

Tema 0 - Programación básica con Python

Curso de Física Computacional

M. en C. Gustavo Contreras Mayén

9 de agosto de 2012

Contenido

- 1 Introducción
- 2 Iniciando con Python
- 3 Operadores relacionales
- 4 Variables
- 5 Listas y Tuplas

Contenido

- 1 Introducción
- 2 Iniciando con Python
- 3 Operadores relacionales
- 4 Variables
- 5 Listas y Tuplas

Contenido

- 1 Introducción
- 2 Iniciando con Python
- 3 Operadores relacionales
- 4 Variables
- 5 Listas y Tuplas

Contenido

- 1 Introducción
- 2 Iniciando con Python
- 3 Operadores relacionales
- 4 Variables
- 5 Listas y Tuplas

Contenido

- 1 Introducción
- 2 Iniciando con Python
- 3 Operadores relacionales
- 4 Variables
- 5 Listas y Tuplas

Para el curso de Física Computacional será necesario que usemos un lenguaje de programación para apoyarnos en la solución de los problemas y algoritmos.

El lenguaje de nuestra elección es un medio para alcanzar nuestro objetivo del curso, más no el fin, por lo que revisaremos lo más básico de Python, dando la oportunidad de que por tu cuenta, logres un mayor conocimiento y práctica con Python.



- Lenguaje de programación de alto nivel, interpretado.
- Desarrollado por Guido van Rossum a principios de los años 90.
- Es multiplataforma (UNIX, Solaris, Linux, DOS, Windows, OS/2, Mac OS, etc.)
- Software libre: Python Software Foundation License (PSFL)
- Tipado dinámico.
- Fuertemente tipado.
- Orientado a objetos.

Tipado dinámico

Cada dato es de un tipo determinado y sólo se puede operar con él de formas bien definidas.

La ventaja es que **NO** hay que declarar variables antes de **SU USO**.

Fuertemente tipado

Se dice que es un lenguaje cuyos tipos son estrictos. Java y Python son fuertemente tipados.

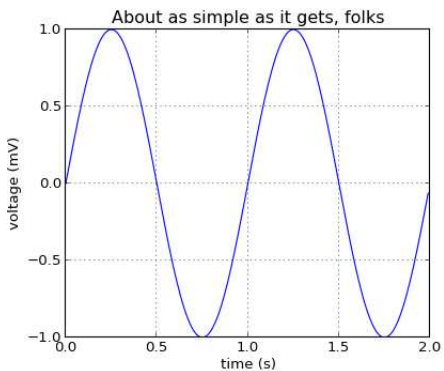
Si tiene un tipo de dato entero, no puede tratarlo como una cadena de texto sin convertirlo explícitamente.

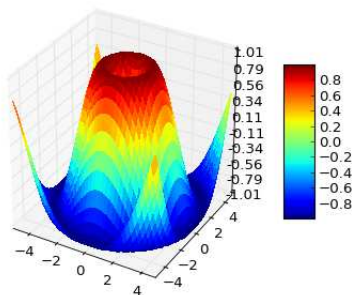
Programación orientada a objetos

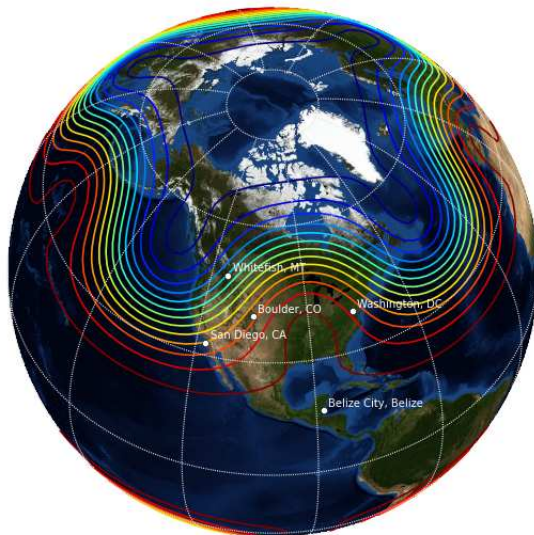
La programación orientada a objetos es un paradigma de programación que busca representar entidades u objetos agrupando datos y métodos que puedan describir sus características y comportamientos.

Complementos para Python

- NumPy: paquete fundamental para computación científica.
- SciPy: librería para computación científica (extiende a NumPy)
- matplotlib: librería para gráficos 2D (soporta gráficos 3D también)
- Mayavi: librería para gráficos y visualización de datos 3D.
- iPython: consola interactiva para Python.





