Universidad de La Laguna
E.T.S. Ingeniería Informática
Área: Lenguajes y Sistemas Informáticos

Lenguajes y Paradigmas de Programación
Práctica de laboratorio **#3**

1. Iniciar una sesión de trabajo en GNU-Linux.

2. Abra una terminal.

3. Muestre el árbol de directorios de su HOME (`tree`).

4. Sitúese en el **directorio** de la asignatura "Lenguajes y Paradigmas de Programación" esto es en el directorio *LPP* (`cd LPP`).

5. Muestre el contenido del directorio actual (`ls -la`).

6. Cree un nuevo directorio denominado *prct03* (`mkdir prct03`). Este será el **directorio de trabajo** durante la realización de esta práctica.

7. Sitúese en el directorio *prct03* (`cd prct03`)

8. Ponga el directorio *prct03* bajo el control de versiones, es decir, cree un repositorio *git* (`git init`).

9. Cree un fichero de texto, `respuestas.txt`, en el que responda las preguntas de esta práctica de laboratorio.

10. Añada todo el contenido del directorio *prct03* al *índice del repositorio git.* (`git add .`)

11. Confirme los cambios del índice en el repositorio `git` local. (`git commit -m "1ra version"`)

12. Cree un repositorio en *GitHub* con el nombre `prct03`

13. Muestre los repositorios remotos. (`git remote -v`)

14. Cree un repositorio remoto con nombre corto *origin*
    (`git remote add origin git@github.com:aluXXXXXXXX/prct03.git`)

15. Empuje los cambios en el repositorio remoto denominado *origin.*
    (`git push -u origin master`)

16. Muestre la versión del intérprete de Ruby disponible (`ruby -v`)

17. A diferencia de los lenguajes compilados, existen dos formas de ejecutar Ruby. Se pueden crear ficheros y ejecutarlos con el intérprete como se hizo en la práctica de laboratorio anterior, y también, se puede introducir el código de forma interactiva. Ejecute el Ruby interactivo (*Interactive Ruby*) (`irb`)

18. Escriba el programa que muestra por la consola la frase `"Hola Mundo"`. ¿Qué significado tiene la salida? (`puts 'Hola Mundo'`)

19. Salga de la sesión interactiva. Para ello escriba el caracter de final del fichero del sistema operativo. (`Ctrl + D`)

20. Para realizar los siguientes ejercicios utilice el intérprete interactivo. Copie **manualmente** los comandos. No corte y pegue desde el fichero PDF, esto puede introducir carácteres extraños e invisibles dependiendo de la codificación que se esté utilizando.

21. ¿Qué diferencia hay entre `"\t\n"` y `'\t\n'`?

22. ¿Cómo funciona `%q`? ¿Qué es `%q{hello world\n}`? ¿Qué es `%q{'a' 'b' 'c'}`?

23. ¿Cómo funciona `%Q`? ¿Qué es `%Q{hello world\n}`? ¿Qué es `%Q{"a" "b" "c"}`?

24. ¿Qué queda en `c`?

```
irb(main):001:0> a = 4
=> 4
irb(main):002:0> b =2
=> 2
irb(main):003:0> c = <<HERE
irb(main):004:0" --#{a}--
irb(main):005:0" --#{b}--
irb(main):006:0" HERE
```

25. ¿Qué queda en `c`?

```
irb(main):001:0> a = 4
=> 4
irb(main):002:0> b =2
=> 2
irb(main):008:0> c = <<'HERE'
irb(main):009:0' --#{a}--
irb(main):010:0' --#{b}--
irb(main):011:0' HERE
```

26. `s = "hello"`. ¿Cuál es el valor de las siguientes expresiones?

    - `s[0,2]`
    - `s[-1,1]`
    - `s[0,10]`

27. ¿Qué queda en `g`?

```
>> g = "hello"
=> "hello"
>> g << " world"
```

28. ¿Qué queda en `e`?

```
>> e = '.'*3
```

29. ¿Cuál es el resultado?

```
>> a = 1
=> 1
>> "#{a=a+1} "*3
```

30. ¿Qué es esto? `%w[this is a test]`

31. ¿Qué es esto? `%w[\t \n]`

32. ¿Qué es esto? `%W[\t \n]`

33. ¿Qué contiene `nils`? `nils = Array.new(3)`

34. ¿Qué contiene `zeros`? `zeros = Array.new(3, 0)`

35. ¿Qué queda en `b`?

```
>> x = [[1,2],[3,4]]
=> [[1, 2], [3, 4]]
>> b = Array.new(x)
```

36. ¿Qué queda en `c`?

```
 >> c = Array.new(3) { |i| 2*i }
```

37. ¿Cuál es el resultado de cada una de estas operaciones?

```
>> a = ('a'..'e').to_a
=> ["a", "b", "c", "d", "e"]
>> a[1,1]
=>
>> a[-2,2]
=>
>> a[0..2]
=>
>> a[0...1]
=>
>> a[-2..-1]
=>
```

