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

Técnicas Experimentales
Práctica de laboratorio #4

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 *prct04* (`mkdir prct04`).

6. Sitúese en el directorio *prct04* (`cd prct04`) 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. El directorio de trabajo será `prct04`. Sitúese en él e inicialícelo para que sea un repositorio `git`. (`git init`)

9. Compruebe que se crea el directorio `.git` (`ls -la`).

10. Para comprobar la sintaxis de las sentencias ejecutar el intérprete interactivo de Python (`python`). Recuerde que para salir el intérprete interactivo se ha de teclear un carácter de final de fichero (`Control^+D`).

11. Los ficheros fuente debe crearlos en el directorio `src` y hacerlos ejecutables.

12. Añada todos los ficheros y subdirectorios del directorio actual al *índice del repositorio git*.
    (`git add .`)

13. Registre (*commit*) todos los cambios en el índice del repositorio git.
    (`git commit -m "Comentario del cambio"`)

14. 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 `prct04`

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*)

15. Cree un repositorio remoto con nombre corto *origin*

    (`git remote add origin git@github.com:aluXXXXXXXX/prct04.git`)

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

17. Empuje los cambios en su repositorio remoto, es decir, en *origin*. (`git push -u origin master`)

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

19. ¿Qué hace el siguiente programa? ¿Es correcto? ¿Qué está mal?

```
a = float(raw_input('Valor de a: '))
b = float(raw_input('Valor de b: '))

a * x + b = 0

print 'Solucion: ', x
```

20. ¿Qué hace el siguiente programa? ¿Es correcto? ¿Qué está mal?

```
x = -b/a

a = float(raw_input('Valor de a: '))
b = float(raw_input('Valor de b: '))

print 'Solucion: ', x
```

21. ¿Qué hace el siguiente programa cuando el valor de la variable `a` es cero? Haga una propuesta para solucionar el error que se produce en tiempo de ejecución.

```
a = float(raw_input('Valor de a: '))
b = float(raw_input('Valor de b: '))

x = -b/a

print 'Solucion: ', x
```

22. ¿Qué error se produce en tiempo de compilación?. ¿Cómo se soluciona?

```
a = float(raw_input('Valor de a: '))
b = float(raw_input('Valor de b: '))

if a != 0:
  x = -b/a
  print 'Solucion: ', x
if a = 0:
  print 'La ecuación no tiene solución.'
```

23. Modifique el programa que calcula la solución de una ecuación de primer grado para que contemple que cuando tanto el valor de `a` como el de `b` son cero, la ecuación tiene infinitas soluciones.

24. ¿Qué hace el siguiente programa?

```
from math import sqrt

a = float(raw_input('Valor de a: '))
b = float(raw_input('Valor de b: '))
c = float(raw_input('Valor de c: '))

if a != 0:
  x1 = (-b + sqrt(b**2 - 4*a*c)) / (2 * a)
  x2 = (-b - sqrt(b**2 - 4*a*c)) / (2 * a)
  print 'Las soluciones de la ecuacion son: x1=%4.3f y x2=%4.3f' % (x1, x2)
else:
  if b != 0:
    x = -c / b
    print 'La solucion de la ecuacion es: x=%4.3f' % x
  else:
    if c != 0:
      print 'La ecuacion no tiene solucion'
    else:
      print 'La ecuacion tiene infinitas soluciones'
```

25. ¿Existe alguna diferencia entre el programa de la pregunta 24 y este cuando se ejecutan?

```
from math import sqrt

a = float(raw_input('Valor de a: '))
b = float(raw_input('Valor de b: '))
c = float(raw_input('Valor de c: '))

if a == 0:
  if b == 0:
    if c == 0:
      print 'La ecuacion no tiene solucion'
    else:
      print 'La ecuacion tiene infinitas soluciones'
  else:
    x = -c / b
    print 'La solucion de la ecuacion es: x=%4.3f' % x
else:
  x1 = (-b + sqrt(b**2 - 4*a*c)) / (2 * a)
  x2 = (-b - sqrt(b**2 - 4*a*c)) / (2 * a)
  print 'Las soluciones de la ecuacion son: x1=%4.3f y x2=%4.3f' % (x1, x2)
```