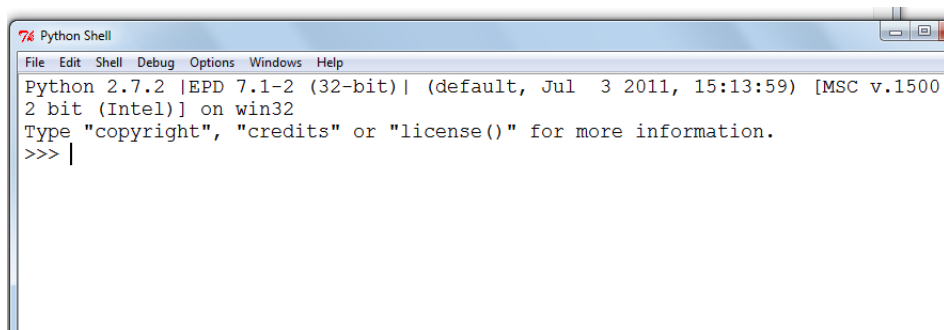


Consola de trabajo en Python

Hay dos modos de trabajo en Python, cada uno de ellos depende de nuestra habilidad:

- **Modo rudo:** trabajo directo en la consola.
- **Modo amigable:** a través de una interface IDLE (Entorno de Desarrollo Integrado)

Modo Rudo



A screenshot of a Windows-style application window titled "Python Shell". The window has a menu bar with "File", "Edit", "Shell", "Debug", "Options", "Windows", and "Help". The main text area displays the following text: "Python 2.7.2 |EPD 7.1-2 (32-bit)| (default, Jul 3 2011, 15:13:59) [MSC v.1500 2 bit (Intel)] on win32", followed by "Type 'copyright', 'credits' or 'license()' for more information.", and finally the prompt ">>> |".

```
Python 2.7.2 |EPD 7.1-2 (32-bit)| (default, Jul 3 2011, 15:13:59) [MSC v.1500
2 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> |
```

Python como calculadora

Una vez abierta la sesión en Python, podemos aprovechar al máximo Python, una de las primeras facilidades que tenemos, es que contamos con una calculadora a la mano, sólo hay que ir escribiendo las operaciones.

Operadores aritméticos

```
>>> 3+4  
7
```

```
>>> 3/4  
0
```

```
>>> 3.0/4.0  
0.75
```

```
>>> 5.0 / 10 * 2 + 5  
6
```

```
>>> 5.0 / (10 * 2 + 5)  
0.2
```

```
>>> 2**3**2  
512
```

```
>>> (2**3)**2  
64
```

```
>>> 17%3%2  
0
```

Operadores aritméticos

```
>>> 3+4  
7
```

```
>>> 3/4  
0
```

```
>>> 3.0/4.0  
0.75
```

```
>>> 5.0 / 10 * 2 + 5  
6
```

```
>>> 5.0 / (10 * 2 + 5)  
0.2
```

```
>>> 2**3**2  
512
```

```
>>> (2**3)**2  
64
```

```
>>> 17%3%2  
0
```

Operadores aritméticos

```
>>> 3+4  
7
```

```
>>> 3/4  
0
```

```
>>> 3.0/4.0  
0.75
```

```
>>> 5.0 / 10 * 2 + 5  
6
```

```
>>> 5.0 / (10 * 2 + 5)  
0.2
```

```
>>> 2**3**2  
512
```

```
>>> (2**3)**2  
64
```

```
>>> 17%3%2  
0
```

Operadores aritméticos

```
>>> 3+4  
7
```

```
>>> 3/4  
0
```

```
>>> 3.0/4.0  
0.75
```

```
>>> 5.0 / 10 * 2 + 5  
6
```

```
>>> 5.0 / (10 * 2 + 5)  
0.2
```

```
>>> 2**3**2  
512
```

```
>>> (2**3)**2  
64
```

```
>>> 17%3%2  
0
```

Operadores aritméticos

```
>>> 3+4  
7
```

```
>>> 3/4  
0
```

```
>>> 3.0/4.0  
0.75
```

```
>>> 5.0 / 10 * 2 + 5  
6
```

```
>>> 5.0 / (10 * 2 + 5)  
0.2
```

```
>>> 2**3**2  
512
```

```
>>> (2**3)**2  
64
```

```
>>> 17%3%2  
0
```

Operadores aritméticos

```
>>> 3+4  
7
```

```
>>> 3/4  
0
```

```
>>> 3.0/4.0  
0.75
```

```
>>> 5.0 / 10 * 2 + 5  
6
```

```
>>> 5.0 / (10 * 2 + 5)  
0.2
```

```
>>> 2**3**2  
512
```

```
>>> (2**3)**2  
64
```

```
>>> 17%3%2  
0
```


