Laboratorio GIT.

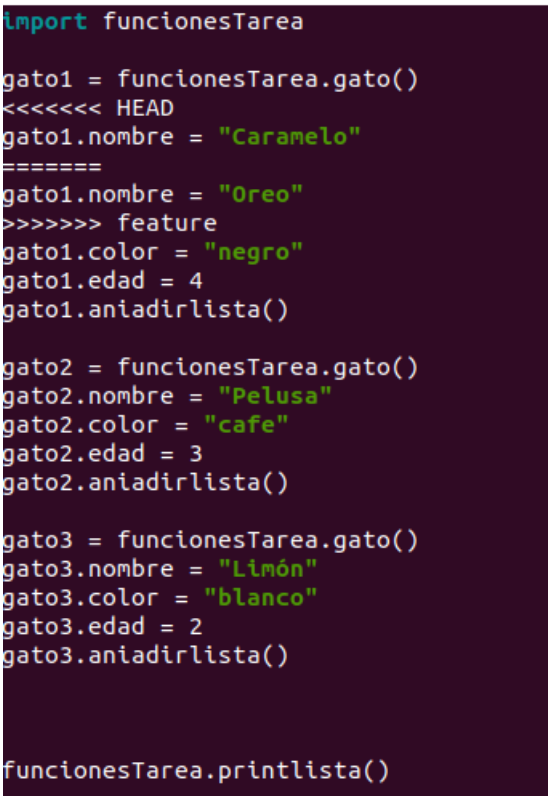
Steven Gerardo Chacón Salazar

Escuela de Ingeniería Eléctrica
Programación Bajo Plataformas Abiertas
Facultad de Ingeniería Eléctrica
Universidad de Costa Rica

M.Sc. Julián Alberto Gairaud Benavides

III Ciclo 2022

1. Tome un screenshot de su editor de texto mostrando los conflictos de ambas ramas (1 punto).



```
import funcionesTarea

gato1 = funcionesTarea.gato()
<<<<<<< HEAD
gato1.nombre = "Caramelo"
=======
gato1.nombre = "Oreo"
>>>>>>> feature
gato1.color = "negro"
gato1.edad = 4
gato1.aniadirlista()

gato2 = funcionesTarea.gato()
gato2.nombre = "Pelusa"
gato2.color = "cafe"
gato2.edad = 3
gato2.aniadirlista()

gato3 = funcionesTarea.gato()
gato3.nombre = "Limón"
gato3.color = "blanco"
gato3.edad = 2
gato3.aniadirlista()



funcionesTarea.printlista()
```

*Figura 1: Conflicto al hacer **merge** en ambas ramas **main** y **feature.***

2. Incluya los cambios en su repositorio remoto, posteriormente corra el comando “git log --oneline --graph” y tome un screenshot. (2 puntos).

```
stevonsa@stevonsa-VirtualBox:~/Git/LaboratorioGit/gatosGit$ git rebase main
First, rewinding head to replay your work on top of it...
Fast-forwarded feature to main.
stevonsa@stevonsa-VirtualBox:~/Git/LaboratorioGit/gatosGit$ git status
On branch feature
Your branch is ahead of 'origin/feature' by 4 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
stevonsa@stevonsa-VirtualBox:~/Git/LaboratorioGit/gatosGit$ cat gatos.py
import funcionesTarea

gato1 = funcionesTarea.gato()
gato1.nombre = "Caramelo"
gato1.color = "negro"
gato1.edad = 4
gato1.aniadirlista()

gato2 = funcionesTarea.gato()
gato2.nombre = "Pelusa"
gato2.color = "cafe"
gato2.edad = 3
gato2.aniadirlista()

gato3 = funcionesTarea.gato()
gato3.nombre = "Limón"
gato3.color = "blanco"
gato3.edad = 2
gato3.aniadirlista()



funcionesTarea.printlista()
stevonsa@stevonsa-VirtualBox:~/Git/LaboratorioGit/gatosGit$ git push
Total 0 (delta 0), reused 0 (delta 0)
To github.com:stevonsa/LaboratorioGit.git
   10ec5c1..a2ccfb3  feature -> feature
```

Figura 2: Inclusión de los cambios en el repositorio.

