MALAD KANDIVALI EDUCATION SOCIETY'S
NAGINDAS KHANDWALA COLLEGE OF COMMERCE,
ARTS & MANAGEMENT STUDIES & SHANTABEN NAGINDAS
KHANDWALA COLLEGE OF SCIENCE
MALAD [W], MUMBAI – 64
(AUTONOMOUS)

(Reaccredited 'A' Grade by NAAC)
(AFFILIATED TO UNIVERSITY OF MUMBAI)
(ISO 9001:2015)

CERTIFICATE

Name: ~~Mr.~~/Ms. Bhavana Phoolchand Prajapati

Roll No: 66 _____ **Programme**: BSc IT/CS **Semester**: II

This is certified to be a bonafide record of practical works done by the above student in the college laboratory for the course **IT platforms, Tools and Practices** (Course Code: **2026UISTP**) for the partial fulfillment of Second Semester of BSc IT/CS during the academic year 2020-2021.

The journal work is the original study work that has been duly approved in the year 2020-2021 by the undersigned.

External Examiner

Subject-In-Charge
(Ms.Sweety Garg)

Date of Examination: (College Stamp)

Name: Bhavana Phoolchand Prajapati **Roll No: 66**

Sr. No.	DATE	TITLE	SIGN
1.	02-Feb-2021	INTRODUCTION and CONTRIBUTING TO WIKIPEDIA a) What is Wikipedia? b) Steps to Create Account on Wikipedia c) Creating Page on Wikipedia d) Edit your page	
2.	09-Feb-2021	Creating account, repository on GitHub and Cloning repository in GitHub Page	
3.	16-Feb-2021	BASIC UNDERSTANDING ON FREE AND OPEN-SOURCE SOFTWARE a) Describe Open-Source Software with Example. b) Describe Free Software with Example c) Difference between Free and Open-Source Software.	
4.	23-Feb-2021	WRITING EMAIL	
5.	25-Feb-2021	Using practical examples, describe green computing. List and explain the steps that you take to contribute to green computing	
6.	02-Mar-2021	WRITING BLOGS	
7.	09-Mar-2021	Implementing coding practices in Python using PEP8.	
8.	20-Mar-2021	PRESENTATION: PEP-8 Style Guide	

Name: Bhavana Prajapati **Class: FYIT** **Roll no: 66**

Practical No: 1 Introduction about Contribution to Wikipedia

- **Description about Wikipedia and its features**
- **Description about Wikipedia:**

Wikipedia is a free, open content online encyclopaedia created through the collaborative effort of a community of users known as *Wikipedians*. Anyone registered on the site can create an article for publication; registration is not required to edit articles. The site's name comes from wiki, a server program that enables anyone to edit Web site content through their Web browser. Jimmy Wales and Larry Sanger co-founded Wikipedia as an offshoot of an earlier encyclopaedia project, Nupedia, in January 2001. Originally, Wikipedia was created to provide content for Nupedia. However, as the wiki site became established it soon grew beyond the scope of the earlier project. As of January 2015, the website provided well over five million articles in English and more than that number in all other languages combined. At that same time, Alexa ranked Wikipedia as the seventh-most popular site on the Internet. Wikipedia was the only non-commercial site of the top ten.

Wikipedia was one of the first social media websites.

- **Features of Wikipedia:**

1. Wikipedia is a free, multilingual open-collaborative online encyclopedia created and maintained by a community of volunteer editors using a wiki-based editing system.
2. Wikipedia has openness - anyone could create articles, and any Wikipedia article could be edited by any reader, even those who did not have a Wikipedia account.
3. Wikipedia has restrictions for publishing the articles
4. Changes can be reviewed by others. The software that powers Wikipedia provides tools allowing anyone to review changes made by others.
5. Wikipedia has a power to remove the Vandalism on the content.
6. Wikipedia is a great place to start your research, giving you background information on your topic and possible keywords to help you conduct more in-depth research elsewhere.

- **Creating Account on Wikipedia:**

Step 1:

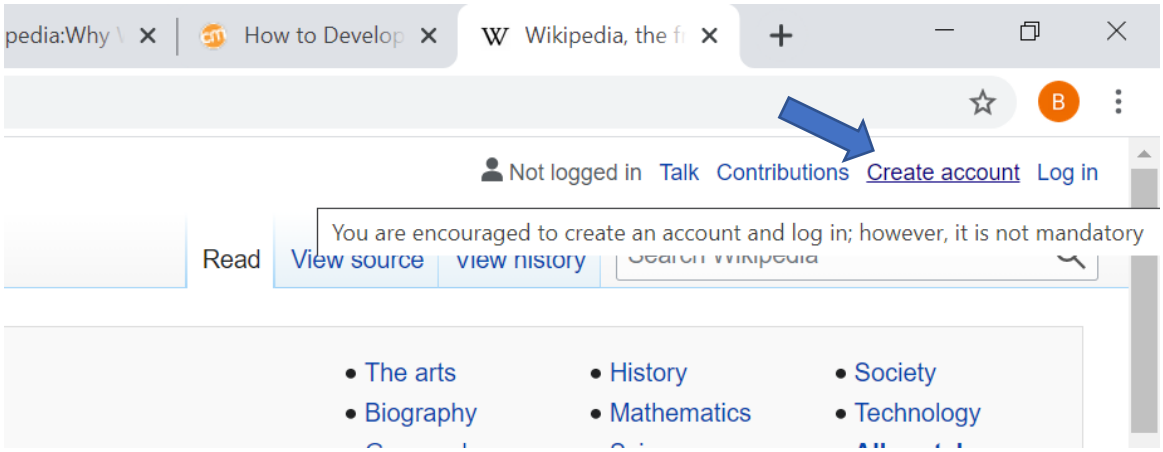
For Publishing a Page on Wikipedia, first you should have an account on Wikipedia.

Go to https://www.wikipedia.org/

Click on the Language as per your need. We are selecting English for the simplicity

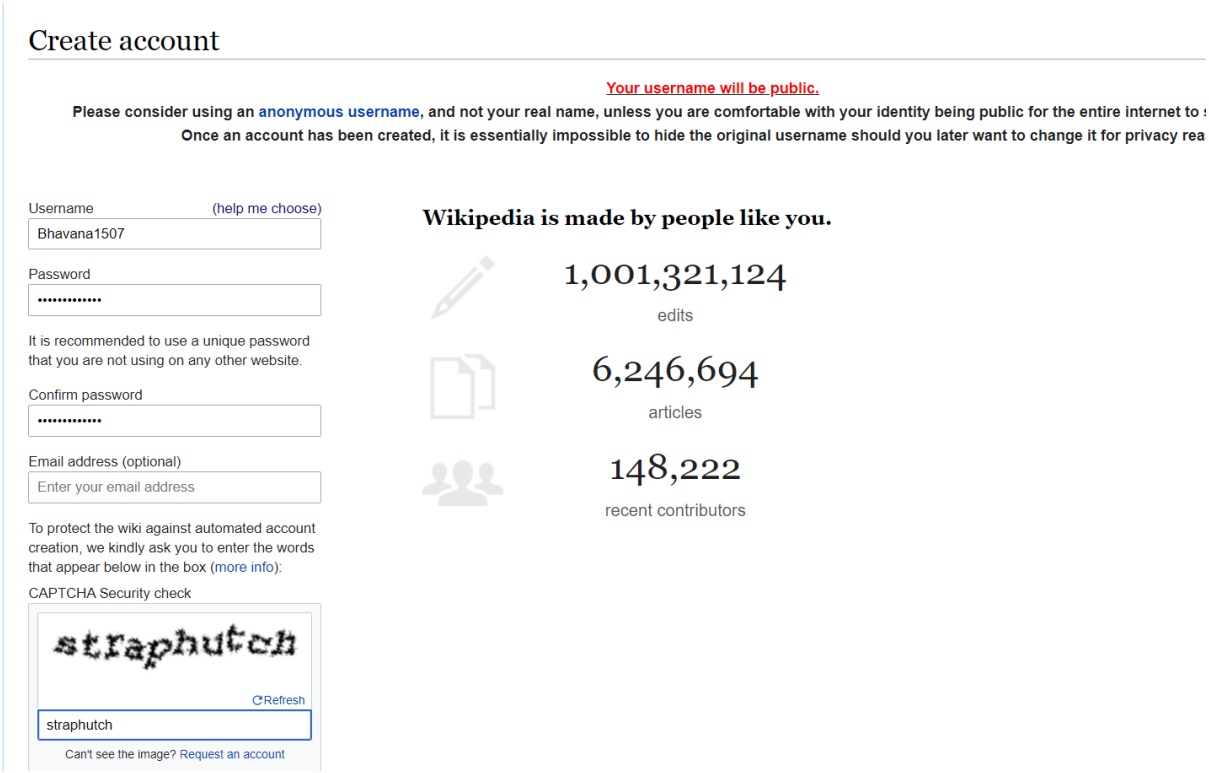
Step2:

Click on ***Create Account***



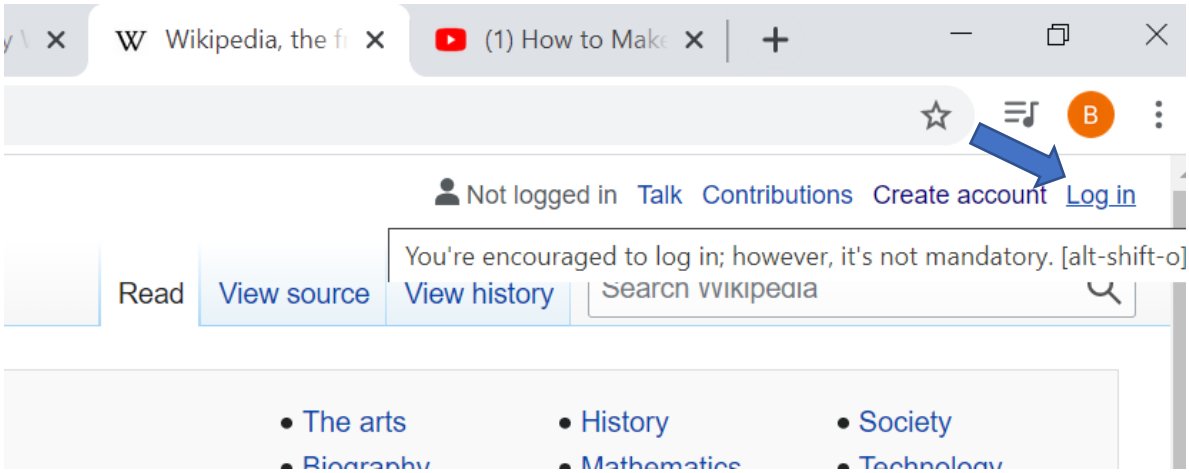
Step 3:

Provide proper Username and Password as per the policy of Wikipedia, then Submit it and your account will be created.



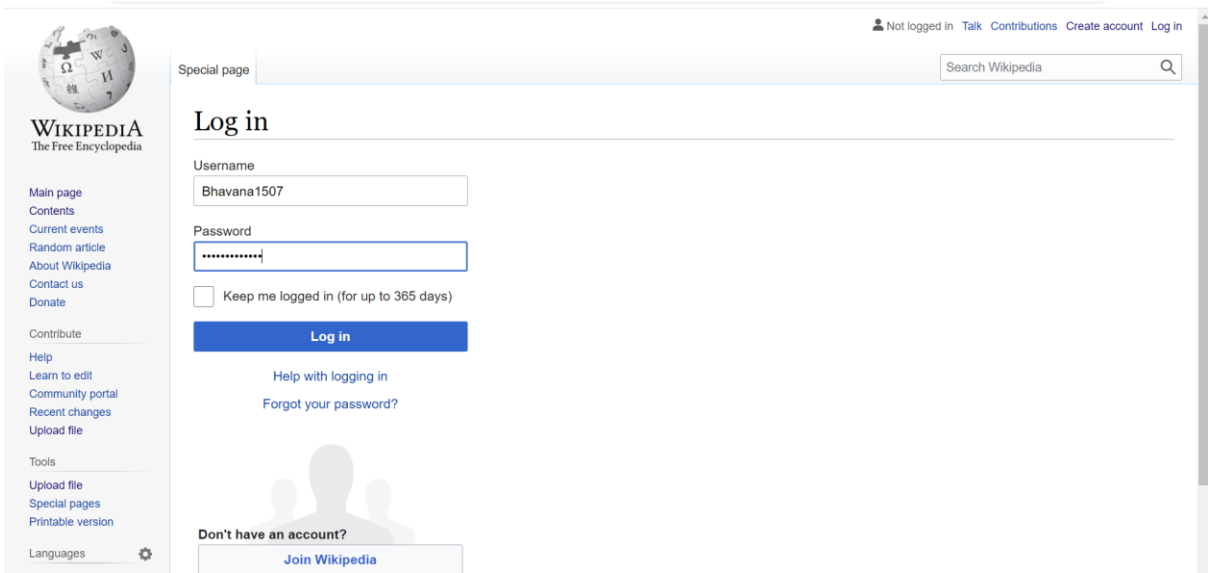
Step 4:

Once your account is created, now we should login using the credentials.



Step 5:

Give proper credentials and login to Wikipedia.