Operadores aritméticos

```
>>> 3+4  
7
```

```
>>> 3/4  
0
```

```
>>> 3.0/4.0  
0.75
```

```
>>> 5.0 / 10 * 2 + 5  
6
```

```
>>> 5.0 / (10 * 2 + 5)  
0.2
```

```
>>> 2**3**2  
512
```

```
>>> (2**3)**2  
64
```

```
>>> 17%3%2  
0
```

Operadores aritméticos

```
>>> 3+4  
7
```

```
>>> 5.0 / (10 * 2 + 5)  
0.2
```

```
>>> 3/4  
0
```

```
>>> 2**3**2  
512
```

```
>>> 3.0/4.0  
0.75
```

```
>>> (2**3)**2  
64
```

```
>>> 5.0 / 10 * 2 + 5  
6
```

```
>>> 17%3%2  
0
```

Operadores aritméticos

```
>>> 3+4  
7
```

```
>>> 5.0 / (10 * 2 + 5)  
0.2
```

```
>>> 3/4  
0
```

```
>>> 2**3**2  
512
```

```
>>> 3.0/4.0  
0.75
```

```
>>> (2**3)**2  
64
```

```
>>> 5.0 / 10 * 2 + 5  
6
```

```
>>> 17%3%2  
0
```

Tabla de operadores

Operador	Operación	Ejemplo	Resultado
**	Potencia	$2 * 3$	8
*	Multiplicación	$7 * 3$	21
/	División	$10.5 / 2$	5.25
//	División entera	$10.5 // 2$	5.0
+	Suma	$3 + 4$	7
-	Resta	$6 - 8$	-2
%	Módulo	$15 \% 6$	3

Precedencia de operadores 1

- 1 Las expresiones contenidas dentro de pares de paréntesis son evaluadas primero. En el caso de expresiones con paréntesis anidados, los operadores en el par de paréntesis más interno son aplicados primero.
- 2 Las operaciones de exponentes son aplicadas después. Si una expresión contiene muchas operaciones de exponentes, los operadores son aplicados de derecha a izquierda.

Precedencia de operadores 2

- 3 La multiplicación, división y módulo son las siguientes en ser aplicadas. Si una expresión contiene muchas multiplicaciones, divisiones u operaciones de módulo, los operadores se aplican de izquierda a derecha.
- 4 Suma y resta son las operaciones que se aplican por último. Si una expresión contiene muchas operaciones de suma y resta, los operadores son aplicados de izquierda a derecha. La suma y resta tienen el mismo nivel de precedencia.

Operadores relacionales (de comparación)

Tipos de datos lógicos: False (0) y True (1)

$1+2>7-3$

False

$3>4<5$

False

$1<2<3$

True

$1.0/3<0.33333$

False

$1>2==2<3$

False

$5.0/3>=11/7.0$

True

$1>(2==2)<3$

False

$2**(2/3)<3**(3/4)$

False

Operadores relacionales (de comparación)

Tipos de datos lógicos: False (0) y True (1)

$1+2>7-3$

False

$3>4<5$

False

$1<2<3$

True

$1.0/3<0.33333$

False

$1>2==2<3$

False

$5.0/3>=11/7.0$

True

$1>(2==2)<3$

False

$2**(2/3)<3**(3/4)$

False

Operadores relacionales (de comparación)

Tipos de datos lógicos: False (0) y True (1)

$1+2>7-3$
False

$3>4<5$
False

$1<2<3$
True

$1.0/3<0.33333$
False

$1>2==2<3$
False

$5.0/3>=11/7.0$
True

$1>(2==2)<3$
False

$2**(2/3)<3**(3/4)$
False

Operadores relacionales (de comparación)

Tipos de datos lógicos: False (0) y True (1)

$1+2>7-3$

False

$3>4<5$

False

$1<2<3$

True

$1.0/3<0.33333$

False

$1>2==2<3$

False

$5.0/3>=11/7.0$

True

$1>(2==2)<3$

False

$2**(2/3)<3**(3/4)$

False

Operadores relacionales (de comparación)

Tipos de datos lógicos: False (0) y True (1)

$1+2>7-3$

False

$3>4<5$

False

$1<2<3$

True

$1.0/3<0.33333$

False

$1>2==2<3$

False

$5.0/3>=11/7.0$

True

$1>(2==2)<3$

False

$2**(2/3)<3**(3/4)$

False

Operadores relacionales (de comparación)

Tipos de datos lógicos: False (0) y True (1)

$1+2>7-3$

False

$3>4<5$

False

$1<2<3$

True

$1.0/3<0.33333$

False

$1>2==2<3$

False

$5.0/3>=11/7.0$

True

$1>(2==2)<3$

False

$2**(2/3)<3**(3/4)$

False

Operadores relacionales (de comparación)

Tipos de datos lógicos: False (0) y True (1)

