# Guía de Referencia Rápida de Python

## Generalidades

- Una declaración debe estar toda en una sola línea. Para romper una declaración en múltiples líneas debe usarse '\' al final de cada una de ellas (salvo la última).

  *Excepción:* siempre se puede romper dentro de cualquier par (), [] o {}, o en una cadena delimitada por triple comillas.

- En una línea pueden aparecer más de una declaración separándolas por ';'.
- Los comentarios comienzan con '#' y continúan hasta el final de la línea.
- Un identificador está formado por una letra o símbolo '_' seguido de más letras, números o símbolos '_'.
- Python distingue mayúsculas de minúsculas.

## Tipos de datos

Los tipos de datos en Python se dividen en mutables, si su contenido puede cambiarse, e inmutables, si su contenido no puede cambiarse.

### Números

- Enteros (int): 1234, −123456789.
- Reales o números en coma flotante (float): 0.001, 10., 3.14e−10.
- Complejos (complex): 2+3j, .5−1.4j, 1j.
- Todos los tipos de números son datos inmutables.

### Constantes lógicas (bool)

- True (verdadero).
- False (falso).
- Las constantes lógicas son datos inmutables.

## Secuencias

- Cadenas (string): delimitadas por comillas simples (‘Hola’), dobles ("Hola") o triples ( ‘‘‘ Hola’’’, """Hola""").
- Tuplas (tuple): delimitadas por paréntesis y los elementos separados por comas ((), (1,), (1, 2, 3)).
- Listas ( list ): delimitadas por corchetes y los elementos separados por comas ([], [1], [1, 2, 3]).
- Las cadenas y tuplas son datos inmutables. Las listas son datos mutables.

## Diccionarios (dict)

- Pares clave : valor separados por coma y delimitados por llaves: ({1:‘primero’, ‘segundo’:2}).
- Las claves deben ser datos inmutables.
- Los diccionarios son datos mutables.

## Conjuntos

- Los conjuntos son colecciones no ordenadas de elementos no duplicados.
- Los elementos de un conjunto deben ser datos inmutables.
- set(secuencia) crea un conjunto mutable con los elementos de la secuencia especificada, descartando las repeticiones.
- frozenset(secuencia) crea un conjunto inmutable con los elementos de la secuencia especificada, descartando las repeticiones.

## Otros tipos de datos

- La constante None es el dato nulo.

# Operaciones sobre los tipos de datos básicos

## Operadores de comparación

| *Operador* | *Significado* |
|---|---|
| < | Menor estricto que |
| <= | Menor o igual que |
| > | Mayor estricto que |
| >= | Mayor o igual que |
| == | Igual que |
| != | Distinto que |
| **is** | Idéntico a |
| **is not** | No idéntico a |

- Están definidos entre cualesquiera tipos de datos.

## Operadores lógicos

| *Declaración* | *Evalúa a* |
|---|---|
| bool(expr) | True si expr es verdadera, False en caso contrario |
| **not** expr | True si expr es falsa, False en caso contrario |
| expr1 **or** expr2 | False si expr1 y expr2 son falsos, True en caso contrario |
| expr1 **and** expr2 | True si expr1 y expr2 son verdaderos, True en caso contrario |

- La constante None, los ceros numéricos, las secuencias vacías y los diccionarios y conjuntos vacíos se consideran falsos.
- El resto de datos se consideran verdaderos.
- Los operadores **or** y **and** solo evalúan expr2 en caso necesario.

## Operadores numéricos

| *Operación* | *Resultado* |
|---|---|
| abs(x) | Valor absoluto de x |
| int (x) | x convertido a entero |
| float (x) | x convertido a real |
| complex(x) | x convertido a complejo |

| | |
|---|---|
| −x | x negado |
| x+y | Suma de x e y |
| x−y | Diferencia de x e y |
| x∗y | Producto de x e y |
| x/y | División de x por y (cociente de la división, en el caso de x e y enteros) |
| x %y | Resto de la división de x por y |
| divmod(x,y) | La tupla (x/y, x %y) |
| x∗∗y | x elevado a y |

