Universidad Zaragoza

Inteligencia Artificial (30223)

Material Adicional. Introducción a Python orientada al trabajo práctico

Universidad Zaragoza

Curso 2015-2016

Javier Civera
Dpto. Informática e Ingeniería de Sistemas.

Índice

- Introducción
- Historia
- Distribuciones / IDEs
- Tutoriales
- Sintaxis

Python - Introducción

- Lenguaje interpretado, de propósito general y alto nivel.
- Fuertemente tipado
- Dinámicamente tipado.
- Soporta varios paradigmas: Imperativo, orientado a objetos (todo es un objeto) y funcional.
- Diseñado con el objetivo de facilitar la legibilidad del código y la productividad del programador.
- Las distribuciones de referencia de Python son libres y de código abierto.
- Ampliamente usado, sobre todo en prototipado y ámbitos científicos.

Python - Releases

- Concebido en 1980 y lanzado por primera vez en 1989 por Guido van Rossum (Holanda).
- 1994: Python 1.0
- 2000: Python 2.0
- 2008: Python 3.0
- La versión recomendada es Python 2.7.

Distribuciones - IDEs

- Anaconda
  - Distribución orientada a procesamiento de datos, analítica predictiva y computación científica.
  - Contiene la librería BSD **scikit-learn** para aprendizaje automático, que utilizaremos en el trabajo práctico.
  - **Scikit-learn** tiene dependencias con las librerías de cálculo numérico y cálculo científico NumPy y SciPy.
  - Anaconda tiene su propia consola y su editor.
- Python is integrated in Eclipse
  - PyDev: Open Source plugin for Eclipse
- PyCharm: Python IDE
- Spyder
  - Intérprete libre de Python "Matlab-like" bastante amigable.

Tutoriales

- https://docs.python.org/2/tutorial/ (Tutorial oficial)
- http://www.learnpython.org/ (Tutorial interactivo, con un intérprete de Python en la propia web)
- http://learnpythonthehardway.org/book/
- …

Sintaxis

- Indentación (sangrado): Python usa indentación en lugar de corchetes o paréntesis para delimitar bloques de código.

```
>>> x = int(raw_input("Please enter an integer: "))
Please enter an integer: 42
>>> if x < 0:
...     x = 0
...     print 'Negative changed to zero'
... elif x == 0:
...     print 'Zero'
... elif x == 1:
...     print 'Single'
... else:
...     print 'More'
...
More
```

Comentarios

- Comentarios de una línea: #
- Comentarios de múltiples líneas: “”” “”””

Python Comments

```
1
2  price = 10.2
3  # increase price to 5%
4  price = price * 1.05
5
6  salary = 5000
7  salary = salary * 1.02  # increase salary 2% for the employee
8
9  def increase_salary(sal,rating,percentage):
10     """ increase salary base on rating and percentage
11     rating 1 - 2 no increase
12     rating 3 - 4 increase 5%
13     rating 4 - 6 increase 10%
14
15     """
```

Strings

- Los strings pueden concatenarse con el operador +
- Los caracteres se acceden con []

```
>>> # 3 times 'un', followed by 'ium'
>>> 3 * 'un' + 'ium'
'unununium'
```

```
>>> word = 'Python'
>>> word[0]  # character in position 0
'P'
>>> word[5]  # character in position 5
'n'
```

Listas

```
>>> letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
>>> letters
['a', 'b', 'c', 'd', 'e', 'f', 'g']
>>> # replace some values
>>> letters[2:5] = ['C', 'D', 'E']
>>> letters
['a', 'b', 'C', 'D', 'E', 'f', 'g']
>>> # now remove them
>>> letters[2:5] = []
>>> letters
['a', 'b', 'f', 'g']
>>> # clear the list by replacing all the elements with an empty list
>>> letters[:] = []
>>> letters
[]
```

```
>>> letters = ['a', 'b', 'c', 'd']
>>> len(letters)
4
```

Bucles (for)

- Los bucles for iteran sobre los componentes de una secuencia (string, lista)

```
>>> # Measure some strings:
... words = ['cat', 'window', 'defenestrate']
>>> for w in words:
...     print w, len(w)
...
cat 3
window 6
defenestrate 12
```

Funciones

```
>>> def fib(n):    # write Fibonacci series up to n
...     """Print a Fibonacci series up to n."""
...     a, b = 0, 1
...     while a < n:
...         print a,
...         a, b = b, a+b
...
>>> # Now call the function we just defined:
... fib(2000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597
```

Módulos

```
# Fibonacci numbers module

def fib(n):    # write Fibonacci series up to n
    a, b = 0, 1
    while b < n:
        print b,
        a, b = b, a+b

def fib2(n): # return Fibonacci series up to n
    result = []
    a, b = 0, 1
    while b < n:
        result.append(b)
        a, b = b, a+b
    return result
```

```
>>> import fibo
```

```
>>> fibo.fib(1000)
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
>>> fibo.fib2(100)
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
>>> fibo.__name__
'fibo'
```

Universidad Zaragoza
1542

Inteligencia Artificial

(30223) Grado en Ingeniería Informática

Lección 14. Aprendizaje
AIMA 18.1 a 18.7
Tema 5 de www.ai-class.com