



Resumen de Python básico

1. Variables:

TIPOS BASICOS DE DATOS NO ESTRUCTURADOS	
En general	Python
Enteros	int long
Reales	float
Carácter simple	string
Booleana	boolean
TIPOS BASICOS DE DATOS ESTRUCTURADOS	
Cadenas de caracteres	string
Arreglos	tuples list sets

2. Tipos de operadores:

Operadores aritméticos			
Operador	Nombre	Ejemplo	Resultado
+	Suma	1 + 3	4
-	Resta	7 - 10	-3
*	Multiplicación	3*5	15
/	División	8/5	1.6
//	División entera	8//3	2
%	Modulo (Residuo)	15%7	1
** ó ^	Potencia	2**3	8

Operadores relacionales			
Operador	Nombre	Ejemplo	Resultado
>	Mayor	1 > 3	False (F)
>=	Mayor o igual	2 >= 1	True (V)
<	Menor	-5 < -1	True (V)
<=	Menor o igual	3 <= 3	True (V)
!=	Diferente	13 != 4	True (V)
==	Igual (Comparación)	0 == 1	False (F)

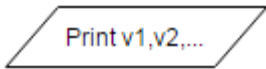
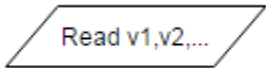
Operadores logicos			
Operador	Nombre	Ejemplo	Resultado
not	No	not(T)	F
and ó %	y	T and F	F
or ó	o	T or F	V

Operaciones lógicas			
Operación	Operadores		Resultado
not	x		not(x)
	F		V
	V		F
and	x	y	x and y
	F	F	F
	F	V	F
	V	F	F
	V	V	V
or	x	y	x or y
	F	F	F
	F	V	V
	V	F	V
	V	V	V

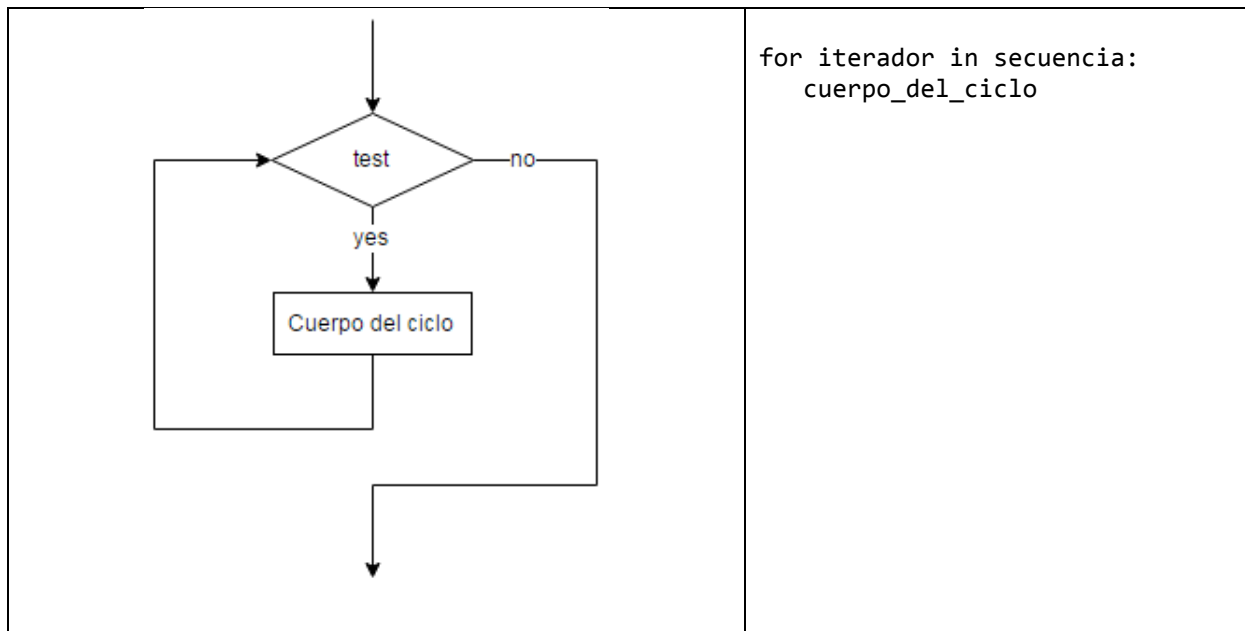
3. Tabla de prioridad y asociatividad en Python:

Operación	Operador	Tipo	Asociatividad	Precedencia
Paréntesis	()	--	Adentro → Afuera	1
Potenciación	**	Binario	D → I	2
Identidad	+	Unitario	----	3
Cambio de signo	-	Unitario	----	3
Multiplicación	*	Binario	I → D	4
División	/	Binario	I → D	4
División entera	//	Binario	I → D	4
modulo	%	Binario	I → D	4
Suma	+	Binario	I → D	5
Resta	-	Binario	I → D	5
Igual que	==	Binario	----	6
Diferente	!=	Binario	----	6
Menor que	<	Binario	----	6
Menor o igual que	<=	Binario	----	6
Mayor que	>	Binario	----	6
Mayor o igual que	>=	Binario	----	6
Negación	not	Unitario	----	7
Conjunción	and	Binario	I → D	8
Disyunción	or	Binario	I → D	9

4. Estructuras de programación en Python:

PRINCIPALES ESTRUCTURAS DE PROGRAMACION EN PYTHON	
Estructura en bloques	Estructura en Python
	Sintaxis: print("Expresion")
	Sintaxis: V = input('cadena')
	Sintaxis:

<div data-bbox="430 170 690 233"><code>variable = expresion</code></div>	<div data-bbox="938 184 1247 247"><code>variable = constante;</code> <code>variable = expresion;</code></div>
<div data-bbox="321 352 808 678"></div>	<div data-bbox="938 289 1177 415">Sintaxis: <code>if(condicion):</code> <code>instrucciones</code></div>
<div data-bbox="300 772 820 1077"></div>	<div data-bbox="938 730 1221 919">Sintaxis: <code>if condicion:</code> <code>instrucciones_Si</code> <code>else:</code> <code>instrucciones_No</code></div>
<div data-bbox="251 1171 844 1623"></div>	<div data-bbox="938 1129 1250 1476">Sintaxis: <code>if icondicion_1:</code> <code>instrucciones_1</code> <code>elif condicion_2:</code> <code>instrucciones_2</code> <code>...</code> <code>elif condicion_N:</code> <code>instrucciones_N</code> <code>else:</code> <code>instrucciones_FALSO</code></div>
	<div data-bbox="938 1644 1230 1770">Sintaxis: <code>while test:</code> <code>cuerpo_del_ciclo;</code></div> <div data-bbox="938 1896 1047 1927">Sintaxis:</div>



5. Diagrama de flujos y python:

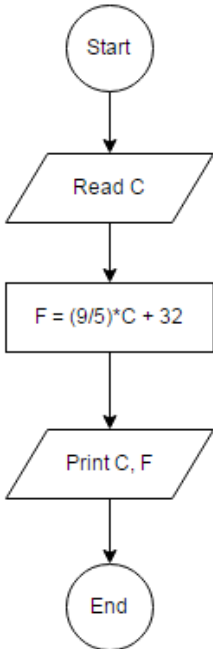
ALGUNOS EJEMPLOS DEL USO DE LAS DIFERENTES ESTRUCTURAS EN PYTHON	
Código Python	Salida en pantalla
<code>print("El tripa seca")</code>	<div style="border: 1px solid black; padding: 2px; display: inline-block;">El tripa seca</div>
<code>edad = input("Digite la edad:")</code>	<div style="border: 1px solid black; padding: 2px; display: inline-block;">Digite la edad: 6</div>
<code>x = 4; z = x + y;</code>	No hay salida en pantalla.
<code>edad = 23 if edad >= 18: print("Viejo")</code>	<div style="border: 1px solid black; padding: 2px; display: inline-block;">Viejo</div>
<code>edad = 15 if edad >= 18: print("Viejo") else: print("Culicagao")</code>	<div style="border: 1px solid black; padding: 2px; display: inline-block;">Culicagao</div>
<code>dia = 'j' if(dia == 'l' dia == 'L'): print("Lunes") elif(dia == 'm' dia == 'M'): print("Martes") elif(dia == 'w' dia == 'W'): print("Miercoles") elif(dia == 'j' dia == 'J'):</code>	<div style="border: 1px solid black; padding: 2px; display: inline-block;">Jueves</div>