## Operaciones sobre secuencias

| *Operación* | *Resultado* |
|---|---|
| x **in** s | True si un elemento de s es igual a x, False en caso contrario |
| x **not in** s | False si un elemento de s es igual a x, True en caso contrario |
| s1+s2 | Concatenación de s1 y s2 |
| s∗n, n∗s | n copias de s concatenadas |
| s[i] | i-ésimo elemento de s |
| s[i:j:salto] | Porción de s con los elementos en los índices de la progresión aritmética desde i (incluido) hasta j (excluido) con diferencia salto |
| len(s) | Longitud de s |
| min(s) | Menor elemento de s |
| max(s) | Mayor elemento de s |

- Si i o j son negativos, entonces son relativos al final de la secuencia; es decir, se considera len(s)+i o len(s)+j en su lugar.
- Si i o j son mayores que len(s), entonces se considera len(s) en su lugar.
- El valor por defecto de j es len(s). El valor por defecto de salto es 1.

## Operaciones sobre cadenas

| *Operación* | *Resultado* |
|---|---|
| s. capitalize () | Una copia de s con solo el primer carácter en mayúsculas |
| s.count(sub,ini, fin) | Número de ocurrencias de la subcadena sub en s[ini:fin] |

| | |
|---|---|
| s.find(sub, ini, fin) | El menor índice donde se encuentra sub como subcadena de s[ini:fin], $-1$ si no se encuentra |
| s.index(sub, ini, fin) | Igual que s.find(sub, ini, fin), pero provoca un error si la subcadena no se encuentra |
| s.isalnum() | True si todos los caracteres de s son alfanuméricos, False en caso contrario |
| s.isalpha() | True si todos los caracteres de s son alfabéticos, False en caso contrario |
| s. isdigit () | True si todos los caracteres de s son dígitos, False en caso contrario |
| s.islower() | True si todos los caracteres de s están en minúsculas, False en caso contrario |
| s.isspace() | True si todos los caracteres de s son espacios en blanco, False en caso contrario |
| s.isupper() | True si todos los caracteres de s están en mayúsculas, False en caso contrario |
| s.join(sec) | Concatenación de las cadenas contenidas en sec, separadas por s |
| s.lower() | Una copia de s con todos los caracteres de s en minúsculas |
| s.replace(vie, nue, max) | Una copia de s con todas (o max) las ocurrencias de la subcadena vie reemplazadas por la subcadena nue |
| s.rfind(sub, ini, fin) | El mayor índice donde se encuentra sub como subcadena de s[ini:fin], $-1$ si no se encuentra |
| s.rindex(sub, ini, fin) | Igual que s.rfind(sub, ini, fin), pero provoca un error si la subcadena no se encuentra |
| s. split (sep,max) | La lista de todas (o max) palabras obtenidas de s usando sep como cadena delimitadora |
| s. rsplit (sep,max) | Igual que s. split (sep,max), pero desde el final de la cadena |
| s.swapcase() | Copia de s transformando mayúsculas en minúsculas y viceversa |
| s.upper() | Copia de s con todos los caracteres en mayúsculas |

- El valor por defecto de ini es 0, y el valor por defecto de fin es len(s).
- Los métodos isalnum, isalpha, isdigit e isspace devuelven False si la cadena no contiene al menos un carácter. Los métodos islower e isupper devuelven False si la cadena no contiene al menos un carácter alfabético.
- El valor por defecto de sep es espacio en blanco. El argumento max es opcional.

## Operaciones sobre listas

| *Operación* | *Resultado* |
|---|---|
| s[i]=x | El i-ésimo elemento de s es reemplazado por x |
| s[i:j:salto]=t | La porción de s con los elementos en las posiciones de la progresión aritmética desde i (incluido) hasta j (excluido) con diferencia salto es reemplazada por t |
| **del** s[i] | Elimina el i-ésimo elemento de s |
| **del** s[i:j:salto] | Lo mismo que s[i:j:salto]=[] |
| s.append(x) | Lo mismo que s[len(s):len(s)]=[x] |
| s.extend(x) | Lo mismo que s[len(s):len(s)]=x |
| s.count(x) | El cardinal del conjunto {i : s[i]==x } |
| s.index(x,ini,fin) | El menor i tal que ini $\leq$ i $<$fin y s[i]==x |
| s.insert(i,x) | Lo mismo que s[i:i]=[x] |
| s.remove(x) | Lo mismo que **del** s[s.index(x)] |
| s.pop(i) | Lo mismo que x=s[i]; **del** s[i]; **return** x |
| s.reverse() | Invierte el orden de los elementos de s |
| s.sort() | Ordena los elementos de s |

- Los métodos reverse y sort modifican la lista. No devuelven el resultado para poner de manifiesto este efecto lateral.
- El argumento del método pop toma $-1$ como valor por defecto, por lo que por defecto el último elemento de la lista es eliminado y devuelto.

