

Práctica de construcción de software

TEMA: Programación por pares

III. Actividades

Usted y su par implementaran un proyecto en Python para cambiar un numero natural ($z \geq 0$) a su equivalente en base b ($2 \leq b \leq 16$) utilizando la programación por pares, el flujo de trabajo Gitflow, el desarrollo guiado por pruebas y el lenguaje de programación Python.

Ejemplo, si el usuario introduce $z=984$ y $b=16$, entonces el programa mostrará la leyenda "3D8 representa al entero 984 en base 16".

Las tareas siguientes son para que pueda trabajar junto con su socio. Durante el ejercicio, asegúrese que usted cambie los roles a menudo; se sugiere cambiar cada diez minutos.

3.1. Diseñe los casos de prueba para el caso presentado.

1. Caso básico con valores pequeños:

$z = 10, b = 2$

$z = 15, b = 8$

$z = 16, b = 16$

2. Caso de número natural grande:

$z = 984, b = 16$

3. Caso con el límite inferior de la base:

$z = 5, b = 2$

4. Caso con el límite superior de la base:

$z = 1000, b = 16$

5. Caso con un número natural igual a cero:

$z = 0, b = 8$

6. Caso con el límite superior de z :

$z = 9999, b = 2$

3.2. Si es necesario revise el marco teórico sobre programación por pares.

```
def convertir_a_base(z, b):  
  