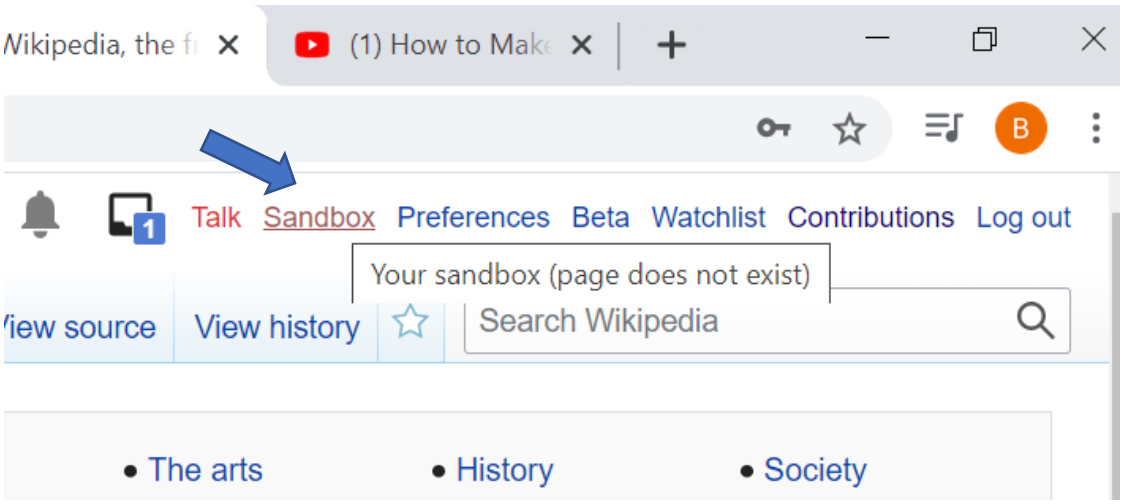
Creating your page on Wikipedia:

Step1:

1. **Create an account**
2. **Create the page**
3. **Go to the Wikipedia Article Wizard**
4. **The Wizard and the Sandbox**
5. **Submit your page for review**

Step1:

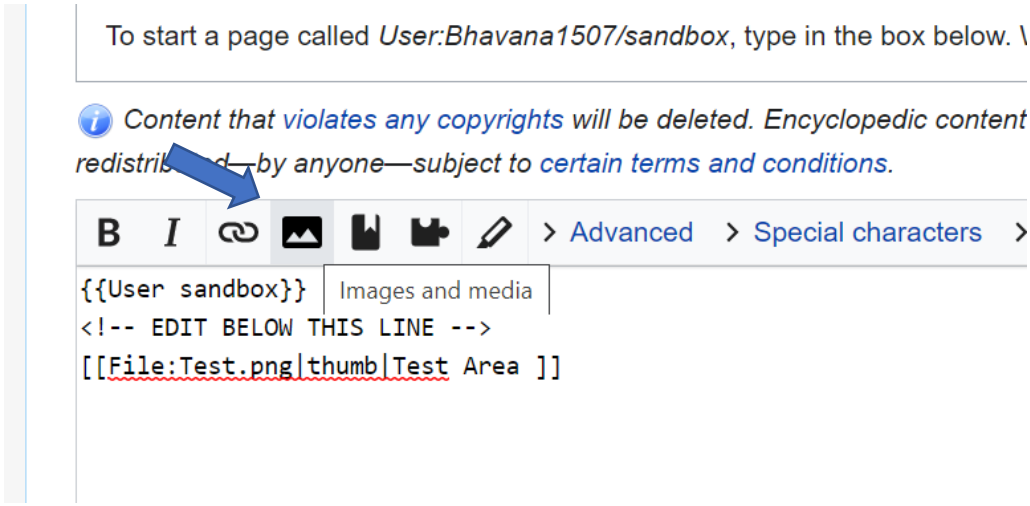
Click on ***Sandbox***

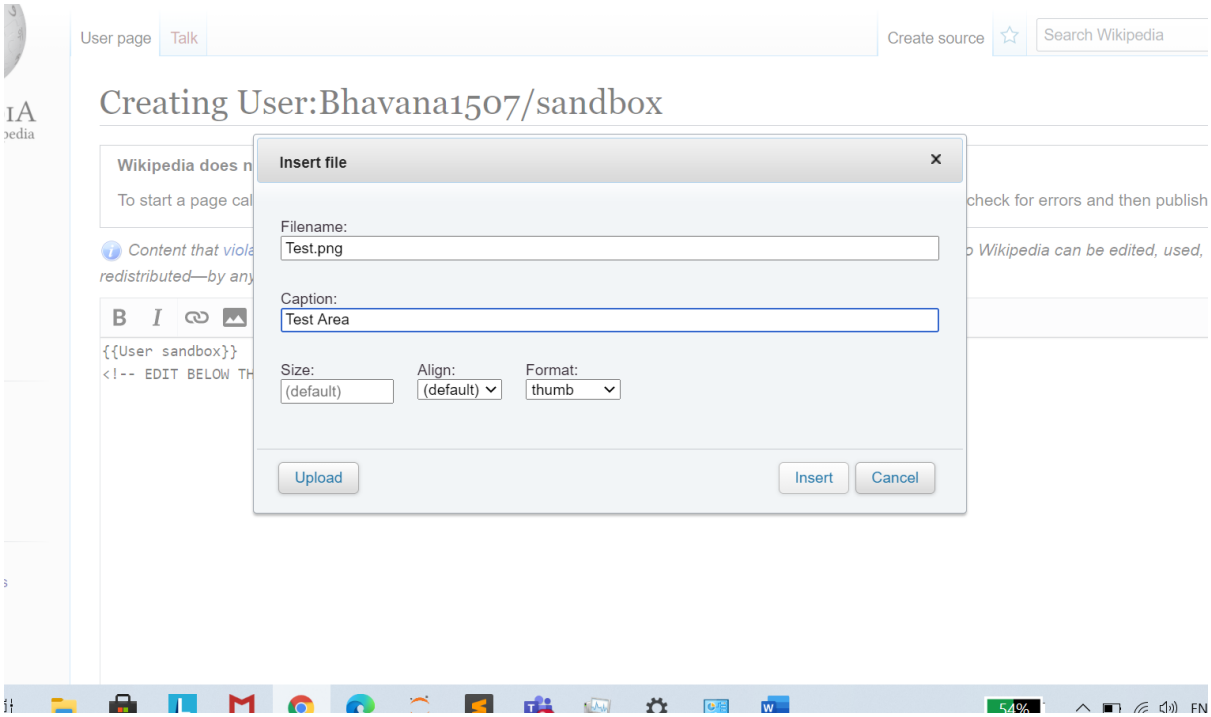


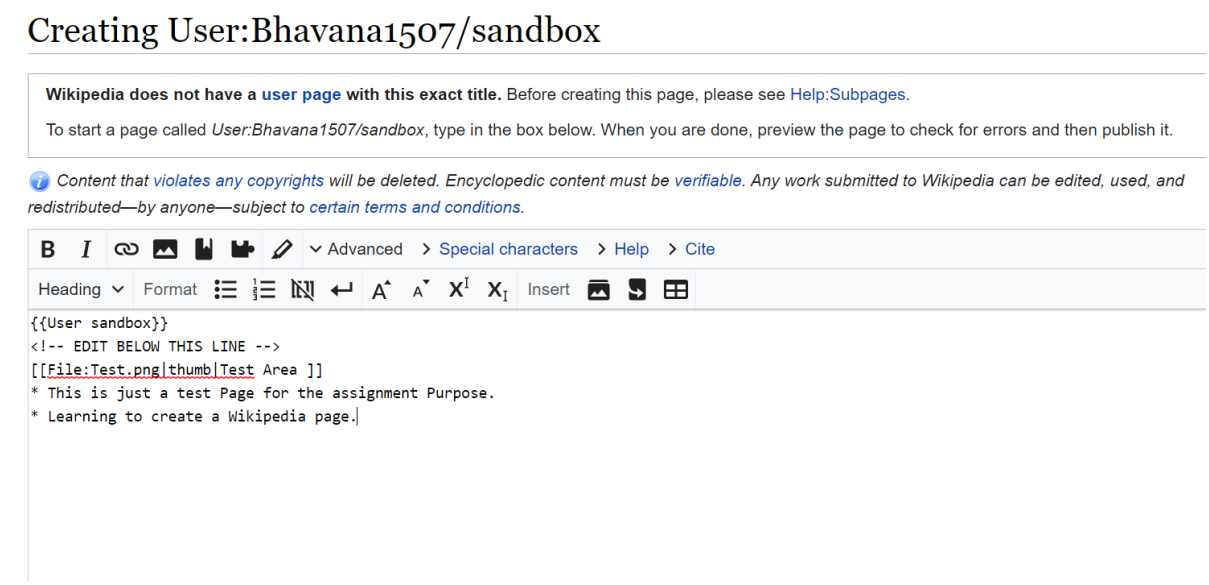
Step 2:

This is the main page where we provide the content that we want to Publish over Wikipedia.

We can include images, links, format the texts like we do in Microsoft Word, we can link the references and many more

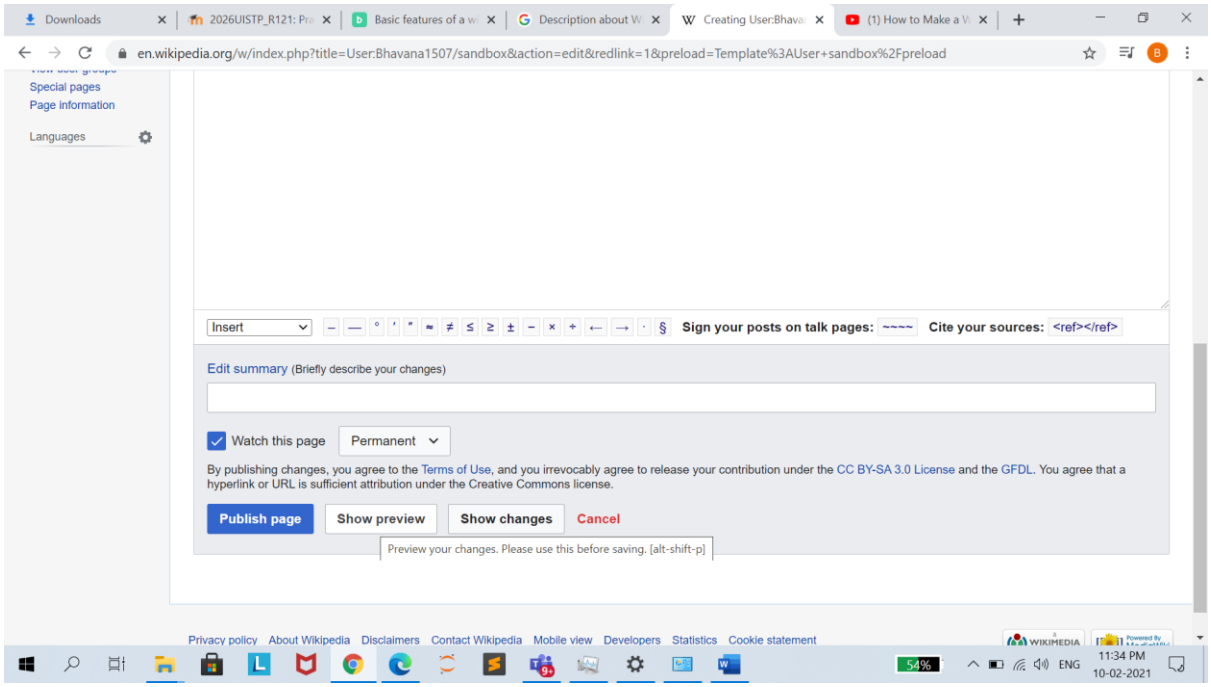

To start a page called User:Bhavana1507/sandbox, type in the box below.
Content that violates any copyrights will be deleted. Encyclopedic content
redistrib by anyone—subject to certain terms and conditions.
Advanced
Special characters
Images and media
{{User sandbox}}
<!-- EDIT BELOW THIS LINE -->
[[File:Test.png|thumb|Test Area]]


User page
Talk
Create source
Search Wikipedia
Creating User:Bhavana1507/sandbox
Insert file
Filename:
Test.png
Caption:
Test Area
Size:
(default)
Align:
(default)
Format:
thumb
Upload
Insert
Cancel



Step3:

After you have arranged your content in the editor, then click on ***Show Preview***, once you find that Preview is the same like you want the content to be. Then comes the Final stage, you click on ***Publish Page***



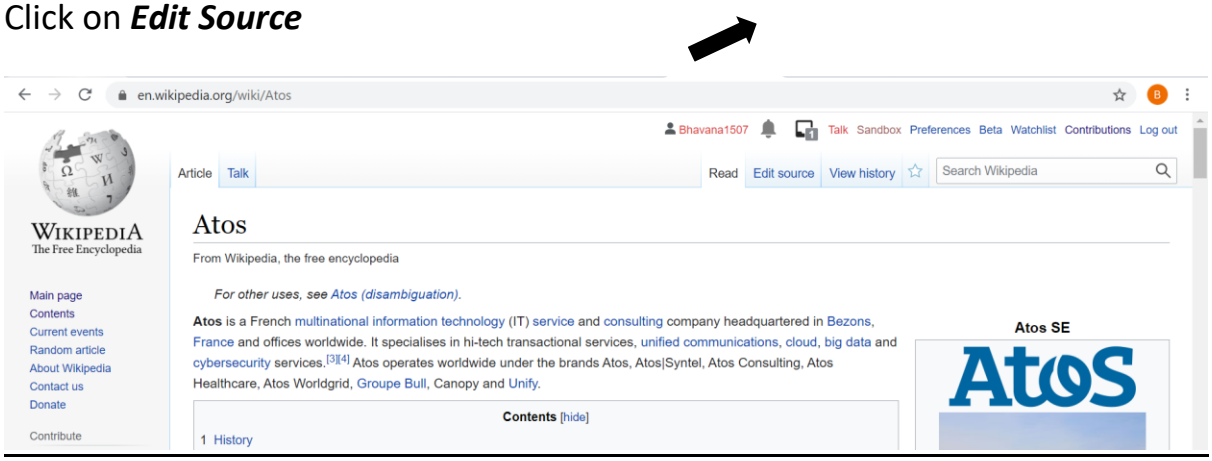
Then your content will be reviewed by Wikipedia community, once it is reviewed and passed by them, then it will be publicly accessible, otherwise they will provide you feedback based on your content and suggest you what is make your content not publishable.

- **Editing your page on Wikipedia:**

Step1:

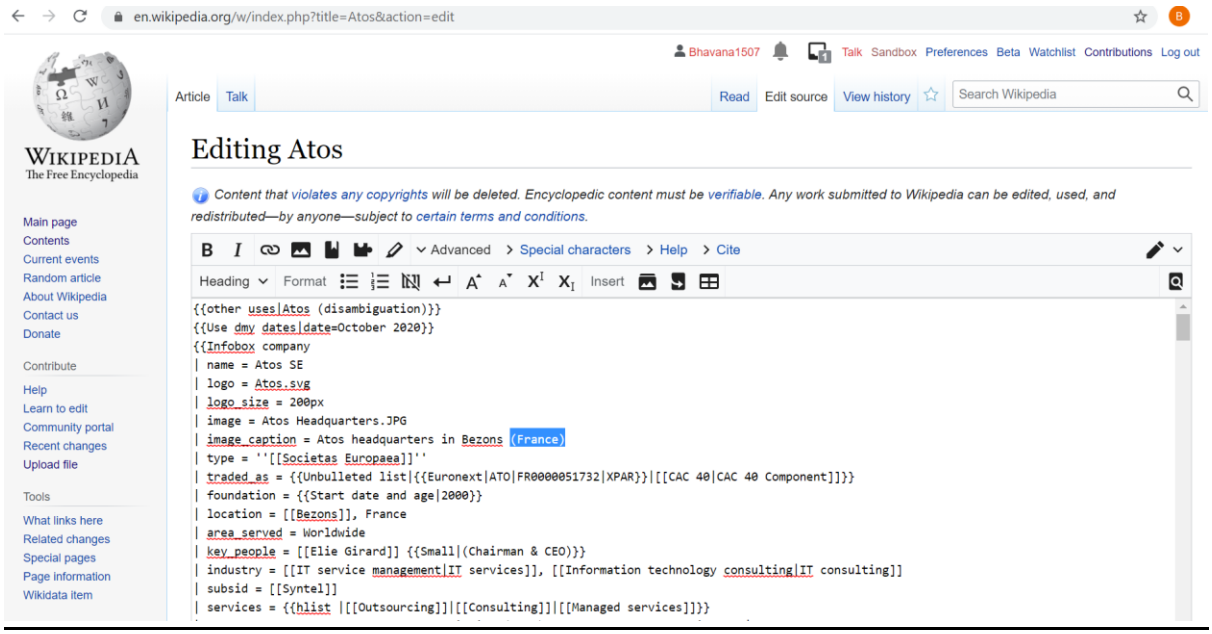
Editing an Unprotected Page

Click on ***Edit Source***



Step2:

Make the necessary changes under the Editing Section



Step3:

Click on Show preview. It will give you the chance to take a look at the article with your edits without saving the edit.

