Guia de ejercicios obligatoria Semana 2: incluye GIT, GITHUB, Sistema Operativo e introducción a la programación

Semana 2. GIT Y GITHUB

Slide 6. [tutorial github dia1 y dia 2]

Ejercicios.
1 . Explorar los comandos anteriores con “git help comando”
2. Crearse una cuenta en GitHub y obtener un token para autenticarse en las sucesivas operaciones que se hagan.

Slide 10.[tutorial github dia1 y dia 2]

Ejercicio:
Hacer cambios o incorporaciones nuevas y ver la historia
1 . Crear el archivo show_atribs.py
2. Agregar código Python para mostrar propiedades de objetos

```
print(f'Mostrando Métodos y Atributos del objeto float\n {dir(float)}\n')
print(f'Mostrando Métodos y Atributos del objeto list\n {dir(list)}')
```

3. preparar los cambios para un nuevo snapshot (pasar al stage)
git add show_atribs.py
3.a Si me equivoqué, puedo sacar el archivo del stage y volver a 1
git restore –staged show_atribs.py / git reset show_atribs.py
4. confirma el snapshot (pasar al repositorio)
git commit -m “trabajado con el método dir de python”
4.a Si lo necesito, aquí puedo revertir un commit y volver al paso 1
git log
Obtener el id_commit
git revert id_commit
5 . veamos la historia
git log

Ejercicio:
analizar la salida de git log para estudiar qué información nos brinda

Slide 12. [tutorial github dia1 y dia2]

Ejercicio: Trabajar con ramas (branchs)

1 . Creamos una carpeta: prueba_branchs
2. Creamos 4 archivos:
echo “linea1” > file1.txt
echo “linea1” > file2.txt
echo “linea1” > file3.txt
echo “linea1” > file4.txt
3. inicializamos el repositorio:
git init
4. Crear dos branchs:
4.a Hacemos un commit inicial
git add . (este commit incluye todo, para eso se usa el punto y no un nombre de archivo)
git branch feature_1
git branch feature_2
5. Veamos en qué branch estamos parados:
git branch
(debemos estar en main, si no es así, cambiar a main)
6. Cambiamos a feature_1
git checkout feature_1
7. Modificamos file1.txt y file2.txt
echo "linea2" >> file1.txt
echo "linea2" >> file2.txt
8. Confirmamos los cambios:
git add file1.txt file2.txt
git commit -m “file1 y file2 modificados”
9 . Cambiamos al branch main y revisemos file1 y file2. Estos archivos,tienen los cambios realizados en el punto 7?
10. Veamos en qué branch estamos parados:
git branch
11. Cambiamos a feature_2
git checkout feature_2
12. Modificamos file3.txt y file4.txt
echo "linea2" >> file3.txt
echo "linea2" >> file4.txt
13. Confirmamos los cambios:
git add file3.txt file4.txt
git commit -m “file3 y file4 modificados”
14. Cambiamos a main
git checkout main
15. Hacemos un merge de feature_1 con main
git merge feature_1
16. Hacemos un merge de feature_2 con main

git merge feature_2

Revisemos el contenido de file1.txt, file2.txt, file3.txt y file4.txt . ¿Qué podemos notar?

Slide 13. [tutorial github dia1 y dia2]

Ejercicio:
1 . Describa gráficamente el flujo de trabajo anterior indicando los ID de commits
2. Repita el ejercicio pero creando el branch feature_2 luego de haber fusionado las ramas feature_1 con master. Que nota? Qué contenidos tienen los archivos file1.txt y file2.txt y porque?

Slide 15. [tutorial github dia1 y dia2]

Ejercicio.
GIT PUSH. Hacer una copia de seguridad del código local en el remoto (GitHub)

Push nuestro repositorio local al remoto:

1. Ir a https://github.com/ crear una cuenta y obtener un token (settings -> developer settings-> personal access tokens -> generate new token)
2. Crear un nuevo repositorio Privado
3. Cambiar a mi work directory:
   cd mi_proyecto_local
4. Apuntar a ese remoto (origin) desde la copia local
   a. git remote add origin https://github.com/pjzgeo/prueba2
   b. git remote -v
5. Enviar commits al repositorio remoto
   a. git push origin main
   b. puedo seguir haciendo push de mis branchs si así lo requiero…
      git checkout feature_1
      git push origin feature_1

Ejercicio:
Revisar en línea (GitHub) que los branchs correspondientes con sus archivos, estén presentes. Hacer una captura de pantalla.

Slide 15. [tutorial github dia1 y dia2]

Ejercicio.
GIT CLONE. Descargar código fuente desde un repositorio hacia nuestra computadora, para instalarlo y usarlo, para modificarlo, para mejorarlo, entre otros.

