UNIVERSITY OF SCIENCE,
HO CHI MINH

Introduction to Git & Github and Python Object-Oriented Programming

TECHNICAL REPORT
W03

submitted for the Solo Project in Semester 2 – First Year

IT DEPARTMENT

Computer Science

by

Phan Tuan Kiet

Full name: Phan Tuan Kiet
Class: 23APCS2
ID: 23125062
Task achieved: 07/07

Lecturer:
Professor Dinh Ba Tien (PhD)
Teaching Assistants:
Mr. Ho Tuan Thanh (MSc)
Mr. Nguyen Le Hoang Dung (MSc)

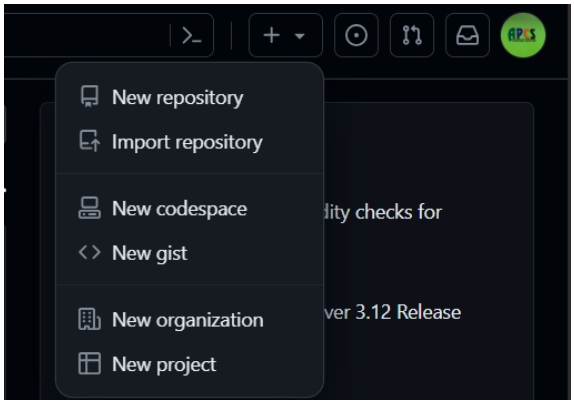
2024

Contents

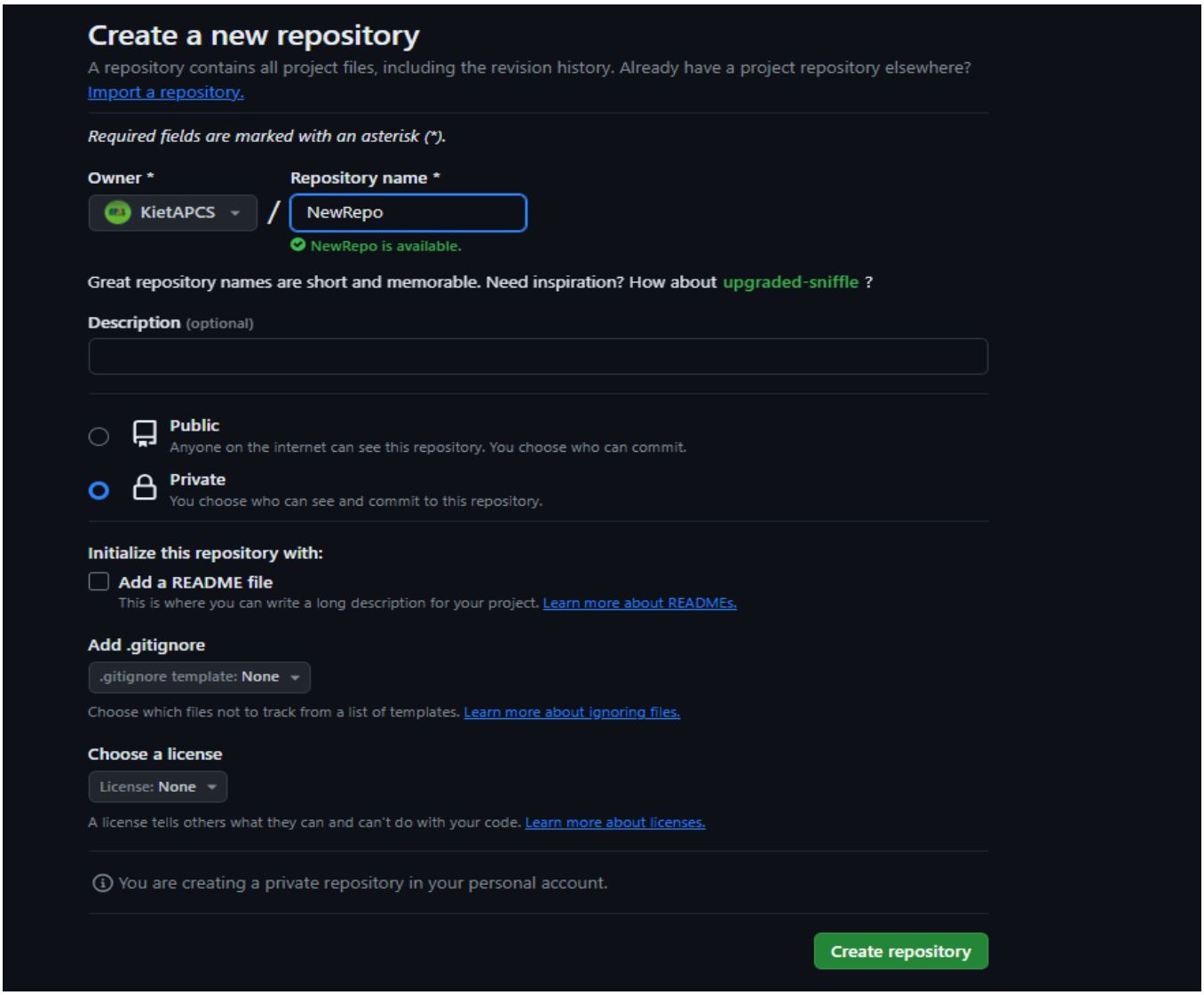
I) Create a Github Private Repository.