<pre> print("Jueves") elif(dia == 'v' dia == 'V'): print("Viernes"); else: print("Fin de semana") </pre>	
<pre> l = 'a' while l <= 'm': print("%d - %c\n"%(l,l)) l +=2 } print("\nAdios pues") </pre>	<pre> 97 - a 99 - c 101 - e 103 - g 105 - i 107 - l Adios pues </pre>
<pre> N = 10 for i in range(1,11): print(i) print("\nAdios pues") </pre>	<pre> 1 2 3 4 5 6 7 8 9 10 Adios pues </pre>

6. Comentarios:





COMENTARIOS EN PYTHON	
Comentarios	Python
Única línea	# Comentario de una linea
Varias líneas	''' Comentario de varias líneas '''

7. Ejemplo de codificación típico:

IMPLEMENTACION DE UN PROGRAMA EN PYTHON	
Diagrama de flujo	Programa en Python
	<pre>''' Titulo: celcius_to_fahrenheit.py Autor: Informatica 2 Fecha: 14/04/2016 Descripción: Este programa convierte un valor en grados celcius a fahrenheit sin hacer uso de funciones. ''' C = float(input("Ingrese los °C: ")) F = (9.0/5)*C + 32 print(C,"°C =",F,"°F")</pre>

8. Funciones en Python:

Las funciones en Python involucran dos aspectos importantes los cuales son, la definición y la invocación. La siguiente tabla aclara el tema

FUNCIONES EN EN PYTHON	
Estructura en bloques	Estructura en Python
Definición	
	<p>Sintaxis:</p> <pre>def nameFuction(p1, p2, ..., pN): instrucciones</pre> <p>Ejemplo: Hacer una función que retorne el mínimo de dos números.</p>  <pre>def mini(x, y): if (x < y): r = x else: r = y return r</pre>
Invocación	
	<p>Sintaxis:</p> <pre>r = nameFuction(v1, v2, ..., vp)</pre> <p>Ejemplo: Invocar la función previamente creada pasando 3 y 4 como argumentos.</p>  <pre>resp = mini(3, 4):</pre>

9. Ejemplo de codificación típico con funciones:

IMPLEMENTACION DE UN PROGRAMA EN PYTHON	
Diagrama de flujo	Programa en Python
<pre> graph TD Start((Start)) --> ReadC[/Read C/] ReadC --> cC[c = C] cC --> celcius_to_fahrenheit{celcius_to_fahrenheit} celcius_to_fahrenheit --> Ff[F = f] Ff --> PrintCF[/Print C, F/] PrintCF --> End((End)) </pre>	<pre> ''' Titulo: celcius_to_fahrenheit.py Autor: Informatica 2 Fecha: 14/04/2016 Descripción: Este programa convierte un valor en grados celcius a fahrenheit sin hacer uso de funciones. ''' # Funcion que convierte de celcuis a # fahrenheit (Definicion) def celcius_to_fahrenheit(c): f = (9.0/5)*c + 32 return f C = float(input("Ingrese los °C: ")) F = celcius_to_fahrenheit(C) # Invocacion de # la funcion print(C,"°C =",F,"°F") </pre>
	<p>Salida en pantalla</p> <pre> Ingrese los °C: 45 45° = 113.0° </pre>

10. Datos estructurados:

Strings

Los strings son inmutables (no se pueden modificar).

Con comillas sencillas:

s = 'Hello Goodbye!'

Con comillas dobles:

s = "Hello Goodbye!"

Efecto:

s

-14

-13

-12

-11

-10

-9

-8

-7

-6

-5

-4

-3

-2

-1

'H'

'e'

'l'

'l'

'o'

' '

'G'

'o'

'o'

'd'

'b'

'y'

'e'

'!'