Then Click Publish changes. When you edited the page and clicked Show preview, click Publish changes to save your edits.

Always do your best to leave an edit summary to describe that changes you have made.

Name: Bhavana prajapati **Class: FYIT** **Roll No: 66**

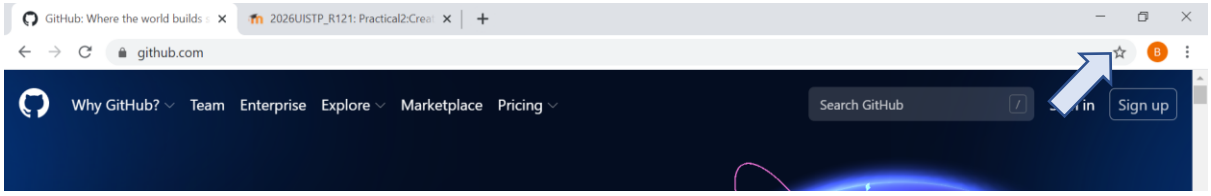
Practical No: 2

[Creating account, repository on Github and Cloning repository in Github]

1.Creating a GitHub Account:

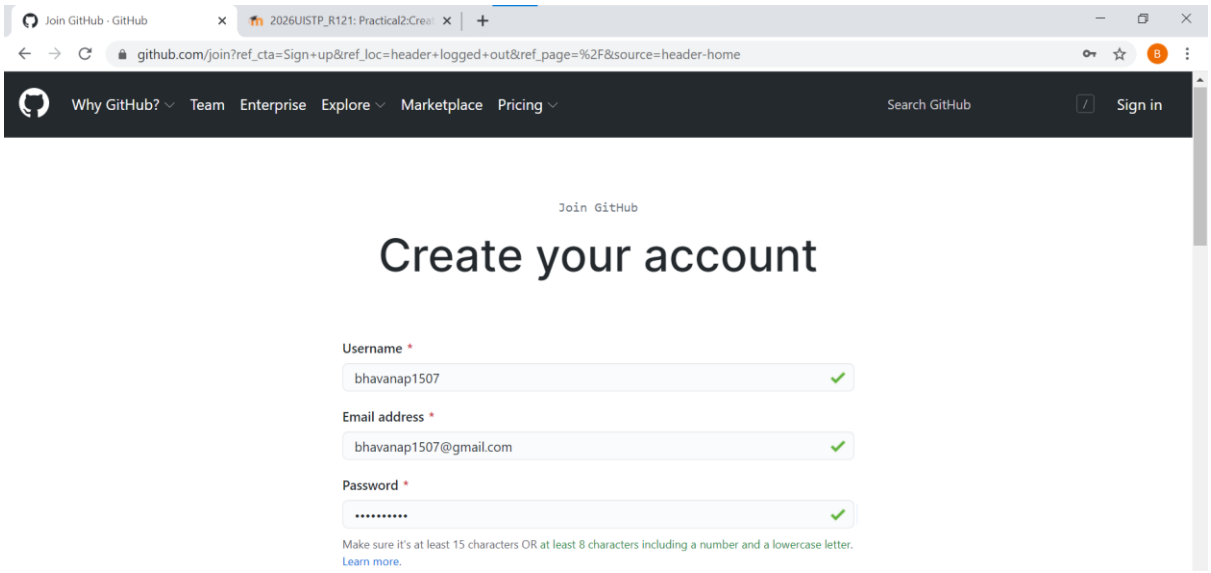
Step1:

- Go to GitHub.com and click on ***Sign Up***



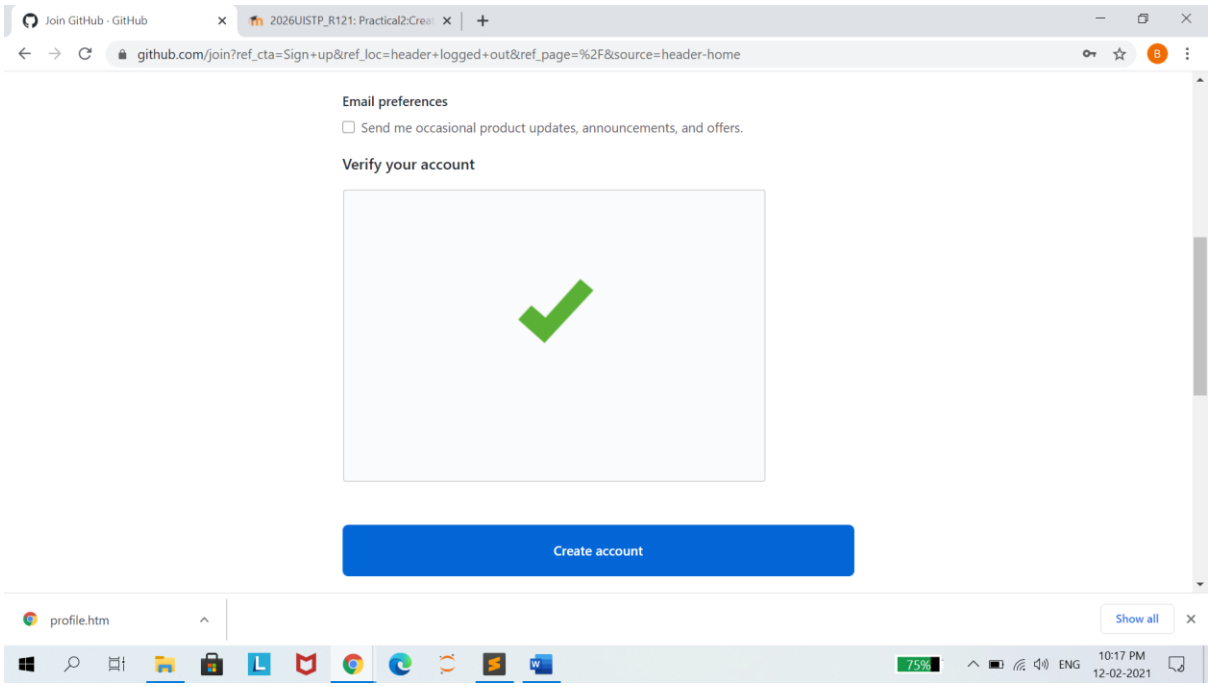
Step2:

Fill in the Proper details like Username, Email Address and Password



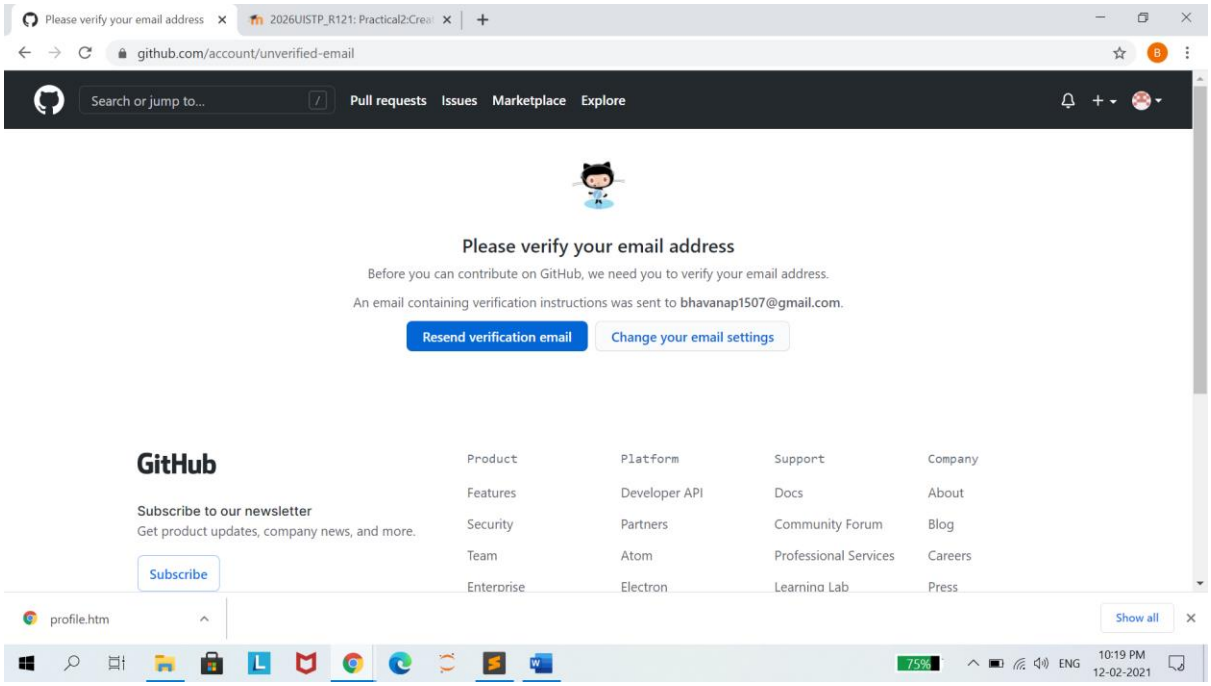
Step3:

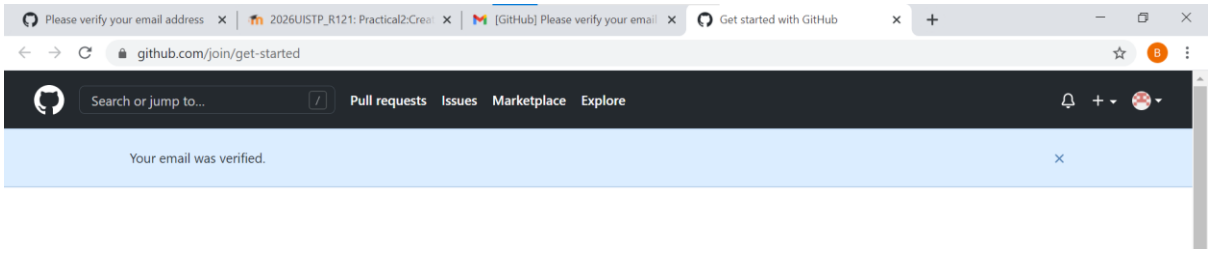
Verify the captcha and then click on ***Create Account***



Step4:

We will get an email on registered mail ID to verify the github account

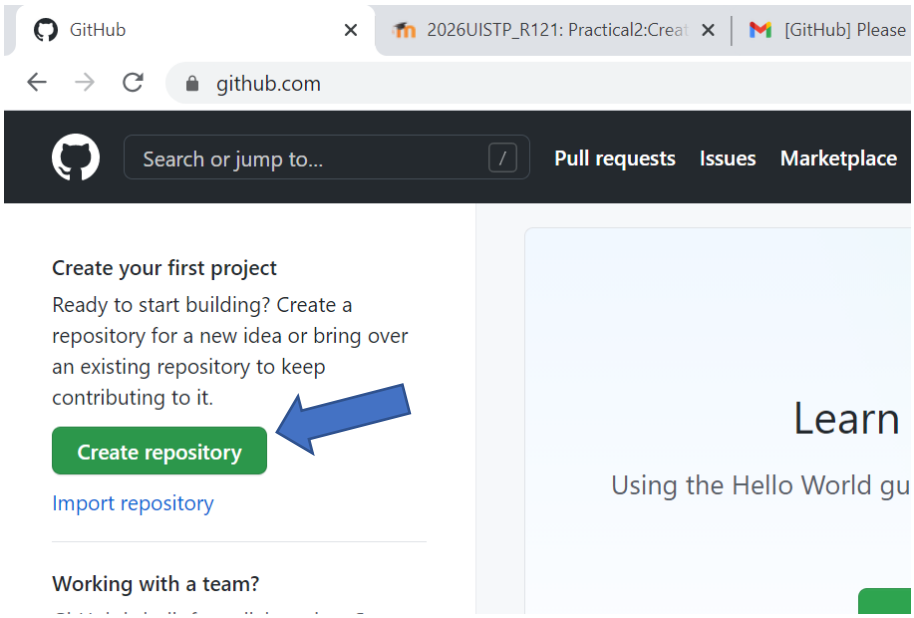




2.Creating a Repository:

Step1:

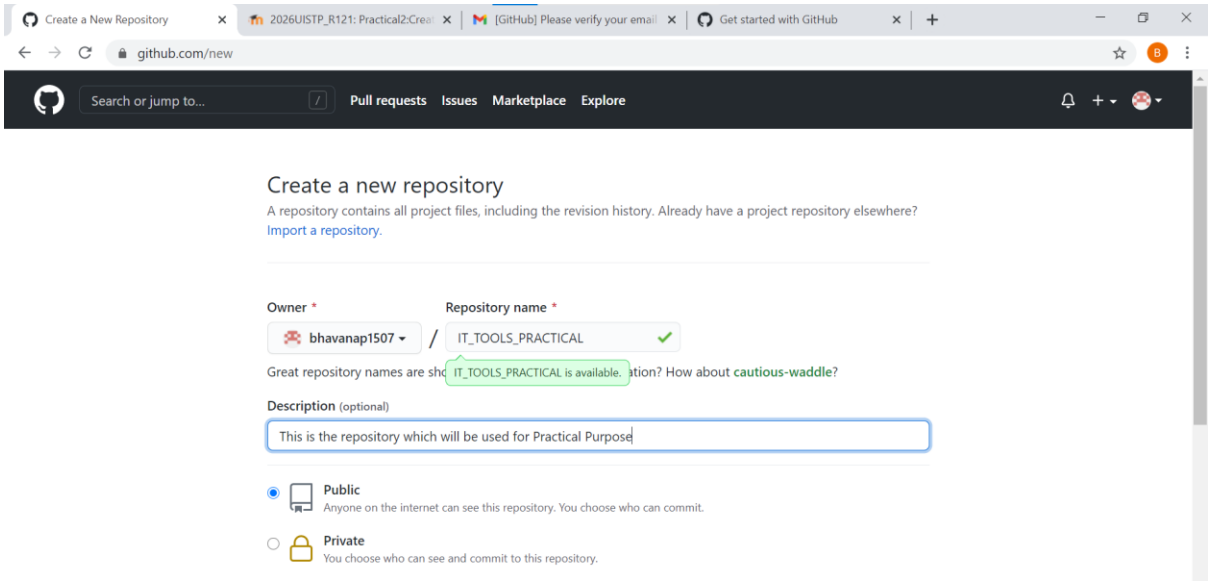
After Verifying the GitHub account, then Logon to Github account and create a new Repository by clicking on Button ***Create Repository.***



Step2:

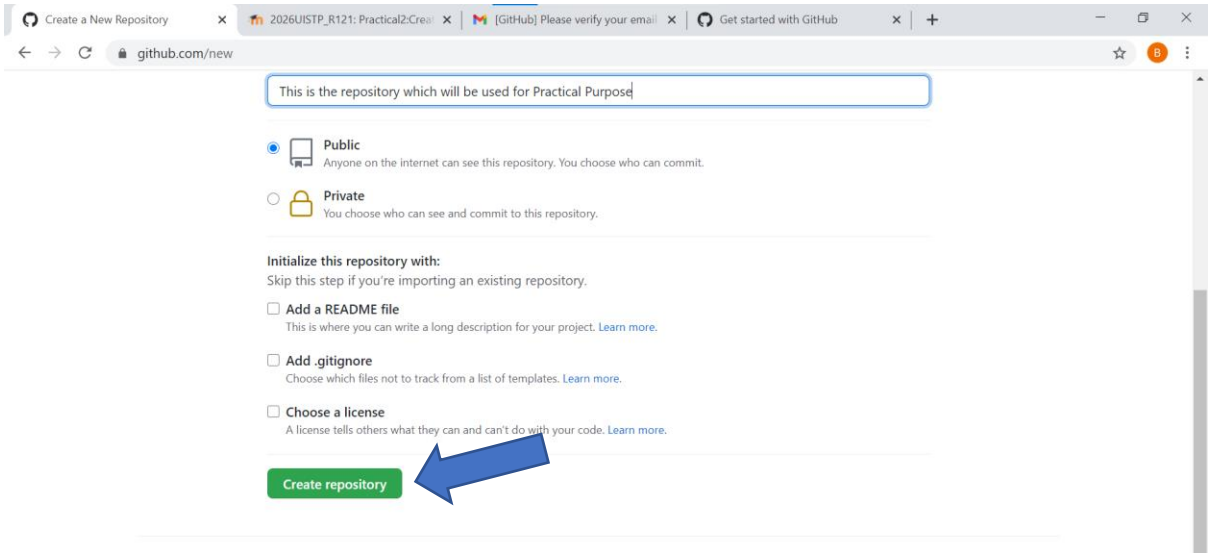
Give a ***Repository Name*** and select the option as shown below

We are creating a repository named ***IT_TOOLS_PRACTICAL***



Step3:

Click on ***Create Repository*** after filling on all the necessary details

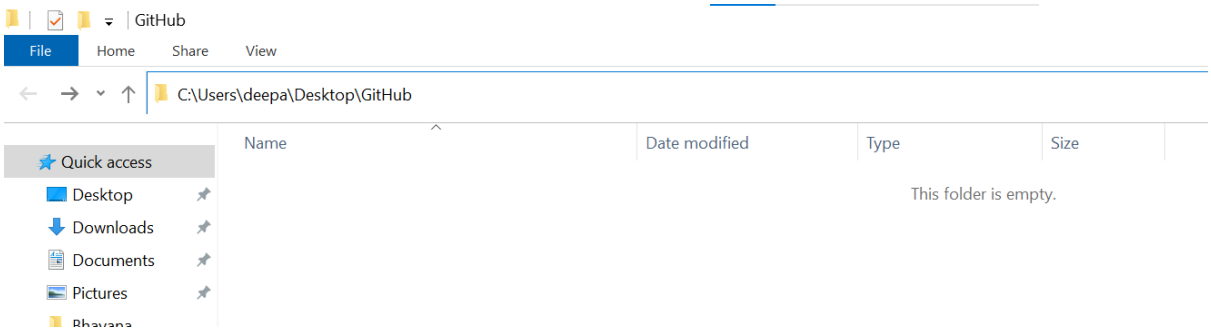


3.Cloning a Repository:

Step1:

Create a Folder on your computer where you want to clone the repository

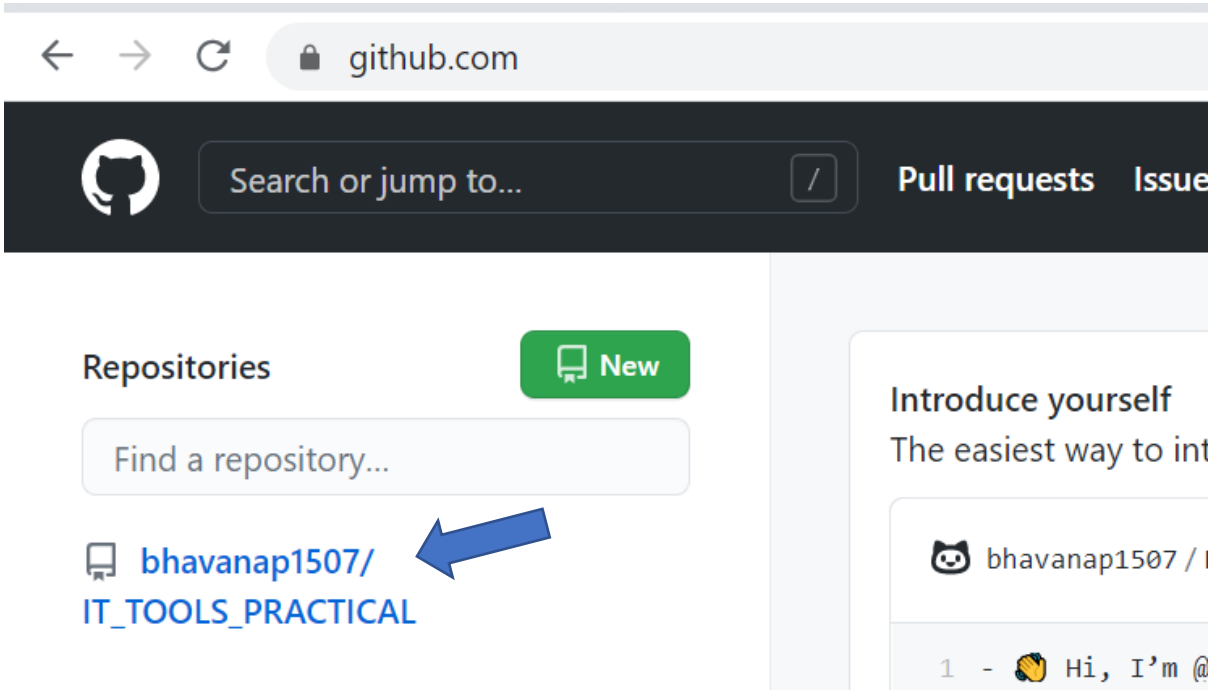
I created a ***GitHub*** Folder on my Desktop



Step2:

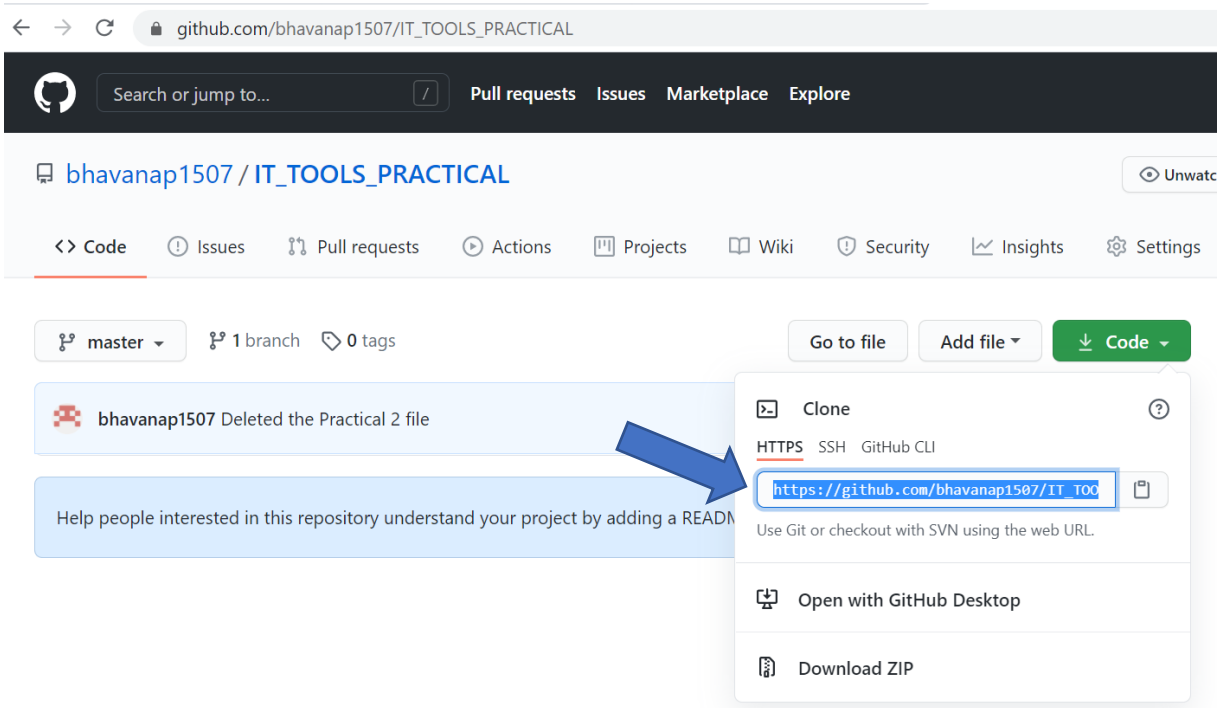
We will clone the repository ***IT_TOOLS_PRACTICAL*** which we created above.

First, We will go to the repository on the Github account which we want to clone.



Step3:

Drop down the ***Code*** button and Copy the link under ***HTTPS*** Section



Step 4:

Open ***Command Prompt*** and go to the Path where you want to clone the Repository.

Command: **`git clone <HTTPS_PATH_OF_THE_REPOSITORY>`**

Execute this command

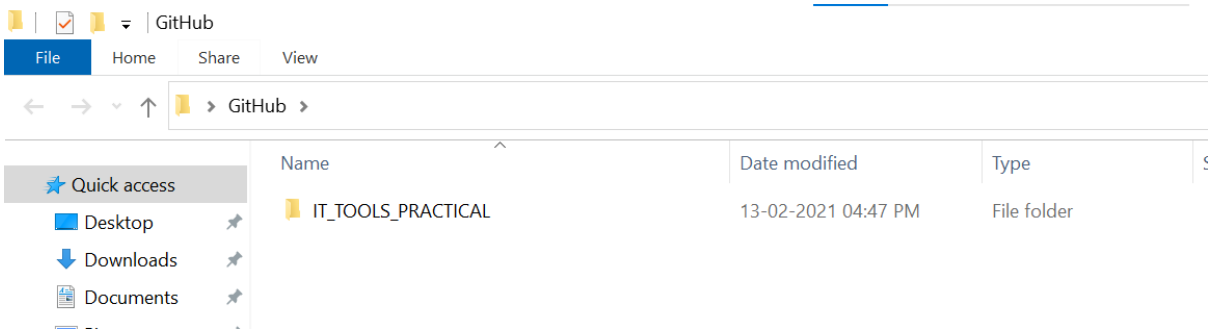
```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19041.264]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\deepa>cd Desktop/github

C:\Users\deepa\Desktop\GitHub>git clone https://github.com/bhavanap1507/IT_TOOLS_PRACTICAL.git
Cloning into 'IT_TOOLS_PRACTICAL'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 9 (delta 1), reused 8 (delta 0), pack-reused 0
Receiving objects: 100% (9/9), 5.82 MiB | 1.95 MiB/s, done.
Resolving deltas: 100% (1/1), done.

C:\Users\deepa\Desktop\GitHub>
```

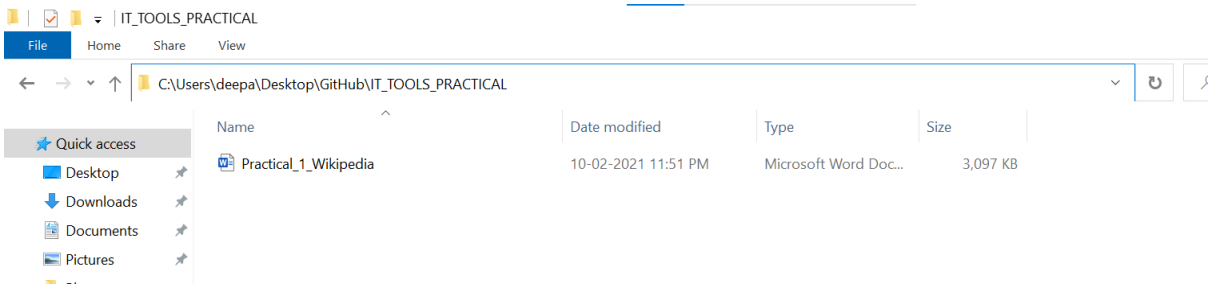
Now you can see that the Repository has been cloned to your computer



Step5:

Now, you can do the changes in the local repository.

For e.g: We are adding a ***Practical_1_Wikipedia.docx*** in the local repository



Step6:

Check the Status of the local Repository

```
C:\Users\deepa\Desktop\GitHub\IT_TOOLS_PRACTICAL>git status
On branch master
Your branch is up to date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Practical_1_Wikipedia.docx

nothing added to commit but untracked files present (use "git add" to track)

C:\Users\deepa\Desktop\GitHub\IT_TOOLS_PRACTICAL>
```

Now we will try to add the newly made changes in the Staging area before pushing the final changes to the Github repository

Adding changes to Staging Area

Command: **`git add <FILE_NAME_SHOWN_IN_GIT_STATUS>`**

Execute this command

```
C:\Users\deepa\Desktop\GitHub\IT_TOOLS_PRACTICAL>git add Practical_1_Wikipedia.docx

C:\Users\deepa\Desktop\GitHub\IT_TOOLS_PRACTICAL>git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   Practical_1_Wikipedia.docx
```

Committing the changes

Command: **`git commit -m "Meaningful Message relevant to changes"`**

```
C:\Users\deepa\Desktop\GitHub\IT_TOOLS_PRACTICAL>git commit -m "Adding Practical 1 to the Github Repository"
[master 39c27d1] Adding Practical 1 to the Github Repository
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Practical_1_Wikipedia.docx

C:\Users\deepa\Desktop\GitHub\IT TOOLS PRACTICAL>_
```

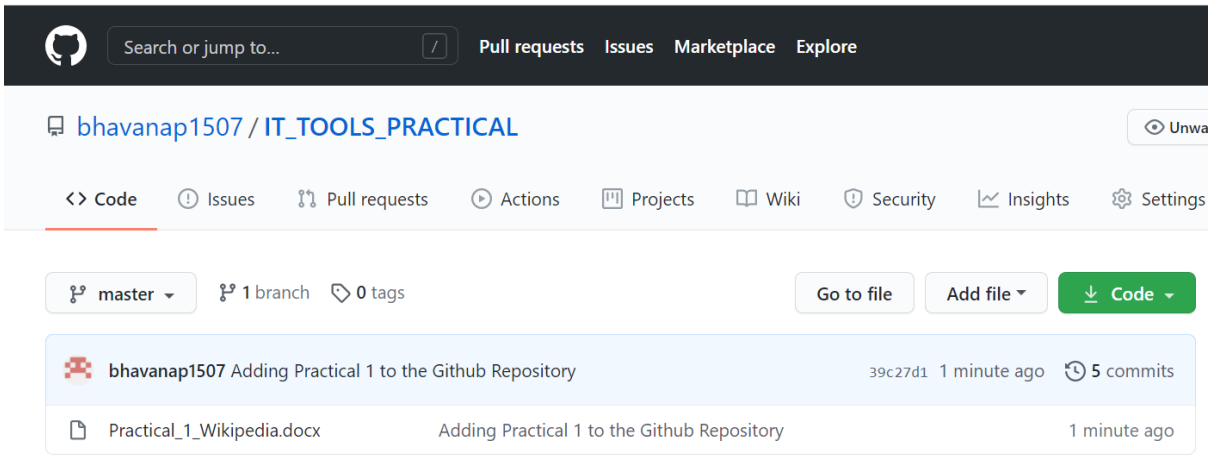
This is the final stage, Pushing the changes to the GitHub Repository

Command: **`git push`**

```
C:\Users\deepa\Desktop\GitHub\IT_TOOLS_PRACTICAL>git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 6 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 2.91 MiB | 1.22 MiB/s, done.
Total 3 (delta 0), reused 1 (delta 0), pack-reused 0
To https://github.com/bhavanap1507/IT_TOOLS_PRACTICAL.git
   d38ac87..39c27d1  master -> master

C:\Users\deepa\Desktop\GitHub\IT_TOOLS_PRACTICAL>_
```

Now we can see that the changes have been reflected in the GitHub Repository.