$1+2>7-3$
False

$3>4<5$
False

$1<2<3$
True

$1.0/3<0.33333$
False

$1>2==2<3$
False

$5.0/3>=11/7.0$
True

$1>(2==2)<3$
False

$2**(2/3)<3**(3/4)$
False

Operadores relacionales (de comparación)

Tipos de datos lógicos: False (0) y True (1)

$1+2>7-3$

False

$3>4<5$

False

$1<2<3$

True

$1.0/3<0.33333$

False

$1>2==2<3$

False

$5.0/3>=11/7.0$

True

$1>(2==2)<3$

False

$2**(2/3)<3**(3/4)$

False

Operadores relacionales (de comparación)

Tipos de datos lógicos: False (0) y True (1)

$1+2>7-3$

False

$3>4<5$

False

$1<2<3$

True

$1.0/3<0.33333$

False

$1>2==2<3$

False

$5.0/3>=11/7.0$

True

$1>(2==2)<3$

False

$2**(2/3)<3**(3/4)$

False

Tabla de operadores relacionales

Operador	Operación	Ejemplo	Resultado
<code>==</code>	Igual a	<code>4 == 5</code>	False
<code>!=</code>	Diferente de	<code>2 != 3</code>	True
<code><</code>	Menor que	<code>10 < 4</code>	False
<code>></code>	Mayor que	<code>5 > -4</code>	True
<code>>=</code>	Menor o igual que	<code>7 <= 7</code>	True
<code>>=</code>	Mayor o igual que	<code>3.5 >= 10</code>	False

Operadores lógicos (booleanos)

Operador	Operación	Ejemplo	Resultado
and	conjunción	False and True	False
or	disyunción	False or True	True
not	negación	not True	False

Tabla de verdad

A	B	A and B	A or B	not A
True	True	True	True	True
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True

Tipos de datos

Cada lenguaje de programación requiere de un conjunto de tipos de datos para operar, cada uno está caracterizado por un nombre, un tamaño de espacio en memoria y un intervalo.

Realizar operaciones entre diferentes tipos de datos nos va a generar un error, ya que como hemos mencionado, Python es un lenguaje fuertemente tipado, moraleja: sumar peras con peras y manzanas con manzanas.

Tabla de tipos de datos

Tipo	Descripción	bits	Rango	Ejemplo
bool	booleano	8	sin rango	True o False
int	entero	16	$[-2^{15}, 2^{15} - 1]$	327
long int	entero largo	32	$[0, 2^{32} - 1]$	24334253234L
float	real (punto flotante)	32	$[-2^{31}, 2^{31} - 1]$	3.1416
string	string (cadena)	32	$[-2^{31}, 2^{31} - 1]$	'hola'
tuple	tupla	32	$[3.4 \times 10^{-38}, 3.4 \times 10^{38}]$	1, 'aja', 2.0)
list	lista	64	$[1.7 \times 10^{-308}, 1.7 \times 10^{308}]$	[1, 'aja', 2.0]
dict	diccionario	80	$[3.4 \times 10^{-4932}, 3.4 \times 10^{4932}]$	'a':7.0, 23: True

Palabras reservadas

No se pueden utilizar dentro del código

and	assert	break	class	continue	def
del	elif	else	except	exec	finally
for	from	global	if	import	in
is	lambda	not	or	pass	print
raise	return	try	while	yield	del

Identificadores

Son nombres que hacen referencia a los objetos que componen un programa: constantes, variables, funciones, etc.

Reglas para construir identificadores:

- 1 El primer carácter debe ser una letra o el carácter de subrayado (guión bajo)
- 2 El primer carácter puede ir seguido de un número variable de dígitos numéricos, letras o caracteres de subrayado.
- 3 No pueden utilizarse espacios en blanco, ni símbolos de puntuación.
- 4 Python distingue mayúsculas y minúsculas.
- 5 No pueden utilizarse palabras reservadas del lenguaje.

Variables

```
>>> base = 2
```

```
>>> print base
```

```
2
```

```
>>> print "base"
```

```
base
```

```
>>> base = base + 1
```

```
>>> base
```

```
3
```

```
>>> alt = 4
```

```
>>> area = base+alt; a= 3
```

```
>>> a= 2*a
```

Variables

```
>>> base = 2
```

```
>>> print base  
2
```

```
>>> print "base"  
base
```

```
>>> base = base + 1
```

```
>>> base  
3
```

```
>>> alt = 4
```

```
>>> area = base+alt; a= 3
```

```
>>> a= 2*a
```


Variables

```
>>> base = 2
```

```
>>> print base  
2
```

```
>>> print "base"  
base
```

```
>>> base = base + 1
```

```
>>> base  
3
```

```
>>> alt = 4
```

```
>>> area = base+alt; a= 3
```

```
>>> a= 2*a
```