0

1

2

3

4

5

6

7

8

9

10

11

12

13

Operadores

Operador	Acción	Ejemplo	Resultado
s[i]	Retorna el elemento i-ésimo del string s.	C = 'Hello world'	C[0] → 'H' C[-11] → 'H' C[-3] → 'r' C[8] → 'o'
len(s)	retorna la longitud del string s.	A = 'Hello world'	len(A) → 11
s1 + s2	Retorna la concatenación de las cadenas de caracteres s1 y s2.	B1 = 'Hello' B2 = 'nena'	B1 + B2 → 'Hellonena' B1 + ' ' + B2 → 'Hello nena'
n*s	repite n veces la cadena s.	D = 'ja'	3*D → 'jajaja'
s[start:end]	Retorna el segmento de la cadena s entre start y end - 1.	C = 'Hello world'	C[2:7] → 'llo w' C[:5] → 'Hello' C[6:] → 'world' C[:] → 'Hello world' C[-5:-1] → 'worl' C[-6:] → ' world' C[:-6] → 'Hello'
e in s	Retorna True si e está en el string s.	voc = 'a' frase = 'centella'	voc in frase → False 'a' in frase → True 'va' in 'vaca' → True 'va ' in 'vaca' → False
e not in L	Retorna True si e no está en el string s.	l = 'i' c = 'z' voc = 'aeiou'	l not in voc → False c not in voc → True
min(L)	Retorna el elemento más pequeño del string s.	frase = 'centella'	Max(frase) → 't' Max('aeiou') → 'u'
max(L)	Retorna el elemento más grande del string s.	frase = 'centella'	Min(frase) → 'a' Min('aeiou') → 'a'
for e in s	Itera sobre los elementos de s.	U = 'Antioquia'	for l in U: print(l) →

			A n t i o q u i a
Métodos			
Verificando el contenido de un string			
str.isalpha()	Devuelve True si str contiene solo letras.	s = 'Hello'	s.isalpha() → True
		s = 'Hello!'	s.isalpha() → False
str.isdigit()	Devuelve True si str contiene solo dígitos.	s = '124'	s.isdigit() → False
		s = '124A'	s.isdigit() → False
str.islower()	Devuelve True si str contiene solo letras minúsculas(o mayúsculas).	s = 'hello'	s.islower() → True
str.isupper()		s = 'Hello'	s.isupper() → False
Buscando el contenido de un string			
str.find()	Retorna el índice de la primera ocurrencia de w en str o -1 si w no está en str .	s = 'Hello!'	s.find('o') → 4
		s = 'Bye'	s.find('o') → -1
Reemplazando el contenido de un string			
str.replace(w,t)	Retorna una copia de str con todas las ocurrencias de w reemplazadas por t .	s = 'Hello'	s.replace('H', 'J') → 'Jello'
		s = 'Hello'	s.replace('ll', 'r') → 'Hero'
str.lower()	Retorna la versión minúscula (o mayúscula) de str .	s = 'Hello!'	s.lower() → 'hello!'
str.upper()		s = 'Hello!'	s.upper() → 'HELLO!'
str.strip()	Retorna una copia de str con los caracteres iniciales y finales en los que aparece w removidos.	s = ' Hello! '	s.strip(' !') → 'Hello'
		s = 'Hello\n'	s.strip('\n') → 'Hello'
		s = 'ababcdeababfghba'	s.strip('ba') → 'cdeababfgh'
str.lstrip()	Retorna una copia de str con los caracteres iniciales en los que aparece w removidos.	s = ' Hello! '	s.lstrip(' !') → 'Hello! '
		s = 'Hello\n'	s.lstrip('\n') → 'Hello\n'
		s = 'ababcdeababfghba'	s.lstrip('ba') → 'cdeababfghba'
str.rstrip()	Retorna una copia de str con los caracteres finales en los que aparece w removidos.	s = ' Hello! '	s.rstrip(' !') → ' Hello'
		s = 'Hello\n'	s.rstrip('\n') → 'Hello'
		s = 'ababcdeababfghba'	s.rstrip('ba') → 'ababcdeababfgh'
Dividiendo un string			
str.split(w)	Retorna una lista que contiene todos los strings delimitados por w .	s = 'gallo, gato'	s.split(',') → ['gallo', 'gato']