Name: Bhavana Prajapati **Class: FYIT** **Roll No: 66**

Practical No: 3

BASIC UNDERSTANDING ON FREE AND OPEN-SOURCE SOFTWARE

- **Describe Open-Source Software with Example.**

Open-Source Software:

Open-source software (OSS) is any computer software that's distributed with its source code available for modification. That means it usually includes a license for programmers to change the software in any way they choose: They can fix bugs, improve functions, or adapt the software to suit their own needs. Open-source software (OSS) is a type of computer software in which source code is released under a license in which the copyright holder grants users the rights to use, study, change, and distribute the software to anyone and for any purpose. Open-source software may be developed in a collaborative public manner.

The “open source” label was created at a strategy session held on February 3rd, 1998 in Palo Alto, California, shortly after the announcement of the release of the Netscape source code.

Open-Source Software Example:

1. Firefox

2.Gimp

3.Open Office

4.PHP

5.Moodle

6.Linux Operating System

7.Android by google

8.VlC media

9.Blender

10. Python

The Most Popular Open-Source Software Licenses:

1. MIT license (MIT)
2. Apache License 2.0 (Apache-2.0)
3. 3-clause BSD license (BSD-3-Clause)
4. GNU General Public License (GPL)
5. Common Development and Distribution License 1.0 (CDDL-1.0)

- **Describe Free Software with Example:**

Free Open Source:

The concept of **free software** is the brainchild of Richard Stallman, head of the GNU Project. The best known example of free software is Linux, an operating system. Debian is an example of a distributor of a Linux package. According to the Free Software Foundation (FSF), a non-profit organization that supports the development of free software, "free software is the software that grants the user the freedom to share, study, and modify it." The FSF coined the term in the 1980s. It is used it without any restriction. Software available without any payment is not necessarily free software. According to the definition as presented by the Free Software Foundation, the word Free in "free software" implies the idea of freedom rather than not having a cost. If software is available to be downloaded without being paid for, but the user is not able to modify the source then it is not free software. It is important to note the distinction here. Freeware is software that you don't have to pay for. Free software is software you are free to modify and use for your own purposes. Freeware does not necessarily have to be free software as the source code can still be protected.

Free Source Software Example:

1.Linux

2.Apache

3.Google Chrome

4.Internet Explorer

5.Whatsapp

The Most Popular Free Source Software Licenses:

1.GPL (general public Licenses)

2.Apache Licenses

- **Difference between Free and Open-Source Software.**

Free Source Software	Open-Source Software
1.free source software foundation "Richard Stallman".	1. Open-Source initiative "Eric Raymond".
2.GPL (General Public License)	2.BSL (BSD-style licenses)
3.Free software is a software for which every one has a right not only to inspect and study the source code but also to use it for any desired purpose without monitory or other restriction.	3.Open-source software whose source code is freely available. That is without any requirement for payment or any other obstacle for any one to inspect and study.
4. Freedom to modify/improve program and release improvements to public.	4.Intergrity of authors Source Code.
5.free Software is a social movement.	5. open source is a development methodology.
6.Free software focuses on providing a moral/ethical argument for open source.	6.open source tends to focus on providing an economic/Business argument for free software.
7.Free software is a good, morally right, thing to do.	7. Open-source software is beneficial to you and your business.
8.all existing free software would qualify as open source. (according to Richard Stallman, GNU free software movement.	8.Nearly all open-source software is free software (according to Richard Stallman, GNU free software movement.

9.Freedom of information.	9.Better quality software.
10.Examples: The free software directory maintains a large database of free software packages. some of the best-known examples include the Linux kernel, the BSD and Linux operating systems, the GNU Complier Collection and C library; the MySQL relational database; the Apache web server; and the sendal mail transport again.	10.Examples: prime Examples of open-source products are the Apache HTTP server, the e-commerce platform commerce, internet browsers Mozilla Firefox and Chromium (the project where the vast majority of development of the freeware google chrome is done and the full office suite LibreOffice.

Name: Bhavana Prajapati **Class: FYIT** **Roll No: 66**

Practical No: 4 [Writing E-Mail]

Job Application Request for Web Developer

66_FYIT_ Bhavana_Prajapati <bhavanap1507@gmail.com> 12:26 PM
to deepakp9513

Hello Deepak,

Good Morning! I hope you are well in this pandemic.

I would like to apply for the subjected profile in your company.

I heard about this vacancy on Naukri.com and I found the job description and job location suitable for me.

Please find the enclosed attachment of my Resume.

Looking forward to hear from you.

Thank You,
Bhavana Prajapati

Name: Bhavana Prajapati **Class: FYIT** **Roll No: 66**

Practical No: 5 [Green Computing]

Green Computing: Green computing is the environmentally responsible and eco-friendly use of computers and their resources. In broader terms, it is also defined as the study of designing, engineering, manufacturing, using and disposing of computing devices in a way that reduces their environmental impact. Many IT manufacturers and vendors are continuously investing in designing energy-efficient computing devices, reducing the use of dangerous materials and encouraging the recyclability of digital devices. Green computing is also known as green information technology (green IT).

History of Green Computing:

The starting of Green computing was named as Energy Star, and it was originated in 1992.This Energy Star was using in all electronic product like as Printers, television and refrigerators, in that time saved more energy but that is not used in computers. After spending some time Green computing name was converted into Energy Star, after that was used in computers for saving energy.

Types of Green Computing:

Solar Power System

In this program we utilize the sunlight and produce the Solar Power for personal and commercial usage. Canada, Spain and California have first position for implementing this technology. This is great achievement for green technology.

Wind Turbine Program

Other great type is Wind Turbine system because with the help of this system anyone can generate electricity power. After embedding wind turbine has no bad effect to environment. It decreases the carbon dioxide emissions. But require huge money for set up of wind turbine, so it is not possible to everyone.

Geothermal Power

This is also exclusive type of green technology. With the help of this Geothermal plant can be generated electricity, and people can utilize of this power in daily usage such as heating and cooling house.

Need of Green Computing

- Save huge money
- Save environment
- Decrease the risks in further life
- More consumption of energy
- for recycle of waste product
- Inspiring to worker
- For retaining high ticketing customers

Advantages of Green Computing:

- Lessened vitality utilization by green registering advances converts into low carbon dioxide emanations, which emerge because of the absence of petroleum derivatives utilized as a part of intensity plants and transportation.
- Conservation of resources means less energy is required to produce, use and dispose of products.
- Saving energy and resources saves money.
- Green processing includes changing government arrangement to empower reusing by people and organizations and to lessen vitality utilization.
- Reduce existing exposure in laptops such as chemical, cancer, nerve damage, and is known due to immune responses in humans.

Disadvantage of Green Computing:

- It can really be quite expensive.
- Some green computers may be very low.
- Rapid technology change.

Explain the steps that you take to contribute to green computing:

- Use the hibernate or sleep mode when away from a computer for extended periods
- Buy energy-efficient notebook computers, instead of desktop computers
- Activate the power management features for controlling energy consumption
- Make proper arrangements for safe electronic waste disposal
- Turn off computers at the end of each day
- Refill printer cartridges, rather than buying new ones
- Instead of purchasing a new computer, try refurbishing an existing device
- Buy "Energy Star" labelled monitors, desktops, laptops, and printers. The "Energy Star" devices can be programmed to "power-down" to a low power state when they are not in use, helping you save energy and run cooler which helps them last even longer.
- E-cycle used computer equipment. Find a recycler in your area. Also, Staples, the office supply retailer, has now started a recycling program. They will accept any brands of used desktop and notebook computers, monitors, printers, fax machines and all-in-one devices with a fee of $10. Smaller items like keyboards, mice and speakers are free to drop off.
- It is relatively easy for an organization to centralize its information technology (IT) system. With server virtualization, carbon footprints can be significantly reduced.
- A good example is checking total power consumption for each month. If it has significantly dropped, then one can say that we have effectively reduced your organization's carbon footprint.
- Participate in electronic recycling programs.
- Look for green packaging solutions.
- Replace old CRT and LCD monitors with efficient OLED monitors.

Conclusion:

Various IT organizations in the world started adopting green computing practices and it has now become the main part of their day-to-day business activities. In India IT giant Wipro has already launched its eco-friendly series of computing systems called greenware. Samsung, Apple, and other IT companies have their own successful recycling programs. By adopting green computing practices we can get an eco-friendly environment, along with other benefits such as cost reduction, energy conservation, and waste minimization.

Now it's our turn, we must understand the importance of green computing and work collaboratively for a healthier and greener environment for our future generations.

Name: Bhavana Prajapati **Class: FYIT** **Roll No: 66**

Practical No: 6 [Blog]

Who am I?

SHARE

Hello Reader,

Not having that much hands on as a Blog Writer, So Pardon my immaturity in the writing. If you are Grammar Nazis, then I think this is the Stop sign for you, because if you went further, I am not sure you placed your time on the right bet or not?

I am Bhavana. I am born and brought in Mumbai (Bombay...Sssh!!!! Keeping it in Bracket because We all know we might enrage some Political Personas on this). So Ahem, I am from Mumbai.

Well, I realize over the time that having some leisure hobby or interests other than Academics can make you think beyond the conformity and can trigger your intellectual trait and can make you think and understand out of box (Well, sometimes Inside the box also works, you know!). I deliberately wanted a Hobby so I chose the safest Hobby which fits up the Resume and won't cost that much to maintain that hobby till your lifetime, Any Guesses? Don't Worry! It's not a brain stormer, so let's reveal the card, it's READING ! and I also want to take up some more space in Interest section of Social Media so I chose one more hobby apart from Reading and that is CRAFTING !

The Diary of Young Girl ! was my first book which I read at a perfect time, perfect as in during the Lockdown. Events were so relatable in the book and the events which were happening around thus during that time. Anne, her family and Many more were hiding in their respective houses during the World War II and similarly we as well, Not Hiding as per se but were not going out of the house during the Pandemic. To know more about **Anne Frank**, go to a

library and ask for **The Diary of Young Girl**, Librarian might judge you there, but wait you can use the Proverb here "**Don't Judge a Book by it's cover".**

I won't be able to show any Crafting Skills as it requires great picture of it to be clicked, but Unfortunately, Photography is not the area where I shine. But You can Trust me, I know Crafting !

PS: Some Says My Witty Nature Makes them Laugh ! Do you agree on that? I know you agree!

THANK YOU!

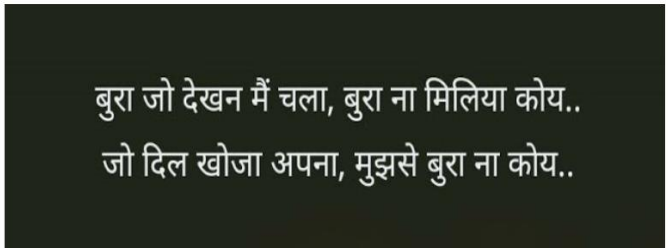
SHARE

Comments

F.L.Y

March 21, 2021

Philosophers, Sufis, Saints, Gurus or mentors all preach to F.L.Y, it can mean to fly as per the dictionary definition or we can add an another meaning to it, i.e First Love Yourself.



SHARE

Labels

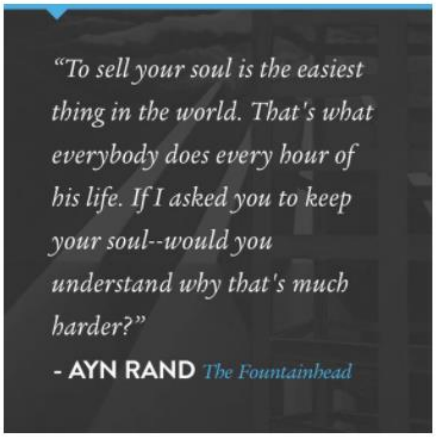
Ayn
Doha
First
FLY
Fountainhead
Kabir
Love
Rand
Soul
yourself

Kabir, once stated in his couplet (doha) : *Bura Jo Dekhan Main Chala, Bura Naa Milya Koye, Jo Mann Khoja Aapna Mujhse bura na Koye.* To start the Journey of loving oneself, to understand yourself, one should accept their shortcomings, insecurities and try to improve yourself slowly and gradually. To be the better version of yourself is not a one day thing, it is a process which you have to follow as a discipline. Rather than pointing out other's fault, it's always good and right to improve yourself.

"The man who does not value himself, cannot value anything or anyone."
– *The Virtue of Selfishness: A New Concept of Egoism*

Ayn Rand, One of the Modern Classic Writers took a philosophical take on words like *Selfishness* and *Egotism.* She totally turned the tables around these words and gave a new perspective to these words. As per her, Selfishness is not the selfishness we know, she states that Being Selfish is to know take your time to know and love yourself because if One Doesn't know himself then how one can do understand and love others. Loving yourself give a depth to the character, one can emphatize, symphatize and can love others as well. It's like First you own the thing that you want to give others. So Love Yourself and then you can love others.

This quote shows the Ayn Rand's Perspective of loving yourself. It is really harder than we thought. Honestly, it is very easier to do something virtuous and good deed for others and feel happy for that as compared to think about yourself and improve yourself.



LABELS: AYN, DOHA, FIRST, FLY, FOUNTAINHEAD, KABIR, LOVE, RAND, SOUL, YOURSELF SHARE

Comments

Link: https://iambhavna.blogspot.com

Name: Bhavana Prajapati Class: FYIT Roll No: 66

Practical No:7

[Implementing coding practices in Python using PEP8.]

PEP8 is a style guide for python code.

PEP stands for Python Enhancement Proposal, and they describe and document the way python language evolves. It was written in 2001 by Guido van Rossum, Barry Warsaw, and Nick Coghlan. A PEP is a document that describes new features proposed for Python and documents aspects of Python, like design and style, for the community. They also provide a reference point (and a standard) for the pythonic way to write code It also has a lot of programming recommendations and useful tips on various topics, which aim to improve readability and reliability of your code.

