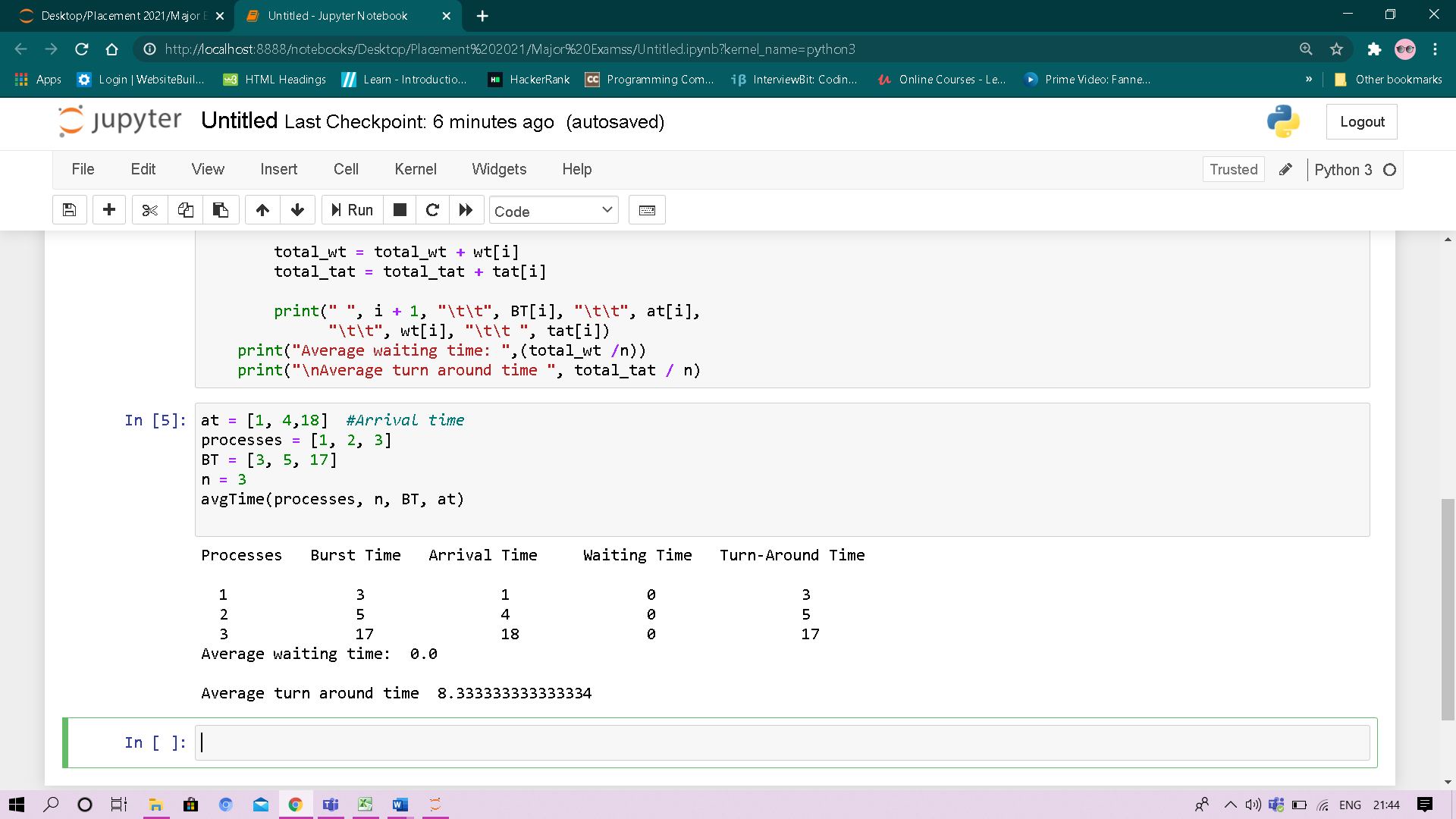
With Arrival Time:



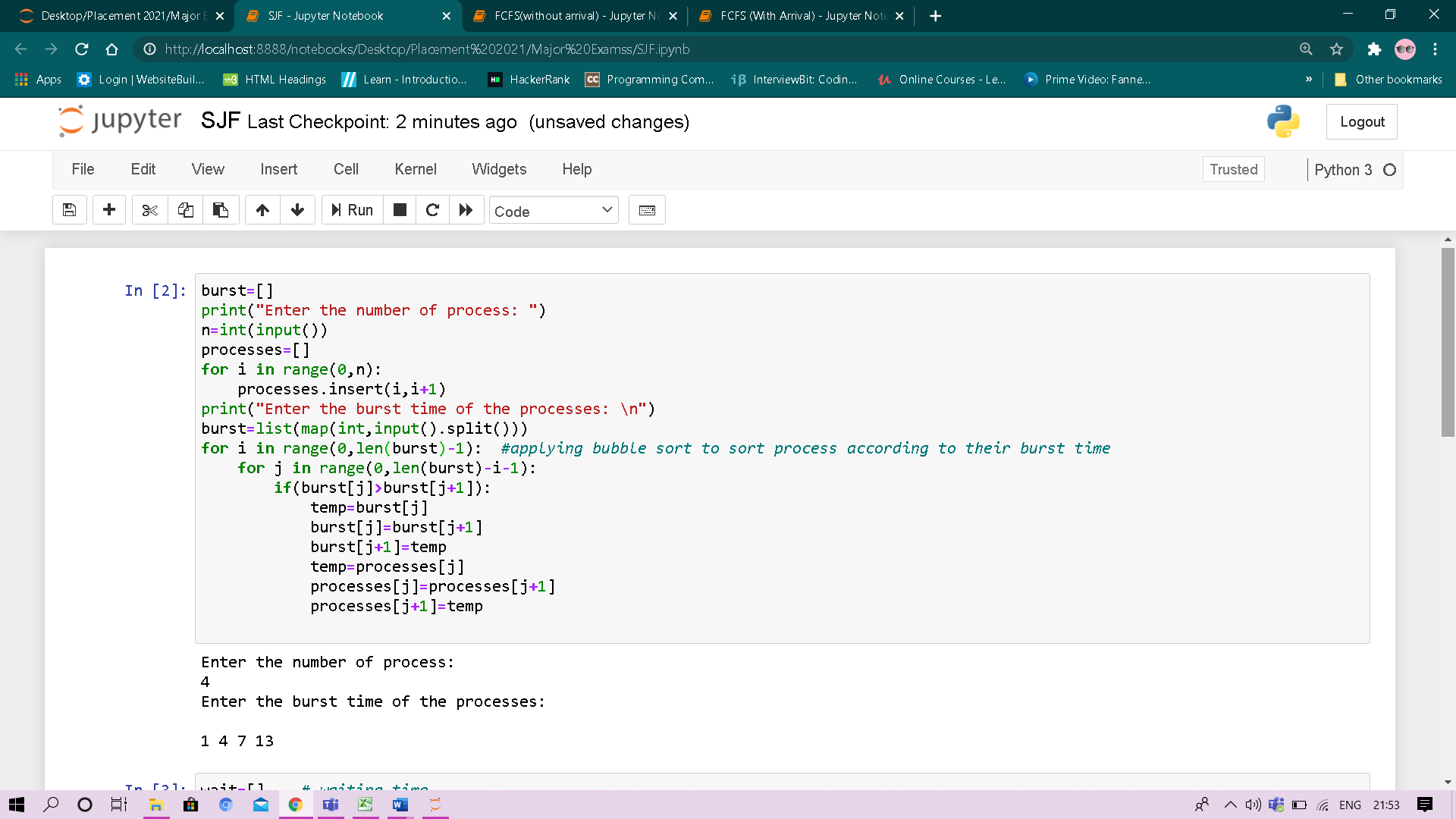


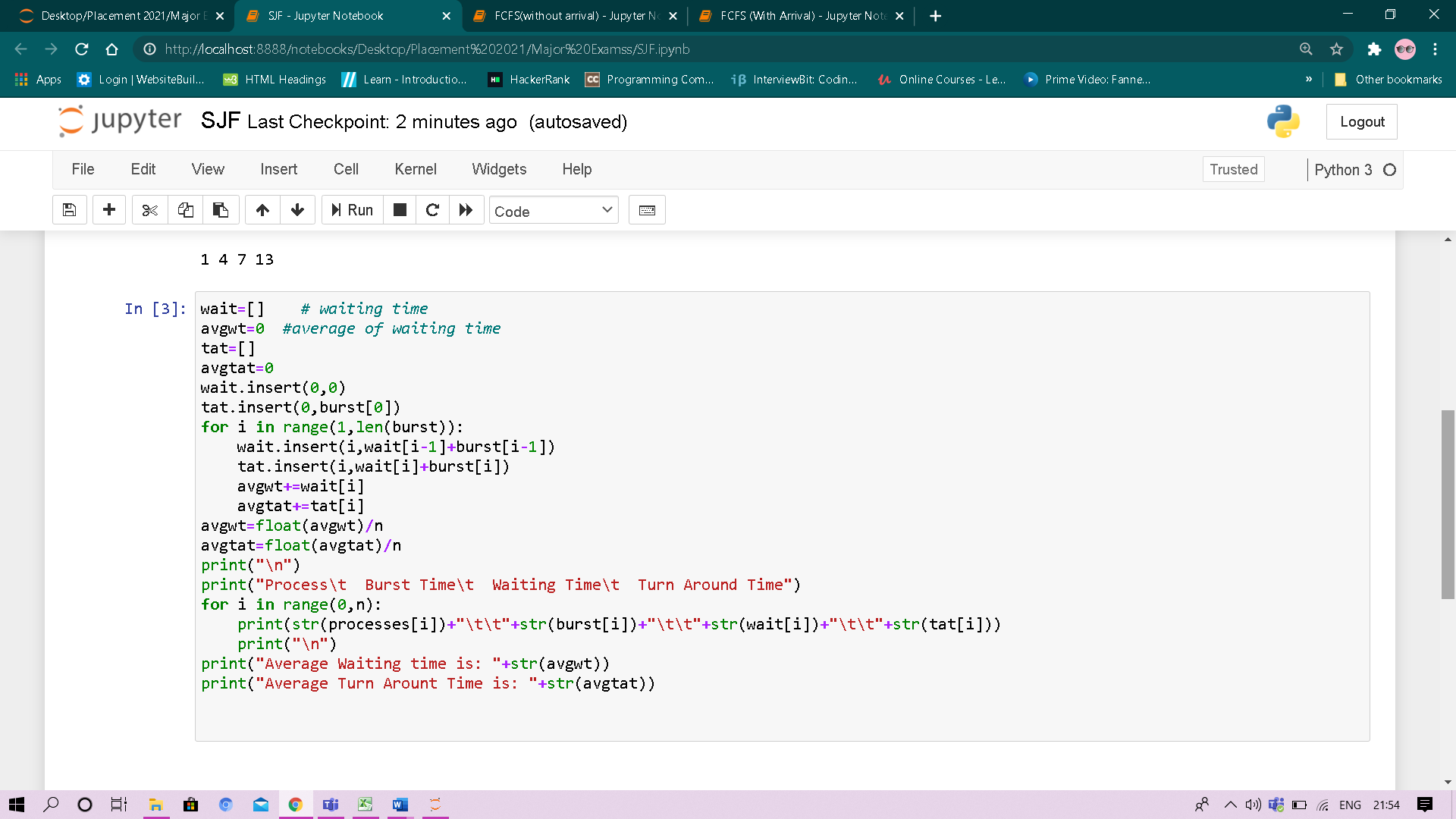


Output:

