

# Programación en Python

Jesús Espino García  
jespinog@gmail.com  
@jespinog

Gul UC3M Nov 2011



**Kaleidos**  
beautiful code

7 de Noviembre de 2011

# ¿Qué es Python?

- Lenguaje de programación.
- Creado por Guido Van Rossum en las navidades de 1989.
- Basado en ABC y Modula-3.
- En febrero de 1991 pasa a USENET.
- A partir de entonces el lenguaje no ha dejado de crecer.
- Actualmente coexisten dos versiones paralelas, 2.7 y 3.2.

# ¿Por qué Python?



- Fácil de aprender.
- Sencillo de usar.
- Potente.

# ¿Por qué Python?

- Libre (y gratis).
- Fácil de escribir.
- Fácil de leer.
- Fácil de mantener.
- De proposito general.
- Alto nivel.
- Orientado a objetos.
- Interpretado.
- Introspectivo.

# ¿Por qué Python?

- Extensible.
- Completo.
- Dinamico.
- Robusto.
- Multiplataforma.
- Herencia multiple.
- List comprehensions.
- Funciones lambda.
- Clausuras.
- Bien documentado.
- Documentación integrada en el language.

# ¿Quién lo usa?

- Google.
- Microsoft.
- IBM.
- NASA.
- MIT.
- Yahoo!.
- HP.
- DOD.
- En resumen, todo el mundo.

# Variables

- Variables dinamicas.
- Principales tipos de datos:
  - Booleanos (bool)
  - Numericos (int, float, long, complex)
  - Secuencias (str, unicode, list, tuple, bytearray)
  - Sets (set, frozenset)
  - Internos (module, function, instancemethod, instance, classobj)
- Distincion entre mutables y no mutables:
  - Mutable: Se cambia el valor del objeto. (Listas, ByteArrays y Sets)
  - Inmutable: Al cambiar se sobrescribe el objeto. (Numericos, Booleanos, Strings, Tuplas y Fronzensets)

# Listas

- Son mutables.
- Se identifica por []
- Lista vacía: []
- Lista con datos homogéneos: [1,2,3,4]
- Lista con datos heterogéneos:  
[1,(2,4),"cadena",["gul","linux","python"]]
- Acceso a un elemento: lista[indice]
- Listas dentro de listas: lista[indice1][indice2]..[indiceN]
- Acceso desde el final: lista[-indice]



# Tuplas

- Son inmutables.
- Se identifica por ()
- Tupla vacía: ()
- Tupla de un elemento: (1,)
- Tupla con datos homogéneos: (1,2,3,4)
- Tupla con datos heterogéneos:  
(1,(2,4),"cadena",["gul","linux","python"])
- Acceso a un elemento: `tupla[indice]`
- Tuplas dentro de tuplas: `tupla[indice1][indice2]..[indiceN]`
- Acceso desde el final: `tupla[-indice]`

# Diccionario

- Son mutables.
- Se identifica por {}
- Diccionario vacío: {}
- Diccionario con datos: {"nombre": "Jesus", "apellido": "Espino"}
- Acceso a un elemento: `diccionario[clave]`
- Diccionesarios dentro de diccionarios:  
`diccionario[clave1][clave2]..[claveN]`

# Introspección con dir()

- Python es introspectivo.
- La funcion `dir()` nos muestra el contenido de un objeto.
- Metodos, atributos, modulos, todo son objetos y pueden estar contenidos unos en otros.
- `dir()` sin parametros nos muestra información del contexto principal.

```
>>> dir()  
['__builtins__', '__doc__', '__name__', '__package__']
```

# Obteniendo ayuda con help()

- La función `help()` nos muestra la documentación de un objeto
- Por ejemplo `help([].append)`

```
>>> help([].append)
```

```
Help on built-in function append:
```

```
append(...)
```

```
    L.append(object) -- append object to end
```

# Indentado

- En python en indentado es obligatorio.
- Forma parte de la sintaxis.
- Sirve para definir donde empieza y termina un bloque.

Ejemplo:

```
if variable == 10:  
    print "La variable es 10"  
else:  
    print "La variable no es 10"
```

# Condiciones

```
if variable == 10:  
    print "La variable es 10"  
elif variable == 20:  
    print "La variable es 20"  
else:  
    print "La variable no es ni 10, ni 20"
```

# Bucle while

```
while value < 10:  
    print value  
    value += 1  
else:  
    print "End"
```

- break sale el bucle.
- continue pasa a la siguiente iteración.

# Bucle for

```
for value in lista:  
    print value  
else:  
    print "End"
```

- break sale el bucle.
- continue pasa a la siguiente iteración.



# Definición de funciones

```
def funcion(arg1, arg2="default", *args, **kwargs):  
    'Documentación de la funcion'  
    print arg1  
    print arg2  
    print args  
    print kwargs  
    return "Valor"  
  
funcion(1)  
funcion(1,2,3,4,prueba=5)
```

# Ficheros de python

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

def mifuncion():
    print "Hola Mundo"

if __name__=='__main__':
    mifuncion()
```

# Entrada por teclado

```
x=input("Obtener valor: ")  
pritrn x  
y=raw_input("Obtener valor: ")  
print y
```

# Parámetros del programa

```
import sys
nombre_de_script = sys.argv[0]
primer_parametro = sys.argv[1]
segundo_parametro = sys.argv[2]
```

# Acceso a ficheros

```
fichero = open('fichero.txt','r')
datos = fichero.read()
fichero.close()
fichero2 = open('fichero2.txt','w')
fichero2.write(datos)
fichero2.close()
```

# Clases

```
# Estilo nuevo, hereda de object
class MiClase(object):
    dato = None

    def set_dato(self, dato):
        self.dato=dato

    def display(self):
        print self.dato

    def get_dato(self):
        return self.dato
```

# Herencia

```
class OtraClase(MiClase):
    def display(self):
        print "El valor actual es %s" % (self.dato)

    def get_dato(self):
        return super(OtraClase, self).get_dato(self) + 1

# Con estilo antiguo
class OtraClase(MiClase):
    def get_dato(self):
        return MiClase.get_dato(self) + 1
```

# Metodos especiales

- `__init__`: Constructor.
- `__del__`: Destructor.
- `__add__`: Operador de suma.
- `__or__`: Operador Or lógico.
- `__getitem__`: Indexación.
- `__setitem__`: Asignación indexada.
- `__getslice__`: Seleccionar una parte.
- `__repr__`: Para salida por pantalla.
- `__len__`: Longitud.
- `__cmp__`: Comparación.
- `__unicode__`: Representación unicode.
- `__str__`: Representación en string.



# Excepciones

```
try:
    x = int(variable)
except ValueError:
    print "La variable no es un entero"
else:
    print "Todo ha ido ok"
finally:
    print "Pase lo que pase se ejecuta esto"

raise "Excepcion personalizada"
```

# Módulos

- Bloques de código que agrupan funciones, clases y variables.
- Pueden ser ficheros .py.
- Pueden ser directorios que contengan un `__init__.py`.

```
fichero1.py:  
def mifuncion():  
    print "Hola Mundo"
```

```
fichero2.py  
import fichero1  
fichero1.mifuncion()
```

# Referencias

- Proyecto python: <http://www.python.org>.
- Documentación de python: <http://docs.python.org>.
- Python-ES: [python-es@python.org](mailto:python-es@python.org)

Introducción  
Tipos básicos  
Buceando dentro de python  
Escribiendo código  
Escribiendo programas  
**Para terminar**

# Dudas

...