



USMP
UNIVERSIDAD DE
SAN MARTIN DE PORRES

CIXOS-FIA
"COMUNIDAD QUE INVESTIGA EL PROYECTO OLPC"



Software Freedom Day

“Lenguaje de Programación Python”

Expositor: Hernán Martín Inga Saavedra

PYTHON

- Python es un lenguaje, dinámico, SIMPLE (Fácil de aprender) y PODEROSO.
- Creado en Holanda en 1991 por Guido van Rossum.
- Soporta tres estilos o paradigmas de programación:
 - Estructurada
 - Orientada a objetos
 - Funcional



PYTHON

Usado en todo tipo de aplicaciones: científicas, administración de sistemas, procesamiento de texto, paginas web, bases de datos, visualización 3D y videojuegos, inteligencia artificial, etc.



PYTHON

- Presenta una sintaxis compacta, sencilla e intuitiva, una curva de aprendizaje mínima, junto a una potente librería de funciones y clases.
- Lo anterior hace posible programar una aplicación completa en cosa de horas o incluso minutos.



¿POR QUÉ USAR PYTHON?

- Diseño simple e internamente consistente.
- Código usualmente muy claro y legible.
- Alto nivel: operaciones complejas en muy pocas líneas.
- Módulos que facilitan un espectro de tareas.
- Versatilidad: distintos paradigmas de programación.



CARACTERÍSTICAS DE PYTHON

- Python es un lenguaje interpretado
- Un programa Python no se compila, simplemente se ejecuta
- Esto hace posible cosas prácticamente imposibles en otros lenguajes:
 - Ejecutar instrucciones de Python interactivamente.
 - Interpretar un string como código Python y ejecutarlo.
 - Crear funciones y clases “al vuelo”, es decir, mientras el programa corre.
 - Introspeccion: un programa Python puede “analizarse a sí mismo”

IMPRIMIR DATOS

Hay 2 formas simples de visualizar en Python

- 1era Forma

"Hola Mundo"

'Hola Mundo'

- 2da Forma, empleando el comando **print**

print "Hola Mundo"

Hola Mundo

VARIABLES EN PYTHON

- En Python no hace falta declarar las variables, solo hay que inicializarlas con algún valor antes de usarlas:

```
n1 = 112
```

```
n2 = 23.1
```

```
nombre1 = 'SFD09'
```

```
nombre2 = "SFD09"
```

```
bool = True
```

Esto no quiere decir que Python
no tenga tipos de variables!!

VARIABLES EN PYTHON

- Tipos de Variable
 - ✓ **Entero(int)**: 2, 3, 4, etc.
 - ✓ **Cadena(str)**: “Hola Mundo”, 'Hola Python', etc.
 - ✓ **Punto Flotante(float)**: 2.1, 3.5, etc.
 - ✓ **Boolean(bool)**: True, False.

INGRESO DE VARIABLES

- En Python existen 2 formas de ingresar variables:
 1. `raw_input("abcdef")` Recibe todo tipo de variables pero las almacena como str.
 2. `input("abcdef")` Recibe solo variables de tipo numérico pero las almacena como int.

INGRESO DE VARIABLES

- Ejemplos:

```
a = raw_input("Ingrese su nombre: ")
```

```
b = raw_input("Ingrese su edad: ")
```

```
print a+b
```

```
a = raw_input("Ingrese su nombre: ")
```

```
b = input("Ingrese su edad: ")
```

```
print a, " ",b
```

OPERADORES DE CONDICIÓN

Son los operadores que se usan en sentencias especiales.

$x == y$ # x es igual a y

$x != y$ # x no es igual a y

$x > y$ # x es mayor que y

$x < y$ # x es menor que y

$x >= y$ # x es mayor o igual que y

$x <= y$ # x es menor o igual que y

OPERADORES LÓGICOS

Son los operadores que se emplean para relacionar 1 o más condiciones.

$x \geq 1$ **and** $x \leq 5$

$x \geq 1$ y $x \leq 5$

$x == 1$ **or** $x == 2$

$x == 1$ o $x == 2$

not ≤ 4

no condición1

SENTENCIAS DE DECISIÓN

- Existen 3 sintaxis de sentencias de decisión en Python:
 1. Sentencias de decisión Simple
 2. Sentencias de decisión Doble
 3. Sentencias de decisión Múltiple

SENTENCIAS DE DECISIÓN

- Sentencias de decisión Simple

if condición:

instrucción 1

instrucción n

Ejemplo:

if a >= 0:

print "Es positivo!!!"