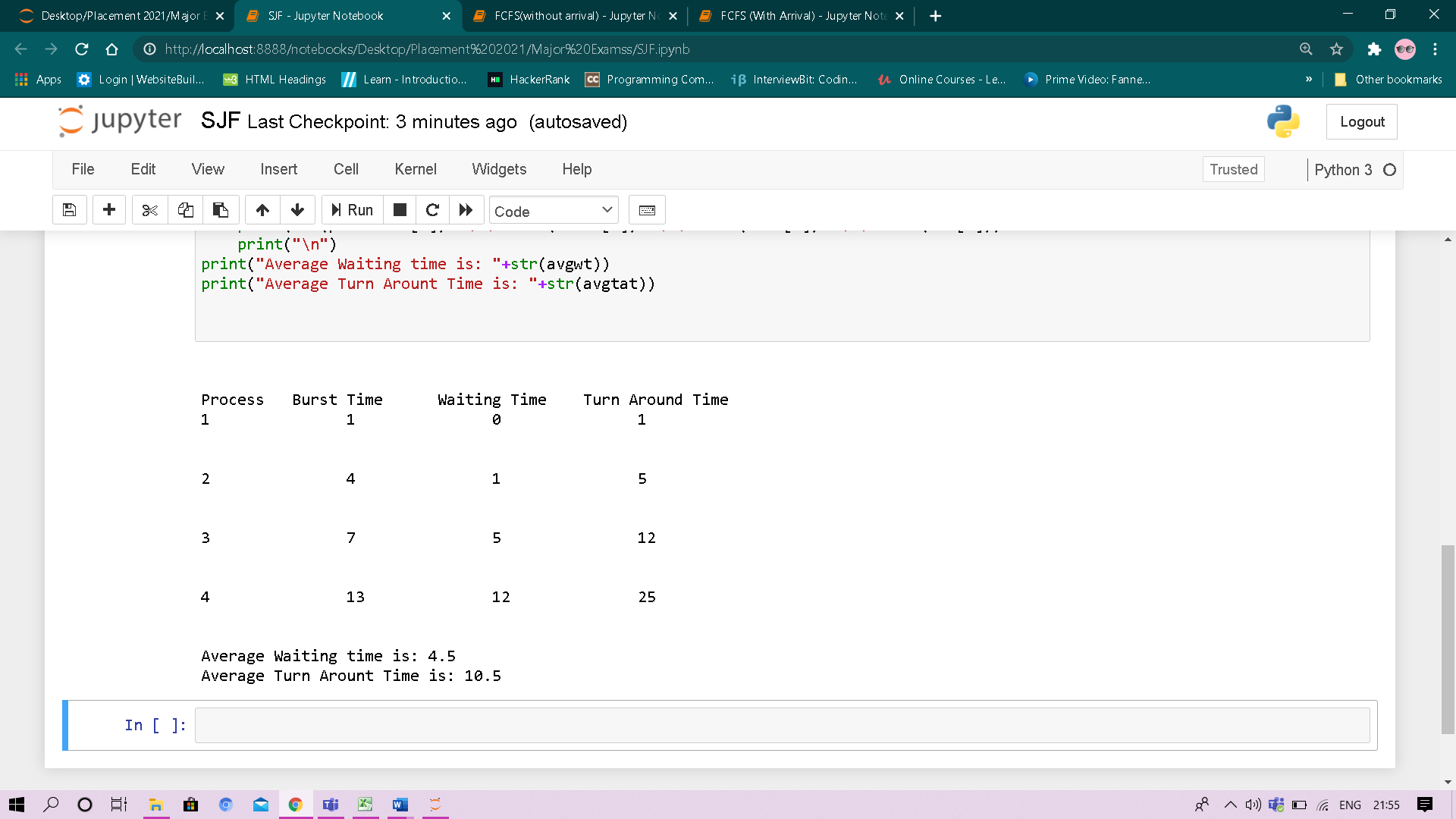


**SJF**





**Output:**



**Experiment No: #**

Student Name and Roll Number:Laxmikant 18csu117

Semester /Section:5th/B

Link to Code:

Date:21/11/2020

Faculty Signature:

Remarks:

**Objective**

To familiarize the students about CPU scheduling Algorithms.

**Program Outcome**

* The students will understand the round robin and priority based algorithm.

**Problem Statement**

Implement the following CPU scheduling Algorithms.

1. Round Robin Scheduling
2. Priority Scheduling

**Background Study:**

**Round Robin Scheduling:**

* It is simple, easy to implement, and starvation-free as all processes get fair share of CPU.
* One of the most commonly used technique in CPU scheduling as a core.
* It is preemptive as processes are assigned CPU only for a fixed slice of time at most.
* The disadvantage of it is more overhead of context switching.

**Priority Based Scheduling:**

* Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems.
* Each process is assigned a priority. Process with highest priority is to be executed first and so on.
* Processes with same priority are executed on first come first served basis.
* Priority can be decided based on memory requirements, time requirements or any other resource requirement.

**Algorithm/ flowchart:**

**Round Robin:**

1. Create an array **rem\_bt[]** to keep track of remaining burst time of processes. This array is initially a copy of bt[] (burst times array)
2. Create another array **wt[]** to store waiting times of processes. Initialize this array as 0.
3. Initialize time: t = 0
4. Keep traversing all processes while all processes are not done. Do following for i'th process if it is not done yet.
   1. If rem\_bt[i] > quantum
      1. t = t + quantum
      2. bt\_rem[i] -= quantum;
   2. Else // Last cycle for this process
      1. t = t + bt\_rem[i];
      2. wt[i] = t - bt[i]
      3. bt\_rem[i] = 0;

**flowchart:**

Diagram

Description automatically generated

**Priority Based Algorithm**

1. First input the processes with their burst time and priority.
2. Sort the processes, burst time and priority according to the priority.
3. Input the processes along with their burst time (bt).
4. Find waiting time (wt) for all processes.
5. As first process that comes need not to wait so waiting time for process 1 will be 0 i.e. wt[0] = 0.
6. Find **waiting time** for all other processes i.e. for
   1. process i ->
   2. wt[i] = bt[i-1] + wt[i-1] .
7. Find **turnaround time** = waiting\_time + burst\_time for all processes.
8. Find **average waiting time** = total\_waiting\_time / no\_of\_processes.
9. Similarly, find **average turnaround time** =total\_turn\_around\_time / no\_of\_processes.