26. ¿Existe alguna diferencia entre el programa de la pregunta 24 y este cuando se ejecutan?

```
from math import sqrt

a = float(raw_input('Valor de a: '))
b = float(raw_input('Valor de b: '))
c = float(raw_input('Valor de c: '))

if a == 0 and b == 0 and c == 0:
  print 'La ecuacion tiene infinitas soluciones'
else:
  if a == 0 and b == 0:
    print 'La ecuacion no tiene solucion'
  else:
    if a == 0:
      x = -c / b
      print 'La solucion de la ecuacion es: x=%4.3f' % x
    else:
```

```
x1 = (-b + sqrt(b**2 - 4*a*c)) / (2 * a)
x2 = (-b - sqrt(b**2 - 4*a*c)) / (2 * a)
print 'Las soluciones de la ecuacion son: x1=%4.3f y x2=%4.3f' % (x1, x2)
```

27. Modifique el programa de la pregunta 26 para utilizar la sentencia `elif`.

28. ¿Qué ocurre cuando ejecuta el programa de la pregunta 26 con los siguientes valores `a = 1`, `b = 1` y `c = 1`? Proponga una solución.

29. ¿Qué hace el siguiente programa?

```
sumatorio = 0
i = 1
while i <= 10:
  i += 1
  sumatorio += i
print sumatorio
```

30. ¿Cuál es la diferencia entre este programa y el de la pregunta 29? ¿Producen ambos el mismo resultado? ¿Por qué?

```
sumatorio = 0
i = 1
while i <= 10:
  sumatorio += i
  i += 1
print sumatorio
```

31. ¿Qué hace el siguiente programa?

```
numero = int( raw_input('Introduzca un numero '))

for potencia in [2,3,4,5]
  print '%d elevado a %d es %d' % (numero, potencia, numero**potencia)
```

32. ¿Qué muestran las siguientes sentencias? ¿Cuál es el tipo de datos de la variable a?

```
>>> a = ['pan', 'huevos', 100, 1234]
>>> a
?
>>> a[0]
?
>>> a[3]
?
>>> a[-2]
?
>>> a[1:-1]
?
>>> a[:2] + ['carne', 2*2]
?
>>> 3*a[:3] + ['Boo!']
?
>>> a
?
>>> a[2] = a[2] + 23
>>> a
?
>>> a[0:2] = [1, 12]
>>> a
?
>>> len(a)
?
```

```
>>> q = [2, 3]
>>> p = [1, q, 4]
>>> len(p)
?
>>> p[1]
?
>>> p[1][0]
?
>>> p[1].append('extra')
>>> p
?
>>> q
?
```

33. ¿Qué hacen las siguientes sentencias?

```
>>> a = ['gato', 'ventana', 'defenestrado']
>>> for x in a:
>>>   print x, len(x)
>>>
>>>?
>>> for x in a[:]:
>>>   if len(x) > 6: a.insert(0,x)
>>>     print a
>>>
>>>?
```

34. ¿Qué hacen las siguientes sentencias?

```
>>> r = range(5,10)
>>> print r
>>> ?
>>> s = range(0, 10, 3)
>>> print s
>>> ?
>>> a = ['gato', 'ventana', 'defenestrado']
>>> for i in range(len(a)):
>>>    print i, a[i]
>>> ?
```

35. ¿Qué hace el siguiente programa?

```
for i in range(0,5):
  for j in range(0,3):
    print i, j
```

36. ¿Qué hace el siguiente programa?

```
for i in range(0,5):
  for j in range(i,5):
    print i, j
```

37. ¿Qué hace el siguiente programa?

```
for i in range(0,5):
  for j in range(0,i):
    print i, j
```

38. ¿Qué hace el siguiente programa?

```
for i in range(0,4):
  for j in range(0,4):
    for k in range(0,2):
      print i, j, k
```

39. ¿Qué hace el siguiente programa?

```
for i in range(0,4):
  for j in range(0,4):
    for k in range(i,j):
      print i, j, k
```

40. ¿Qué hace el siguiente programa?

```
for i in range(1,5):
  for j in range(0,10,i):
    print i, j
```

41. ¿Qué hay mal en la siguiente función?

```
def es_perfecto(n):
  for i in range(1,n):
    sumatorio = 0
    if n % i == 0:
      sumatorio += i
  return sumatorio == n
```

42. ¿Qué hace el siguiente programa?

```
def tabla_perfectos(m):
  for i in range(1, m+1):
    if es_perfecto(i):
      print i, 'es perfecto'

x = int( raw_input('Introduzca un numero ') )
tabla_perfectos(x)
```

43. Cierre la sesión.