- Step 1: Sign up for a GitHub account and log in.

- Step 2: Click on the plus sign button in the top-right corner of the screen, then select New Repository.



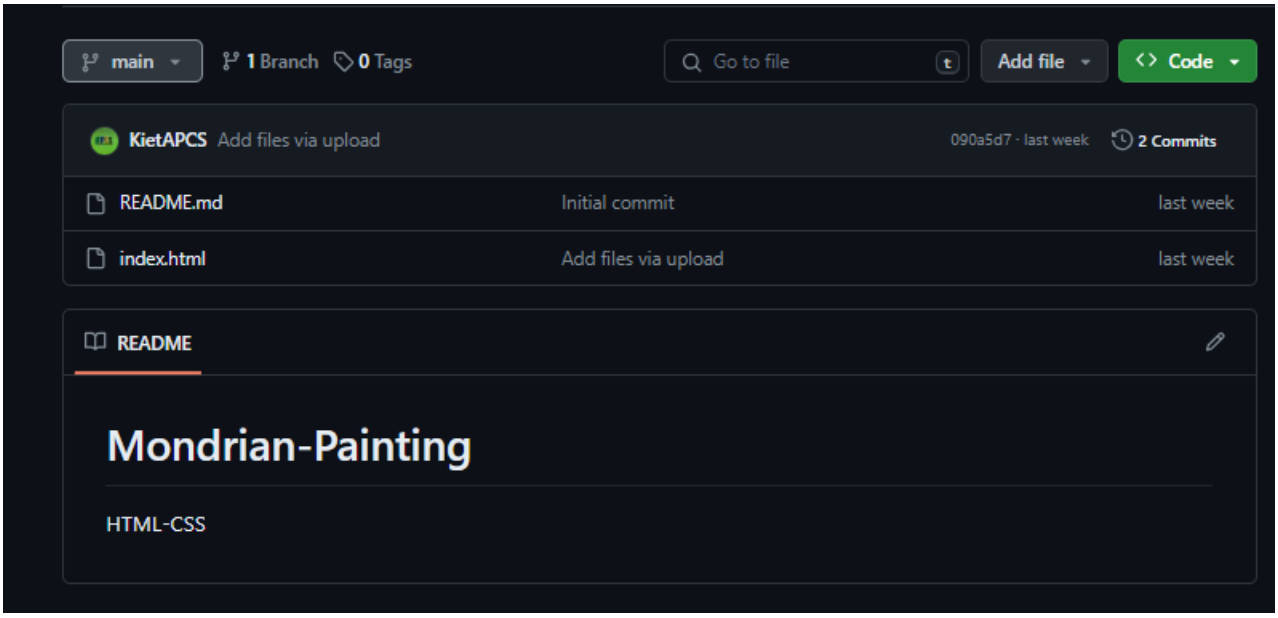
- Step 3: Enter the repository name, select Private, then click on the Create repository button → Successful creation.



II) Clone Github to the computer.