38. ¿Cuál es el resultado de cada una de estas operaciones?

```
>> a
=> ["a", "b", "c", "d", "e"]
>> a[0,2] = %w{A B}
=> ["A", "B"]
>> a
=>
>> a[2..5] = %w{C D E}
=> ["C", "D", "E"]
>> a
=>
>> a[0,0] = [1,2,3]
=> [1, 2, 3]
>> a
=>
>> a[0,2] = []
=> []
>> a
```

```
=>
>> a[-1,1] = [ 'Z' ]
=> ["Z"]
>> a
=>
>> a[-2,2] = nil
=> nil
>> a
=>
```

39. ¿Cuál es el resultado de cada una de estas operaciones?

```
>> a = (1...4).to_a
=>
>> a = a + [4, 5]
=>
>> a += [[6, 7, 8]]
=>
>> a = a + 9
```

40. ¿Cuál es el resultado de cada una de estas operaciones?

```
>> x = %w{a b c b a}
=>
>> x = x - %w{b c d}
=>
```

41. ¿Cuál es el resultado de cada una de estas operaciones?

```
>> z = [0]*8
=>
```

42. ¿Cuál es el resultado de cada una de estas operaciones?

```
>> a = []
=> []
>> a << 1
=>
>> a << 2 << 3
=>
>> a << [4, 5, 6]
=>
>> a.concat [7, 8]
=>
```

43. ¿Cuál es el resultado de cada una de estas operaciones?

```
>> a = [1, 1, 2, 2, 3, 3, 4]
=> [1, 1, 2, 2, 3, 3, 4]
>> b = [5, 5, 4, 4, 3, 3, 2]
=> [5, 5, 4, 4, 3, 3, 2]
>> c = a | b
=>
>> d = b | a
=>
>> e = a & b
=>
>> f = b & a
=>
```

44. ¿Qué resultados dan las siguientes operaciones?

```
>> a = 1..10
=> 1..10
>> a.class
=> Range
>> a.to_a
=>
>> b = 1...10
=> 1....10
>> b.to_a
=>
>> b.include? 10
=>
>> b.include? 8
=>
>> b.step(2) {|x| print "#{x} " }

>> 1..3.to_a
```

45. ¿Qué resultados dan las siguientes operaciones?

```
>> r = 0...100
=> 0....100
>> r.member? 50
=>
>> r.include? 99.9
=>
>> r.member? 99.9
=>
```

46. ¿Qué resultados dan las siguientes operaciones?

```
>> true.class
=>
>> false.class
=>
>> puts "hello" if 0
=>
>> puts "hello" if nil
=>
>> puts "hello" if ""
=>
```

47. ¿Qué resultados dan las siguientes operaciones?

```
>> x = :sym
=> :sym
>> x.class
=>
>> x == 'sym'
=>
>> x == :sym
=>
>> z = :'a long symbol'
=> :"a long symbol"
>> z.class
=>
>> x == 'sym'.to_sym
=>
>> x.to_s == 'sym'
=>
```

48. ¿Qué resultados se dan?

```
>> s = "Ruby"
=> "Ruby"
```

```
>> t = s
=> "Ruby"
>> t[-1] = ""
=> ""
>> print s

>> t = "Java"
=> "Java"
>> print s, t
```

49. ¿Cuál es el resultado?

```
>> "%d %s" % [3, "rubies"]
=>
```

50. ¿Cuáles son los resultados?

```
>> x, y = 4, 5
=>
>> z = x > y ? x : y
=>
>> x,y,z = [1,2,3]
=>
```

51. ¿Qué resultados dan las siguientes operaciones?

```
>> x = { :a => 1, :b => 2 }
=> {:b=>2, :a=>1}
>> x.keys
=>
>> x.values
=>
>> x[:c] = 3
=> 3
>> x
=>
>> x.delete('a')
=> nil
>> x
=>
>> x.delete(:a)
=> 1
>> x
=>
>> x = { :a => 1, :b => 2, :c => 4 }
=> {:b=>2, :c=>4, :a=>1}
>> x.delete_if { |k,v| v % 2 == 0 }
=>
>> x
=>
```

52. ¿Qué hace `yield 4, 5`?

53. ¿Qué hace la siguiente sentencia? `counts = Hash.new(0)` ¿Qué diferencia hay con la asignación `counts = {}`?

54. ¿Qué retorna esta expresión? `'hello world, hello LPP'.scan /\w+/`

55. Explique que hacen estas sentencias:

```
c = { :a => 3, :b => 2, :c => 1 }
c.keys.sort.each { |k| puts c[k] }
```

56. Escriba la dirección del repositorio que ha creado en GitHub en la tarea habilitada en el campus virtual.

57. Cierre la sesión.