    if z == 0:  
  
        return '0'
```

```
digitos = "0123456789ABCDEF"

resultado = ""

while z > 0:

    resultado = digitos[z % b] + resultado

    z //= b

return resultado

def main():

    z = int(input("Introduce el número natural z (z >= 0): "))

    b = int(input("Introduce la base b (2 <= b <= 16): "))

    if not (2 <= b <= 16):

        print("La base debe estar entre 2 y 16.")

        return

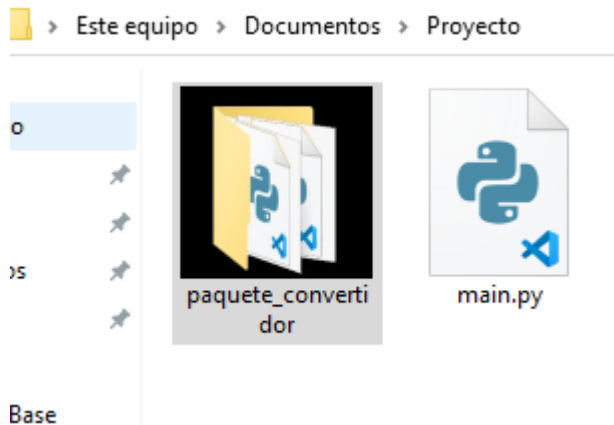
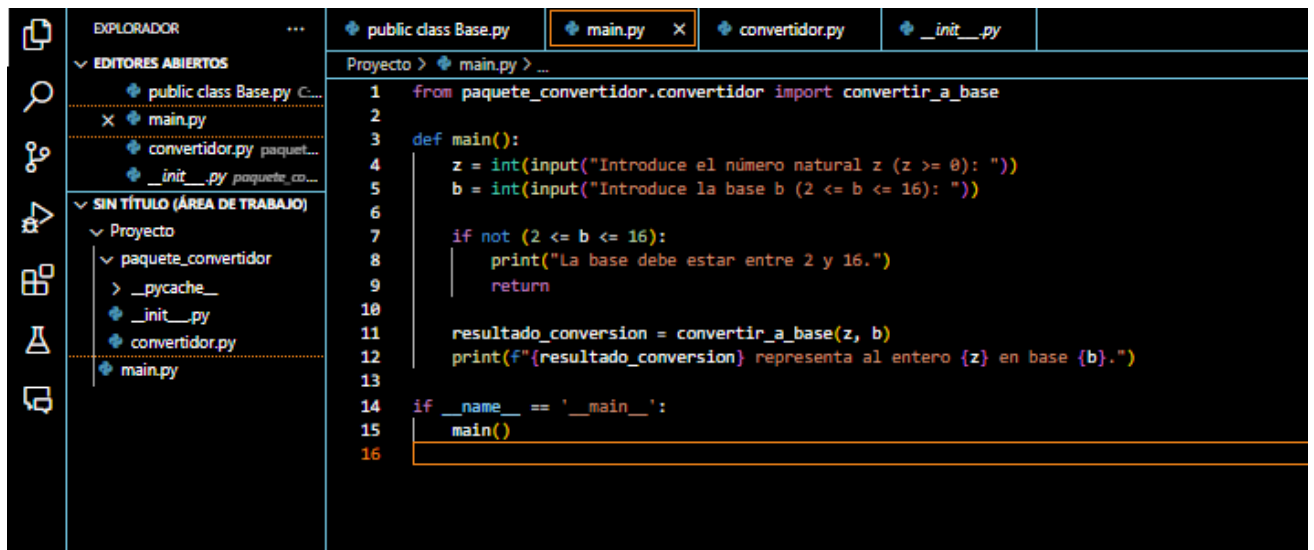
    resultado_conversion = convertir_a_base(z, b)

    print(f"{resultado_conversion} representa al entero {z} en base {b}.")

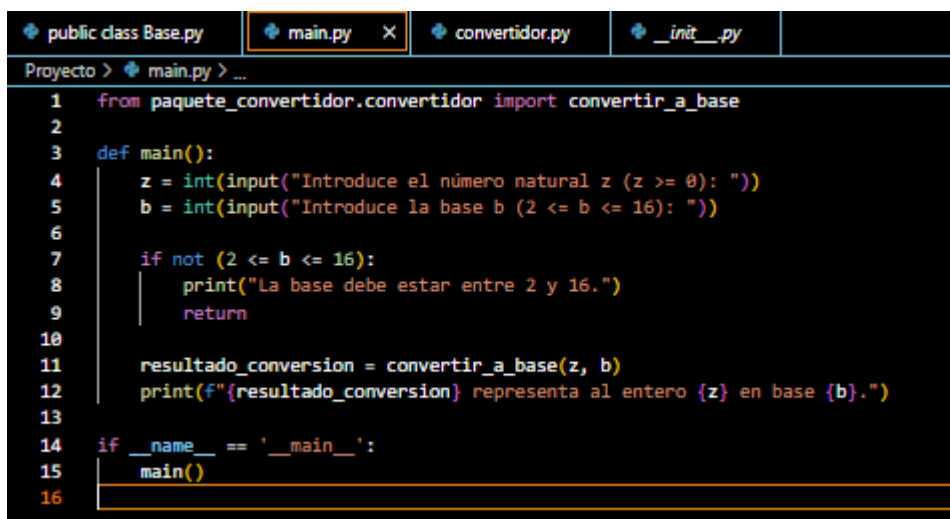
if __name__ == '__main__':

    main()
```

3.3. Organice el proyecto usando paquetes.



3.4. Divida la implementación en partes que sean fáciles de implementar, use los conceptos de modularidad y reutilización.



```
public class Base.py | main.py | convertidor.py X
Proyecto > paquete_convertidor > convertidor.py > ...
1 def convertir_a_base(z, b):
2     if z == 0:
3         return '0'
4
5     digitos = "0123456789ABCDEF"
6
7     resultado = ""
8     while z > 0:
9         resultado = digitos[z % b] + resultado
10        z //= b
11
12    return resultado
13
```

