```
>>> import time
>>> print "Python CeibalJAM", time.ctime()
   Python CeibalJAM Sat Nov 7 08:29:03 2009
```

Números:

```
>>> a = 15
>>> print a, 15.0, 0xF, 017, (15+0j), hex(a)
```

Cadenas: "..", '..'

```
>>> print 'Hooola',"Mundo"
Hooola Mundo
>>> print 'Hola %s' % 'mundo'
Hola mundo
>>> s = 'Hola %s%s%s' % ('m','u'*8,'ndo')
>>> print s
Hola muuuuuuuundo
```

Listas: []

```
>>> pares = [0,2,4,6,8]
>>> pares[0]
0
>>> pares[0:2]
[0, 2]
>>> pares[-1]
8
>>> pares[2:]
[4, 6, 8]
>>> pares.append(9)
>>> pares.remove(9)
>>> 6 in pares
True
```

Bucles: for x in y, while x

```
>>> for par in pares:
...     print par
...
>>> for l in 'hola':
...     print l
...
>>> pares.reverse() # sobre la misma lista
>>> pares
[8, 6, 4, 2, 0]
>>> pares.sort()    # sobre la misma lista
>>> pares
[0, 2, 4, 6, 8]
>>> while pares:
...     pares = pares[:-1]
...     print pares
[0, 2, 4, 6]
[0, 2, 4]
[0, 2]
[0]
[]
>>> range(10)       # for i in range(10)....
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Listas por comprensión:

```
>>> [x*2 for x in range(5)]
[0, 2, 4, 6, 8]
>>> [x for x in range(10) if x % 2 == 0]
[0, 2, 4, 6, 8]
```

Tuplas:

```
>>> punto = (1,2)
>>> x,y = punto
>>> print punto, x, y
(1, 2) 1 2
>>> print list(punto)
[1, 2]
```

Documentación: help, __doc__

```
>>> help(pares.append)
>>> print pares.append.__doc__
L.append(object) -- append object to end
>>> for f in dir(pares):
...     print f
...
```

Diccionarios: {}, { 'a':1, 2:'3', 15.0 : [] }

```
>>> dic = {}
>>> dic[5] ='Dicese de numero que precede al 6'
>>> dic['jam'] = 'jalea','improvisacion'
>>> dic['jam']
('mermelada', 'improvisacion')
>>> dic
{'jam': ('jalea', 'improvisacion'), 5: 'Dicese
de numero que precede al 6'}
>>> dic.keys()
['jam', 5]
>>> dic.values()
[('jalea', 'improvisacion'), 'Dicese de numero
que precede al 6']
>>> dic.items()
```

Funciones: def x(a1,a2):, def x(a1=2), def x(a1, *args):

```
>>> def check(x):
...     if x:          # condicional... obvio.
...             return True
...     else:
...             return False
>>> check([]) or check(0) or check('') or
check(False) or check(None)
False
>>> check(['a',3]) and check(15) and
check('foo!') and check(True) and check(check)
True
>>> def compleja(x,y=2,*args, **kwargs):
...     print x, y, args, kwargs
>>> compleja(1)
1 2 () {}
>>> compleja(1,4)
1 4 () {}
>>> compleja(1,y=4)
1 4 () {}
>>> compleja(1,5,'a','b')
1 5 ('a', 'b') {}
>>> compleja(1,5,'a','b',muy='compleja')
1 5 ('a', 'b') {'muy': 'compleja'}
>>> f = compleja # referencias a funciones!
>>> f(1,foo='bar')
1 2 () {'foo': 'bar'}
```

Módulos: import time, from re import match

```
>>> import time
>>> time.time()
1243684019.0555091
>>> from re import match #expresiones regulares
>>> match('j[Aa]*m', 'bar?') and 'matched!'
>>> match('j[Aa]*m', 'jaAaAAam') and 'matched!'
'matched!'
```

Clases: class Foo(object):

```
>>> class XO(object): # object == superclase
...     def __init__(self, id): # Constructor
...             # Llama a constructor del padre
...             super(XO,self).__init__()
...             self.id = id
...             self.prendida = False
...     def prender(self): # Metodo
...             """
...             Prende una XO
...             """
...             assert not self.prendida
...             self.prendida = True
>>> xo = XO(10)
>>> xo.prender()
>>> xo.prendida
True
```

Métodos especiales

```
__init__(self, ...) # Constructor
__str__(self)# Pasar a cadena (ej. para print)
__eq__(self, other) # True si es igual a other
#...etc... para verlos -> dir(XO())
```