1. Buscar un repositorio de interés en github.

2. Clonarlo. Nos ubicamos en el home en nuestra computadora:
   cd ~
3. clonar.
   git clone https://github.com/NouveauNu/parity_code.git
4. Cambiar al directorio de repositorio descargado
   cd parity_code
5. Revisar su contenido
   ls

Slide 15. [tutorial github dia1 y dia2]

Ejercicio: colaborar con un proyecto de un tercero en GitHub

Separarse en dos grupos
El grupo 1: Crear un proyecto en GitHub que contenga un archivo de texto con las indicaciones de “Como contribuir a un proyecto de GitHub”. Elija a uno de los alumnos para usar su cuenta de GitHub como repositorio.
El grupo 2: Contribuir al proyecto del repositorio del Grupo 1:
Va a revisar las indicaciones y las va corregir o completar o hacer alguna observación.
Va a agregar un diagrama que represente las instrucciones definidas en el archivo de texto.

El grupo 2 genera un pull request para que el Grupo 1 incorpore los cambios
El grupo 1 atiende el pull request fusionando los cambios recibidos, previos revisión para determinar que lo que llega está correcto y se debe fusionar o mergear.

Semana 2. Sistemas Operativos e introducción a la programación

Las siguiente preguntas son en referencia al contenido dado en los slides sem2-Dia3 y sem2-Dia5a

1. ¿Cómo se define un sistema operativo? ¿Existe una definición concreta o su definición está dada en base a aspectos?
2. ¿Cual es la diferencia entre un proceso y un programa?
3. ¿Con qué comando puede listar los programas instalados en la PC?
4. Listar los procesos de sistema. Hacer un print de pantalla. Observar el estado de algún par de procesos elegidos al azar. ¿Observa que cambian de estado? ¿Cómo podría forzar a que esto ocurra? Lograr ver el cambio de

estados es una acción que involucra que la CPU está trabajando de forma más intensiva sobre el proceso, o bien, el proceso tiene más tiempo de CPU.

5. Veamos la asignación de recursos de algunos procesos
    a. Abrir gedit y vscode (o elegir cualquiera dos programas en ejecución)
    b. ps aux | grep gedit (obtenemos el PID de gedit)
    c. ps aux | grep vscode (obtenemos el PID de vscode)
    d. Ejecutar htop -p pid1,pid2
    e. Observar CPU y Memoria asignada a los procesos
    f. Pregunta: ¿Porqué en la salida de htop, se repiten los nombres de los procesos y con diferentes PIDs? ¿Qué pasa si ejecuto htop -t -p pid1,pid2 ?
6. ¿Qué es compilar?
7. ¿Que es un compilador?
8. En la división lógica de una RAM, que partes existen que son usadas por los programas escritos en lenguajes tipo C? Que se almacenan en estos espacios?
9. Definir que es variable local y variable global
10. Definir que es una función recursiva
11. Implementar la función factorial en Python y ejecutarla para diferentes entradas. Muestre print de pantalla con el código y las salidas de las ejecuciones para factorial evaluado en: n = -1, 5, 5000
    ¿Qué cree que sucede cuando n = -1 ? Ayuda: matemáticamente cuál es el dominio e imagen de la función factorial?
12. En lenguajes tipo python o java, como se organiza y gestiona la memoria a diferencia de los lenguajes tipo C? Ayudese con la herramienta https://pythontutor.com/
13. ¿Qué es un lenguaje natural y uno formal?
14. ¿Qué elementos mínimos en común tienen dos lenguajes que pertenecen a estos grupos?
15. De una diferencia fundamental que presentan los lenguajes naturales de los formales
16. Suponga que está programando en Python. Quiere crear este diccionario pero algo no esta correcto:

    ```
    mi_diccionario = { "compilar": "proceso de traducir código escrito por el usuario a código    en lenguaje de máquina"
          "compilador": "programa para compilar"
       }
    ```

    ¿Qué tipo de error encuentra? en qué parte del proceso desde la escritura del programa por parte del programador hasta la ejecución, se determina este tipo de errores?
17. ¿Que es un algoritmo?
18. ¿Qué es un lenguaje de scripting y cuáles conoces?

19. ¿En qué contextos recomendaría el uso de un lenguaje de scripting vs un lenguaje compilado tipo C?
20. ¿Qué características tiene un lenguaje de scripting?
21. ¿Que debe tener en cuenta a la hora de pensar (no de implementar) un algoritmo?
22. Implementa un ejemplo en Python de un programa que use variables contadores
    Implementa un ejemplo en Python de un programa que use variables acumuladores
    Para ambos, hacer print de pantalla del código y mostrar la salida de la ejecución.
23. Liste los operadores relacionales y lógicos de Python
24. Liste los operadores aritméticos de Python
25. Explica como funciona un IF y cómo funciona un loop en Python. Cree que funcionan igual en cualquier lenguaje de programación?