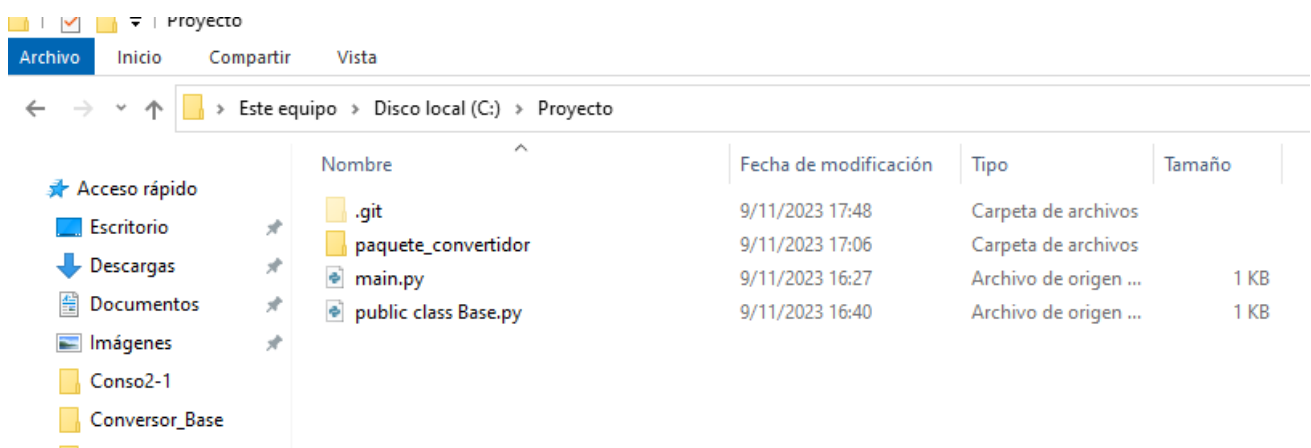
3.5. Cree el repositorio local (Git) y el remoto (GitHub).

En el github

3.6. Realice la presentación del proyecto usando el archivo Readme.MD. La presentación debe incluir: el título del proyecto, los objetivos del proyecto, las lista de integrantes del equipo de desarrollo y un diagrama de HIPO (Hierarchical Input Process Output, [enlace](#)) de la implementación.

En el github

3.7. En el repositorio local, crea las ramas dev y master, y sincronízalo con el repositorio remoto



3.8. En el repositorio local, crea una rama para cada funcionalidad o feature e implemente la funcionalidad aplicando TDD.

```
create mode 100644 main.py
create mode 100644 paquete_convertidor/__init__.py
create mode 100644 paquete_convertidor/__pycache__/con
create mode 100644 paquete_convertidor/convertidor.py
create mode 100644 public class Base.py

kened@DESKTOP-7294C5B MINGW64 /c/Proyecto (master)
$ git branch dev

kened@DESKTOP-7294C5B MINGW64 /c/Proyecto (master)
$ git checkout dev
Switched to branch 'dev'

kened@DESKTOP-7294C5B MINGW64 /c/Proyecto (dev)
$ git add .

kened@DESKTOP-7294C5B MINGW64 /c/Proyecto (dev)
$ git commit -m "Cambios realizados en la rama dev"
On branch dev
nothing to commit, working tree clean

kened@DESKTOP-7294C5B MINGW64 /c/Proyecto (dev)
$ git checkout master
Switched to branch 'master'

kened@DESKTOP-7294C5B MINGW64 /c/Proyecto (master)
$ git merge dev
Already up to date.

kened@DESKTOP-7294C5B MINGW64 /c/Proyecto (master)
$ git push origin dev
fatal: 'origin' does not appear to be a git repository
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
```



- 3.8.1. Una vez probada la funcionalidad (para nuestro caso es la definición de terminado), sincronice la rama feature con el repositorio remoto. No elimine la rama feature.
- 3.8.2. Realice un merge de la rama implementada con las ramas dev y master. 3.9.

```
kened@DESKTOP-7294C5B MINGW64 /c/Proyecto (master)
$ git checkout master
Already on 'master'

kened@DESKTOP-7294C5B MINGW64 /c/Proyecto (master)
$ git merge dev
Already up to date.

kened@DESKTOP-7294C5B MINGW64 /c/Proyecto (master)
$ git push origin master|
```

Comparta la ruta URL del proyecto en GitHub: <https://forms.gle/KFBjUQ5euLaPPEkDA>.