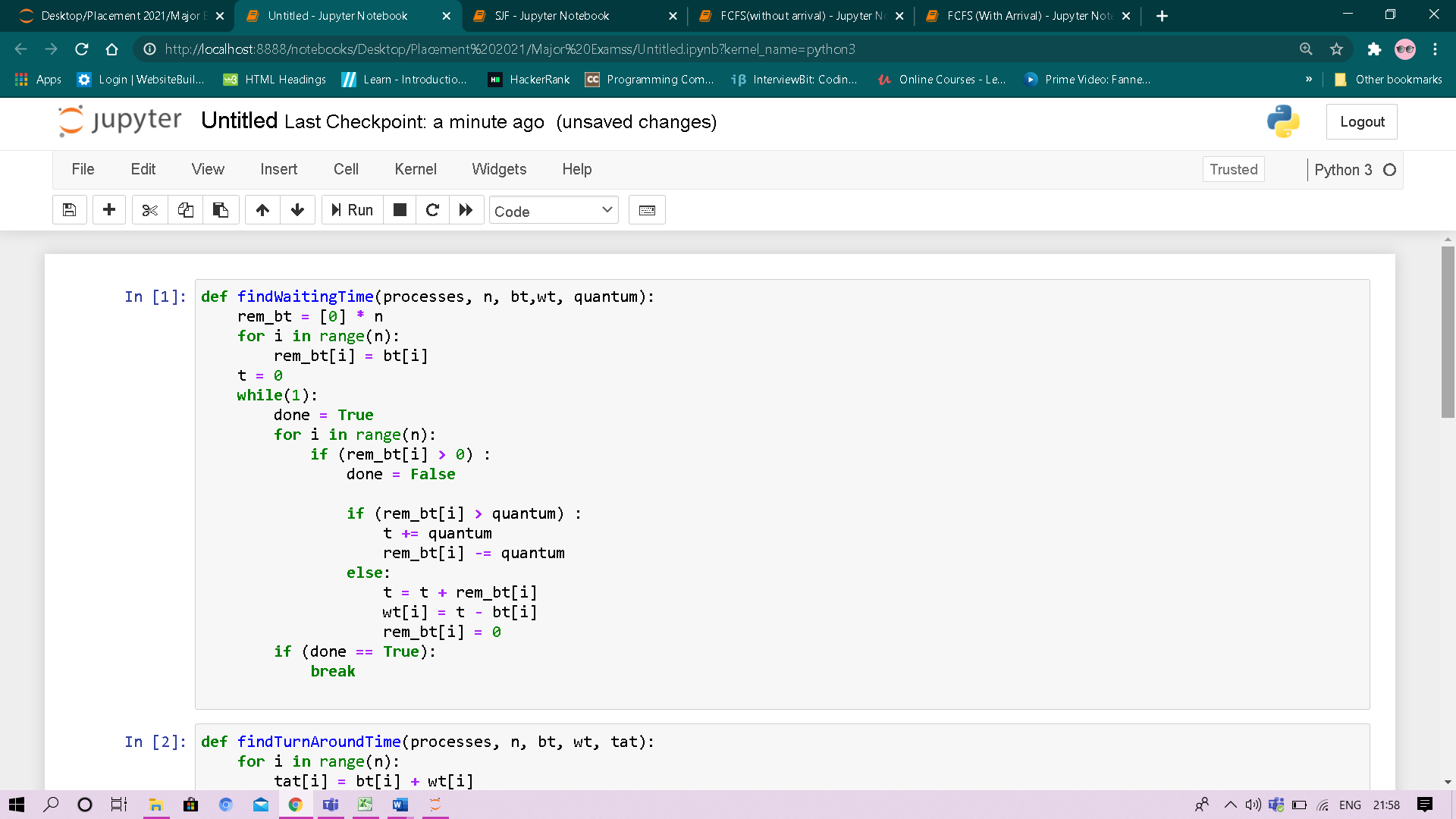
**flowchart:**

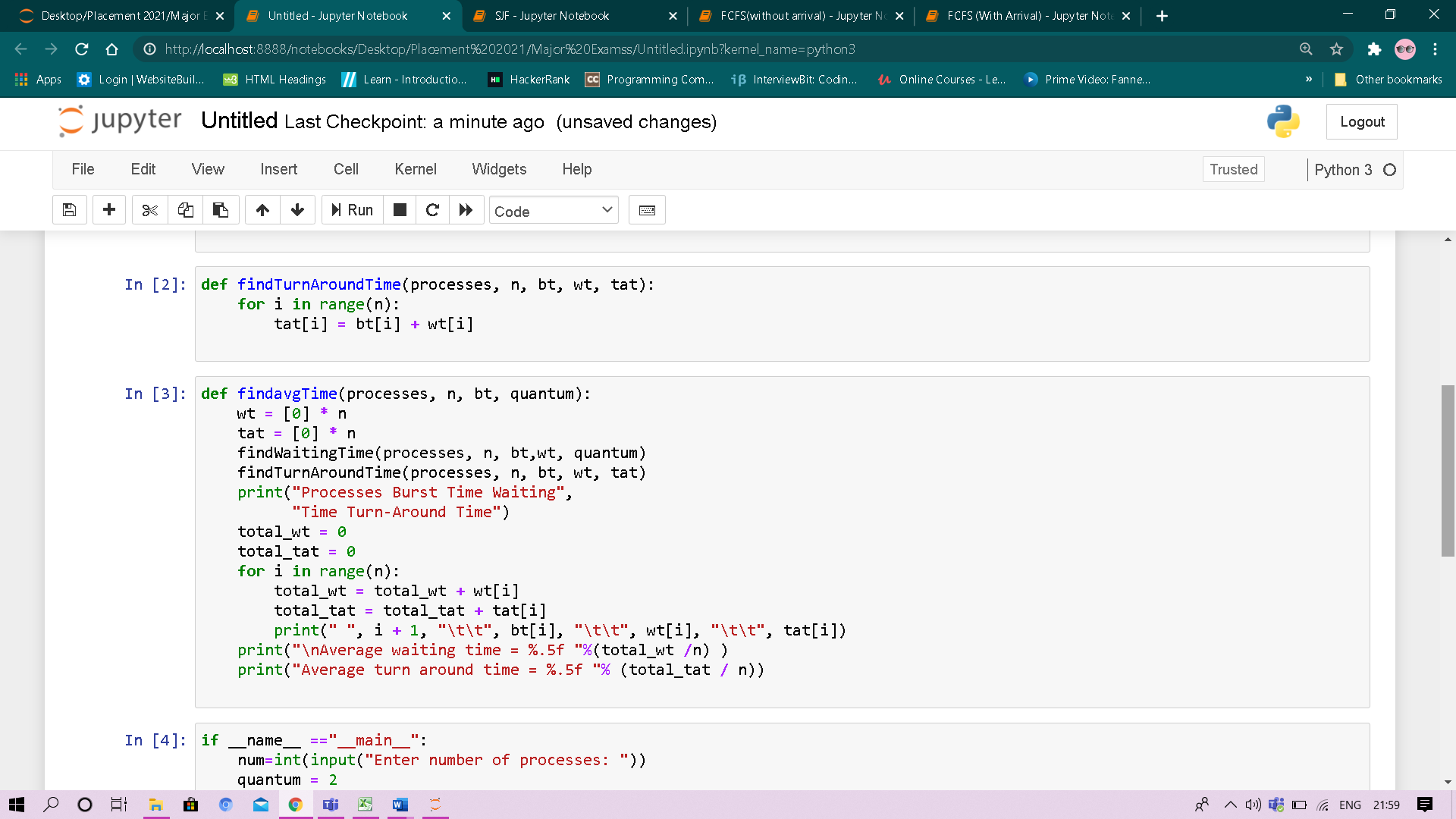
Chart

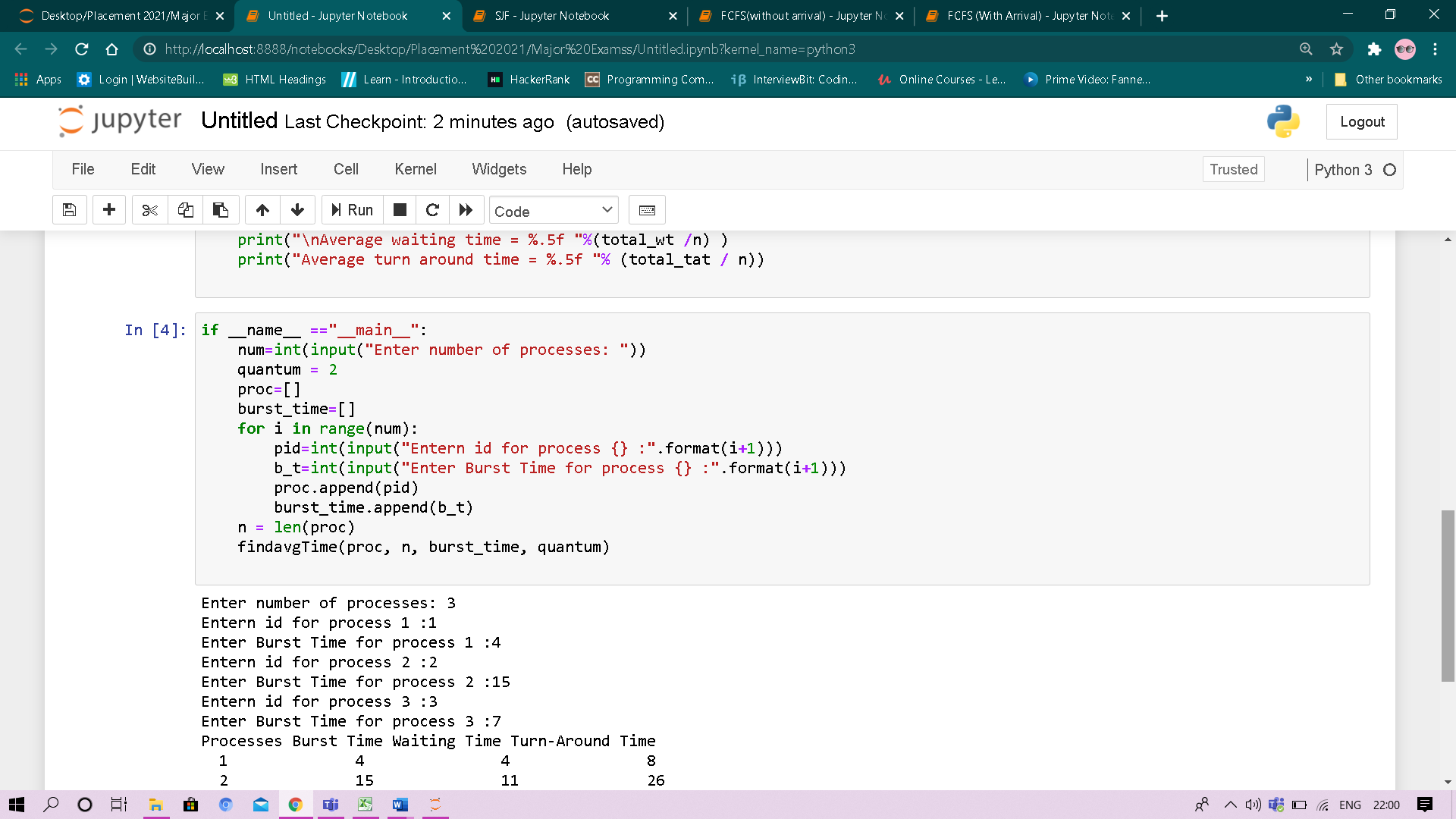
Description automatically generated

**Code**

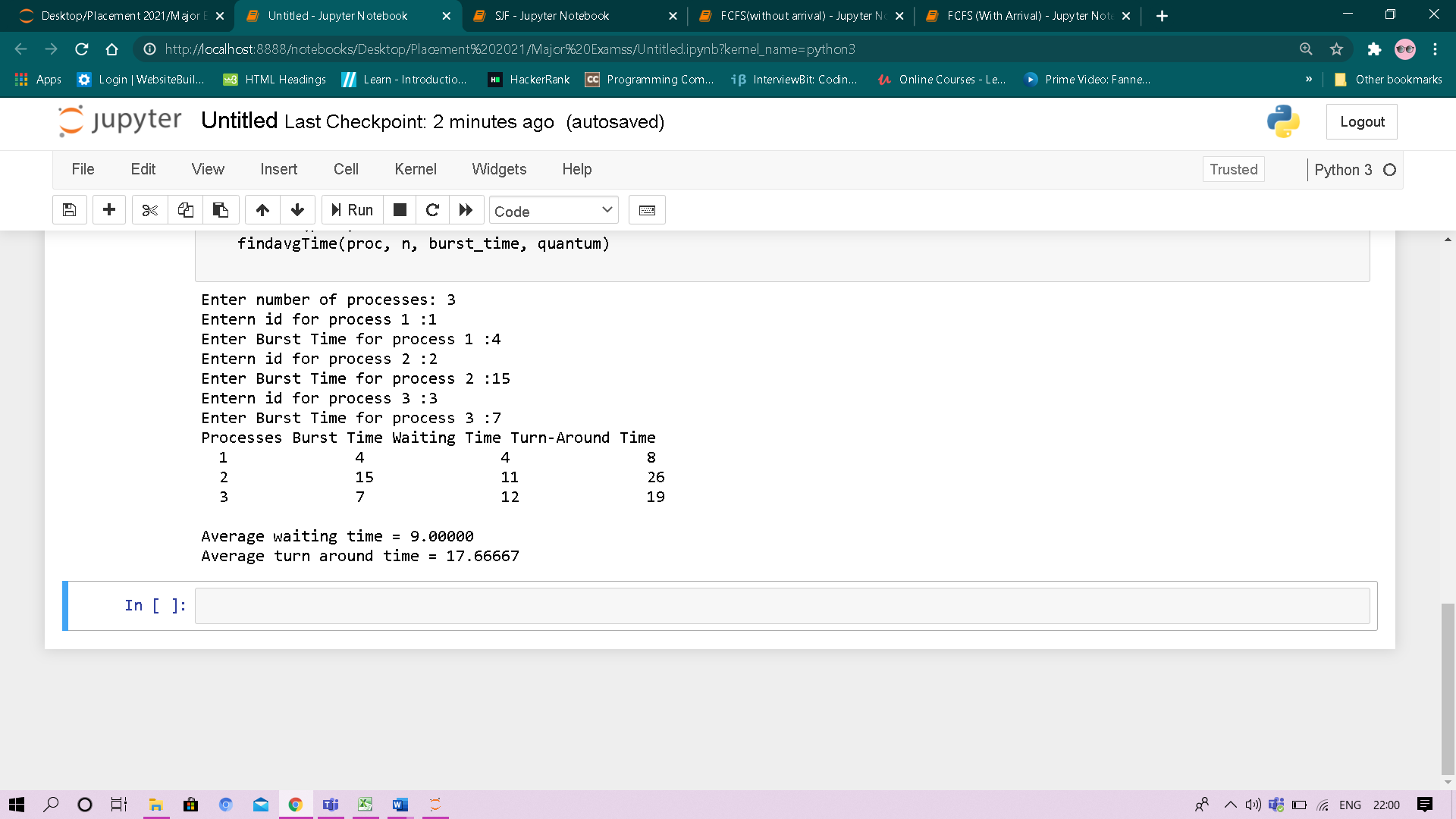