## Operaciones sobre diccionarios

| *Operación* | *Resultado* |
|---|---|
| len(d) | El número de elementos en d |
| d[k] | El elemento de d con clave k |
| d[k]=x | Establece a x el valor del elemento de d con clave k |
| **del** d[k] | Elimina el elemento de d con clave k |
| d.clear() | Elimina todos los elementos de d |
| d.copy() | Una copia de d |
| d.has_key(k) | True si d contiene un elemento con clave k, False en caso contrario |
| k **in** d | Lo mismo que d.has_key(k) |
| d.items() | La lista de los pares (clave, valor) de d |
| d.keys() | La lista de las claves de d |
| d.values() | La lista de los valores de d |

### Operaciones sobre conjuntos

| *Operación* | *Resultado* |
|---|---|
| len(s) | Cardinalidad del conjunto s |
| elt **in** s | True si elt pertenece a s, False en caso contrario |
| elt **not in** s | True si elt no pertenece a s, False en caso contrario |
| s1.issubset(s2) | True si todo elemento de s1 pertenece a s2, False en caso contrario |
| s1.issuperset(s2) | True si todo elemento de s2 pertenece a s1, False en caso contrario |
| s.add(elt) | Añade elt al conjunto mutable s |
| s.remove(elt) | Elimina elt del conjunto mutable s |
| s.clear() | Elimina todos los elementos del conjunto mutable s |
| s1.intersection(s2) | Nuevo conjunto con los elementos comunes a s1 y s2 |
| s1 & s2 | Lo mismo que s1.intersection(s2) |
| s1.union(s2) | Nuevo conjunto con los elementos de s1 y de s2 |
| s1 \| s2 | Lo mismo que s1.union(s2) |
| s1.difference(s2) | Nuevo conjunto con los elementos que pertenecen a s1, pero no a s2 |
| s1 − s2 | Lo mismo que s1.difference(s2) |
| s1.symmetric_difference(s2) | Nuevo conjunto con los elementos que pertenecen a s1 o a s2, pero no a ambos |
| s1 ^ s2 | Lo mismo que s1.symmetric_difference(s2) |
| s.copy() | Nuevo conjunto copia del conjunto s |
| s.update(sec) | Añade los elementos de sec al conjunto s |

# Declaraciones

| *Declaración* | *Resultado* |
|---|---|
| **pass** | Declaración nula |
| **del** nombre | Borra el dato llamado nombre |
| **global** nombre | Establece como global la variable nombre |
| **print** s1 ,..., sN, | Escribe en pantalla representaciones de s1, ..., sN separadas por un espacio en blanco |

| | |
|---|---|
| **print** s1 ,..., sN | Escribe en pantalla representaciones de s1, ..., sN separadas por un espacio en blanco y terminando con un salto de línea |
| **raise** TipoError | Provoca un error de tipo TipoError |

## Operadores de asignación

| *Operador* | *Resultado* |
|---|---|
| a=b | Asigna el dato b a la etiqueta a |
| a+=b | Lo mismo que a=a+b |
| a−=b | Lo mismo que a=a−b |
| a∗=b | Lo mismo que a=a∗b |
| a/=b | Lo mismo que a=a/b |
| a%=b | Lo mismo que a=a%b |
| a∗∗=b | Lo mismo que a=a∗∗b |

- El operador de asignación puede desempaquetar cadenas, tuplas y listas:

  (a,b)=range(2) es lo mismo que a=0;b=1
  x,y=y,x intercambia los valores de x e y

- Es posible realizar asignaciones múltiples:

  a=b=c=0 es lo mismo que a=0; b=0; c=0

## Declaraciones de control de flujo

| *Declaración* | *Significado* |
|---|---|
| `if condicion:`<br>`  consecuencias`<br>`elif condicion:`<br>`  consecuencias`<br>`else:`<br>`  alternativas` | Condicional simple (**if**), doble (**if**/**else**) y múltiple (**if**/**elif**/**else**) |
| `while condicion:`<br>`  acciones` | Bucle mientras |
| `for elt in secuencia:`<br>`  acciones` | Bucles para y desde |
| **break**<br>**continue** | Interrupción de un bucle<br>Continuación de un bucle |