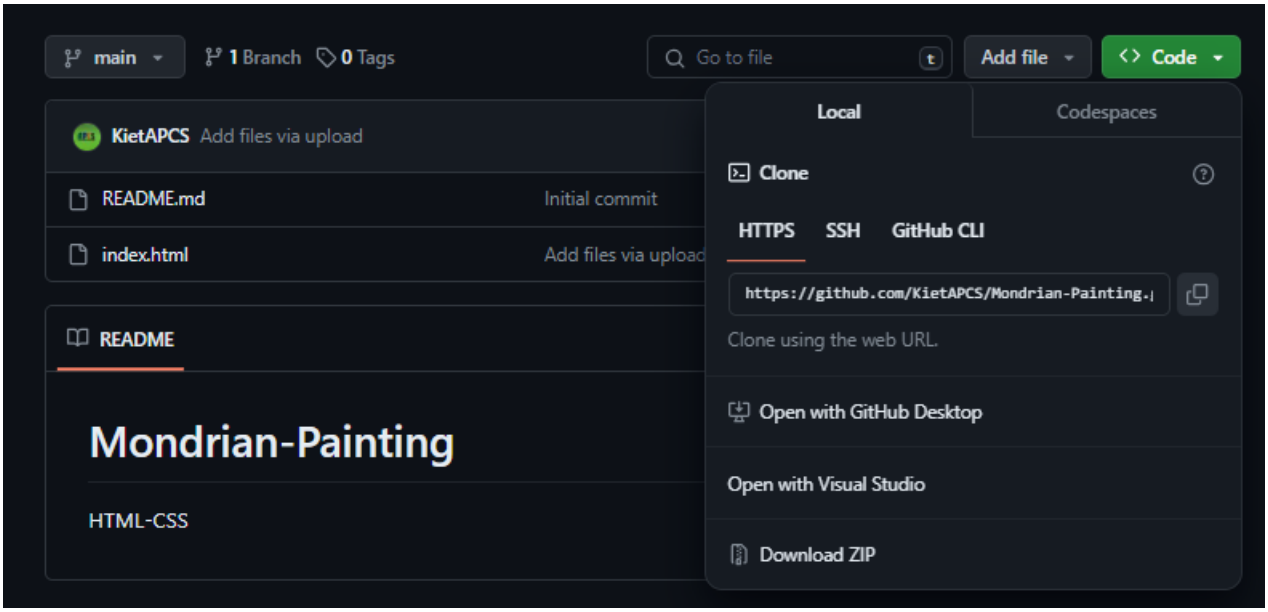
A) Clone public repo

- Step 1: Access the project on GitHub that you want to clone.



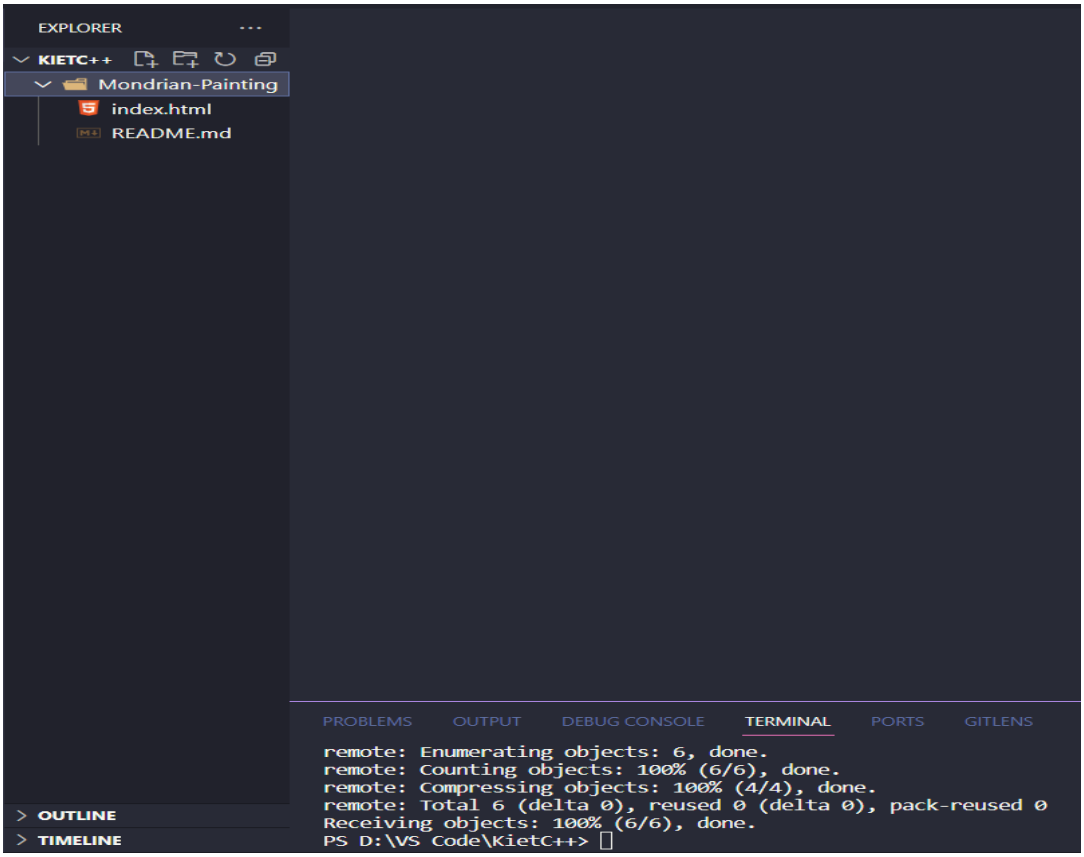
- Step 2: Open the IDE where you want to store the code from GitHub (e.g., VS Code).

