UD 05_05 – GNU/Linux.
Introducción a Python y uso con Bash.

CONTENIDO

1 INTRODUCCIÓN

Python es un lenguaje de programación creado por Guido van Rossum a principios de los años 90 cuyo nombre está inspirado en el grupo de cómicos ingleses “Monty Python”. Es un lenguaje similar a Perl, pero con una sintaxis muy limpia y que favorece un código legible.

Algunas características de Python son:

- Permite **tipado dinámico** (es decir permite la creación y transformación del tipo de variable en el momento de la asignación del valor a la misma, no hay que declararla con antelación)
- Es un lenguaje i**nterpretado**, no necesita compilarse para probarse o ejecutarse.
- Es **multiplataforma**, puede ejecutarse en cualquier sistema operativo.
- Cabe destacar que hay que ser cuidadosos con la **indentación** (sangría), ya que, a diferencia de otros lenguajes, esta no se utiliza únicamente para ayudar al programador/a sino que tiene un significado a la hora de anidar sentencias.

Solo vamos a ver una pequeña introducción a Python para adaptar algunos de nuestros scripts en Linux a este lenguaje. Python ofrece muchísimas más posibilidades.

2 WEB Y DOCUMENTACIÓN

- Puedes encontrar mucha más información y documentación en la web de Python:

  https://www.python.org/

- Tutoriales:

  https://wiki.python.org/moin/BeginnersGuide/Programmers

- También es posible utilizar Python para scripting en Windows:

  https://docs.microsoft.com/es-es/windows/python/scripting

3 INSTALACIÓN DE PYTHON 3.9

Utilizaremos la versión 3 de Python (hay diferencias en cuanto a la sintaxis con la versión 2, así que, si no tienes instalada alguna versión 3.x, deberás hacerlo para no tener problemas con los comandos que se incluyen en esta práctica).

Comprueba si tu distribución ya tiene Python instalado, prueba a poner en la terminal python3. Si está instalado, obtendrás algo parecido a esto:

```
$python3

Python 3.8.10 (default, Nov 26 2021, 20:14:08)

[GCC 9.3.0] on linux

Type "help", "copyright", "credits" or "license" for more information.

>>>
```

Para salir del entorno Python, escribe:

```
>>>exit()
```

Para instalar la versión 3.9 de Python, escribimos en la terminal:

```
$ sudo apt update

$ sudo apt install python3.9
```

4 PRIMER PROGRAMA EN PYTHON

Utilizaremos un editor de textos, como el que usamos para escribir los scripts de Bash. Puede ser nano, el editor por defecto de nuestra distribución (escribe en el lanzador de aplicaciones “editor” y te aparecerá) u otro como idle, emacs, Bluefish, etc.

Escribimos lo siguiente y guárdalo como hola.py, por ejemplo:

```
#!/usr/bin/python3.9

print("Hola, mundo.")
```

NOTA: En caso de que el intérprete no esté en /usr/bin, usa esto como primera línea:

```
#!/usr/bin/env python3.9
```

Para ejecutarlo, escribe en la terminal:

```
$ python3.9 hola.py
```

o da permisos de ejecución a hola.py y, estando en el directorio donde se encuentra el fichero, ejecútalo con:

```
$ ./hola.py
```

Para probar el funcionamiento de los ejemplos que se plantean en el resto del tema, incluye las líneas del ejemplo en un fichero (o varios) y ejecútalo.

5 VARIABLES

#Declarar una variable y sacar su valor por pantalla:

```
num=6

print(num)

texto=Hola

print(texto)
```

#Devolver por pantalla el tipo de la variable num:

```
print(type(num),type(texto))
```

#Para sacar varias líneas por pantalla, escribir cada línea y encerrarlas entre """ """

```
varias="""primera linea

segunda linea"""

print(varias)
```

6 INTRODUCCIÓN DE INFORMACIÓN: INPUT

#Quedar a la espera de que el usuario introduzca algo por teclado. Para finalizar la introducción, el usuario debe pulsar intro.

```
input()
```

#Solicitar información en tiempo de ejecución y guardarla en una variable:

```
centro="IES Abastos"

num=4

nombre=input("¿Cómo te llamas?")

print(nombre,"está matriculado/a en",centro,"de Valencia")
```

7 OPERACIONES Y NÚMEROS

#Mostrar por pantalla el resultado de una operación:

```
print(17 / 3)
```

#Pedir un número que debe ser un entero, si no, dará error.

```
print("Dime un numero\n")

num=int(input())
```

#Para numeros reales, `float` en lugar de `int`

7.1. OPERADORES

7.1.1. OPERADORES DE COMPARACIÓN

Operador	Descripción
>	Mayor que. `True` si el operando de la izquierda es estrictamente mayor que el de la derecha; `False` en caso contrario.
>=	Mayor o igual que. `True` si el operando de la izquierda es mayor o igual que el de la derecha; `False` en caso contrario.
<	Menor que. `True` si el operando de la izquierda es estrictamente menor que el de la derecha; `False` en caso contrario.
<=	Menor o igual que. `True` si el operando de la izquierda es menor o igual que el de la derecha; `False` en caso contrario.
==	Igual. `True` si el operando de la izquierda es igual que el de la derecha; `False` en caso contrario.
!=	Distinto. `True` si los operandos son distintos; `False` en caso contrario.