## Definición de funciones

```
def nombre_funcion (parametros):
    'documentacion'
    acciones
    return resultado
```

- parametros es una sucesión de identificadores separados por comas.

```
def nula (x, y, z):
    'Funcion que no hace nada'
    pass
```

- Los argumentos se pasan a la función por posición o por nombre.

```
nula(1, z=3, 2)
```

hace que los parámetros tomen los siguientes valores:

x=1, y=2, z=3

- La sucesión de parámetros puede contener uno de la forma *nombre, en cuyo caso se le asignará la tupla de todos los argumentos proporcionados por posición que no correspondan a otro parámetro.

```
def nula (x, y, z, *args):
    'Funcion que no hace nada'
    pass
nula(1, 2, 3, 4, 5, 6)
```

hace que los parámetros tomen los siguientes valores:

x=1, y=2, z=3, args=(4, 5, 6)

- La sucesión de parámetros puede contener uno de la forma **nombre, en cuyo caso se le asignará un diccionario con todos los argumentos proporcionados por nombre que no correspondan a otro parámetro.

```
def nula (x, y, z, **kwargs):
    'Funcion que no hace nada'
    pass
nula(1, z=3, 2, u=4, v=5, w=6)
```

hace que los parámetros tomen los siguientes valores:

x=1, y=2, z=3, kwargs={u:4, v:5, w:6}

- La declaración **return** devuelve el resultado de aplicar la función a los argumentos proporcionados. Si no se incluye, entonces la función devuelve None (y, entonces, la consideramos un procedimiento).

## Funciones predefinidas

| *Función* | *Resultado* |
|---|---|
| dir() | Devuelve la lista de variables definidas |
| globals() | Devuelve un diccionario con los nombres y valores de las variables globales definidas |
| help() | Invoca el sistema de ayuda |
| input(mensaje) | Escribe mensaje en pantalla, lee una entrada desde el teclado, la evalúa y devuelve el resultado |
| isinstance(dato, tipodato) | Devuelve True si dato es del tipo especificado, False en caso contrario |
| locals() | Devuelve un diccionario con los nombres y valores de las variables locales definidas |
| range(ini, fin, salto) | Devuelve una lista con los enteros de la progresión aritmética que empieza en ini, termina en fin−1 y tiene diferencia salto. El valor por defecto de ini es 0 y el valor por defecto de salto es 1 |
| round(x, n) | Redondea x al valor más cercano con n dígitos tras la coma decimal. El valor por defecto de n es 0 |
| str(dato) | Devuelve una cadena conteniendo una representación de dato |
| sum(sec, valini) | Devuelve la suma de la secuencia de números sec, añadiéndole además valini. El valor por defecto de valini es 0 |
| vars() | Devuelve un diccionario con los nombres y valores de las variables definidas |

## Definición de clases de objetos

```
class nombre_clase:
    definicion de metodos
```

define la clase básica nombre_clase y

```
class nombre_clase (nombre_superclase):
    definicion de metodos
```

define la clase nombre_clase que hereda de la clase nombre_superclase.

## Métodos especiales y redefinición de operadores

| *Método* | *Descripción* |
|---|---|
| __init__(propio,args) | Procedimiento que inicializa la instancia a partir de args |
| __str__(propio) | Función que devuelve una cadena representando la instancia |
| __lt__(propio, otro) | Función utilizada para la comparación propio<otro |
| __le__(propio, otro) | Función utilizada para la comparación propio<=otro |
| __gt__(propio, otro) | Función utilizada para la comparación propio>otro |
| __ge__(propio, otro) | Función utilizada para la comparación propio>=otro |
| __eq__(propio, otro) | Función utilizada para la comparación propio==otro |
| __ne__(propio, otro) | Función utilizada para la comparación propio!=otro |
| __add__(propio, otro) | Función utilizada para la operación propio+otro |
| __sub__(propio, otro) | Función utilizada para la operación propio−otro |
| __mul__(propio, otro) | Función utilizada para la operación propio∗otro |
| __div__(propio, otro) | Función utilizada para la operación propio/otro |
| __mod__(propio, otro) | Función utilizada para la operación propio %otro |
| __pow__(propio, otro) | Función utilizada para la operación propio∗∗otro |
| __neg__(propio) | Función utilizada para la operación −propio |
| __len__(propio) | Función utilizada para la operación len(propio) |