Variables

```
>>> base = 2
```

```
>>> print base  
2
```

```
>>> print "base"  
base
```

```
>>> base = base + 1
```

```
>>> base  
3
```

```
>>> alt = 4
```

```
>>> area = base+alt; a= 3
```

```
>>> a= 2*a
```

Variables

```
>>> base = 2
```

```
>>> print base  
2
```

```
>>> print "base"  
base
```

```
>>> base = base + 1
```

```
>>> base  
3
```

```
>>> alt = 4
```

```
>>> area = base+alt; a= 3
```

```
>>> a= 2*a
```

Variables

```
>>> base = 2
```

```
>>> print base  
2
```

```
>>> print "base"  
base
```

```
>>> base = base + 1
```

```
>>> base  
3
```

```
>>> alt = 4
```

```
>>> area = base+alt; a= 3
```

```
>>> a= 2*a
```

Variables

```
>>> base = 2
```

```
>>> print base  
2
```

```
>>> print "base"  
base
```

```
>>> base = base + 1
```

```
>>> base  
3
```

```
>>> alt = 4
```

```
>>> area = base+alt; a= 3
```

```
>>> a= 2*a
```

Variables

```
>>> base = 2
```

```
>>> print base  
2
```

```
>>> print "base"  
base
```

```
>>> base = base + 1
```

```
>>> base  
3
```

```
>>> alt = 4
```

```
>>> area = base+alt; a= 3
```

```
>>> a= 2*a
```

```
>>> area== 2*a  
False
```

```
>>> x= "uno"; y= "dos"
```

```
>>> x  
uno
```

```
>>> print x  
uno
```

```
>>> x+y  
unodos
```

```
>>> print x+y  
unodos
```

```
>>> area== 2*a  
False
```

```
>>> x= "uno"; y= "dos"
```

```
>>> x  
uno
```

```
>>> print x  
uno
```

```
>>> x+y  
unodos
```

```
>>> print x+y  
unodos
```



```
>>> area== 2*a  
False
```

```
>>> x= "uno"; y= "dos"
```

```
>>> x  
uno
```

```
>>> print x  
uno
```

```
>>> x+y  
unodos
```

```
>>> print x+y  
unodos
```

```
>>> area== 2*a  
False
```

```
>>> x= "uno"; y= "dos"
```

```
>>> x  
uno
```

```
>>> print x  
uno
```

```
>>> x+y  
unodos
```

```
>>> print x+y  
unodos
```

```
>>> area== 2*a  
False
```

```
>>> x= "uno"; y= "dos"
```

```
>>> x  
uno
```

```
>>> print x  
uno
```

```
>>> x+y  
unodos
```

```
>>> print x+y  
unodos
```

```
>>> area== 2*a  
False
```

```
>>> x= "uno"; y= "dos"
```

```
>>> x  
uno
```

```
>>> print x  
uno
```

```
>>> x+y  
unodos
```

```
>>> print x+y  
unodos
```

Listas y Tuplas

```
milista=[a,"hola",3.0,True]
```

```
milista  
[8,"hola",3.0,True]
```

```
milista[0]  
8
```

```
milista[1]  
'hola'
```

```
milista[2]  
3.0
```

```
milista[1:3]  
'hola',3.0
```

```
milista[0] = 2.0
```

```
milista  
[2.0,"hola",3.0,True]
```

```
milista[-1]  
True
```

Listas y Tuplas

```
milista=[a,"hola",3.0,True]
```

```
milista  
[8,"hola",3.0,True]
```

```
milista[0]  
8
```

```
milista[1]  
'hola'
```

```
milista[2]  
3.0
```

```
milista[1:3]  
'hola',3.0
```

```
milista[0] = 2.0
```

```
milista  
[2.0,"hola",3.0,True]
```

```
milista[-1]  
True
```

Listas y Tuplas

```
milista=[a,"hola",3.0,True]
```

```
milista  
[8,"hola",3.0,True]
```

```
milista[0]  
8
```

```
milista[1]  
'hola'
```

```
milista[2]  
3.0
```

```
milista[1:3]  
'hola',3.0
```

```
milista[0] = 2.0
```

```
milista  
[2.0,"hola",3.0,True]
```

```
milista[-1]  
True
```

Listas y Tuplas

```
milista=[a,"hola",3.0,True]
```

```
milista  
[8,"hola",3.0,True]
```

```
milista[0]  
8
```

```
milista[1]  
'hola'
```

```
milista[2]  
3.0
```

```
milista[1:3]  
'hola',3.0
```

```
milista[0] = 2.0
```

```
milista  
[2.0,"hola",3.0,True]
```

```
milista[-1]  
True
```


