Universidad de La Laguna
Facultad de Matemáticas
Área: Lenguajes y Sistemas Informáticos

Técnicas Experimentales
Práctica de laboratorio **#3**

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

2. Muestre el árbol de directorios de su directorio HOME.

3. Sitúese en el **Directorio de Proyecto** de la asignatura Técnicas Experimentales esto es en el directorio *TE* (`cd TE`).

4. Muestre el contenido del directorio de trabajo (`ls -la`).

5. Cree un nuevo directorio denominado *prct03* (`mkdir prct03`).

6. Sitúese en el directorio *prct03* (`cd prct03`) y cree la estructura de directorios que le permita tener subcarpetas para el código y los documentos, es decir:

   - un subdirectorio *src*
   - un subdirectorio *docs*

7. Guarde el fichero PDF que contiene el enunciado de esta práctica en el directorio *docs*.

8. Sitúese en el directorio *src* y clone el repositorio remoto que contiene los ficheros fuente que ponen de manifiesto la diferencia entre compilación e interpretación.

   (`git clone git@github.com:coromoto/CompilacionVSInterpretacion.git`)

9. Compruebe que aparece en su directorio actual un subdirectorio con nombre `CompilacionVSInterpretacion` (`ls -la`).

10. Sitúese en el directorio *CompilacionVSInterpretacion* (`cd CompilacionVSInterpretacion`).

11. Compruebe que aparecen en su directorio actual los siguientes ficheros (`ls -la`).

```
helloWorld.c    - Lenguaje C   / compilado
helloWorld.cc   - Lenguaje C++ / compilado
helloWorld.sh   - Lenguaje Bash / interpretado
helloWorld.py   - Lenguaje Python / interpretado
HelloWorld.java - Lenguaje Java / compilado e interpretado
```

    Cada uno de los ficheros contiene un programa que muestra por pantalla la frase "Hello World".

12. Compilación en C:

    a) Muestre el contenido del fichero `helloWorld.c` sin abrirlo (`cat`).
    b) Compile el fichero `helloWorld.c` con el comando `gcc -o helloWorldC helloWorld.c`
    c) Compruebe que aparece en el directorio actual el fichero `helloWorldC` (`ls -la`)
    d) Ejecute el programa que se ha compilado con el comando `./helloWorldC`

13. Compilación en C++:
    a) Muestre el contenido del fichero `helloWorld.cc` sin abrirlo (`cat`)
    b) Compile el fichero `helloWorld.cc` con el comando `g++ -o helloWorldCPP helloWorld.cc`
    c) Compruebe que aparece en el directorio actual el fichero `helloWorldCPP` (`ls -la`)
    d) Ejecute el programa que se ha compilado con el comando `./helloWorldCPP`
14. Interpretación en Bash:
    a) Muestre el contenido del fichero `helloWorld.sh` sin abrirlo (`cat`)
    b) Ejecute el programa con el comando `bash ./helloWorld.sh`
15. Interpretación en Python:
    a) Muestre el contenido del fichero `helloWorld.py` sin abrirlo (`cat`)
    b) Ejecute el programa con el comando `python ./helloWorld.py`
16. Compilación e interpretación en Java:
    a) Muestre el contenido del fichero `HelloWorld.java` sin abrirlo (`cat`)
    b) Compile el fichero `HelloWorld.java` con el comando `javac HelloWorld.java`
    c) Compruebe que aparece en el directorio actual el fichero `HelloWorld.class` (`ls -la`)
    d) Ejecute el programa que se ha generado con el comando `java HelloWorld`
17. ¿Cuál es la diferencia entre compilación e interpretación?
18. ¿Qué permisos tiene el fichero `helloWorld.sh` ? ¿Se puede ejecutar directamente?
    (`./helloWorld.sh`)
19. ¿Qué permisos tiene el fichero `helloWorld.py`? ¿Se puede ejecutar directamente?
    (`./helloWorld.py`)
20. ¿Cuál es la principal diferencia entre el contenido del fichero `helloWorld.sh` y `helloWorld.py`?
21. ¿En qué directorio está instalado el intérprete de Python? (`which python`)
22. Modifique el fichero `helloWorld.py` para que sea ejecutable. En primer lugar edite el fichero para añadir la primera línea correspondiente y a continuación modifique los permisos del fichero. (`chmod u+x helloWorld.py`)
23. Muestre el estado del repositorio `git` local, esto es, qué ficheros han cambiado, cuáles son nuevos y cuáles han sido borrados. (`git status`)
24. Muestre las diferencias entre los ficheros sin registrar y los del último registro. (`git diff`)
25. Añada los cambios al *índice del repositorio git* y regístrelos.
    (`git add . && git commit -m "Python ejecutable"`)

26. Muestre la historia de los distintos *registros* (`commit`) en la rama actual. (`git log --pretty=oneline`) Copie el número de identificación del registro "Primeros ejemplos".