**Round Robin (Without Arrival Time):**





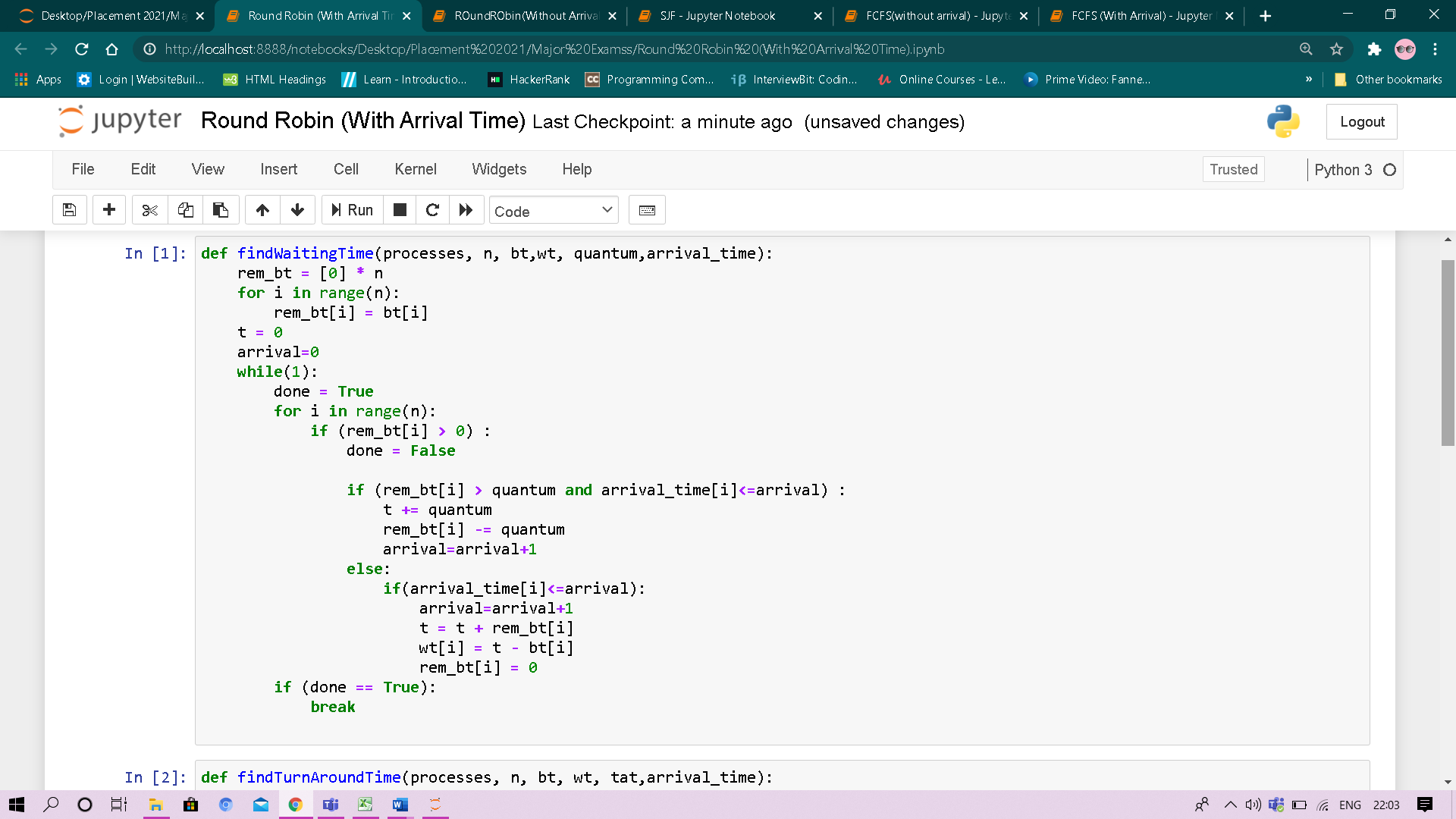


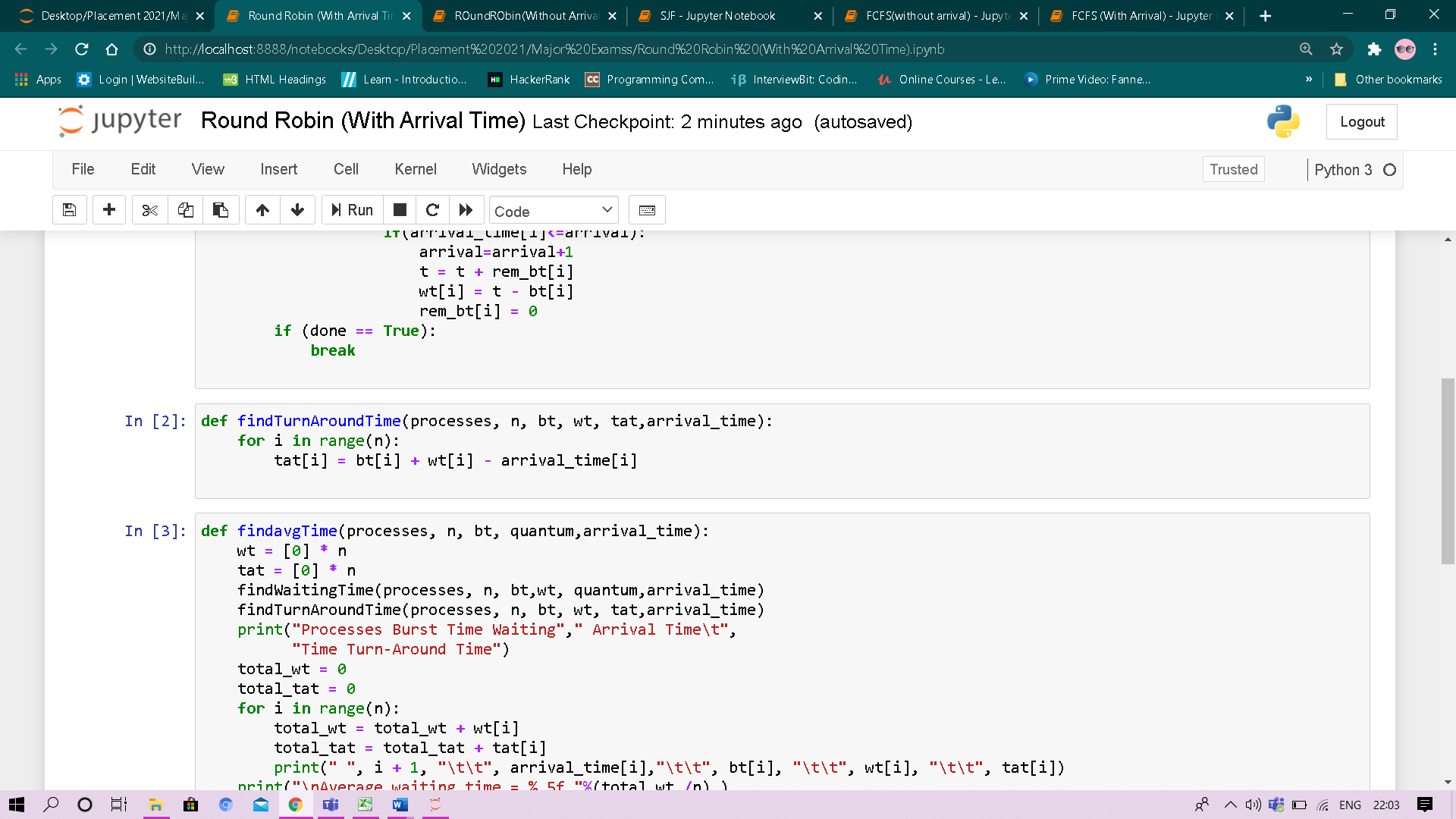
**Output:**

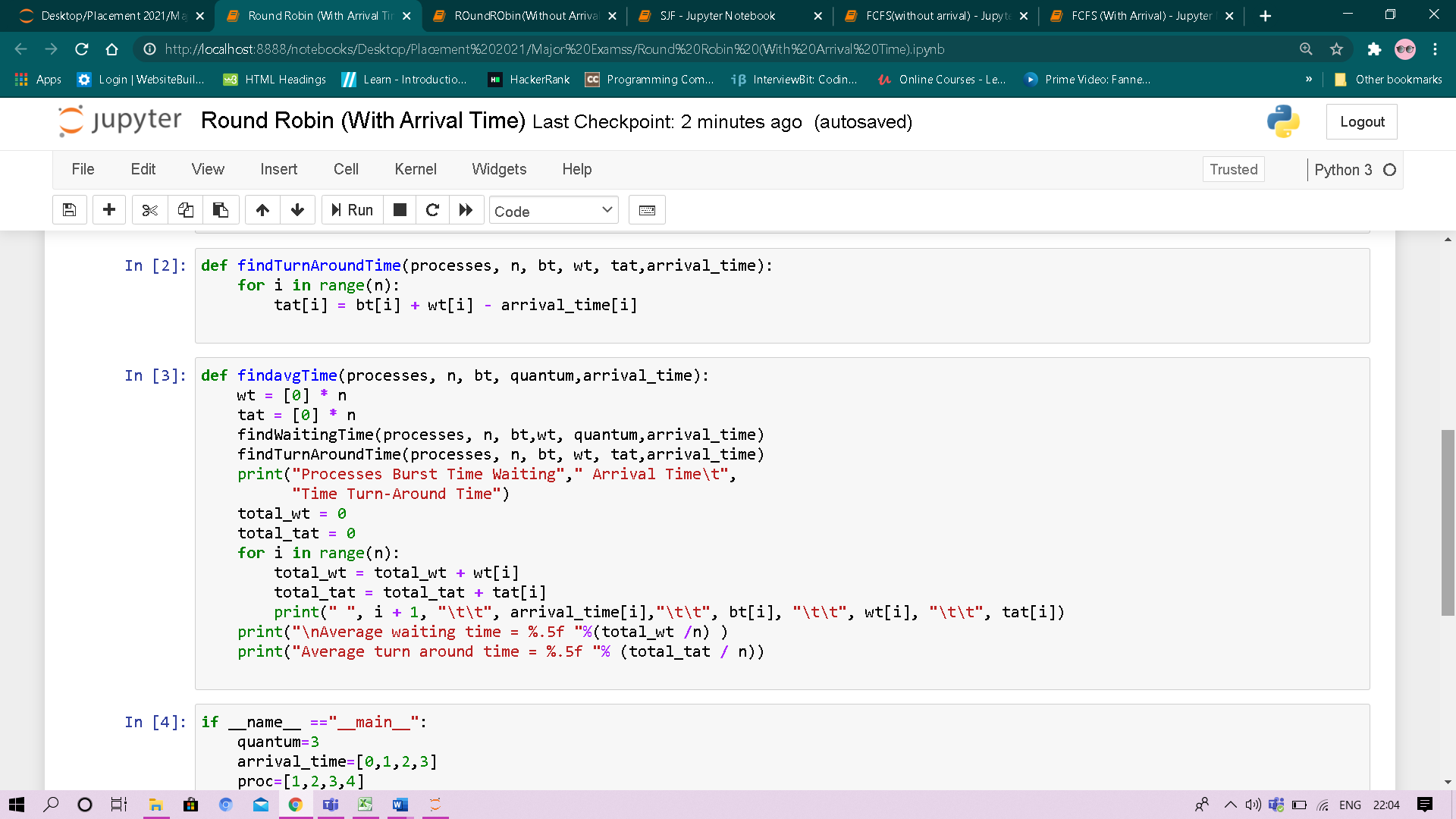


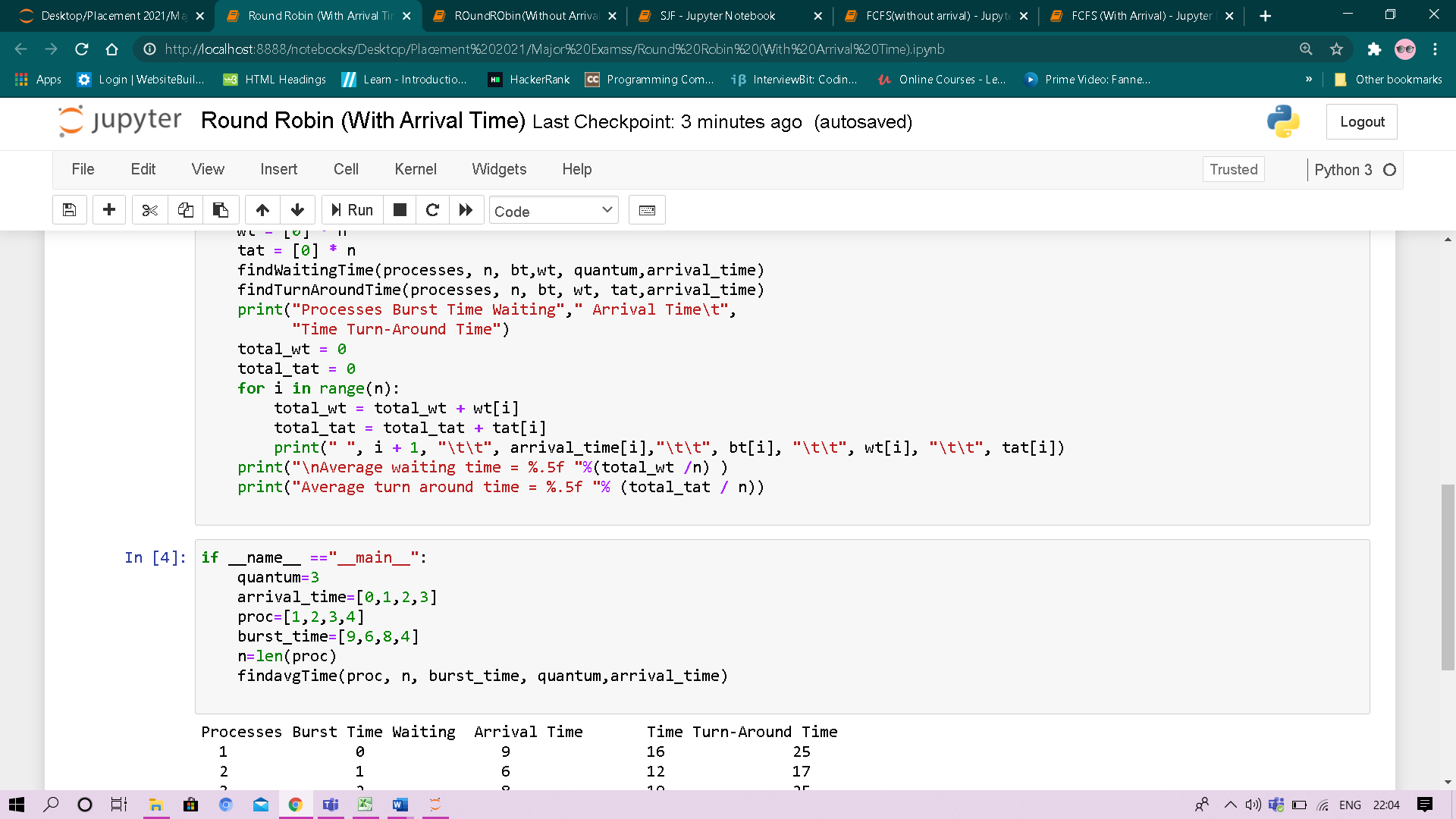
**Round Robin (With Arrival Time)**

**Code:**

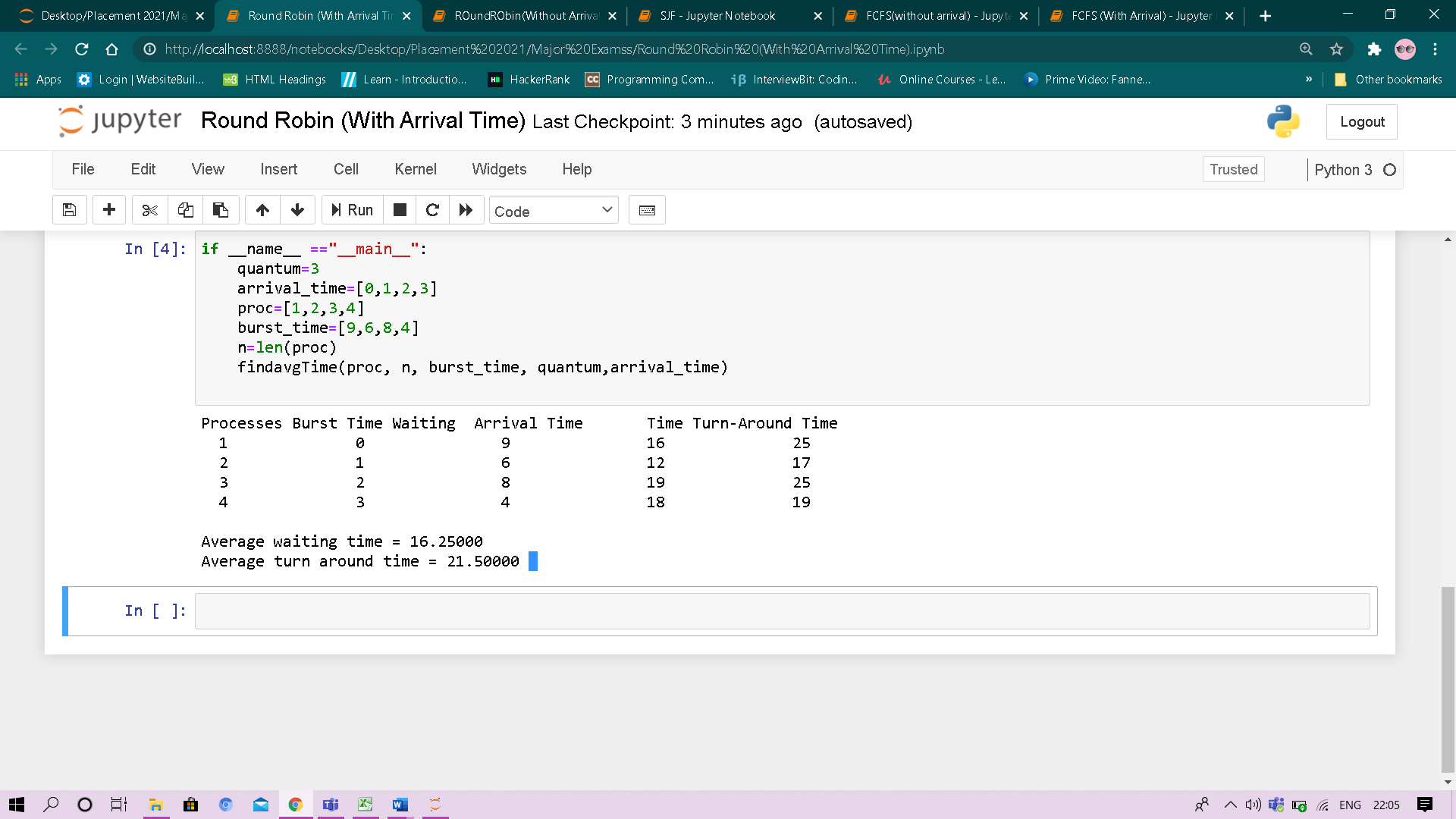