Listas y Tuplas

```
milista=[a,"hola",3.0,True]
```

```
milista  
[8,"hola",3.0,True]
```

```
milista[0]  
8
```

```
milista[1]  
'hola'
```

```
milista[2]  
3.0
```

```
milista[1:3]  
'hola',3.0
```

```
milista[0] = 2.0
```

```
milista  
[2.0,"hola",3.0,True]
```

```
milista[-1]  
True
```

Listas y Tuplas

```
milista=[a,"hola",3.0,True]
```

```
milista  
[8,"hola",3.0,True]
```

```
milista[0]  
8
```

```
milista[1]  
'hola'
```

```
milista[2]  
3.0
```

```
milista[1:3]  
'hola',3.0
```

```
milista[0] = 2.0
```

```
milista  
[2.0,"hola",3.0,True]
```

```
milista[-1]  
True
```

Listas y Tuplas

```
milista=[a,"hola",3.0,True]
```

```
milista  
[8,"hola",3.0,True]
```

```
milista[0]  
8
```

```
milista[1]  
'hola'
```

```
milista[2]  
3.0
```

```
milista[1:3]  
'hola',3.0
```

```
milista[0] = 2.0
```

```
milista  
[2.0,"hola",3.0,True]
```

```
milista[-1]  
True
```

Listas y Tuplas

```
milista=[a,"hola",3.0,True]
```

```
milista  
[8,"hola",3.0,True]
```

```
milista[0]  
8
```

```
milista[1]  
'hola'
```

```
milista[2]  
3.0
```

```
milista[1:3]  
'hola',3.0
```

```
milista[0] = 2.0
```

```
milista  
[2.0,"hola",3.0,True]
```

```
milista[-1]  
True
```

Listas y Tuplas

```
milista=[a,"hola",3.0,True]
```

```
milista  
[8,"hola",3.0,True]
```

```
milista[0]  
8
```

```
milista[1]  
'hola'
```

```
milista[2]  
3.0
```

```
milista[1:3]  
'hola',3.0
```

```
milista[0] = 2.0
```

```
milista  
[2.0,"hola",3.0,True]
```

```
milista[-1]  
True
```

```
milista.append("otro")
```

```
milista  
[2.0, "hola", 3.0, True, 'otro']
```

```
milista[:2]  
[2.0, "hola"]
```

```
milista[1:]  
[2.0, "hola", 3.0, True]
```

```
lista2=[]
```

```
lista2  
[]
```

```
lista2.insert(1,"a")
```

```
lista2  
['a']
```

```
lista2.insert(2,"b")
```

```
milista.append("otro")
```

```
milista  
[2.0,"hola",3.0,True,'otro']
```

```
milista[:2]  
[2.0,"hola"]
```

```
milista[1:]  
[2.0,"hola",3.0,True]
```

```
lista2=[]
```

```
lista2  
[]
```

```
lista2.insert(1,"a")
```

```
lista2  
['a']
```

```
lista2.insert(2,"b")
```

```
milista.append("otro")
```

```
milista  
[2.0,"hola",3.0,True,'otro']
```

```
milista[:2]  
[2.0,"hola"]
```

```
milista[1:]  
[2.0,"hola",3.0,True]
```

```
lista2=[]
```

```
lista2  
[]
```

```
lista2.insert(1,"a")
```

```
lista2  
['a']
```

```
lista2.insert(2,"b")
```



```
milista.append("otro")
```

```
milista  
[2.0,"hola",3.0,True,'otro']
```

```
milista[:2]  
[2.0,"hola"]
```

```
milista[1:]  
[2.0,"hola",3.0,True]
```

```
lista2=[]
```

```
lista2
```

```
[]
```

```
lista2.insert(1,"a")
```

```
lista2  
['a']
```

```
lista2.insert(2,"b")
```

```
milista.append("otro")
```

```
milista  
[2.0,"hola",3.0,True,'otro']
```

```
milista[:2]  
[2.0,"hola"]
```

```
milista[1:]  
[2.0,"hola",3.0,True]
```

```
lista2=[]
```

```
lista2  
[]
```

```
lista2.insert(1,"a")
```

```
lista2  
['a']
```

```
lista2.insert(2,"b")
```

```
milista.append("otro")
```

```
milista  
[2.0,"hola",3.0,True,'otro']
```

```
milista[:2]  
[2.0,"hola"]
```

```
milista[1:]  
[2.0,"hola",3.0,True]
```

```
lista2=[]
```

```
lista2  
[]
```

```
lista2.insert(1,"a")
```

```
lista2  
['a']
```

```
lista2.insert(2,"b")
```

```
milista.append("otro")
```

```
milista  
[2.0,"hola",3.0,True,'otro']
```

```
milista[:2]  
[2.0,"hola"]
```

```
milista[1:]  
[2.0,"hola",3.0,True]
```

```
lista2=[]
```

```
lista2  
[]
```

```
lista2.insert(1,"a")
```

```
lista2  
['a']
```

```
lista2.insert(2,"b")
```

```
milista.append("otro")
```

```
milista  
[2.0,"hola",3.0,True,'otro']
```

```
milista[:2]  
[2.0,"hola"]
```

```
milista[1:]  
[2.0,"hola",3.0,True]
```

```
lista2=[]
```

```
lista2  
[]
```

```
lista2.insert(1,"a")
```

```
lista2  
['a']
```

```
lista2.insert(2,"b")
```

```
milista.append("otro")
```

```
milista  
[2.0,"hola",3.0,True,'otro']
```

```
milista[:2]  
[2.0,"hola"]
```

```
milista[1:]  
[2.0,"hola",3.0,True]
```

```
lista2=[]
```

```
lista2  
[]
```

```
lista2.insert(1,"a")
```

```
lista2  
['a']
```

```
lista2.insert(2,"b")
```

```
lista2  
['a','b']
```

```
lt = ( 1,2,True, "python"
```

```
lt  
(1,2, True, 'python'  
)
```

```
lt[0]=3  
Ups, hay un error!
```

```
3 in lt  
False
```

```
lista2  
['a','b']
```

```
lt = ( 1,2,True, "python"
```

```
lt  
(1,2, True, 'python'  
)
```

```
lt[0]=3  
Ups, hay un error!
```

```
3 in lt  
False
```



```
lista2  
['a','b']
```

```
lt = ( 1,2,True, "python"
```

```
lt  
(1,2, True, 'python'  
)
```

```
lt[0]=3
```