- Step 3: Copy the project link from Github (Under the "local" section: HTTPS).



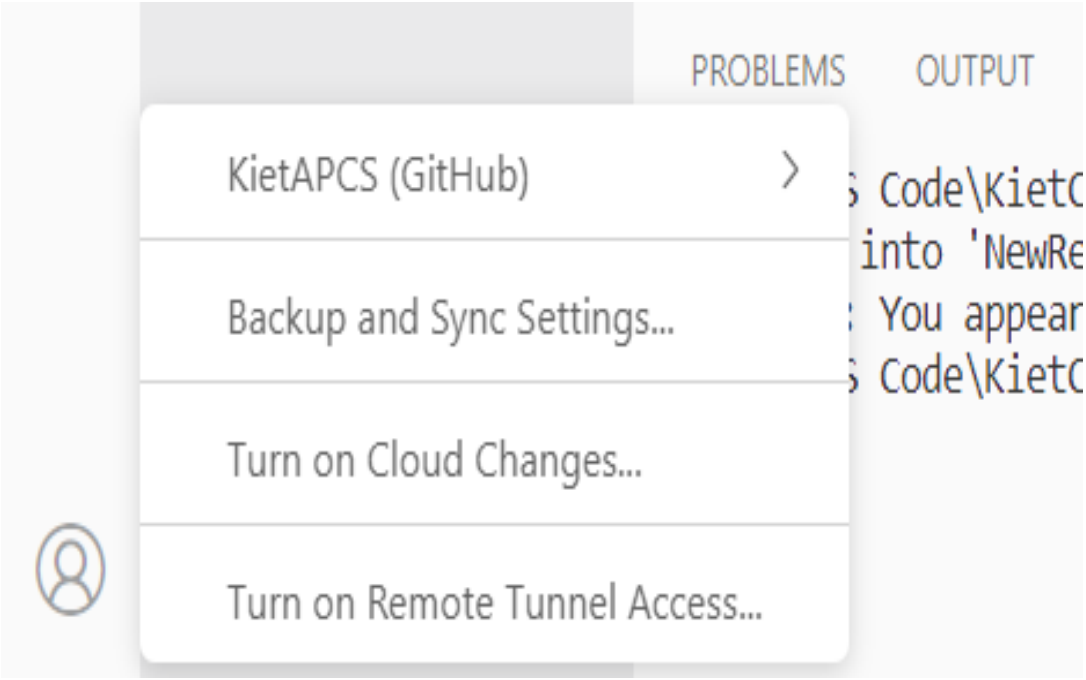
- Step 4: Open the terminal of the IDE and enter the command: git clone <url>

- Step 5: Completion. Begin editing the code.



B) Clone private repo

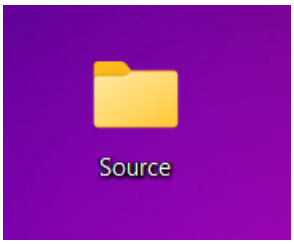
- Step 1: Log in to your GitHub account within the IDE.