PEP8 features:-

I. Plugin architecture: Adding new checks is easy.

2.Parseable output: Jump to error location in your editor.

3.Small: Just one Python file, requires only stdlib. You can use just the pep8.py file for this purpose.

Naming Conventions

1.Variable

2.Function

3.Class

4.Method

5.Constant

6.Module

7.Package`

Variable: A variable is created the moment you first assign a value to it

```
#Wrong Way to Initialize or assigning a name to a variable
#Name Should not start with a number
#Name should be intuitive and not too common.

1variable=2 #Variable name started with a number (Wrong Way)
print(1variable)
```

```
  File "<ipython-input-1-d1860915d72c>", line 5
    1variable=2
     ^
SyntaxError: invalid syntax
```

```
#Wrong Way to Initialize or assigning a name to a variable
#Name Should not start with a number
#Name should be intuitive and not too common.

first_name='Bhavana' #Variable name is self-explanatory and has a readibility, and it is seperated using underscores
print(x)
```

```
Bhavana
```

Function: A function is a block of code which only runs when it is called.

```
#Wrong Way to Initialize or assigning a name to a function
#Name Should not start with a number
#Name should be intuitive and not too common.

def ^function(): #Function name should not be started with a Number or special characters
    print("Not a correct way to represent a function name")

^function()
```

```
  File "<ipython-input-5-5f84f1733e34>", line 5
    def ^function():
        ^
SyntaxError: invalid syntax
```

```
#Wrong Way to Initialize or assigning a name to a function
#Name Should not start with a number
#Name should be intuitive and not too common.

def x(): #Function name is too generic and it can create a confusion in enterprise programming
    print("Function Name is too generic, you can use it but it is not recommended as it is not self-explanatory and intuitive")

x()
```

```
Function Name is too generic, you can use it but it is not recommended as it is not self-explanatory and intuitive
```

```
#Wrong Way to Initialize or assigning a name to a function
#Name Should not start with a number
#Name should be intuitive and not too common.

def display_function(): #Function name is self explanatory
    print("Function Name is self explanatory, name can be more intuitive in case of proper functionality")

display_function()
```

```
Function Name is self explanatory, name can be more intuitive in case of proper functionality
```

Class: class definitions begin with a class keyword.

```
#Wrong Way to Initialize or assigning a name to a class
#Name Should not start with a number
#Name should be intuitive and not too common.

1class x:
def display_function(): #Function name is self explanatory
    print("Function Name is self explanatory, name can be more intuitive in case of proper functionality")

display_function()
```

```
  File "<ipython-input-9-0547726683a1>", line 5
    1class x:
     ^
SyntaxError: invalid syntax
```

```
class Employee:
     def accept(self):
      print("Enter Id:")
      self.Id=int(input())
      print("Enter Name:")
      self.name= str(input())
     def display(self):
          print("ID: %d \nName: %s"%(self.Id,self.name))

emp=Employee()
emp.accept()
emp.display()
```

```
Enter Id:
66
Enter Name:
bhavana
ID: 66
Name: bhavana
```

Method: A Python method is a label that you can call on an object; it is a piece of code to execute on that object.

```
#Wrong Way to Initialize or assigning a name to a method
#Name Should not start with a number
#Name should be intuitive and not too common.


1class Method:
    def display(self):
        print("This is method function. ")

c = Method()
c.display()
```

```
  File "<ipython-input-26-3e88b14da450>", line 6
    1class Method:
     ^
SyntaxError: invalid syntax
```

```
#Wrong Way to Initialize or assigning a name to a class
#Name Should not start with a number
#Name should be intuitive and not too common


class Product:
    def __init__(self):
        self.prod_id = input("Enter the Product ID: ")
        self.prod_name = input("Enter the Product Name: ")
        self.total_no = int(input("Enter the total no. of Items Purchase: "))
        self.unit_price=float(input("Enter the unit Price: "))
    def display(self):
        print("Total Price of %d units of Product %s is: %0.2f" %(self.total_no,self.prod_name,self.total_no*self.unit_price))

p1=Product()
p1.display()
```

```
Enter the Product ID: A20134
Enter the Product Name: Choclate
Enter the total no. of Items Purchase: 7
Enter the unit Price: 75.50
Total Price of 7 units of Product Choclate is: 528.50
```

Constant: A constant is a type of variable whose value cannot be changed.

```
pi = 3.14                #pi is constant
radius=5
print("Area of circle: %0.2f" %(pi*radius*radius))
```

```
Area of circle: 78.50
```

Modules: Modules refer to a file containing Python statements and definitions.

```
# to import standard module math

import math
print("The value of pi is", math.pi)
```

```
The value of pi is 3.141592653589793
```

Packages: A package is basically a directory with Python files and a file with the name __init__ . py

Code layout:

WITHOUT SPACE:

These conventions lead to text that you can read easily, like this:
This would become increasingly hard to read.

For example have a look at the example below

```
howwillitlookifwedonothavethespace
```

WITH SPACE:

Now here, we will use space and write it in regular English language, so it will be very easy to read.

How will it look if we do not have the space

Maximum line length and line breaking:

PEP 8 guidelines suggest that each line of code (as well as comment lines) should be 79 characters wide or less. This is a common standard that is also used in other languages including R.

CORRECT:

```
# Perform some math
a = 1+2
b = 3+4
c = a+b

# Read in and plot some
precip_timeseries = pd.readcsv("precip-2019.csv")
precip_timeseries.plot()
```

#WRONG

```
#Perform some math and do some things
a=1+2
b=3+4
c=a+b
data=pd.readcsv("precip-2019.csv")
data.plot()
```

Should a line break Before or After a Binary Operator

Here, it's harder to see which variable is being added and which is subtracted.

```
# WRONG
Total = (Number 1+
         Number 2-
         Number 3)
```

You can immediately see which variable is being added or subtracted, as the operator is right next to the variable being operated on.
In the below Example

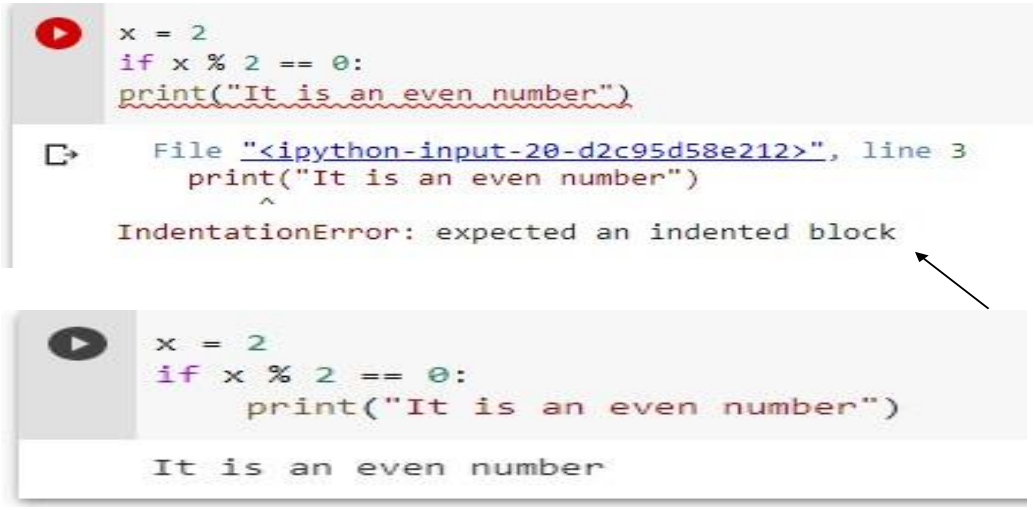
```
#CORRECT
Total = (Number 1
         + Number 2
         - Number 3)
```

Indentation

•Indentation is extremely important in Python.

•The Indentation level of lines of code in python determines how statements are grouped together.

1. Expected an indented block



2. Unexpected Indent:

```
x = 2
 if x % 2 == 0:
    print("It is an even number")
```

```
  File "<ipython-input-24-a296ed44a7f2>", line 2
    if x % 2 == 0:
    ^
IndentationError: unexpected indent
```

3. Unindent does not match any outer indentation level:

```
def greeting():
    print("Greetings of the day")
  return

greeting()
```

```
  File "<ipython-input-30-698032a46f85>", line 3
    return
         ^
IndentationError: unindent does not match any outer indentation level
```

```
def greeting():
    print("Greetings of the day")
    return

greeting()
```

```
Greetings of the day
```

Tabs vs. Spaces:

The key indentation rules laid out by PEP 8 are the following:

- Use 4 consecutive spaces to indicate indentation.
- Prefer spaces over tabs.

Indentation following line breaks:

- Add a comment after the final condition. Due to syntax highlighting in most editors, this will separate the conditions from the nested code:

Not Recommended

```
x = 5
if (x > 3 and
    x < 10):
    print(x)
```

Recommanded

```
x = 5
if (x > 3 and
    x < 10):
    # Both conditions satisfied
    print(x)
```

```
x = 5
if (x > 3 and
        x < 10):
    print(x)
```

Not Recommended

```
def function(
    arg_one, arg_two,
    arg_three, arg_four):
    return arg_one
```

Recommended

```
def function(
        arg_one, arg_two,
        arg_three, arg_four):
    return arg_one
```

Where to put the closing Braces:

Not Recommended

```
list_of_numbers = [ 1, 2, 3,
                    4, 5, 6,
                    7, 8, 9]
```

Recommended Method

```
list_of_numbers = [
    1, 2, 3,
    4, 5, 6,
    7, 8, 9
    ]
```

Method

```
list_of_numbers = [
    1, 2, 3,
    4, 5, 6,
    7, 8, 9
]
```

COMMENTS:

Comments are lines that exist in computer programs that are ignored by compilers and interpreters.
Comment begins with a hash mark (#) Generally, comment looks like this:
 # this a comment
Because comment does not execute ,when you will run program you will not see any indication of the comment there.

BLOCK COMMENTS:

Each line of block comments starts with a # and a single space
Paragraphs inside a block comment are separated by a line containing a single #.

Anti-pattern

```
#This comment needs a space
def print_name(self):
    print(self.name)
```

Best practice

```
# Comment is correct now
def print_name(self):
    print(self.name)
```

INLINE COMMENTS:

Inline comment should be separated by at least two spaces from the comment.
They should start with a # and a single space
Inline comments are unnecessary and in fact distracting if they state the obvious

Anti-pattern

```
def print_name(self):
    print(self.name)  #This comment needs a space
```

Best practice

```
def print_name(self):
    print(self.name)  # Comment is correct now
```

DOCSTRING COMMENTS:

A docstring is added as a comment string right below the function,module,or object.

Rules: A docstring is either a single line, or a multi-line comment In latter case, the first line is short description, and after the first line an empty line follows

This is a basic example of what it looks like:

```
def add(value1, value2):
    """Calculate the sum of value1 and value2."""
    return value1 + value2
```

In the Python interactive help system, the docstring is then made available via the `__doc__` attribute.

```
>>> print add.__doc__
Calculate the sum of value1 and value2.
```

Whitespace in Expressions and Statements

1.Whitespace Around Binary Operators

Surround the following binary operators with a single space on either side:

- Assignment operators (=, +=, -=, and so forth)
- Comparisons (==, !=, >, <. >=, <=) and (is, is not, in, not in)
- Booleans (and,or not)

Note: When = is used to assign a default value to a function argument, do not surround it with spaces.

Python

```
# Recommended
def function(default_parameter=5):
    # ...


# Not recommended
def function(default_parameter = 5):
    # ...
```

Adding space when there is more than one operator in a statement

```python
# Recommended
y = x**2 + 5
z = (x+y) * (x-y)

# Not Recommended
y = x ** 2 + 5
z = (x + y) * (x - y)
```

Adding space to if statements where there are multiple conditions.

```python
# Not recommended
if x > 5 and x % 2 == 0:
    print('x is larger than 5 and divisible by 2!')
```

```python
# Recommended
if x>5 and x%2==0:
    print('x is larger than 5 and divisible by 2!')
```

Note : Use the same amount of whitespace either side of the operator.

The following is not acceptable :

```python
# Definitely do not do this!
if x >5 and x% 2== 0:
    print('x is larger than 5 and divisible by 2!')
```

When to Avoid Adding Whitespace

Trailing space Immediately inside parentheses, brackets, or braces:

```python
# Recommended
my_list = [1, 2, 3]

# Not recommended
my_list = [ 1, 2, 3, ]
```

Before a comma, semicolon, or colon:

```python
x = 5
y = 6

# Recommended
print(x, y)

# Not recommended
print(x , y)
```

```python
# Recommended
list[3]

# Not recommended
list [3]
```

Before the open parenthesis that starts the argument list of a function call:

```python
def double(x):
    return x * 2

# Recommended
double(3)

# Not recommended
double (3)
```

Before the open bracket that starts an index or slice:
Between a trailing comma and a closing parenthesis:

```python
# Recommended
tuple = (1,)

# Not recommended
tuple = (1, )
```

To align assignment operators:

```python
# Recommended
var1 = 5
var2 = 6
some_long_var = 7

# Not recommended
var1          = 5
var2          = 6
some_long_var = 7
```

Programming Recommendations by PEP-8:

C)
```python
# Not recommended
my_list = []
if not len(my_list):
    print('List is empty!')
```

D)
```
# Recommended
my_list = []
if not my_list:
    print('List is empty!')
```

In the above program D is recommended over C by the PEP-8

When to Ignore PEP-8

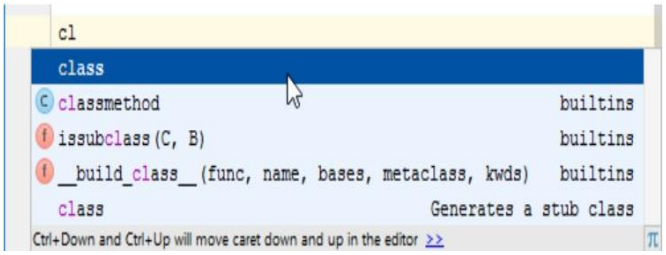
Answer: never.

Though, there are some guidelines in PEP-8 that are inconvenient in some instances:

- Complying with PEP-8
- Code surrounding
- Code compatibility

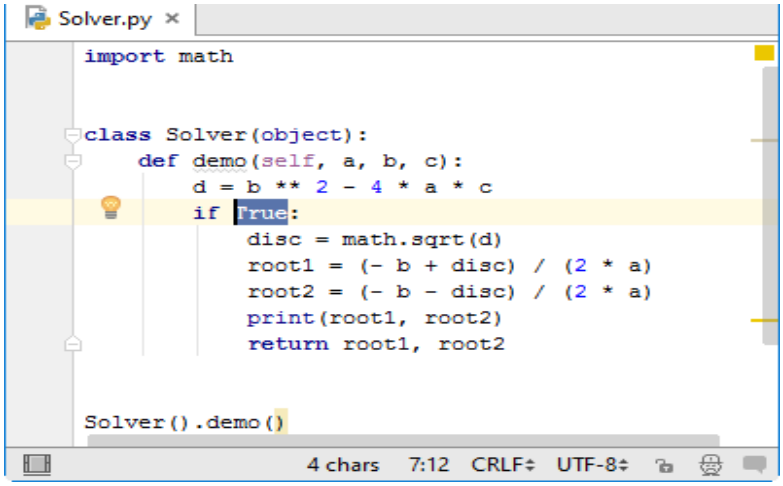
Tips and Tricks to Help Ensure Your Code Follows PEP 8

Highliting code style violations:



Generating Source code:

Select if option from the suggestion list. As you see, PyCharm automatically adds if True: and indents the selected lines:



Linter-python-pep8 package

This linter-python-pep8 plugin or Linter provides an interface to pep8. It will be used with files that have the Python syntax.