| | |
|---|---|
| __getitem__(propio, k) | Función utilizada para la operación propio[k] |
| __setitem__(propio, k, valor) | Función utilizada para la operación propio[k]=valor |
| __delitem__(propio, k) | Función utilizada para la operación **del** propio[k] |
| __contains__(propio, elt) | Función utilizada para la operación elt **in** propio |

# Módulos

| *Declaración* | *Resultado* |
|---|---|
| **import** modulo1, ..., moduloN | Importa los módulos especificados |
| **from** modulo **import** nombre1, ..., nombreN | Importa los nombres especificados del módulo indicado |
| **from** modulo **import** $*$ | Importa todos los nombres definidos en el módulo especificado |

## Módulo string

| *Variable* | *Significado* |
|---|---|
| digits | La cadena '0123456789' |
| letters | La cadena con todos los caracteres alfabéticos en minúsculas y en mayúsculas |
| lowercase | La cadena con todos los caracteres alfabéticos en minúsculas |
| uppercase | La cadena con todos los caracteres alfabéticos en mayúsculas |

| *Función* | *Resultado* |
|---|---|
| find(s, sub, ini, fin) | El menor índice en s donde se encuentra la subcadena sub |
| rfind(s, sub, ini, fin) | El mayor índice en s donde se encuentra la subcadena sub |
| lower(s) | Una copia de s con todos los caracteres en minúsculas |
| upper(s) | Una copia de s con todos los caracteres en mayúsculas |

| | |
|---|---|
| split (s, sep, max) | Una lista conteniendo todas (o max) las palabras de s, usando la cadena sep como separador |
| rsplit (s, sep, max) | Igual que split , pero empezando por el final de la cadena |
| join(s, sep) | Concatena las palabras contenidas en la lista o tupla s, usando sep como separador |
| replace(s, nue, ant, max) | Una copia de s en la que todas (o max) las ocurrencias de la subcadena nue se han sustituidos por la subcadena ant |

**Plantillas**

Las instancias de Template poseen los métodos substitute y safe_substitute para sustituir las variables indicadas por $ en la cadena proporcionada como plantilla por los valores indicados mediante un diccionario. La diferencia entre ambos métodos es que el primero produce un error si no se proporciona un valor para todas las variables, mientras que el segundo no.

```
plantilla = Template('Hola $nombre, tienes $cant euros')
plantilla.substitute({'nombre':'Antonio', 'cant':100})
'Hola Antonio, tienes 100 euros'
```

## Módulo math

| *Constante* | *Valor* |
|---|---|
| pi | 3.1415926535897931 |
| e | 2.7182818284590451 |

| *Función* | *Resultado* |
|---|---|
| acos(x) | El arco-coseno de x, en radianes |
| asin(x) | El arco-seno de x, en radianes |
| atan(x) | El arco-tangente de x, en radianes |
| atan2(y, x) | El arco-tangente de y/x, en radianes. El resultado está entre −pi y pi, y se consideran los signos de x e y, al contrario que atan(y/x) |
| ceil (x) | El menor entero mayor o igual que x. El resultado se devuelve como número real |
| cos(x) | El coseno del ángulo en radianes x |
| degrees(x) | Convierte el ángulo x de radianes a grados |
| exp(x) | e**x |

| | |
|---|---|
| fabs(x) | Valor absoluto de x como un número real |
| floor (x) | El mayor entero menor o igual que x. El resultado se devuelve como número real |
| hypot(x,y) | La distancia Euclídea sqrt(x∗x+y∗y) |
| log(x, base) | Logaritmo de x en la base especificada (por defecto e) |
| log10(x) | Logaritmo en base 10 de x |
| modf(x) | Tupla con la parte decimal y la parte entera (como número real) de x. Ambos números conservan el signo de x |
| radians(x) | Convierte el ángulo x de grados a radianes |
| sin (x) | El seno del ángulo en radianes x |
| sqrt (x) | La raíz cuadrada de x |
| tan(x) | La tangente del ángulo en radianes x |