*Figura 3: Utilización del comando "**git log –oneline –graph**" en git.*

3. Del manual de git log que aparece al usar el comando --help, tome un screenshot de su terminal mostrando para qué sirve el comando --graph, otro mostrando qué significa el comando --oneline y otro más mostrando para qué fue "diseñado" oneline (3 screenshots en total para el punto 8) (3 puntos).

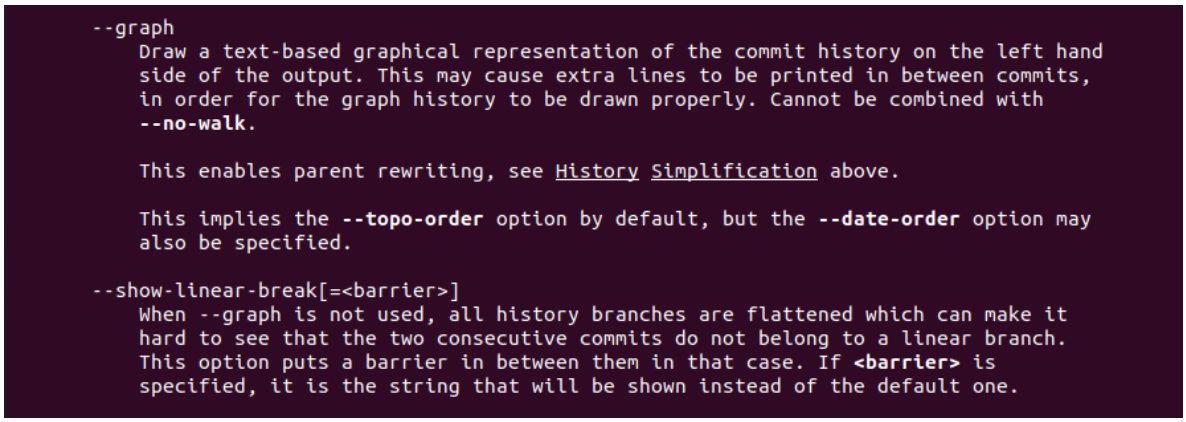


*Figura 4: funcionalidad del comando –**graph**.*

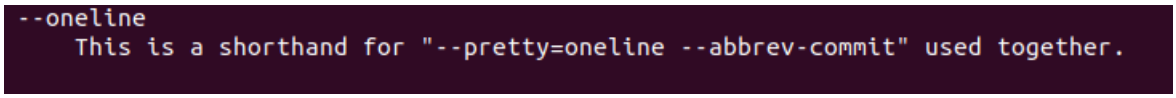


*Figura 5: significado del comando –**oneline**.*

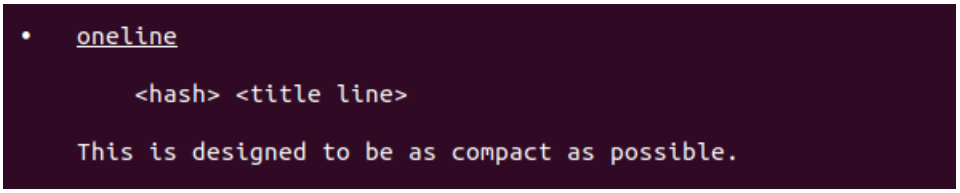


*Figura 6: por qué fue "diseñado" el comando –**oneline**.*

4. Tome un screenshot del archivo gatos.py resultante corriendo en su computadora (1 punto).

```
stevonsa@stevonsa-VirtualBox:~/Git/LaboratorioGit/gatosGit$ python3 gatos.py
Caramelo, 4 annos, color: negro
Pelusa, 3 annos, color: cafe
Limón, 2 annos, color: blanco
stevonsa@stevonsa-VirtualBox:~/Git/LaboratorioGit/gatosGit$ cat gatos.py
import funcionesTarea

gato1 = funcionesTarea.gato()
gato1.nombre = "Caramelo"
gato1.color = "negro"
gato1.edad = 4
gato1.aniadirlista()

gato2 = funcionesTarea.gato()
gato2.nombre = "Pelusa"
gato2.color = "cafe"
gato2.edad = 3
gato2.aniadirlista()

gato3 = funcionesTarea.gato()
gato3.nombre = "Limón"
gato3.color = "blanco"
gato3.edad = 2
gato3.aniadirlista()



funcionesTarea.printlista()
```

Figura 7: screenshot del archivo.py siendo ejecutado.