Installation:

Before using this plugin, you should make sure that pep8 is installed on your system. You can follow following instructions to install pep8:

Install python.
Install pep8 by typing the following in a terminal:

Black:

Black can be installed by running pip install black. It requires Python 3.6.0+ to run. Once Black is installed, you will have a new command-line tools called black available to you in your shell, and you're ready to start.

```
$ pip install black
```

Example: Write a program using PEP8 rules.

Answer:

```
# Names of class and methods are self-explantory as per the naming convention

class Rectangle:
    def Take_input(self):
        print("Enter the Length:")
        self.l=float(input())
        print("Enter the breadth:")
        self.b=float(input())

    def CalculateArea(self):
        area=self.l*self.b
        print("Area of rectangle is =%f" %(area))

    def CalculatePerimeter(self):
        perimeter=2*(self.l*self.b)
        print("Perimeter of rectangle is =%f" %(perimeter))

c= Rectangle()
c.Take_input()
c.CalculateArea()
c.CalculatePerimeter()
```

```
Enter the Length:
5
Enter the breadth:
7
Area of rectangle is =35.000000
Perimeter of rectangle is =70.000000
```

2.

Used Proper Whitespaces and line breaks so that it can be read easily.

```
data = []

n = int(input('Enter the number of elements: '))

for i in range (0, n):

    element = int(input('Enter the element: '))

    data.append(element)

print('The data set is: ', data)
```

```
Enter the number of elements: 7
Enter the element: 21
Enter the element: 66
Enter the element: 67
Enter the element: 110
Enter the element: 20
Enter the element: 65
Enter the element: 80
The data set is:  [21, 66, 67, 110, 20, 65, 80]
```

3.

Imported the module into a Python script using keyword *import*.

```
import fractions
print(fractions.Fraction('1.5'))
print(fractions.Fraction(5))
print(fractions.Fraction(4,30))
```

```
3/2
5
2/15
```

4.

Using the comments while developing a code/script can give the information about the functionality and documentation to other user who is reading the code.

```
# Python program to find simple interest

p = float(input("Enter the principle amount : "))
r = float(input("Enter the rate of interest : "))
t = float(input("Enter the time in the years: "))

si = (p*r*t)/100                    # calculating simple interest

print("Principle amount: ", p)
print("Interest rate   : ", r)
print("Time in years   : ", t)
print("Simple Interest : ", si)
```

```
Enter the principle amount : 25000
Enter the rate of interest : 3
Enter the time in the years: 12
Principle amount:  25000.0
Interest rate   :  3.0
Time in years   :  12.0
Simple Interest :  9000.0
```

5.

Make the Proper alignment of statements

```
students = [
 ("Bhavana", ["CompSci", "Physics"]),
 ("Vanshu", ["Maths", "CompSci", "Stats"]),
 ("Jess", ["CompSci", "Accounting", "Economics", "Management"]),
 ("Sarah", ["InfSys", "Accounting", "Economics", "CommLaw"]),
 ("Zuki", ["Sociology", "Economics", "Law", "Stats", "Music"])]
```

```
# Print all students with a count of their courses.
for name, subjects in students:
 print(name, "takes", len(subjects), "courses")
```

```
Bhavana takes 2 courses
Vanshu takes 3 courses
Jess takes 4 courses
Sarah takes 4 courses
Zuki takes 5 courses
```

PEP 8 STYLE GUIDE

GROUP No. 8

ANNE VERONICA	1
PRIYA GUPTA	20
SHUBH PATEL	60
BHAVANA PRAJAPATI	66
POOJA PRAMANIK	67
HEETA TALAVIYA	92
GLORY LITHIYAL	95
NITESH GUPTA	111
AMAN UPADHYAY	132
DEEPAK KESHRI	134
SURYASEN VISHWAKARMA	139

PEP8

PEP8 is a style guide for python code.

→ PEP stands for Python Enhancement Proposal, and they describe and document the way python language evolves.

→ It was written in 2001 by Guido van Rossum, Barry Warsaw, and Nick Coghlan.

→ A PEP is a document that describes new features proposed for Python and documents aspects of Python, like design and style, for the community.

→ They also provide a reference point (and a standard) for the pythonic way to write code

→ It also has a lot of programming recommendations and useful tips on various topics, which aim to improve readability and reliability of your code.

→ PEP8 features:-

I. Plugin architecture: Adding new checks is easy.

2. Parseable output: Jump to error location in your editor.

3. Small: Just one Python file, requires only stdlib. You can use just the pep8.py file for this purpose.

Naming Conventions

Naming Conventions:

1.Variable

2.Function

3.Class

4.Method

5.Constant

6.Module

7.Package

`

Variable: A variable is created the moment you first assign a value to it

```
#Wrong Way to Initialize or assigning a name to a variable
#Name Should not start with a number
#Name should be intuitive and not too common.

1variable=2 #Variable name started with a number (Wrong Way)
print(1variable)
```

```
  File "<ipython-input-1-d1860915d72c>", line 5
    1variable=2
     ^
SyntaxError: invalid syntax
```

```
#Wrong Way to Initialize or assigning a name to a variable
#Name Should not start with a number
#Name should be intuitive and not too common.

x='Bhavana' #Variable name is too common and not intuitive (Not a Good Way)
print(x)
```

```
Bhavana
```

```
#Wrong Way to Initialize or assigning a name to a variable
#Name Should not start with a number
#Name should be intuitive and not too common.

first_name='Bhavana' #Variable name is self-explanatory and has a readibility, and it is seperated using underscores
print(x)
```

```
Bhavana
```

Function: A function is a block of code which only runs when it is called.

```
#Wrong Way to Initialize or assigning a name to a function
#Name Should not start with a number
#Name should be intuitive and not too common.

def ^function(): #Function name should not be started with a Number or special characters
    print("Not a correct way to represent a function name")

^function()
```

```
  File "<ipython-input-5-5f84f1733e34>", line 5
    def ^function():
        ^
SyntaxError: invalid syntax
```

```
#Wrong Way to Initialize or assigning a name to a function
#Name Should not start with a number
#Name should be intuitive and not too common.

def x(): #Function name is too generic and it can create a confusion in enterprise programming
    print("Function Name is too generic, you can use it but it is not recommended as it is not self-explanatory and intuitive")

x()
```

```
Function Name is too generic, you can use it but it is not recommended as it is not self-explanatory and intuitive
```

```
#Wrong Way to Initialize or assigning a name to a function
#Name Should not start with a number
#Name should be intuitive and not too common.

def display_function(): #Function name is self explanatory
    print("Function Name is self explanatory, name can be more intuitive in case of proper functionality")

display_function()
```

```
Function Name is self explanatory, name can be more intuitive in case of proper functionality
```

Class: class definitions begin with a class keyword.

```
#Wrong Way to Initialize or assigning a name to a class
#Name Should not start with a number
#Name should be intuitive and not too common.

1class x:
def display_function(): #Function name is self explanatory
    print("Function Name is self explanatory, name can be more intuitive in case of proper functionality")

display_function()
```

```
  File "<ipython-input-9-0547726683a1>", line 5
    1class x:
     ^
SyntaxError: invalid syntax
```

```
class Employee:
     def accept(self):
      print("Enter Id:")
      self.Id=int(input())
      print("Enter Name:")
      self.name= str(input())
     def display(self):
          print("ID: %d \nName: %s"%(self.Id,self.name))

emp=Employee()
emp.accept()
emp.display()
```

```
Enter Id:
66
Enter Name:
bhavana
ID: 66
Name: bhavana
```

Method: A Python method is a label that you can call on an object; it is a piece of code to execute on that object.

```
#Wrong Way to Initialize or assigning a name to a method
#Name Should not start with a number
#Name should be intuitive and not too common.


1class Method:
    def display(self):
        print("This is method function. ")

c = Method()
c.display()
```

```
  File "<ipython-input-26-3e88b14da450>", line 6
    1class Method:
     ^
SyntaxError: invalid syntax
```

```
#Wrong Way to Initialize or assigning a name to a class
#Name Should not start with a number
#Name should be intuitive and not too common


class Product:
    def __init__(self):
        self.prod_id = input("Enter the Product ID: ")
        self.prod_name = input("Enter the Product Name: ")
        self.total_no = int(input("Enter the total no. of Items Purchase: "))
        self.unit_price=float(input("Enter the unit Price: "))
    def display(self):
        print("Total Price of %d units of Product %s is: %0.2f" %(self.total_no,self.prod_name,self.total_no*self.unit_price))

p1=Product()
p1.display()
```

```
Enter the Product ID: A20134
Enter the Product Name: Choclate
Enter the total no. of Items Purchase: 7
Enter the unit Price: 75.50
Total Price of 7 units of Product Choclate is: 528.50
```

Constant: A constant is a type of variable whose value cannot be changed.

```
pi = 3.14                    #pi is constant
radius=5
print("Area of circle: %0.2f" %(pi*radius*radius))
```

```
Area of circle: 78.50
```

Modules: Modules refer to a file containing Python statements and definitions.

```
# to import standard module math

import math
print("The value of pi is", math.pi)
```

```
The value of pi is 3.141592653589793
```

Packages: A package is basically a directory with Python files and a file with the name __init__ . py

Code layout

WITHOUT SPACE

These conventions lead to text that you can read easily, like this:

This would become increasingly hard to read. For example have a look at the example below

```
howwillitlookifwedonothavethespace
```

WITH SPACE

Now here, we will use space and write it in regular English language, so it will be very easy to read.

```
How will it look if we do not have the space
```

Maximum line length and line breaking

PEP 8 guidelines suggest that each line of code (as well as comment lines) should be 79 characters wide or less. This is a common standard that is also used in other languages including **R**.

CORRECT

```
# Perform some math
a = 1+2
b = 3+4
c = a+b

# Read in and plot some
precip_timeseries = pd.readcsv("precip-2019.csv")
precip_timeseries.plot()
```

#WRONG

```
#Perform some math and do some things
a=1+2
b=3+4
c=a+b
data=pd.readcsv("precip-2019.csv")
data.plot()
```

Should a line break Before or After a Binary Operator

Here, it's harder to see which variable is being added and which is subtracted.

```
# WRONG
Total = (Number 1+
         Number 2-
         Number 3)
```

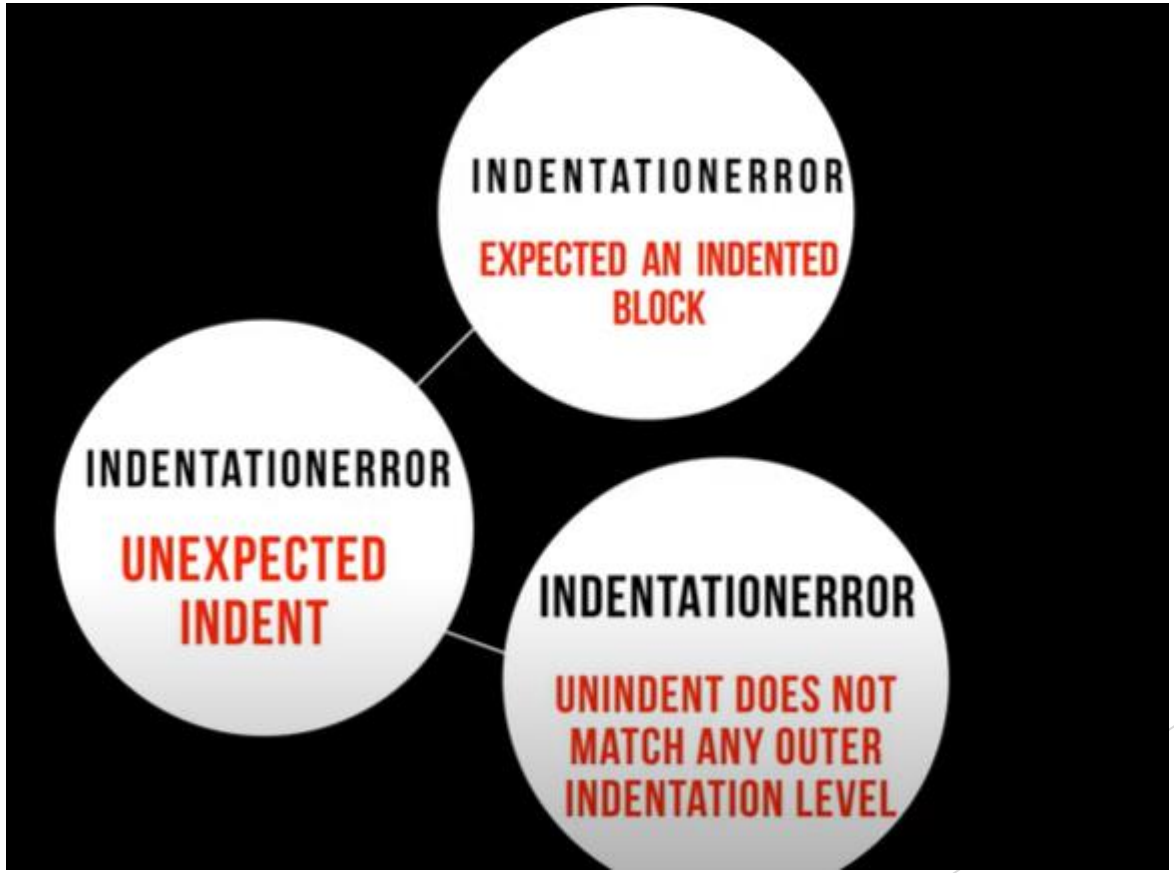
You can immediately see which variable is being added or subtracted, as the operator is right next to the variable being operated on.

In the below Example

```
#CORRECT
Total = (Number 1
         + Number 2
         - Number 3)
```

Indentation

- Indentation is extremely important in Python.
- The Indentation level of lines of code in python determines how statements are grouped together.