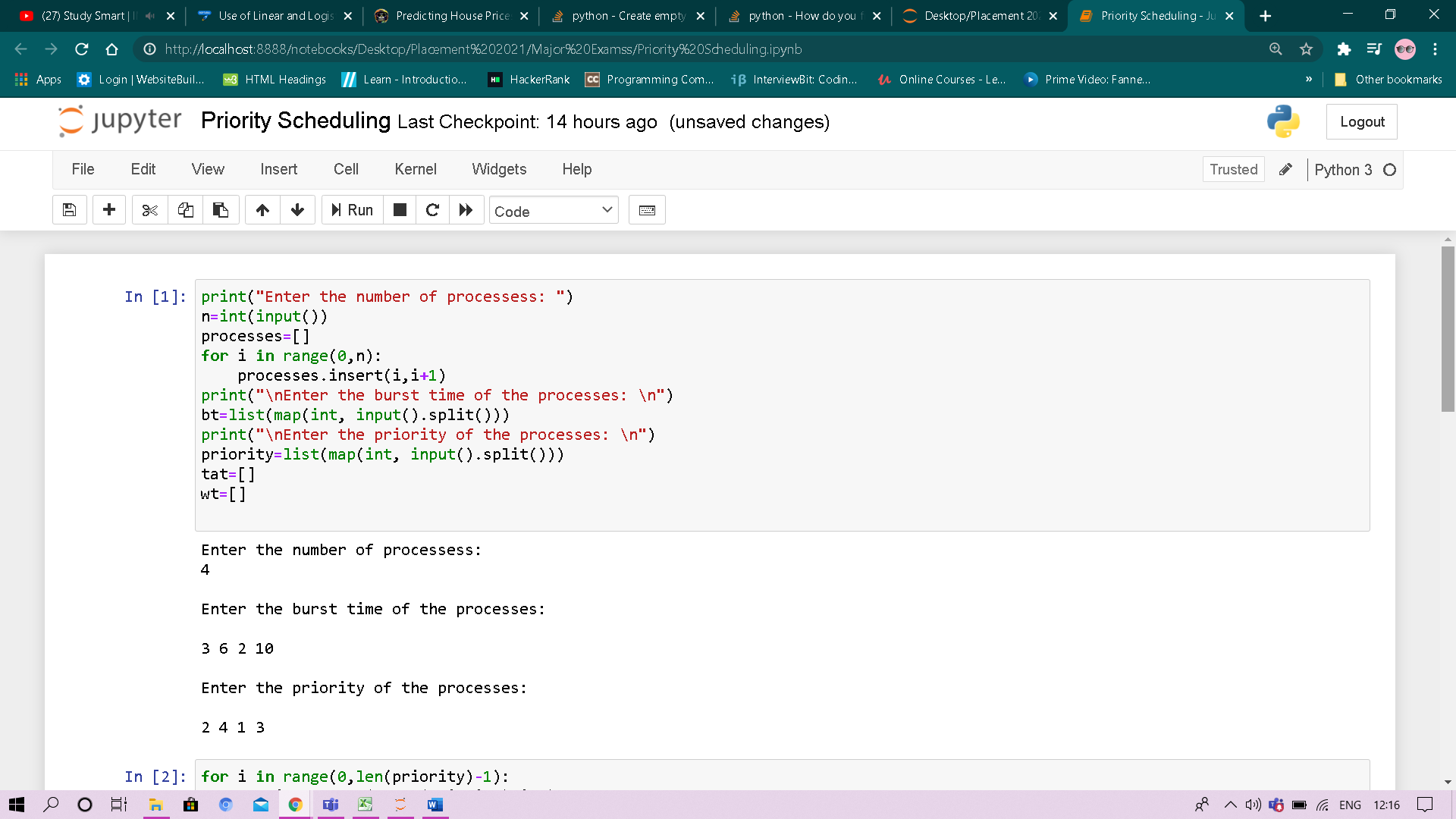


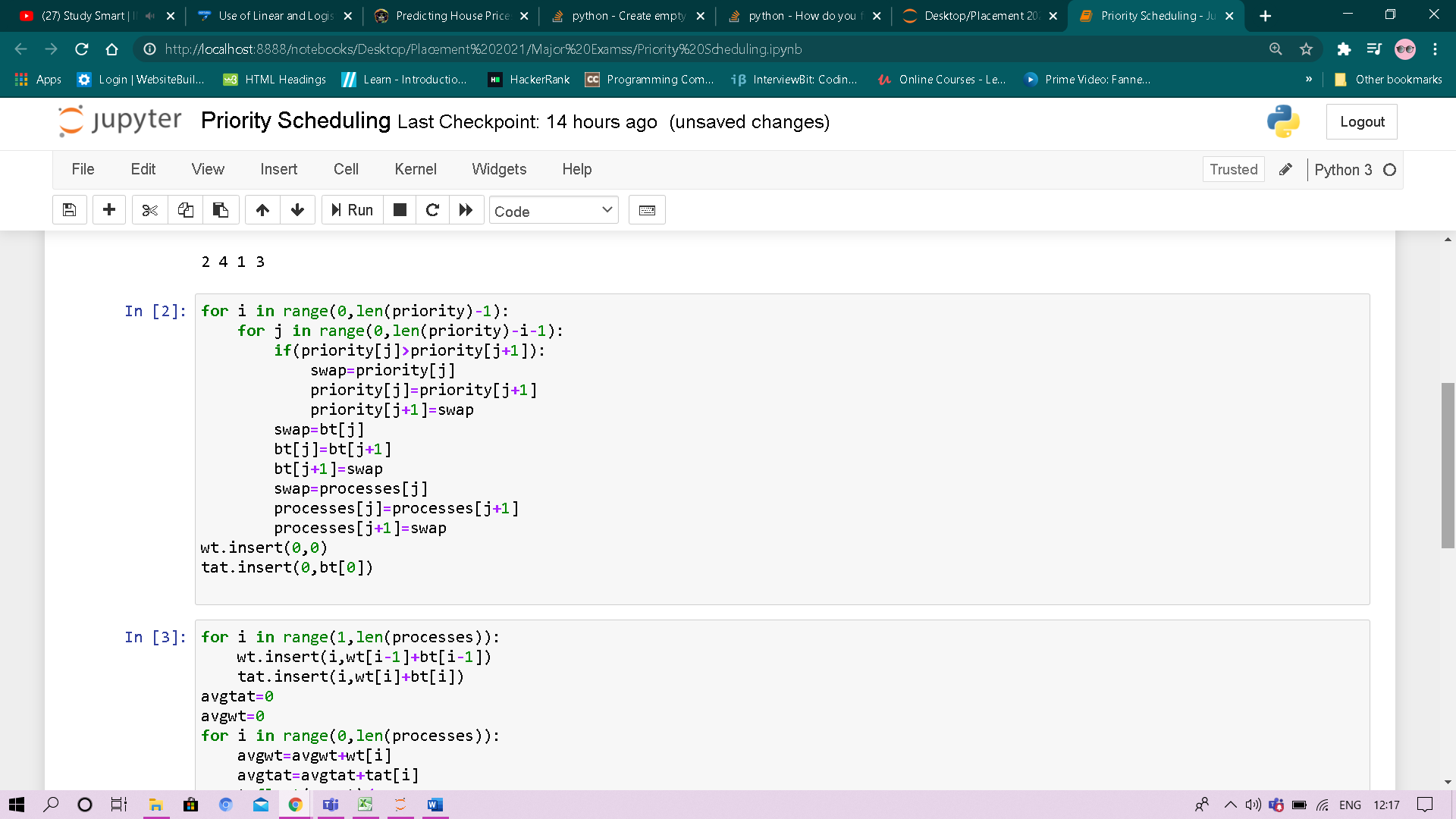


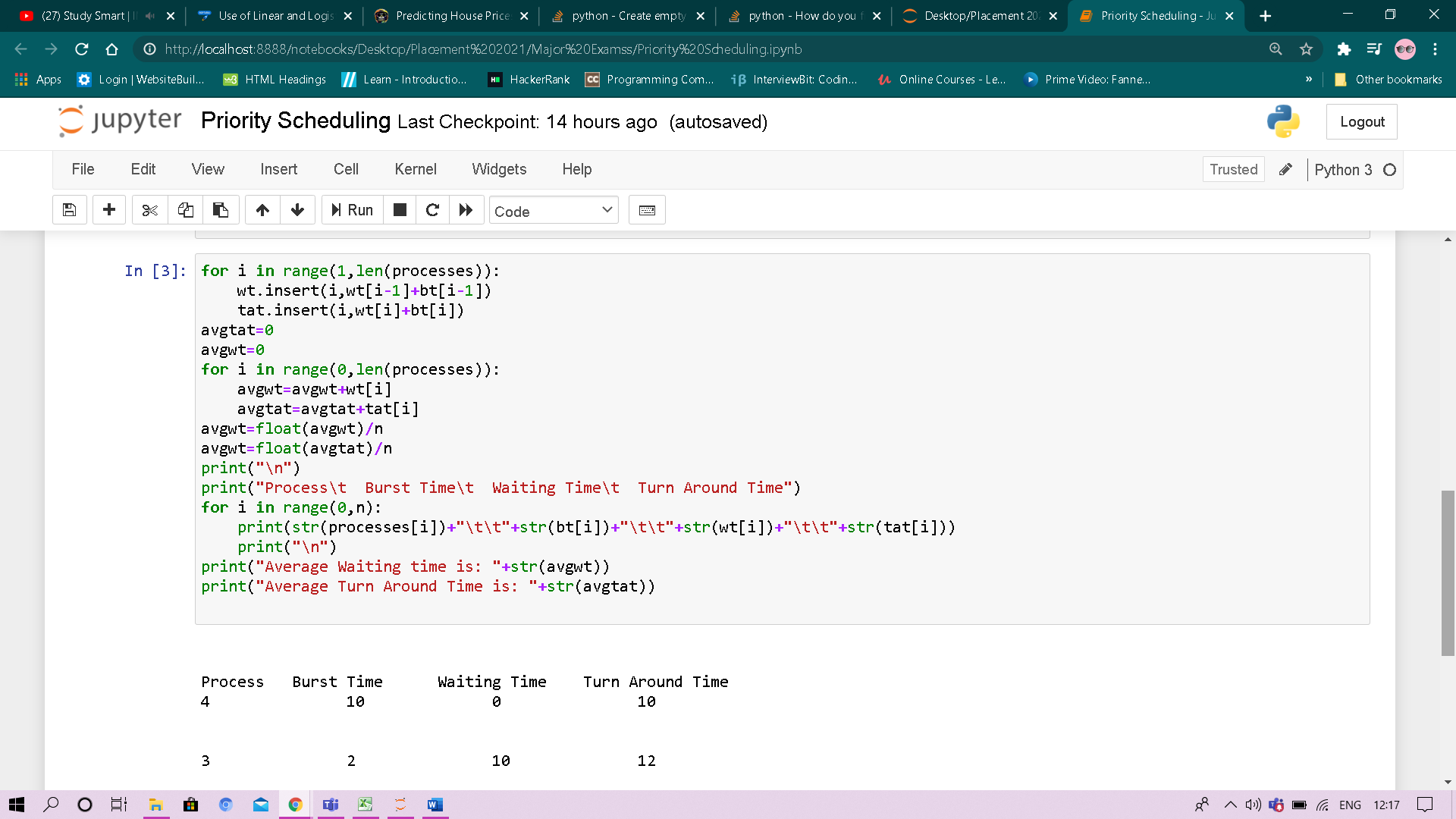
**Output:**



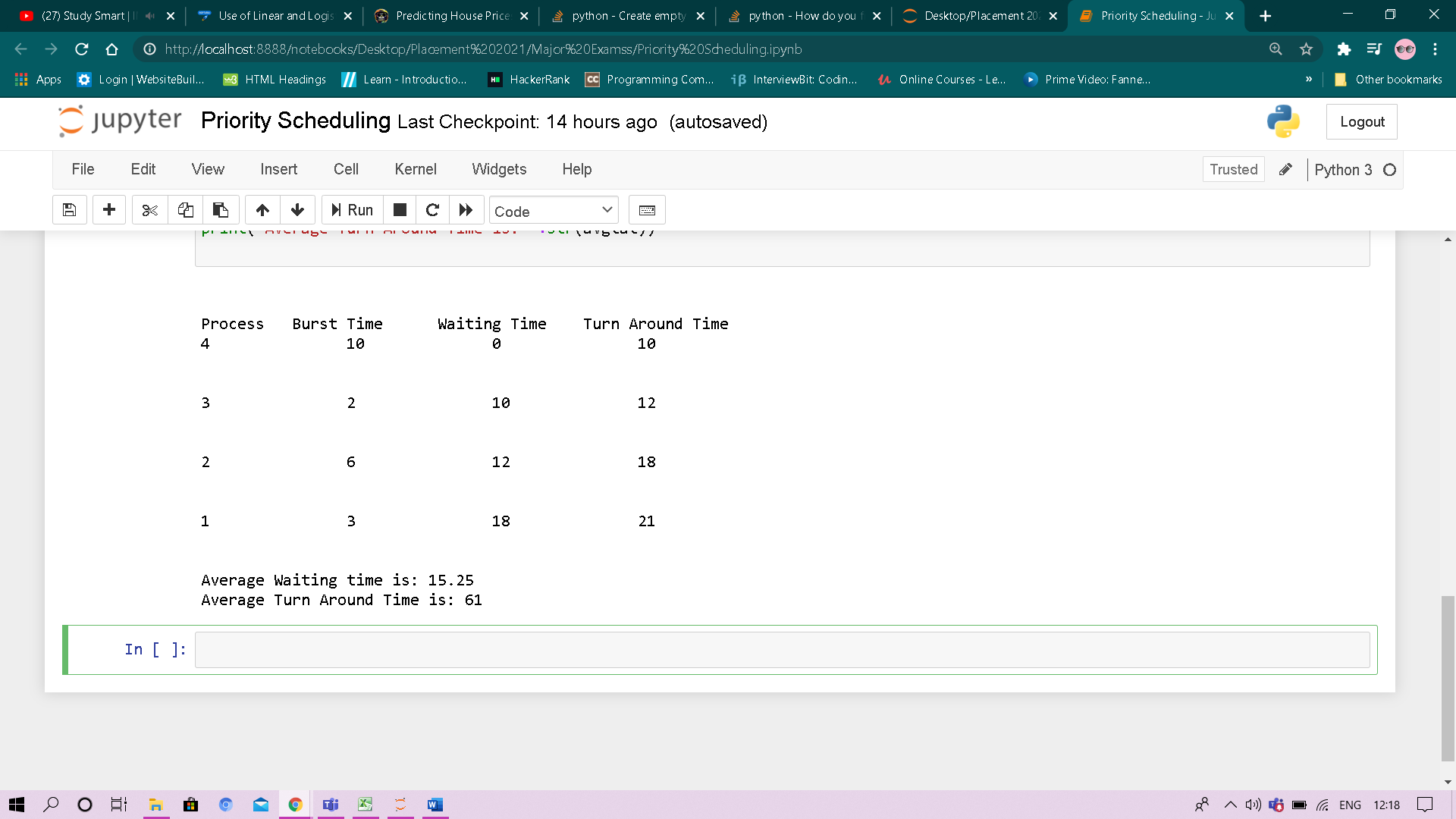
**Priority Scheduling**







**Output:**



**Experiment No: 8**

Student Name and Roll Number: Laxmikant and 18csu116

Semester /Section: Vth / B

Link to Code: CSL 303

Date: 27 october 2020

Faculty Signature:

Remarks:

**Objective**

To familiarize the students about Reader-Writer Problem.