Listas																										
L1 = [1,2,3]																										
	L1	<table><tr><td>-3</td><td>-2</td><td>-1</td></tr><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>0</td><td>1</td><td>2</td></tr></table>	-3	-2	-1	1	2	3	0	1	2															
-3	-2	-1																								
1	2	3																								
0	1	2																								
L2 = ["yellow", "blue", "red"]																										
	L2	<table><tr><td>-3</td><td>-2</td><td>-1</td></tr><tr><td>'yellow'</td><td>'blue'</td><td>'red'</td></tr><tr><td>0</td><td>1</td><td>2</td></tr></table>	-3	-2	-1	'yellow'	'blue'	'red'	0	1	2															
-3	-2	-1																								
'yellow'	'blue'	'red'																								
0	1	2																								
L3 = ["ala", -7, 3.3, True]																										
	L3	<table><tr><td>-4</td><td>-3</td><td>-2</td><td>-1</td></tr><tr><td>'ala'</td><td>-7</td><td>3.3</td><td>True</td></tr><tr><td>0</td><td>1</td><td>2</td><td>3</td></tr></table>	-4	-3	-2	-1	'ala'	-7	3.3	True	0	1	2	3												
-4	-3	-2	-1																							
'ala'	-7	3.3	True																							
0	1	2	3																							
L4 = [[2, 3, 4], 5, ['a', 'b']]																										
	L4	<table><tr><td>-3</td><td>-2</td><td>-1</td><td>-2</td><td>-1</td></tr><tr><td>2</td><td>3</td><td>4</td><td>5</td><td>a</td><td>b</td></tr><tr><td>0</td><td>1</td><td>2</td><td></td><td>0</td><td>1</td></tr><tr><td></td><td>0</td><td></td><td>1</td><td></td><td>2</td></tr></table>	-3	-2	-1	-2	-1	2	3	4	5	a	b	0	1	2		0	1		0		1		2	
-3	-2	-1	-2	-1																						
2	3	4	5	a	b																					
0	1	2		0	1																					
	0		1		2																					

Operadores			
Operador	Acción	Ejemplo	Resultado
L[i]	retorna el elemento i-ésimo de lista L.	A = [1,2,3,4,-3]	A[0] → 1 A[5] → -3 A[-2] → 4
Len(L)	retorna la longitud de la lista L.	A = [1,2,3,'oso'] B = [[1,2],3] C = []	len(A) → 4 len(B) → 2 len(C) → 0
L1 + L2	Retorna la concatenación de las listas L1 y L2.	x = ['a', 'b'] y = [1,2,3]	x + y → ['a', 'b', 1, 2, 3]
n*L	repite n veces la lista L.	z = [-1,-2,0]	2*z → [-1, -2, 0, -1, -2, 0]
L[start:end]	Retorna el segmento de la lista L entre start y end - 1.	A = [1,2,5,4,-3]	A[1:3] → [2,5] A[:4] → [1,2,5,4] A[3:] → [4,-3] A[:] → [1,2,5,4,-3]
e in L	Retorna True si e está en la lista L.	A = ['a', 'bc']	'bc' in A → True 'b' in A → False
e not in L	Retorna True si e no está en la lista L	A = ['a', 'bc']	'bc' not in A → False 'b' not in A → True
min(L)	Retorna el elemento más pequeño de la lista L	t = [-10,4,-2,0]	min(t) → -10

max(L)	Retorna el elemento más grande de la lista L	t = [-10,4,-2,0]	max(t) → 4
for e in L	Itera sobre los elementos de L	B = [1,2,3]	for num in B: print(num) → 1 2 3
Métodos			
list.append(e)	Agrega el objeto e al final de la lista	L = [1, 2, 3] M = ['a', 'b']	L.append(-5) → [1, 2, 3, -5] M.append([2,3]) → ['a', 'b', [2,3]]
list.count(e)	Retorna el número de veces que e se encuentra en la lista	L = ['a', 'b', 'c', 'b']	L.count('b') → 2
list.insert(i, e)	Inserta el objeto e en la posición i de la lista	L = ['a', 'b', 'c']	L.insert(1, 'x') → ['a', 'x', 'b', 'c']
list.extend(list2)	Agrega todos los elementos de list2 al final de la lista	M = ['a', 'b']	M.extend([2,3]) → ['a', 'b', 2, 3]
list.remove(e)	borra el primer elemento e que encuentre	L = ['a', 'b', 'c', 'b']	L.remove('b') → ['a', 'c', 'b']
list.index(e)	retorna la posición del primer elemento e que encuentre	L = ['a', 'b', 'c', 'b']	L.index('b') → 1
list.pop(i)	borra el elemento de la posición i	L = ['a', 'b', 'c']	L.pop(1) → ['a', 'c']
list.sort()	ordena la lista	L = [2, 1, 3, 2]	L.sort() → [1, 2, 2, 3]
list.reverse()	invierte el orden de la lista	L = [1, 2, 3]	L.reverse() → [3, 2, 1]