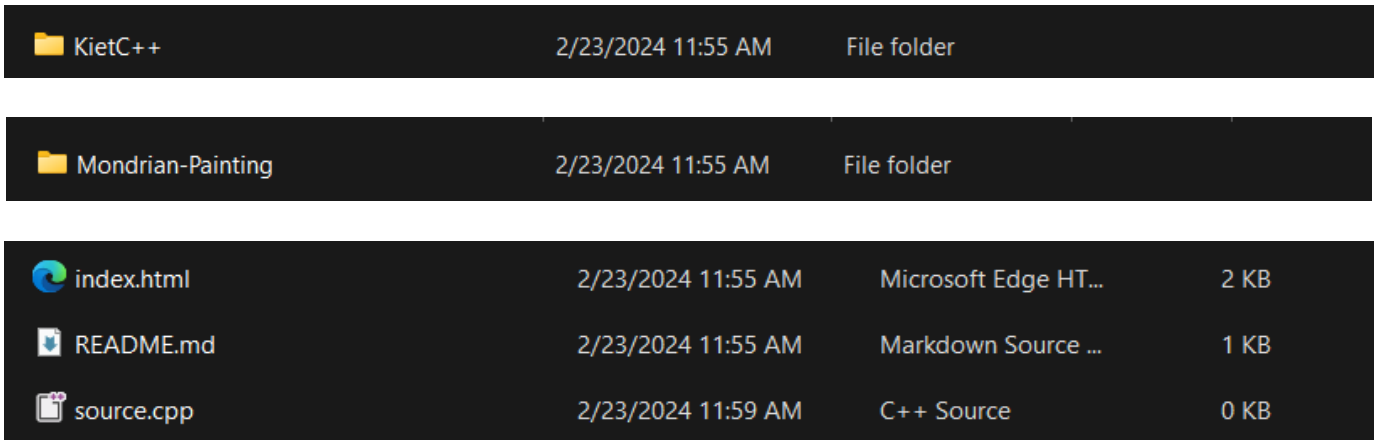
- Step 2: Follow the same steps as in section A.

III) Create a folder to contain the source code.

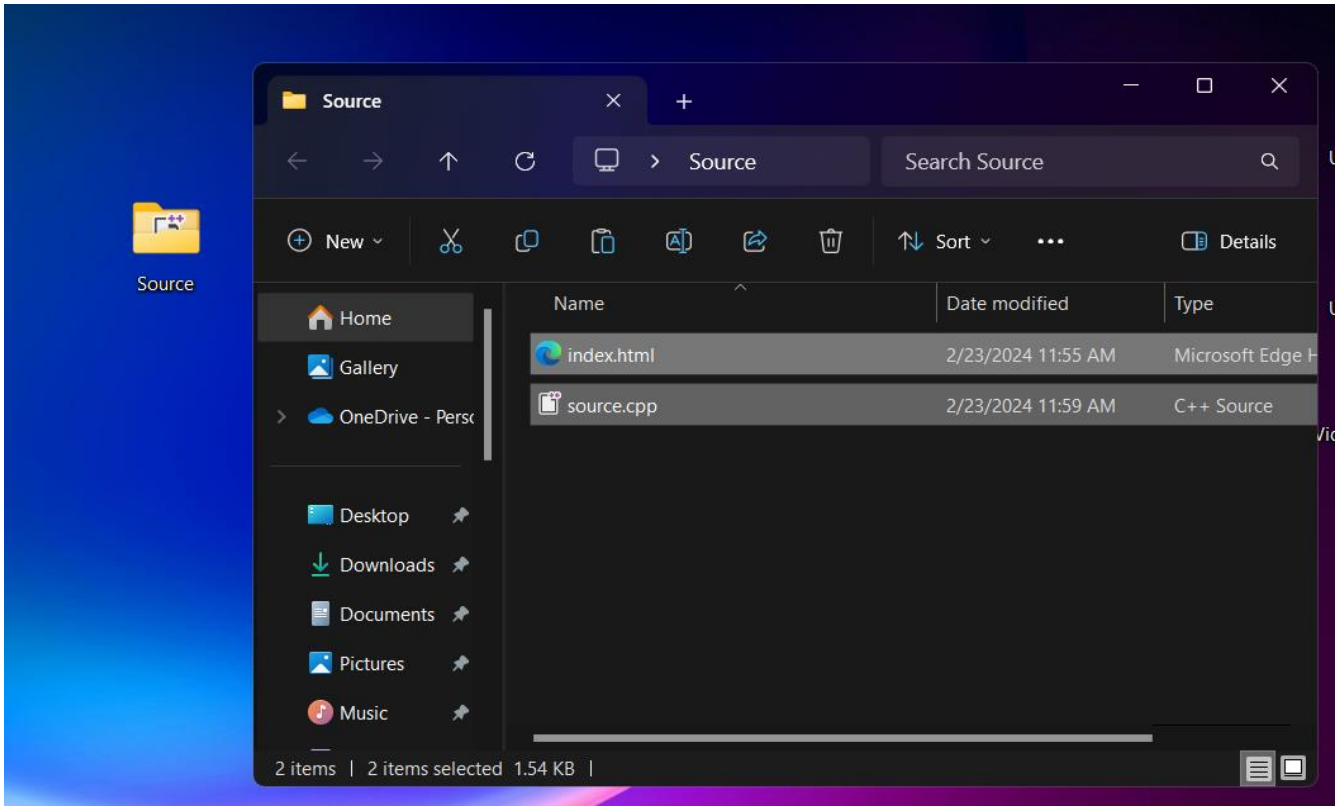
- Step 1: Create a folder.