**Program Outcome**

* The students will understand the Reader-Writer Problem.

**Problem Statement**

Implement the following Reader-Writer Problem.

i) Reader-Writer Problem.

**Background Study:**

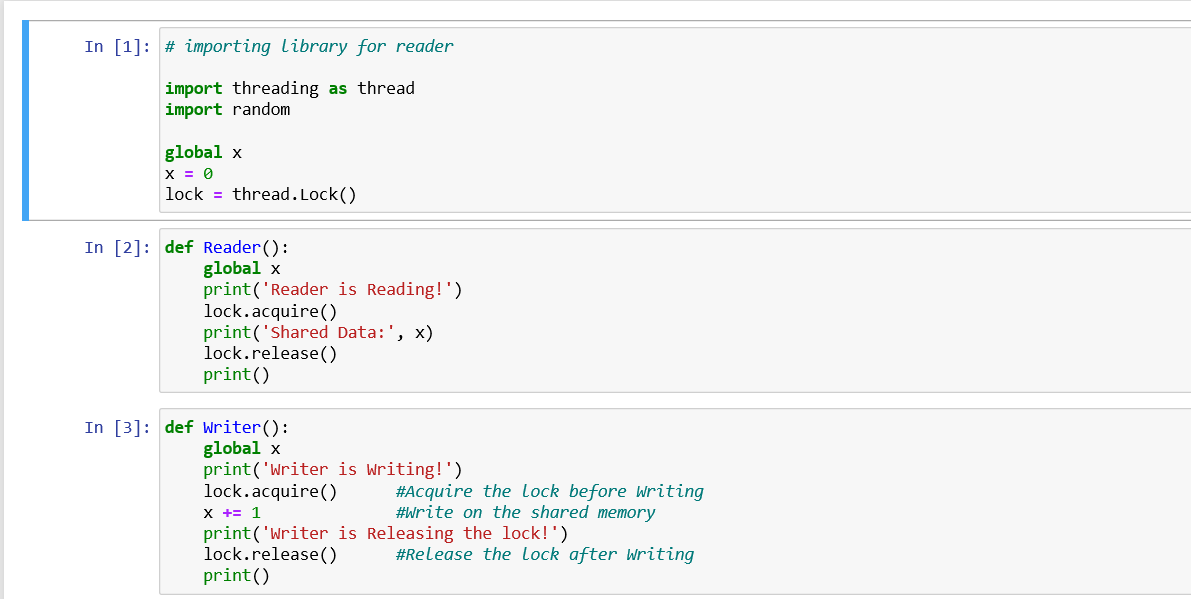
* The readers-writers problem relates to an object such as a file that is shared between multiple processes.
* To solve this situation, a writer should get exclusive access to an object i.e. when a writer is accessing the object, no reader or writer may access it.

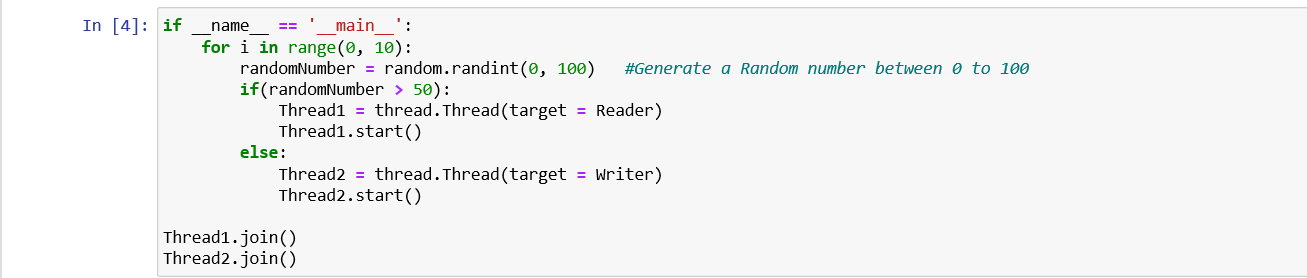
**Algorithm:**

1. Reader will run after Writer because of read semaphore.
2. Writer will stop writing when the write semaphore has reached 0.
3. Reader will stop reading when the read semaphore has reached 0.

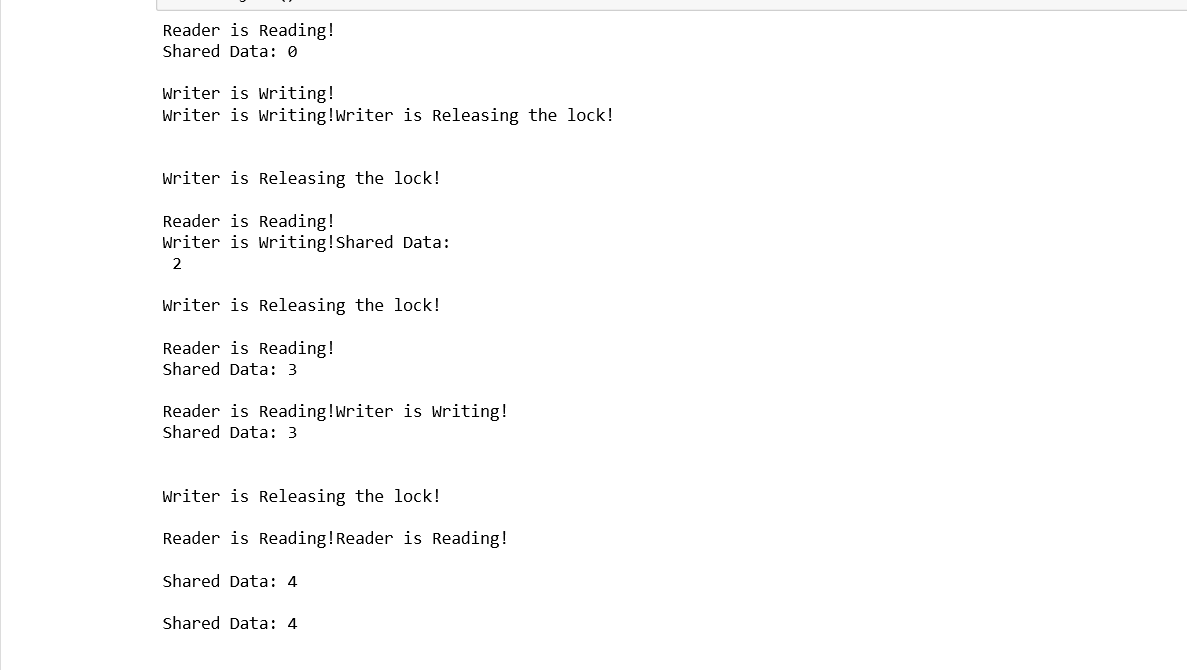
In writer, the value of write semaphore is given to read semaphore and in reader, the value of read is given to write on completion of the loop.

**Code: Screenshot:**

****

****

**Output: Screenshot**

****

**Experiment No: 9**

Student Name and Roll Number: Laxmikant and 18csu117

Semester /Section: Vth / B

Link to Code: CSL 303

Date: 10 november 2020

Faculty Signature:

Remarks:

**Objective**

To familiarize the students about Banker’s Algorithm.

**Program Outcome**

* The students will understand the Banker’s Algorithm Problem.

**Problem Statement**

Implement the following Banker’s Algorithm Problem.

i) Banker’s Algorithm .

**Background Study:**

* The banker’s algorithm is a resource allocation and deadlock avoidance algorithm that tests for safety by simulating the allocation for predetermined maximum possible amounts of all resources, then makes an “s-state” check to test for possible activities, before deciding whether allocation should be allowed to continue.
* Banker's Algorithm is used majorly in the banking system to avoid deadlock. It helps you to identify whether a loan will be given or not. This algorithm is used to test for safely simulating the allocation for determining the maximum amount available for all resources.

**Drawbacks**:  
  
It requires the number of processes to be fixed; no additional processes can start while it is executing. It requires that the number of resources remain fixed; no resource may go down for any reason without the possibility of deadlock occurring.

**Algorithm:**

**Safety Algorithm:**

1) Let Work and Finish be vectors of length ‘m’ and ‘n’ respectively.  
Initialize: Work = Available  
Finish[i] = false; for i=1, 2, 3, 4….n

2) Find an i such that both  
a) Finish[i] = false  
b) Needi <= Work  
if no such i exists goto step (4)

3) Work = Work + Allocation[i]  
Finish[i] = true  
goto step (2)

4) if Finish [i] = true for all i  
then the system is in a safe state

**Resource-Request Algorithm:**

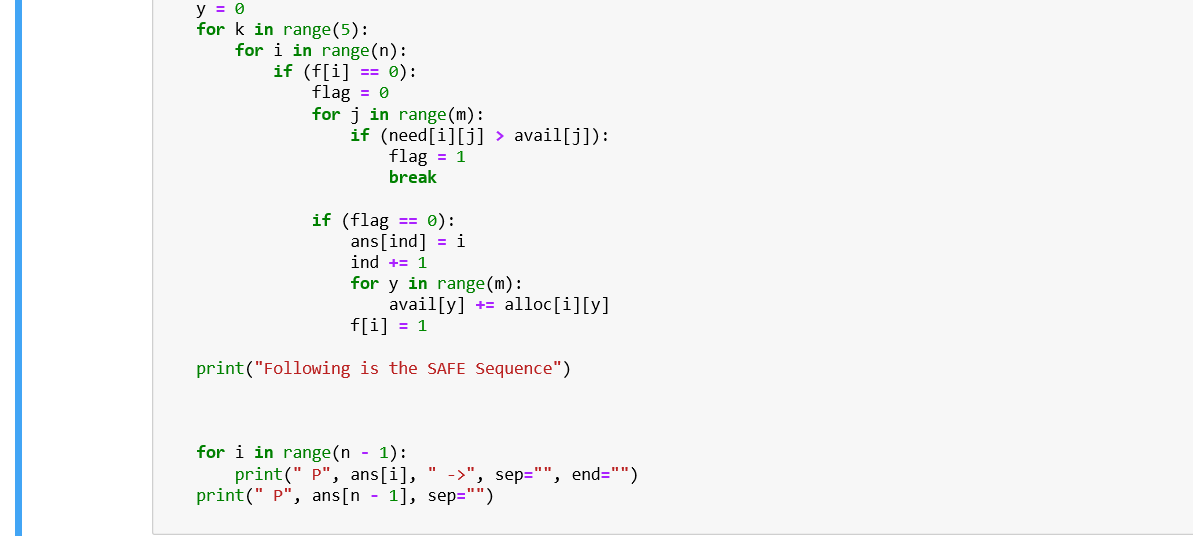
1) If Requesti <= Needi  
Goto step (2) ; otherwise, raise an error condition, since the process has exceeded its maximum claim.