Ups, hay un error!

```
3 in lt
```

```
False
```

```
lista2  
['a','b']
```

```
lt = ( 1,2,True, "python"
```

```
lt  
(1,2, True, 'python'  
)
```

```
lt[0]=3  
Ups, hay un error!
```

```
3 in lt  
False
```

```
lista2  
['a','b']
```

```
lt = ( 1,2,True, "python"
```

```
lt  
(1,2, True, 'python'  
)
```

```
lt[0]=3  
Ups, hay un error!
```

```
3 in lt  
False
```

Función range

La función `range()` crea una lista de números enteros en sucesión aritmética. La función `range()` puede tener uno, dos o tres argumentos numéricos.

La función con un único argumento se escribe `range(n)` y crea una lista creciente de n términos enteros que empieza en 0 y acaba en $n - 1$ (el incremento es unitario)

```
range(8)  
[0,1,2,3,4,5,6,7]
```

```
range(3,7)  
[3,4,5,6]
```

```
range(4,10,2)  
[4,6,8]
```

Función range

La función `range()` crea una lista de números enteros en sucesión aritmética. La función `range()` puede tener uno, dos o tres argumentos numéricos.

La función con un único argumento se escribe `range(n)` y crea una lista creciente de n términos enteros que empieza en 0 y acaba en $n - 1$ (el incremento es unitario)

```
range(8)  
[0,1,2,3,4,5,6,7]
```

```
range(3,7)  
[3,4,5,6]
```

```
range(4,10,2)  
[4,6,8]
```

Función range

La función `range()` crea una lista de números enteros en sucesión aritmética. La función `range()` puede tener uno, dos o tres argumentos numéricos.

La función con un único argumento se escribe `range(n)` y crea una lista creciente de n términos enteros que empieza en 0 y acaba en $n - 1$ (el incremento es unitario)

```
range(8)  
[0,1,2,3,4,5,6,7]
```

```
range(3,7)  
[3,4,5,6]
```

```
range(4,10,2)  
[4,6,8]
```

Función range

La función `range()` crea una lista de números enteros en sucesión aritmética. La función `range()` puede tener uno, dos o tres argumentos numéricos.

La función con un único argumento se escribe `range(n)` y crea una lista creciente de n términos enteros que empieza en 0 y acaba en $n - 1$ (el incremento es unitario)

```
range(8)  
[0,1,2,3,4,5,6,7]
```

```
range(3,7)  
[3,4,5,6]
```

```
range(4,10,2)  
[4,6,8]
```

Funciones intrínsecas

```
x = -5
```

```
y = 4
```

```
p = 3.1416
```

```
z = '6.3'
```

```
print int(p)
```

```
3
```

```
abs(x)
```

```
5
```

```
print float(z)
```

```
6.0
```

```
complex(x)
```

```
(-5+0j)
```

```
complex(x,y)
```

```
(-5+4j)
```

```
print round(p,2)
```

```
3.14
```

```
cmp(x,y)
```

```
-1
```