- Step 2: Access the project from which you want to get the source code.



- Step 3: Copy the source code into the folder containing the source.



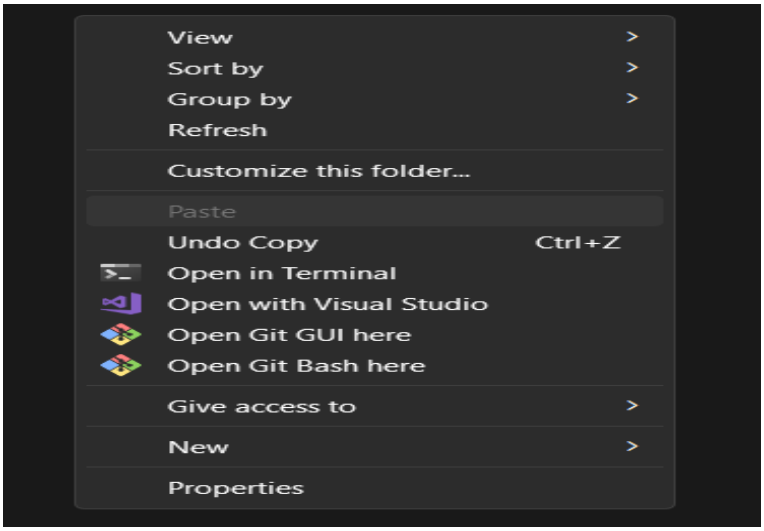
IV) Create a .gitignore file.

- Step 1: Navigate to the project you want to control and create a .git using the command: git init

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    GITLENS

PS D:\VS Code\KietC++\Mondrian-Painting> git init
Initialized empty Git repository in D:/VS Code/KietC++/Mondrian-Painting/.git/
PS D:\VS Code\KietC++\Mondrian-Painting>
```

- Step 2: Access the project and select "Open Git Bash Here" (Git must be downloaded to the machine beforehand).



- Step 3: Create the .gitignore file using the command line: touch .gitignore.

V) Demo how to use git add, git commit, git push, git pull.

a) Git add

- git add is a command used to stage changes, preparing them for confirmation before committing them to the repository.

*** Demo:**

- Use the command: git status to check for any changes that have not been added to the Staging Area.

```
PS D:\VS Code\KietC++\Mondrian-Painting> git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        README.md
        index.html
        source.cpp

nothing added to commit but untracked files present (use "git add" to track)
PS D:\VS Code\KietC++\Mondrian-Painting>
```

- If we want to add each file separately: git add <file name>

```
PS D:\VS Code\KietC++\Mondrian-Painting> git add README.md
```

- Upon rechecking, it appears that only those other files are yet to be added, while the README.md file has already been added.

```
PS D:\VS Code\KietC++\Mondrian-Painting> git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   README.md

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        index.html
        source.cpp

PS D:\VS Code\KietC++\Mondrian-Painting>
```

- To add all files into the Staging Area, use this command: git add.

```
PS D:\VS Code\KietC++\Mondrian-Painting> git add .
```

- Upon rechecking using the git status command, it is evident that all files have been added.