27. Recupere el archivo original de `helloWorld.py`.

    (`git show` *Número de identificación de "Primeros ejemplos"*`:helloWorld.py > helloWorld.bak`)

28. Añada el fichero `helloWorld.bak` al *índice del repositorio git.*

    (`git add helloWorld.bak`)

29. Registre los cambios en el *índice del repositorio git.*

    (`git commit -m "Recuperación de una version anterior"`)

30. Muestre la historia de los distintos registros (*commits*) en la rama actual. (`git log`)

31. Cree un repositorio en *GitHub*

    a) Abra en el navegador el sitio de GitHub: http://github.com

    b) Introduzca su Nombre de Usuario `aluXXXXXXXXX`

    c) Introduzca su contraseña

    d) En la barra de usuario, en la esquina superior derecha de la página, haga clik en el icono de "Crear un repositorio nuevo" (*Create a New Repo*).

    e) Introduzca el nombre `prct03`

    f) Seleccione que quiere hacer el repositorio público.

    g) NO seleccione la casilla de crear el fichero README.md.

    h) Pulse el botón para crear el repositorio (*Create repository*)

32. Cree un repositorio remoto con nombre corto *miremoto*

    (`git remote add miremoto git@github.com:aluXXXXXXXX/prct03.git`)

33. Muestre los repositorios remotos que están definidos. (`git remote -v`)

34. Empuje los cambios en su repositorio remoto, es decir, en *miremoto.*

    (`git push -u miremoto master`)

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

36. Ejecutar el intérprete interactivo de Python. En este modo, se espera a la siguiente orden con el indicador principal, que suelen ser tres signos 'mayor' (`>>>`). Para las líneas adicionales, se utiliza el indicador secundario, que son tres puntos (`...`). El intérprete muestra un mensaje de bienvenida con su número de versión e información de derechos de copia, antes de mostrar el indicador principal. Para salir el intérprete interactivo se ha de teclear un carácter de final de fichero (`Control^+D`)

```
$python
Python 2.5.2 (r252:60911, Jan 20 2010, 21:48:48)
[GCC 4.2.4 (Ubuntu 4.2.4-1ubuntu3)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

37. Utilizar el intérprete interactivo de Python como una calculadora.

```
>>> 2 + 2
4
```

38. Escribir el programa "Hola Mundo" en Python usando el intérprete interactivo.

```
>>> hola = "Hola mundo desde Python"
>>> print hola
Hola mundo desde Python
```

39. ¿Qué sucede cuando se usa el operador de división entera (`//`) con números reales? Por ejemplo: `1.0 // 3.0`

40. ¿Qué hace el operador `**`? Por ejemplo: `2 ** 3`

41. ¿Qué hace el operador `%`? Por ejemplo: `12 % 10`

42. ¿Qué hacen los operadores `+=, -=, *=, /=, **=, //=`?

43. Evaluar las siguientes expresiones:

```
>> 2 + 3 < 2 * 3 or 6 < 10
?
>>> (2+3 < 2) * (3 or 6 < 10)
?
>>> a = 2 < 4
?
>>> b = 2 >= 4
?
>>> a * b
?
>>> True and True
?
>>> True and False
?
>>> False and True
?
>>> False and False
?
>>> True or True
?
>>> True or False
?
>>> False or True
?
>>> False or False
?
```

44. En Python hay números enteros y números reales. Los números enteros pertenecen a la clase denominada `int`, los números reales pertenecen a otra clase llamada `float`. `True` y `False` pertenecen a la clase `bool`. Para determinar la clase de un número o variable se usa la función `type()`. Comprobar su comportamiento con los siguientes ejercicios:

```
>>> type(10)
?
>>> type(10.0)
?
>>> type(False)
?
```

```
>>> type(True)
?
>>> type(10) is int
?
>>> type(10.0) is float
?
>>> type(10<20) is bool
?
>>> type(False) is bool
?
>>> type(False) is int
?
>>> type(10) == type(10.0)
?
>>> type(10+20) == type(10)
?
>>> type(10+20) == type(10<20)
?
```

45. Escriba varias cadenas (`strings`), asígnelas a variables y concaténalas entre sí. Por ejemplo:

```
>>> estrofa = " El patio de mi casa... "
>>> estribillo = " dubididu "
>>> estribillo += "zombie "*4
>>> cancion = estrofa + estribillo
```

46. Pruebe las siguientes expresiones de comparación entre cadenas:

```
>>> a = 'PYTHON'
>>> b = 'python'
>>> a == b
?
>>> a != b
?
>>> a != b
?
>>> a <= b
?
```

47. ¿Qué ocurre al concatenar objetos de tipo cadena con objetos numéricos? Por ejemplo:

```
>>> 'PYTHON' + 3
?
>>> 'PYTHON' + 3.1415
?
```

48. ¿Qué hace la función str()? Por ejemplo:

```
>>> 'PYTHON' + str(3)
?
>>> 'PYTHON' + str(3.1415)
?
```

49. ¿Qué resultado se obtiene al ejecutar las siguientes expresiones?

```
>>> "3" + 3
?
>>> int("3") + 3
?
>>> "3" + str(3)
?
```

50. Se puede acceder a cada una de las letras de una cadena de caracteres usando el operador de indexación `[]`. ¿Qué resultados se obtienen al ejecutar lo siguiente?

```
>>> 'PYTHON'[0]
?
>>> cantante[0]
?
```

51. El mismo operador se utiliza para acceder a segmentos de la cadena (substrings). El primer número indica el índice de inicio y el segundo, separado por dos puntos, la longitud. ¿Qué resultados se obtienen al ejecutar?

```
>>> 'PYTHON'[0:3]
?
>>> 'cantante'[2:4]
?
```

52. ¿Qué ocurre cuando se utilizan índices negativos para acceder a los elementos de una cadena? Por ejemplo: (`'PYTHON'[-1]`)

53. ¿Qué sucede cuando se omite parte del rango de una cadena? Por ejemplo, `'PYTHON'[3:] o 'PYTHON'[:3]`

54. ¿Qué hace la función abs()? Por ejemplo:

```
>>> abs(-3)
?
>>> abs(3)
?
```

55. ¿Qué hace la función float()? Por ejemplo:

```
>>> float(3)
?
>>> float(3.2)
?
>>> float(3.2e10)
?
>>> float(Un Texto)
?
```

56. ¿Qué hace la función int()? Por ejemplo:

```
>>> int(2.1)
?
>>> int(-2.9)
?
>>> int('2')
?
```

57. ¿Qué hace la función round()? Por ejemplo:

```
>>> round(2.1)
?
>>> round(2.9)
?
>>> round(-2.9)
?
>>> round(2)
?
```

58. ¿Cuál es el resultado de evaluar las siguientes expresiones?

```
>>> abs(-23) % int(7.3)
?
>>> abs(round(-34.2765,1))
?
>>> str(int(12.3)) + 0
?
>>> str(float(str(2) * 3 + .123)) + 321
?
>>> str(int(2.1) + float(3))
?
```

59. ¿Cuál es el resultado de ejecutar las siguientes sentencias?

```
>>> from math import sin
>>> sin(0)
?
>>> sin(1)
?
>>> cos(0)
?
>>> from math import cos
>>> cos(0)
?
>>> from math import *
>>> sin(2 * pi)
?
```

60. ¿Cuál es el resultado de evaluar las siguientes expresiones?

```
>>> from math import *
>>> int(exp(2 * log(3)))
?
>>> round(4*sin(3 * pi / 2))
?
>>> abs(log10(.01) * sqrt(25))
?
>>> round(3.21123 * log10(1000), 3)
?
```

61. ¿Cuál es el resultado de ejecutar las siguientes sentencias?

```
>>> print "%d" %1
?
>>> print "%d %d" %(1, 2)
?
>>> print "%d%d" %(1, 2)
?
>>> print "%d, %d" %(1, 2)
?
>>> print 1, 2
?
>>> print "%d 2" %1
?
```

62. Cierre la sesión.