Funciones intrínsecas

```
x = -5
```

```
y = 4
```

```
p = 3.1416
```

```
z = '6.3'
```

```
print int(p)  
3
```

```
abs(x)  
5
```

```
print float(z)  
6.0
```

```
complex(x)  
(-5+0j)
```

```
complex(x,y)  
(-5+4j)
```

```
print round(p,2)  
3.14
```

```
cmp(x,y)  
-1
```

Funciones intrínsecas

```
x = -5
```

```
y = 4
```

```
p = 3.1416
```

```
z = '6.3'
```

```
print int(p)  
3
```

```
abs(x)  
5
```

```
print float(z)  
6.0
```

```
complex(x)  
(-5+0j)
```

```
complex(x,y)  
(-5+4j)
```

```
print round(p,2)  
3.14
```

```
cmp(x,y)  
-1
```

Funciones intrínsecas

```
x = -5
```

```
y = 4
```

```
p = 3.1416
```

```
z = '6.3'
```

```
print int(p)  
3
```

```
abs(x)  
5
```

```
print float(z)  
6.0
```

```
complex(x)  
(-5+0j)
```

```
complex(x,y)  
(-5+4j)
```

```
print round(p,2)  
3.14
```

```
cmp(x,y)  
-1
```

Funciones intrínsecas

```
x = -5
```

```
print float(z)  
6.0
```

```
y = 4
```

```
complex(x)  
(-5+0j)
```

```
p = 3.1416
```

```
complex(x,y)  
(-5+4j)
```

```
z = '6.3'
```

```
print int(p)  
3
```

```
print round(p,2)  
3.14
```

```
abs(x)  
5
```

```
cmp(x,y)  
-1
```

Funciones intrínsecas

```
x = -5
```

```
y = 4
```

```
p = 3.1416
```

```
z = '6.3'
```

```
print int(p)  
3
```

```
abs(x)  
5
```

```
print float(z)  
6.0
```

```
complex(x)  
(-5+0j)
```

```
complex(x,y)  
(-5+4j)
```

```
print round(p,2)  
3.14
```

```
cmp(x,y)  
-1
```

Funciones intrínsecas

```
x = -5
```

```
y = 4
```

```
p = 3.1416
```

```
z = '6.3'
```

```
print int(p)  
3
```

```
abs(x)  
5
```

```
print float(z)  
6.0
```

```
complex(x)  
(-5+0j)
```

```
complex(x,y)  
(-5+4j)
```

```
print round(p,2)  
3.14
```

```
cmp(x,y)  
-1
```

Funciones intrínsecas

```
x = -5
```

```
y = 4
```

```
p = 3.1416
```

```
z = '6.3'
```

```
print int(p)  
3
```

```
abs(x)  
5
```

```
print float(z)  
6.0
```

```
complex(x)  
(-5+0j)
```

```
complex(x,y)  
(-5+4j)
```

```
print round(p,2)  
3.14
```

```
cmp(x,y)  
-1
```

Funciones intrínsecas

```
x = -5
```

```
y = 4
```

```
p = 3.1416
```

```
z = '6.3'
```

```
print int(p)  
3
```

```
abs(x)  
5
```

```
print float(z)  
6.0
```

```
complex(x)  
(-5+0j)
```

```
complex(x,y)  
(-5+4j)
```

```
print round(p,2)  
3.14
```

```
cmp(x,y)  
-1
```


Funciones intrínsecas

```
x = -5
```

```
y = 4
```

```
p = 3.1416
```

```
z = '6.3'
```

```
print int(p)  
3
```

```
abs(x)  
5
```

```
print float(z)  
6.0
```

```
complex(x)  
(-5+0j)
```

```
complex(x,y)  
(-5+4j)
```

```
print round(p,2)  
3.14
```

```
cmp(x,y)  
-1
```

Funciones intrínsecas

```
x = -5
```

```
y = 4
```

```
p = 3.1416
```

```
z = '6.3'
```

```
print int(p)  
3
```

```
abs(x)  
5
```

```
print float(z)  
6.0
```

```
complex(x)  
(-5+0j)
```

```
complex(x,y)  
(-5+4j)
```

```
print round(p,2)  
3.14
```

```
cmp(x,y)  
-1
```

Operación	Descripción
<code>int(x)</code>	Convierte x a entero
<code>long(x)</code>	Convierte x a entero largo
<code>float(x)</code>	Convierte x a punto flotante
<code>complex(x)</code>	Convierte x al complejo $x+0j$
<code>complex(x,y)</code>	Convierte al complejo $x+yj$

Función	Descripción
<code>abs(x)</code>	Valor absoluto de x
<code>max(sucesion)</code>	Mayor elemento de la sucesión
<code>min(sucesion)</code>	Menor elemento de la sucesión
<code>round(x,n)</code>	Redondea x al decimal n
<code>cmp(x,y)</code>	Devuelve -1 , 0 , 1 si $x < y$, $x == y$, $x > y$