2) If Requesti <= Available  
Goto step (3); otherwise, Pi must wait, since the resources are not available.

3) Have the system pretend to have allocated the requested resources to process Pi by modifying the state as  
follows:  
Available = Available – Requesti  
Allocationi = Allocationi + Requesti  
Needi = Needi– Requesti

**Code: Screenshot**

****

****

**Output: Screenshot**

****

**Suggested Question Bank**

**Preparatory Questions**

1. Which module gives control of the CPU to the process selected by the short-term scheduler?
   1. dispatcher
   2. interrupt
   3. scheduler
   4. none of the mentioned
2. The processes that are residing in main memory and are ready and waiting to execute are kept on a list called
   1. job queue
   2. ready queue
   3. execution queue
   4. process queue
3. The interval from the time of submission of a process to the time of completion is termed as
   1. waiting time
   2. turnaround time
   3. response time
   4. throughput
4. Which scheduling algorithm allocates the CPU first to the process that requests the CPU first?
   1. first-come, first-served scheduling
   2. shortest job scheduling
   3. priority scheduling
   4. none of the mentioned
5. In priority scheduling algorithm
   1. CPU is allocated to the process with highest priority
   2. CPU is allocated to the process with lowest priority
   3. equal priority processes can not be scheduled
   4. none of the mentioned
6. Process are classified into different groups in
   1. shortest job scheduling algorithm
   2. round robin scheduling algorithm
   3. priority scheduling algorithm
   4. multilevel queue scheduling algorithm
7. Which one of the following can not be scheduled by the kernel?
   1. kernel level thread
   2. user level thread
   3. process
   4. none of the mentioned
8. CPU scheduling is the basis of \_\_\_\_\_\_\_\_\_\_\_\_.
   1. multiprocessor systems
   2. multiprogramming operating systems
   3. larger memory sized systems
   4. None of these
9. With multiprogramming, \_\_\_\_\_\_ is used productively.

a) time b) space c) money d) All of these

1. The two steps of a process execution are : (choose two)

a) I/O Burst b) CPU Burst c) Memory Burst d) OS Burst

1. An I/O bound program will typically have :
   1. a few very short CPU bursts
   2. many very short I/O bursts
   3. many very short CPU bursts
   4. a few very short I/O bursts
2. A process is selected from the \_\_\_\_\_\_ queue by the \_\_\_\_\_\_\_\_ scheduler, to be executed.

a) blocked, short term b) wait, long term c) ready, short term d) ready, long term

1. In the following cases non – preemptive scheduling occurs : (Choose two)
   1. When a process switches from the running state to the ready state
   2. When a process goes from the running state to the waiting state
   3. When a process switches from the waiting state to the ready state
   4. When a process terminates
2. Dispatch latency is :
   1. the speed of dispatching a process from running to the ready state
   2. the time of dispatching a process from running to ready state and keeping the CPU idle
   3. the time to stop one process and start running another one
   4. None of these
3. Scheduling is done so as to :
   1. increase the throughput
   2. decrease the throughput
   3. increase the duration of a specific amount of work
   4. None of these
4. Turnaround time is :
   1. the total waiting time for a process to finish execution
   2. the total time spent in the ready queue
   3. the total time spent in the running queue
   4. the total time from the completion till the submission of a process
5. Scheduling is done so as to :
   1. increase the turnaround time
   2. decrease the turnaround time
   3. keep the turnaround time same
   4. there is no relation between scheduling and turnaround time

**Case Study of Dos , Windows and linux**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Feature** | **Windows** | **Linux** | **DOS** | **Linux** |
| **Ability to** | You can change the | You can rewrite the | You can download | Overall Linux is the most |
| **Customize** | background and general | code for Linux so | expansions for DOS | customisable, because it is |
|  | colour scheme and | that you can | so that you can | open source and you can |
|  | fonts for your windows | customise it further, | customise it. | alter the codes for it. DOS |
|  | desktop. | meaning that you |  | can also be open source, |
|  |  | can have a |  | but there is no GUI to |
|  |  | completely different |  | customise. Windows is |
|  |  | layout than everyone |  | relatively customisable, but |
|  |  | else. |  | not fully. You can mainly |
|  |  |  |  | just change the colours and |
|  |  |  |  | looks of things slightly. |
| **Support for** | Windows supports | Linux doesn’t have | DOS doesn’t | With Windows being the |
| **connectivity for** | connectivity for | as wide a range of | provide much | most popular OS out of  the |
| **portable media** | different media, | support for | connectivity, from | 3, it has more  connectivity |
|  | including USB drives, | connectivity but you | the fixed HDD only. | available, and expansions |
|  | CD/DVD, Syncing, | can download a |  | available to enable more |
|  | Bluetooth and Wi-Fi. | Software |  | connectivity. You can also |
|  |  | development kit to |  | download expansions to |
|  |  | improve its |  | make Linux more |
|  |  | connectivity. |  | supportive for connectivity, |
|  |  |  |  | but there are not as many |
|  |  |  |  | things available as for |
|  |  |  |  | windows. As DOS is an old |
|  |  |  |  | OS and isn’t really used |
|  |  |  |  | anymore, it doesn’t really |
|  |  |  |  | have any support for |
|  |  |  |  | connecting media. Newer |
|  |  |  |  | versions of DOS might, |
|  |  |  |  | because they are built with |
|  |  |  |  | Windows on top of them. |
| **Security** | Windows has an | For Linux you have | DOS does not | The most secure would be |
|  | administrative user that | to enter the | support multiple | Linux, because of its ability |
|  | can make changes to | administrator’s | users so there are | to customise, viruses |
|  | the computer, but it has | password in order to | no threats from | wouldn’t be able to access |
|  | other users that can | make changes or | other user on the | the computer because it |
|  | perform small tasks. | download things, so | computer. DOS | wouldn’t be specific |
|  | You need the | it is harder for | didn’t really have | enough. Because of |
|  | administrator’s | harmful programs to | much security | Windows’ popularity, it |
|  | password to change | gain access to your | when it first came | makes it the most |
|  | anything if on another | computer. You can | out, because of the | vulnerable to attacks. This |
|  | user. | also change the | low threat. | is also because you can’t |
|  |  | codes to make it |  | edit the code. DOS has the |
|  |  | more secure. |  | least security because it | |
|  |  |  |  | was made before viruses | |
|  |  |  |  | existed. | |
| **Stability** | Windows is relatively | Linux operating | DOS doesn’t have a | Overall I think that Linux is | |
|  | stable, but vista is more | systems are stable, | user interface, so it | the most stable, because it | |
|  | unstable than other | because you can | is much more | doesn’t have as much | |
|  | versions. Vista didn’t | rewrite the codes for | stable because it | software and other things | |
|  | sell because it crashed when it was launched, and Microsoft released windows 7 soon after it. | it. | doesn’t have to produce graphics. | running as Windows, and it  is more popular than DOS, so it is better tested and improved.  Windows crashes a lot and needs rebooting a lot because of the  amount of programs running on it. | |
| **Reliability** | Windows is not that reliable, with it crashing/needing to be restarted all of the time. Windows Vista was particularly unreliable, which is why it didn’t sell very well. | Linux is very reliable and is known for being run for months or even years without needing to be rebooted. | DOS is not very reliable because it crashes easily if a program doesn’t open properly or a wrong command is put in. | Linux is the most reliable out of the 3 because it doesn’t need rebooting or crash all of the time. It can run for months or even years without  need to be rebooted. If programs  don’t open correctly in DOS  then the system can crash,  because it was designed for the correct commands  to be entered and nothing else. | |
| **Ease of management** | Because Windows is so popular, people are more familiar with the way it works so it is fairly easy for everyone to use. | Later versions of Linux have greatly improved in their ease of management side, but because it is open source you have to know how to write programming language to alter things. | DOS isn’t very easy to manage unless you know the commands to perform different tasks. | Windows is probably the easiest to manage because of tutorials available to teach users and how most people are taught to use Windows over any other OS. Linux is simple as well to  an extent, but you have to know the commands to manage things past a certain point. DOS is purely command driven so it is easy to manage things  if you know the codes. | |
| **Associated utilities** | Windows is the most popular OS, so it has a much wider range of utilities. | Linux has a wide range of utilities available to download. | DOS doesn’t have many associated utilities, because it isn’t very popular any more. There aren’t many utilities available because it doesn’t have a GUI as well. | Windows has the most associated  utilities, developed by Microsoft and  many other companies. Linux has  a lot, but not as many, with many  people writing their own because it  is open source. DOS has the least  associated utilities, because it isn’t  used for applications because it  is just code. | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Cost** | The latest version of windows costsaround  £100. You also have to pay for other software you want onwindows. | Linux is free to download, so it doesn’t cost you anything. There are  software downloads that are free, such as open office etc | Because DOS isn’t actively developed any more, it is free of charge. | DOS is free of charge to download, and  so is Linux. This is better than Windows, which costs to download. Further  software for Windows also costs, and with the other operating systems of  software costs  a lot less or nothing  at all. |
| **Support for the** | Windows has its own | Linux has books | There are websites | There is most support |
| **User** | help section, and | available for help | and forums | available for windows, with |
|  | websites/forums where | and also online | available for help | a help section actually in |
|  | you can talk to people | forums. | with DOS, but they | the OS and with many |
|  | and gain information |  | provide more basic | websites and books and |
|  | about windows. There |  | Information | other sources. Linux has a |
|  | are also books available |  | because DOS isn’t | lot of support as well, but |
|  | for each version of |  | very popular. | not as much as Windows. |
|  | windows to help you. |  |  | DOS has the least amount |
|  |  |  |  | of support available. |
|  |  |  |  | Windows users need the |
|  |  |  |  | most support because of |
|  |  |  |  | its stability. |

**Experiment No: (Mini Project)**

Student Name and Roll Number:

Semester /Section:

Link to Code:

Date:

Faculty Signature:

Remarks:

**Project Title:**

**Description of Project:**

**Problem Statement:**

**Problem Analysis:**

**Program Design:**

**Programming Requirements:**

**Data/Input Output Description:**

**Algorithmic Approach/Algorithm/DFD/ER diagram/Program Steps**

**Implementation and Testing (stage/module wise)**

**Output (Screenshots)**