```
PS D:\VS Code\KietC++\Mondrian-Painting> git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   .gitignore
        new file:   README.md
        new file:   index.html
        new file:   source.cpp
```

b) Git commit

- The git commit command confirms the changes made to these files and packages them into the local repository.

***Demo:**

Use this command: git commit -m <message>

```
PS D:\VS Code\KietC++\Mondrian-Painting> git commit -m "Add three file and 1 .gitignore file"
[master (root-commit) e51750d] Add three file and 1 .gitignore file
 4 files changed, 89 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 README.md
 create mode 100644 index.html
 create mode 100644 source.cpp
PS D:\VS Code\KietC++\Mondrian-Painting>
```

Check again by this command: git status

```
PS D:\VS Code\KietC++\Mondrian-Painting> git status
On branch master
nothing to commit, working tree clean
PS D:\VS Code\KietC++\Mondrian-Painting>
```

c) Git push

- The git push command is used to push the changes from the local repository to the global repository.

- Example: From your local computer to Github Repository (global).

Demo:

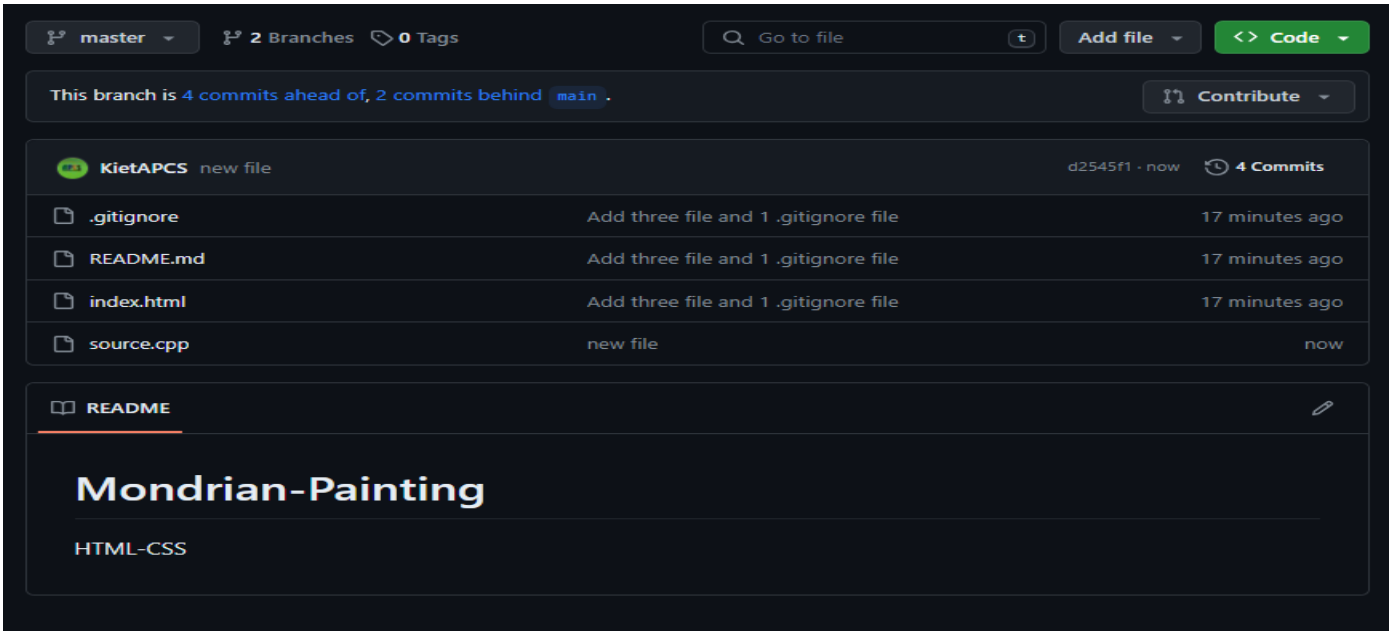
+ Step 1: Use this command: git remote add main <url> to link local with global.

```
PS D:\VS Code\KietC++\Mondrian-Painting> git remote add main https://github.com/KietAPCS/Mondrian-Painting.git
```

+ Step 2: Use this command: git push -u main master

```
PS D:\VS Code\KietC++\Mondrian-Painting> git push -u main master
```

+ Step 3: Check the result in Github.



d) Git pull

- The git pull command is used to update the changes from the global repository to your local repository.

***Demo**

+ Step 1: Change the file in GitHub

```
#include <iostream>

using namespace std;

int main() {

  cout << "Hello World";

  return 0;

}
```

+ Step 2: Use the command git pull to update changes from the global repository.