SENTENCIAS DE DECISIÓN

- Sentencias de decisión Doble

if condición:

instrucción 1

instrucción 2

else:

instrucción 1

instrucción n

SENTENCIAS DE DECISIÓN

Ejemplo:

```
if x >= 0:
```

```
    print 'El número es Positivo'
```

```
else:
```

```
    print 'El número es Negativo'
```

SENTENCIAS DE DECISIÓN

- Sentencias de decisión Múltiple

if condición1:

instrucción 1

instrucción 2

elif condición2:

instrucción 1

instrucción n

else:

instrucción 1

SENTENCIAS DE DECISIÓN

Ejemplo:

```
if x>10:
```

```
    print 'El número es mayor que 10'
```

```
elif x<10:
```

```
    print 'El número es menor que 10'
```

```
else:
```

```
    print 'El número es 10'
```

SENTENCIAS DE REPETICIÓN

- Existen 2 sintaxis de sentencias de repetición.
 1. Sentencia de Repetición for
 2. Sentencia de Repetición while

SENTENCIAS DE REPETICIÓN

- Sentencia de Repetición for

for contador **in** range(inicio,final):

instrucción 1

instrucción n

Ejemplo:

for i **in** range(0,5):

print i

SENTENCIAS DE REPETICIÓN

- Sentencia de Repetición while

while condicion:

instrucción 1

instrucción n

Ejemplo:

while x<10:

print x

x=x+1

CADENAS

- Las cadenas son textos encerrados entre comillas simple ' ' o entre doble comillas " ".
- Entre las comillas se pueden agregar caracteres especiales como \n (nueva línea) y \t (tabulación)

Ejemplos:

✓ `c='Peru'`

`print "SFD09 \t", c`

`print 'SFD09 \n', c`

CADENAS

- Operaciones con cadenas:

`c = "cadena"` `c = 'cadena'`

1. Suma:

`print c+c`

2. Multiplicación

`print c*3`

CADENAS

- Cómo recorrer una cadena:

```
c = 'cadena'
```

```
c = "cadena"
```

```
for i in c:
```

```
    print i
```

```
prefijos = "JKLMNOPQ"
```

```
sufijo = 'ack'
```

```
for letra in prefijos:
```

```
    print letra + sufijo
```

BOOLEANOS

- Los boolean o variables lógicas, solo existen 2 posibles valores:
 1. True
 2. False

BOOLEANOS

- Operadores Lógicos o condicionales:

and # r = True **and** False

print r

False

Rpta.

or # r = True **or** False

print r

True

Rpta.

not # r = **not** True

print r

False

Rpta.

BOOLEANOS

- Expresiones que emplean operadores relacionales (comparación de valores).

print 5 == 2 Rpta. False

print 7 <= 3 Rpta. False

print 9 >= 6 Rpta. True

print 5 > 2 Rpta. True

print 4 < 1 Rpta. False

print 8 != 10 Rpta. True

LISTAS O ARREGLOS

- Las listas pueden contener cualquier tipo de dato: números, cadenas, booleanos y también listas.

```
lista = [22, True, "una lista", [1, 2]]
```

```
print lista[0]      #Rpta. 22
```

```
print lista[1]      #Rpta. True
```

```
print lista[2]      #Rpta. "una lista"
```

```
print lista[3]      #Rpta. [1,2]
```

FUNCIONES

- Una función es:
 - ✓ Un conjunto de instrucciones.
 - ✓ Que se la invoca mediante un nombre.
 - ✓ Luego de ejecutado el algoritmo puede retornar un valor.

FUNCIONES

- Existen 2 tipos de Funciones:
 1. Funciones sin Retorno
 2. Funciones con Retorno

FUNCIONES

- Funciones sin Retorno

Sintaxis:

```
def nombreFuncion(parámetros):
```

```
    instrucción 1
```

```
    instrucción n
```

Ejecución:

```
nombreFuncion(parámetros)
```

FUNCIONES

- Ejemplo:

```
def tabla(n):
```

```
    for i in range(1,13):
```

```
        print n, " x ",i, " = ", n*i
```

```
tabla(7)
```

FUNCIONES

- Funciones con Retorno

def nombreFuncion(parámetros):

instrucción 1

instrucción n

return valor

Ejecución:

a = nombreFuncion(parámetros)

FUNCIONES

- Ejemplo:

```
def mayor(n,m):  
    if n<m:  
        return m  
    else: return n
```

```
a = mayor(4,7)
```

```
print a
```

GRACIAS PREGUNTAS.

Mail: cixosfia@libreusmp.org

Web: <http://cixosfia.libreusmp.org>