1. Expected an indented block

```
x = 2
if x % 2 == 0:
print("It is an even number")
```

```
  File "<ipython-input-20-d2c95d58e212>", line 3
    print("It is an even number")
        ^
IndentationError: expected an indented block
```

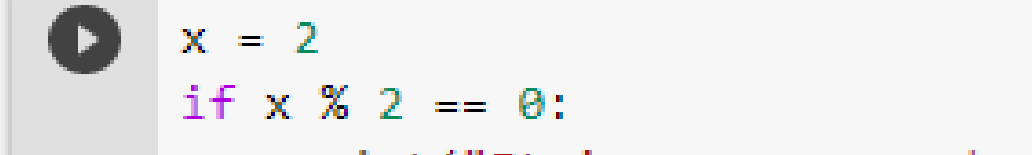
```
x = 2
if x % 2 == 0:
    print("It is an even number")
```

```
It is an even number
```

2. Unexpected Indent

```
x = 2
 if x % 2 == 0:
    print("It is an even number")
```

```
  File "<ipython-input-24-a296ed44a7f2>", line 2
    if x % 2 == 0:
    ^
IndentationError: unexpected indent
```



```
x = 2
if x % 2 == 0:
    print("It is an even number")
```

```
It is an even number
```

3. Unindent does not match any outer indentation level

```
def greeting():
    print("Greetings of the day")
  return

greeting()
```

```
  File "<ipython-input-30-698032a46f85>", line 3
    return
         ^
IndentationError: unindent does not match any outer indentation level
```

```
def greeting():
  print("Greetings of the day")
  return

greeting()
```

```
Greetings of the day
```

Tabs vs. Spaces

➢ Tabs vs. Spaces

The key indentation rules laid out by PEP 8 are the following:

- Use 4 consecutive spaces to indicate indentation.
- Prefer spaces over tabs.

➢ Indentation following line breaks

- Add a comment after the final condition. Due to syntax highlighting in most editors, this will separate the conditions from the nested code:

Not Recommended

```
x = 5
if (x > 3 and
    x < 10):
    print(x)
```

Recommended

```
x = 5
if (x > 3 and
    x < 10):
    # Both conditions satisfied
    print(x)
```

```
x = 5
if (x > 3 and
        x < 10):
    print(x)
```

 # Not Recommended

```
var = function(arg_one, arg_two,
    arg_three, arg_four)
```

 # Recommended

```
var = function(
    arg_one, arg_two,
    arg_three, arg_four)
```

 # Not Recommended

```
def function(
    arg_one, arg_two,
    arg_three, arg_four):
    return arg_one
```

 # Recommended

```
def function(
        arg_one, arg_two,
        arg_three, arg_four):
    return arg_one
```

➢ Where to put the closing Braces

 # Not Recommended

```
list_of_numbers = [ 1, 2, 3,
                    4, 5, 6,
                    7, 8, 9]
```

1. Method

```
list_of_numbers = [
    1, 2, 3,
    4, 5, 6,
    7, 8, 9
    ]
```

2. Method

```
list_of_numbers = [
    1, 2, 3,
    4, 5, 6,
    7, 8, 9
]
```

COMMENTS:

Comments are lines that exist in computer programs that are ignored by compilers and interpreters.

Comment begins with a hash mark (#)

Generally, comment looks like this:

 # this a comment

Because comment does not execute ,when you will run program you will not see any indication of the comment there.

BLOCK COMMENTS:

Each line of block comments starts with a # and a single space

Paragraphs inside a block comment are separated by a line containing a single #.

Anti-pattern

```
#This comment needs a space
def print_name(self):
    print(self.name)
```

Best practice

```
# Comment is correct now
def print_name(self):
    print(self.name)
```

INLINE COMMENTS:

Inline comment should be separated by at least two spaces from the comment.

They should start with a # and a single space

Inline comments are unnecessary and in fact distracting if they state the obvious

Anti-pattern

```
def print_name(self):
    print(self.name)  #This comment needs a space
```

Best practice

```
def print_name(self):
    print(self.name)  # Comment is correct now
```

DOCSTRING COMMENTS:

A docstring is added as a comment string right below the function,module,or object

RULES:

A docstring is either a single line, or a multi-line comment
In latter case, the first line is short description, and after the first line an empty line follows

This is a basic example of what it looks like:

```
def add(value1, value2):
    """Calculate the sum of value1 and value2."""
    return value1 + value2
```

In the Python interactive help system, the docstring is then made available via the `__doc__` attribute.

```
>>> print add.__doc__
Calculate the sum of value1 and value2.
```

Inline Comments vs Block Comments

Inline comments look like this

```
x = x + 1                 # Compensate for border
```

While block comments look like this

```
# Compensate for border.  These comments
# often cover multiple lines.
x = x + 1
```

Whitespace in Expressions and Statements

1) Whitespace Around Binary Operators

Surround the following binary operators with a single space on either side:

- Assignment operators (=, +=, -=, and so forth)
- Comparisons (==, !=, >, <. >=, <=) and (is, is not, in, not in)
- Booleans (and, not, or)

Note: When = is used to assign a default value to a function argument, do not surround it with spaces.

Python

```
# Recommended
def function(default_parameter=5):
    # ...


# Not recommended
def function(default_parameter = 5):
    # ...
```

- Adding space when there is more than one operator in a statement.

```python
# Recommended
y = x**2 + 5
z = (x+y) * (x-y)

# Not Recommended
y = x ** 2 + 5
z = (x + y) * (x - y)
```

- Adding space to if statements where there are multiple conditions.

```python
# Not recommended
if x > 5 and x % 2 == 0:
    print('x is larger than 5 and divisible by 2!')
```

```python
# Recommended
if x>5 and x%2==0:
    print('x is larger than 5 and divisible by 2!')
```

Note : Use the same amount of whitespace either side of the operator.

The following is not acceptable :

Python

```
# Definitely do not do this!
if x >5 and x% 2== 0:
    print('x is larger than 5 and divisible by 2!')
```

When to Avoid Adding Whitespace

- Trailing space
- Immediately inside parentheses, brackets, or braces:

```python
# Recommended
my_list = [1, 2, 3]

# Not recommended
my_list = [ 1, 2, 3, ]
```

- Before a comma, semicolon, or colon:

```python
x = 5
y = 6

# Recommended
print(x, y)

# Not recommended
print(x , y)
```

Before the open parenthesis that starts the argument list of a function call:

```python
def double(x):
    return x * 2

# Recommended
double(3)

# Not recommended
double (3)
```

Before the open bracket that starts an index or slice:

```python
# Recommended
list[3]

# Not recommended
list [3]
```

- Between a trailing comma and a closing parenthesis:

Python

```python
# Recommended
tuple = (1,)

# Not recommended
tuple = (1, )
```

- To align assignment operators:

Python

```python
# Recommended
var1 = 5
var2 = 6
some_long_var = 7

# Not recommended
var1          = 5
var2          = 6
some_long_var = 7
```

Programming Recommendations

- ***Two Programming Recom endations by PEP-8***

A)
```
# Not recommended
my_bool = 6 > 5
if my_bool == True:
    return '6 is bigger than 5'
```

B)
```
# Recommended
if my_bool:
    return '6 is bigger than 5'
```

In the above program B is recommended over A by the PEP-8

C)
```
# Not recommended
my_list = []
if not len(my_list):
    print('List is empty!')
```

D) # Recommended

```
my_list = []
if not my_list:
    print('List is empty!')
```

In the above program D is recommended over C by the PEP-8

Q. When to Ignore PEP-8

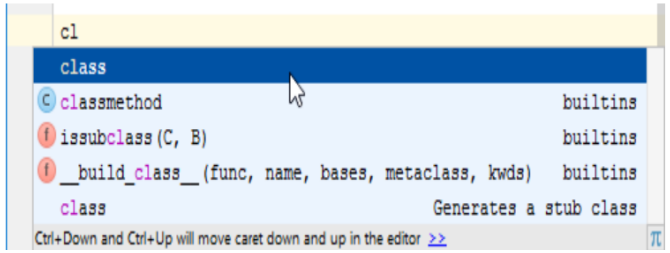
ANSWER: NEVER

Though, there are some guidelines in PEP-8 that are inconvenient in some instances:

- Complying with PEP-8
- Code surrounding
- Code compatibility

Tips and Tricks to Help Ensure Your Code Follows PEP 8

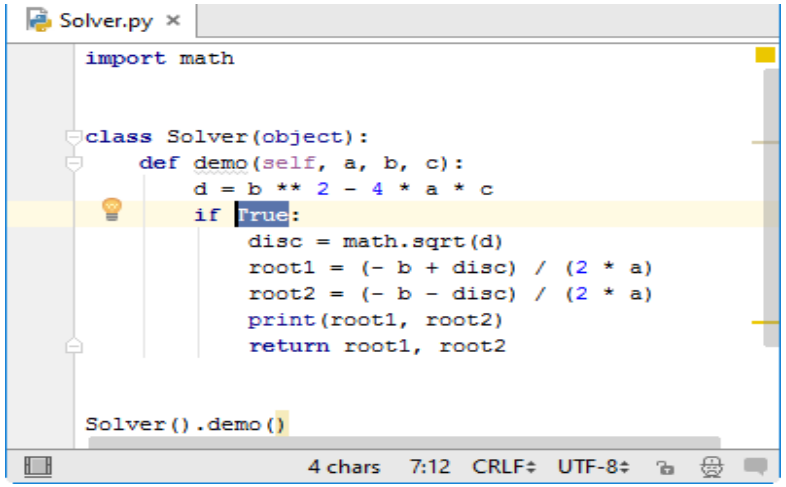
Highliting code style violations:



(Refer to Code Completion page of the product documentation for details.)

Generating Source code:

Select `if` option from the suggestion list. As you see, PyCharm automatically adds `if True:` and indents the selected lines:



Linter-python-pep8 package

This linter-python-pep8 plugin or Linter provides an interface to pep8. It will be used with files that have the Python syntax.

Installation:

Before using this plugin, you should make sure that pep8 is installed on your system. You can follow following instructions to install pep8:

Install python.
Install pep8 by typing the following in a terminal:

```
pip install pep8
```

Black

Black can be installed by running pip install black. It requires Python 3.6.0+ to run. Once Black is installed, you will have a new command-line tools called black available to you in your shell, and you're ready to start.

```
$ pip install black
```

Format a Single File:

Let's look at this simple example: here are my two python functions in my python file called sample_code.py.

```
def add(a,       b):
    answer     =   a    +    b

    return answer


def sub(c    ,
 d):

    answer = c    -   d

    return answer
```

You can use black sample_code.py in the terminal to change the format. After running Black, you will see the following output:

```
reformatted sample_code.py
All done! ✨🍰✨
1 file reformatted.
```

Then you open sample_code.py to see formatted python code:

```python
def add(a, b):
    answer = a + b

    return answer


def sub(c, d):

    answer = c - d

    return answer
```

Example of code and layout.

With space and without space.

WITH SPACE.

*ex- MY NAME IS NITESH

WITHOUT SPACE.

*ex- MYNAMEISNITESH

Maximum line length and line breaking.

 # Recommended

- Ex-

```Python
def function(arg_one, arg_two,
             arg_three, arg_four):
    return arg_one
```

 # Not Recommended

- Ex-

```Python
from mypkg import example1, \
    example2, example3
```

Should a line break Before or After A Binary Operator.

- Ex-

Python

```
# Recommended
total = (first_variable
         + second_variable
         - third_variable)
```

- Ex-

Python

```
# Not Recommended
total = (first_variable +
         second_variable -
         third_variable)
```

Example of comments.

block comment.

Anti-partten.

Example

```
#This is a comment
print("Hello, World!")
```

Best-pratice.

Example

```
#This is a comment
#written in
#more than just one line
print("Hello, World!")
```

Inline comments.

Anti-partten.

```Python
x = 5  # This is an inline comment
```

Best practice.

```Python
x = 'John Smith'  # Student Name
```

Documentation string comment.

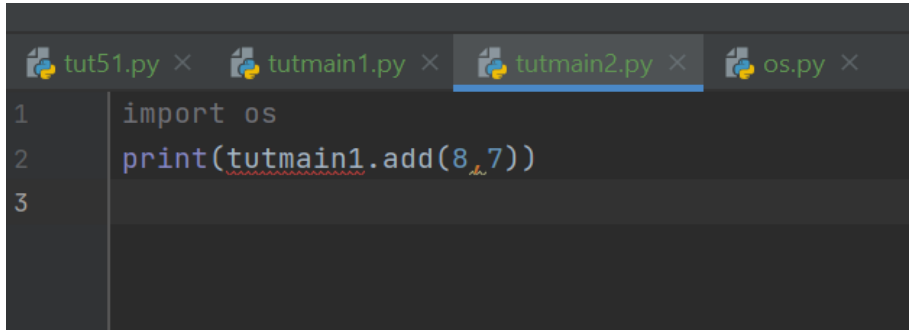
```
"""Return a foobang

Optional plotz says to
frobnicate the bizbaz first.
"""
```

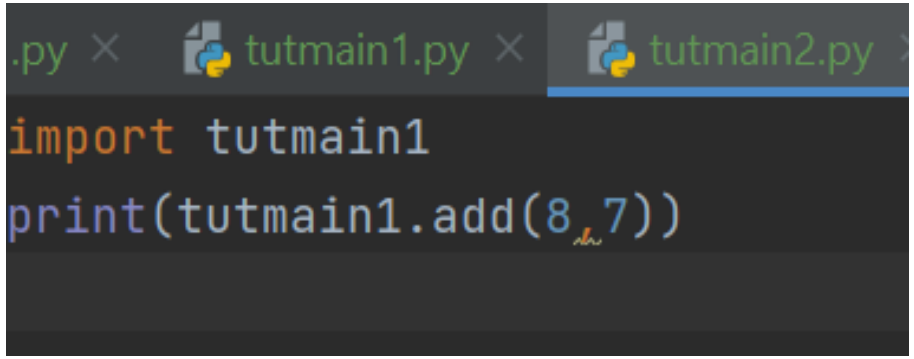
EXAMPLE OF NAMING CONVENTION

NAMING MODULE WITH HELP OF PEP8

```
# Not recommended
```




```
# Recommended
```



NAMING VERIABLE WITH HELP OF PEP8

Veriable:

```
>>> # Not recommended
>>> x = 'John Smith'
>>> y, z = x.split()
>>> print(z, y, sep=', ')
'Smith, John'

>>>
```

```
>>> # Recommended
>>> name = 'John Smith'
>>> first_name, last_name = name.split()
>>> print(last_name, first_name, sep=', ')
'Smith, John'
```

EXAMPLES OF INDENTATION

❖ METHODS OF WHERE TO PUT CLOSING BRACES:-

 # Recommended

```
list_of_flowers = [
    rose,sunflower,
    marigold,jasmine,
    tulips,lavender
]
```

```
list_of_flowers = [
    rose,sunflower,
    marigold,jasmine,
    tulips,lavender
    ]
```

❖ Methods for following line breaks

 # Recommended

```
x=10
if (x < 15 and
    x > 5):
     #Both the conditions satisfied
    print(x)
```

```
x=10
if (x < 15 and
        x > 5):
    print(x)
```

EXAMPLE OF WHITESPACING

1. **Adding space when there is more than one operator in a statement**.

```
#recommended
b = a**8 + 5
c = (a+b) * (a-b)
```

```
#not recommended
b = a ** 8 + 5
c = (a + b) * (a - b)
```

1. **Adding space to if statemennts where there are multiple conditions.**

```
#Recommended
if x>8 and x%2== 0:
  print('x is larger than 8 and divisible by 2!')
```

```
#not Recommended
if x > 8 and x % 2 == 0:
  print('x is larger than 8 and divisible by 2!')
```

THANK YOU