Tuplas

Son similares a las listas solo que no sus valores no se pueden modificar (Propiedad de inmutabilidad).

L1 = (1,2,3)

	-3	-2	-1
L1	1	2	3
	0	1	2

L2 = ("yellow", "blue", "red")

	-3	-2	-1
L2	'yellow'	'blue'	'red'
	0	1	2

L3 = ("ala", -7, 3.3, True)

	-4	-3	-2	-1
L3	'ala'	-7	3.3	True
	0	1	2	3

Operadores y metodos			
Operador	Acción	Ejemplo	Resultado
T[i]	retorna el elemento i-ésimo de lista L .	T = (1,2,3,4,-3)	T[0] → 1 T[5] → -3 T[-2] → 4
Len(T)	retorna la longitud de la lista L .	A = (1,2,3,'oso') B = ([1,2],3) C = ()	len(A) → 4 len(B) → 2 len(C) → 0
T[start:end]	Retorna el segmento de la lista L entre start y end - 1 .	A = (1,2,5,4,-3)	A[1:3] → (2,5) A[:4] → (1,2,5,4) A[3:] → (4,-3) A[:] → (1,2,5,4,-3)
T1 + T2	Retorna la concatenación de las tuplas T1 y T2 .	x = ('a', 'b') y = (1,2,3)	x + y → ('a', 'b', 1, 2, 3)
n*T	repite n veces la tupla T .	z = (-1,-2,0)	2*z → (-1, -2, 0, -1, -2, 0)
e in T	Retorna True si e está en la tupla T .	A = ('a', 'bc')	'bc' in A → True 'b' in A → False
e not in T	Retorna True si e no está en la tupla T	A = ('a', 'bc')	'bc' not in A → False 'b' not in A → True
T.count(e)	Retorna la cantidad de veces que e se encuentra en la tupla T .	A = ('a', 'bc')	A.count('bc') → 1 A.count('b') → 0
T.index(e)	Retorna el índice en el cual e se da la primera ocurrencia de e en la tupla T .	A = ['a', 'bc']	A.index('bc') → 1 A.index('a') → 0
min(T)	Retorna el elemento más pequeño de la tupla T	t = (-10,4,-2,0)	min(t) → -10
max(T)	Retorna el elemento más grande de la tupla T	t = (-10,4,-2,0)	max(t) → 4
for e in T	Itera sobre los elementos de T .	B = (1,'xxx', [1,2],3)	for e in B: print(e) → 1 xxx [1,2] 3

Diccionarios							
D1 = {"111-34-3434":"John", "132-56-6290":"Peter"}							
D1	<table border="1"> <tr> <td>"John"</td><td>"Peter"</td></tr> <tr> <td>"111-34-3434"</td><td>"132-56-6290"</td></tr> </table>	"John"	"Peter"	"111-34-3434"	"132-56-6290"		
"John"	"Peter"						
"111-34-3434"	"132-56-6290"						
D1	<table border="1"> <tr> <th>key</th><th>value</th></tr> <tr> <td>"111-34-3434"</td><td>"Peter"</td></tr> <tr> <td>"132-56-6290"</td><td>"John"</td></tr> </table>	key	value	"111-34-3434"	"Peter"	"132-56-6290"	"John"
key	value						
"111-34-3434"	"Peter"						
"132-56-6290"	"John"						

Representación Forma 1:

D1

"John"	"Peter"
"111-34-3434"	"132-56-6290"

Representación Forma 2:

D1

key	value
"111-34-3434"	"Peter"
"132-56-6290"	"John"

D2 = {'A1': 80, 'A2': 90, 'A3': 90}

Representación Forma 1:

D2

80	90	90
"A1"	"A2"	"A3"

