



UNIVERSIDAD DE COSTA RICA

UNIVERSIDAD DE COSTA RICA
SEDE RODRIGO FACIO

ESCUELA DE INGENIERÍA ELÉCTRICA

Programación bajo plataformas abiertas (IE-0117)

Profesor: Julián Gairaud

Estudiante:

Gabriel Siles C17530

Laboratorio GIT/ GIT hub

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



```
ragnarok@ie0117:~/GITHub/LaboratorioGit/gatosGit$ cat gatos.py
import funcionesTarea

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

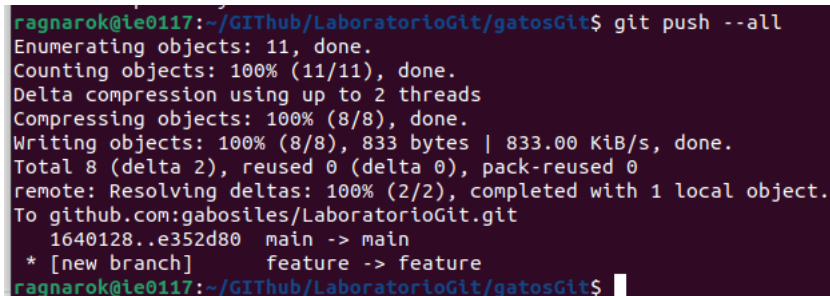
gato2 = funcionesTarea.gato()
gato2.nombre = "Pelusa"
gato2.color = "cafe"
gato2.edad = 3
gato2.aniadirlista()

gato3 = funcionesTarea.gato()
gato3.nombre = "Flipper"
gato3.color = "blanco"
gato3.edad = 1
gato3.aniadirlista()

funcionesTarea.printlista()
ragnarok@ie0117:~/GITHub/LaboratorioGit/gatosGit$
```

Figura 9. Texto del archivo gatos.py en rama main.

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



```
ragnarok@ie0117:~/GITHub/LaboratorioGit/gatosGit$ git push --all
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 2 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (8/8), 833 bytes | 833.00 KiB/s, done.
Total 8 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To github.com:gabosiles/LaboratorioGit.git
 1640128..e352d80  main -> main
 * [new branch]      feature -> feature
ragnarok@ie0117:~/GITHub/LaboratorioGit/gatosGit$
```

Figura 11. Inclusión de cambios.

```
ragnarok@ie0117:~/GITHub/LaboratorioGit/gatosGit$ git log --oneline --graph
* e352d80 (HEAD -> main, origin/main, origin/HEAD) Cambio Gato1, agregando gato 3
* 1640128 included gato2 and list printing
* c2b877a corrected mistakes
* d02a6d2 included gato1
* 7ab0adc included functions file
* a13fd08 first commit
ragnarok@ie0117:~/GITHub/LaboratorioGit/gatosGit$
```

Figura 12. Comando git log --oneline --graph.

8. 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).

```
--graph
  Draw a text-based graphical representation of the commit history on the left hand side of the output. This may cause extra
  lines to be printed in between commits, in order for the graph history to be drawn properly. Cannot be combined with
  --no-walk.

  This enables parent rewriting, see History Simplification above.

  This implies the --topo-order option by default, but the --date-order option may also be specified.
```

Figura 13. Uso comando --graph.

```
--oneline
  This is a shorthand for "--pretty=oneline --abbrev-commit" used together.
  git log, git show and git whatchanged commands when there is no --pretty, --format, or --oneline option given on the
  $ git rev-list --left-right --boundary --pretty=oneline A...B
  If the commit is a merge, and if the pretty-format is not oneline, email or raw, an additional line is inserted before the
  • oneline
  a new line, just as the "oneline" format does. For example:
```

Figura 14. Uso comando --oneline.

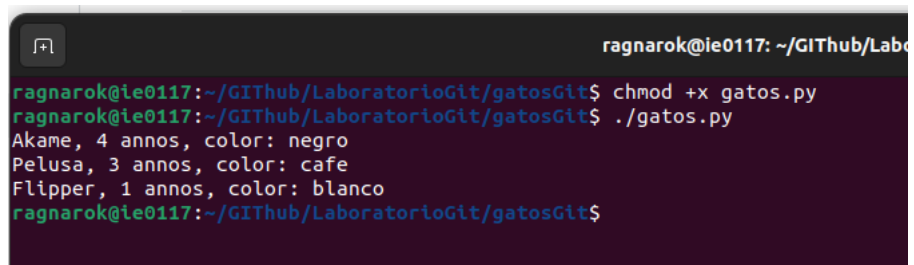
```
• oneline

  <hash> <title line>

  This is designed to be as compact as possible.
```

Figura 15. Razón de diseño del oneline

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



```

ragnarok@ie0117: ~/GITHub/Labo
ragnarok@ie0117:~/GITHub/LaboratorioGit/gatosGit$ chmod +x gatos.py
ragnarok@ie0117:~/GITHub/LaboratorioGit/gatosGit$ ./gatos.py
Akame, 4 annos, color: negro
Pelusa, 3 annos, color: cafe
Flipper, 1 annos, color: blanco
ragnarok@ie0117:~/GITHub/LaboratorioGit/gatosGit$
```

Figura 16. Programa gatos.py