7.1.2. OPERADORES ARITMÉTICOS

Operador	Descripción
+	Suma dos operandos.
-	Resta al operando de la izquierda el valor del operando de la derecha. Utilizado sobre un único operando, le cambia el signo.
*	Producto/Multiplicación de dos operandos.
/	Divide el operando de la izquierda por el de la derecha (el resultado siempre es un `float`).
%	Operador módulo. Obtiene el resto de dividir el operando de la izquierda por el de la derecha.
//	Obtiene el cociente entero de dividir el operando de la izquierda por el de la derecha.
**	Potencia. El resultado es el operando de la izquierda elevado a la potencia del operando de la derecha.

8 IF ELSE

```
if num == 1:
      print("El número es uno")
else:
      print("El número no es uno")
```

9 WHILE

```
edad = 0
while edad < 18:
      edad = edad + 1
      print("Felicidades, tienes ",edad)
```

#Bucle infinito y uso de BREAK. Devolverá por pantalla lo que escriba el usuario. Solo se saldrá si se teclea adios.

```
print("Ahora solo saldrás si escribes adios")
while True:
      #con la siguiente línea se queda a la espera de que el usuario
      #introduzca algo, al principio de línea tendremos >
      entrada = input("> ")
      if entrada == "adios":
            break
      else:
            print(entrada)
```

10 FUNCIONES

#Declaración de la función:

```
def nombre_funcion():
      instrucciones
```

#Invocar la función:

```
nombre_funcion
```

#Por ejemplo:

```
def di_hola():
    print("Hola, mundo.")

di_hola()
```

#Devolver el valor de una función:

```
def suma(a, b):
    return a + b


resul = suma(4, 8)
print(resul)
```

Más sobre funciones: https://python.land/introduction-to-python/functions

11 EJECUTAR UN COMANDO DE LINUX EN PYTHON

#Fijaos dónde hay que escribir el comando, en este caso, `ls -l`:

```
import shlex, subprocess
subprocess.call(shlex.split('ls -l'))
```

#Vamos a crear una función para abreviar un poco el uso de comandos en caso de que tengamos que usarlos varias veces en un script.

```
import shlex, subprocess
def com_bash(comando):
       subprocess.call(shlex.split(comando))


#El comando se solicita al usuario:
comando=input("Dime el comando Bash: ")
com_bash(comando)


#El comando se especifica dentro del script
com_bash('ls -l')
```

12 LECTURA DE FICHEROS

Vamos a ver cómo adaptar los scripts en que recorríamos ficheros con while read. Deberemos importar el módulo de Python csv.

En el siguiente ejemplo se muestra cómo abrir un fichero, leer línea a línea y cada campo en una línea. Se indica el delimitador (en este caso ;) y se utilizar un vector para nombrar cada uno de sus campos, ya que dentro del fichero no hay una primera línea de cabeceras.

```
#!/usr/bin/python3.9

import csv

with open('/etc/passwd', newline='') as fichero:

      campos = ['usuario','x','id_usuario','id_grupo']

      linea = csv.DictReader(fichero, fieldnames=campos, delimiter=':')

      for columna in linea:

            if int(columna['idusu']) >= 1000:

                  print(columna['usuario'])
```

Más información: https://docs.python.org/es/3/library/csv.html

13 MENÚ

#A continuación se muestra un ejemplo para ofrecer un menú de opciones:

```
#!/usr/bin/python3

def pedirNumeroEntero():

    correcto=False
    num=0
    while(not correcto):
        try:
            num = int(input("Introduce un numero entero: "))
            correcto=True
        except ValueError:
            print('Error, introduce un numero entero')

    return num
```

```
salir = False
opcion = 0

while not salir:

    print ("1. Opcion 1")
    print ("2. Opcion 2")
    print ("3. Opcion 3")
    print ("4. Salir")

    print ("Elige una opcion")

    opcion = pedirNumeroEntero()

    if opcion == 1:
        print ("Opcion 1")
    elif opcion == 2:
        print ("Opcion 2")
    elif opcion == 3:
        print("Opcion 3")
    elif opcion == 4:
        salir = True
    else:
        print ("Introduce un numero entre 1 y 3")

print ("Fin")
```

#Fuente: https://www.discoduroderoer.es/crear-un-menu-de-opciones-en-consola-en-python/

14 EJERCICIOS

Adapta algunos de los scripts realizados en Bash a Python. Entrega los ejercicios 1 y 5 del documento pdf UD 05_Act 03.