Representación Forma 2:

D2

key	value
"A1"	80
"A2"	90
"A3"	90

D3 = {'apple': 1, 3: 4}

Representación Forma 1:

D3

1	4
"apple"	3

Representación Forma 2:

D3

key	value
"apple"	1
3	4

Operadores con diccionarios

Operador	Acción	Ejemplo	Resultado
len(Dic)	Retorna la longitud del Diccionario Dic.	A = {'a':1,'b':2,'c':3} B = {'ab':1,'b':[2,3]} C = {}	len(A) → 3 len(B) → 2 len(C) → 0
key in Dic	Retorna True si la clave key está en el Diccionario Dic.	A = {'a':1,'b':2,'c':3}	'a' in A → True 'd' in A → False
key not in Dic	Retorna True si la clave key no está en el Diccionario Dic.	A = {'a':1,'b':2,'c':3}	'a' not in A → False 'd' not in A → True
Dic1 == Dic2	Devuelve True si dos diccionarios contienen los mismos ítems.	d1 = {"red":41, "blue":3} d2 = {"blue":3, "red":41}	d1 == d2 → True
Dic1 != Dic2	Devuelve True si dos diccionarios contienen diferentes ítems.	d1 = {"red":41, "blue":3} d2 = {"blue":3, "red":41}	d1 != d2 → False
Dic[key]	Retorna el valor asociado a la clave key.	d1 = {"red":41, "blue":3}	d1["red"] → 41 d1["r"] → Error
Dic[key] = v	Asocia la clave key con el valor v; si key no existía, crea un nuevo elemento en el diccionario.	d1 = {"A":1, "B":2}	d1["A"] = -1 → d1 = {"A":-1, "B":2} d1["C"] = -1 → d1 = {"A":1, "B":2, "C":-1}
for k in Dic	Itera sobre las claves de Dic.	mdays = { 'enero':31,	for e in modays: print(e, mdays[e])

		'febrero':28, 'marzo':31, 'abril':30 }	→ febrero 28 abril 30 marzo 31 enero 31
Métodos sobre diccionarios			
dic.keys()	Retorna una secuencia con las claves de dic.	A = {'a':1,'b':2,'c':3} B = {'ab':1,'b':[2,3]} C = {}	A.keys() → ['a', 'b', 'c'] B.keys() → ['ab', 'b'] C.keys() → []
dic.values()	retorna una secuencia con los valores de dic.	A = {'a':1,'b':2,'c':3} B = {'ab':1,'b':[2,3]} C = {}	A.values() → [1, 2, 3] B.values() → [1, [2, 3]] C.values() → []
dic.items()	Retorna una secuencia de tuplas, cada tupla es de la forma (key, value) para un item.	A = {'a':1,'b':2,'c':3}	A.items() → [('b',2),('c',3),('a',1)]
dic.get(k,[e])	retorna dic[k] si k está en d, de lo contrario retorna e. e es un parámetro opcional y por defecto es None.	A = {'a':1,'b':2,'c':3}	A.get('c',4) → 3 A.get('f',4) → 4 A.get('b') → 2 A.get('d') → None
dic.pop(k)	Elimina el elemento con clave k y retorna su valor asociado	A = {'a':1,'b':2,'c':3}	A.pop('b') • Retorna: 2 A queda: A = { 'a':1, 'c':3}
dic.popitem()	Retorna un par key/value aleatoriamente seleccionado como una tupla y remueve el item seleccionado.	A = {'a':1,'b':2,'c':3}	A.popitem() • Retorna: ('b',2) A queda: A = { 'a':1, 'c':3}
dic.clear()	Borra todas las entradas del diccionario dic dejándolo vacío, no retorna nada.	A = { 'a':1, 'b':2, 'c':3}	A.clear() → A = {}
dic.update(dic2)	Agrega el contenido (pares key/value) del diccionario dic2 al diccionario dic.	A = {'a':1,'b':2} B = { 'xyz':1, 'm':[2,3]}	A.update(B) → A = { 'a':1, 'b':2, 'xyz':1, 'm':[2,3]} →

	Esta función no retorna nada.		B = {'xyz':1, 'm':[2,3]}
--	----------------------------------	--	-----------------------------