```
int main() {

    cout << "Hello World";

    return 0;

}
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    GITLENS

PS D:\VS Code\KietC++\Mondrian-Painting> git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 1017 bytes | 113.00 KiB/s, done.
From https://github.com/KietAPCS/Mondrian-Painting
   d2545f1..1a1f35f  master     -> main/master
Updating d2545f1..1a1f35f
Fast-forward
 source.cpp | 6 ++++++
 1 file changed, 6 insertions(+)
PS D:\VS Code\KietC++\Mondrian-Painting>
```

VI) How to create a project with multiple files in Python.

+ Step 1: Create a project to contain the Python files.

+ Step 2: Create the necessary .py files.

+ Step 3: Link the files together using the command: from <file name> import <function or class>.

***Demo:**

```
EXPLORER
KIETPYT...
function.py
main.py

main.py    function.py
function.py > ...
class Dog:

    def __init__(self, name):
        self.name = name

    def speak(self):
        print("My name is {}".format(self.name))
```

```python
from function import Dog
```

VII) How to write and link source code across multiple files in Python.

- For files containing classes, we link them to the main file using the command:

from <file name> import <class name>

```python
from function import Dog
```

- For other files, typically those containing functions, you only need to write the command: import <file name>.

```python
import math

def checkPrime(n):
    if (n <= 1): return False

    for i in range (2, (int)(math.sqrt(n)) + 1):
        if (n % i == 0):
            return False

    return True
```

```python
from classFile import Dog
import function

n = (int)(input("Enter n: "))

if (function.checkPrime(n) == True):
    print(n, "is a prime number")
else:
    print (n, "is not a prime number")
```

* Note:

- When calling functions related to other files or libraries, you must follow the syntax:

<library/file name>.<function name>.

VIII) How to create classes, properties, and methods.

a) Create a class

Class Definition Syntax:

```
class ClassName:

  # Statement-1

  …

  # Statement-N
```

Demo:

```python
class Dog:
    pass
```

b) Create properties

- Use the command: def __init__(self, para1, para2):

self.properties1 = para1

self.properties2 = para2

→ Properties 1 and 2 are two properties of an object within their class.

+ Alternatively, global properties can be declared by placing them outside the def __init__() method.

- For example:

+ Let's say we need a class to manage students, each student having two properties: name and age. We need to pass two parameters, the name and age of that student.

+ Both students share a common property: they attend the University of Natural Sciences.

***Demo:**

```python
class Student:

    school = "VNU university"

    def __init__(self, name, age):
        self.name = name
        self.age = age
```

```python
from classFile import Student
import function

st1 = Student("Kiet", 19)
st2 = Student("Hai", 19)

print(st1.__dict__)
print(st2.__dict__)

print(st1.school)
print(st2.school)
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\VS Code\KietPython> & C:/Users/USER/AppData/Local/Programs/Python/Python312/python.exe "d:/VS Code/KietPython/main.py"
{'name': 'Kiet', 'age': 19}
{'name': 'Hai', 'age': 19}
VNU university
VNU university
PS D:\VS Code\KietPython>
```

c) Create methods

- A method is like an action performed by an object within a class. When you call that method, the object in that class will execute that action.

- Example:

+ Both students have a method called "introduction". When calling this method, both students will introduce themselves sequentially: their name, age, and school.

+ Note: In functions, it's recommended to pass at least one parameter, "self", which points to the object belonging to that class.

*** Demo:**

```python
class Student:
    school = "VNU University"

    def __init__(self, name, age):
        self.name = name
        self.age = age

    def introduction(self):
        print("Hello, my name is {}. I am {} years old and currently studying at {}"
              .format(self.name, self.age, self.school))
```

main.py > ...

```python
from classFile import Student
import function

st1 = Student("Kiet", 19)
st2 = Student("Hai", 19)

st1.introduction()
st2.introduction()
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\VS Code\KietPython> & C:/Users/USER/AppData/Local/Programs/Python/Python312/python.exe "d:/VS Code/KietPython/main.py"
Hello, my name is Kiet. I am 19 years old and currently studying at VNU University
Hello, my name is Hai. I am 19 years old and currently studying at VNU University
PS D:\VS Code\KietPython>
```