Universidad del Rosario | Educación Continua y Consultoría

Diplomado en Ciencia de Datos

#URSolucionesInnovadoras

#URConsultoría

Universidad del Rosario | Educación Continua y Consultoría

PROGRAMACIÓN 2

Estructuras de datos básicas

Universidad del Rosario | Educación Continua y Consultoría

Contenido

- Tuplas
- Listas
- Funciones de secuencia
- Diccionarios
- Conjuntos
- Listas, conjuntos y diccionarios por comprensión
- Algunos retos de programación

Universidad del Rosario | Educación Continua y Consultoría

Contenido

- Tuplas
- Listas
- Funciones de secuencia
- Diccionarios
- Conjuntos
- Listas, conjuntos y diccionarios por comprensión
- Algunos retos de programación

Universidad del Rosario | Educación Continua y Consultoría

Contenido

- **Tuplas**
- Listas
- Funciones de secuencia
- Diccionarios
- Conjuntos
- Listas, conjuntos y diccionarios por comprensión
- Algunos retos de programación

Tuplas

Definición

- Es una secuencia de tamaño fijo
- Es inmutable
- Es una secuencia de valores separados por comas

```
tupla = 4, 5, 6
```

- Se pueden crear tuplas con tuplas adentro, hay que usar paréntesis para esto

```
tupla_anidada = (4, 5, 6), (7, 8)
```

Tuplas

Índices

```
tupla_str = tuple('string')
```

- Para consultar los elementos de una tupla se usan los corchetes cuadrados

- Las secuencias están indexadas desde cero

```
tupla_str[0]
tupla_str[1]
tupla_str[2]
tupla_str[3]
tupla_str[4]
tupla_str[5]
```

Tuplas

Inmutabilidad

- Una vez la tupla se crea, no se pueden modificar los objetos de cada espacio.

```
tupla4 = tuple(['Hola', [1,2], True])

tupla4[2] = False
```

- Pero si un objeto dentro de una tupla es mutable, se puede modificar.

```
tupla4[1].append(3)
```

Tuplas

Concatenación

- Se pueden concatenar tuplas para hacer tuplas más largas
- El operador de concatenar es el +

```
(4, None, 'Hola') + (6, 0) + ('Adios', )
```

- Multiplicar una tupla por un entero es similar a concatenar varias veces una tupla

```
(6, 0) * 4
```

Tuplas

Desempaquetado

- Asignar los valores dentro de una tuple a una variable

```
tupla = 4, 5, 6
a, b, c = tupla
```

- Inclusive tuplas con tuplas anidadas se pueden desempacar

```
tupla5 = 4 ,5 (6, 7)
a, b, (c, d) = tupla5
```

Tuplas

Desempaquetado

- En la mayoría de lenguajes, cuando se quiere intercambiar nombres de variables, hay que usar una variable temporal

```
a = 10; b = 5;
temp = a
a = b
b = temp
```

- Usando el desempacado de tuplas se puede hacer fácilmente

```
a, b = 10, 5
b, a = a, b
```

Tuplas

Desempaquetado

- Uso práctico: iterar sobre secuencias de tuplas o listas

```
seq = [(1, 2, 3),(4, 5, 6),(7, 8, 9)]

for a, b, c in seq:
    print('a={0}, b={1}, c={2}'.format(a, b, c))
```

- El arreglo seq tiene tres elementos (que son tuplas)
- Cada elemento es despempacado en las variables a, b y c

Tuplas

Pluck

Pluck: Sacar

- En ocasiones queremos sacar elementos del comienzo de una tupla. Para eso se emplea la síntaxis *rest

```
valores =  1, 2, 3, 4, 5
a, b, *rest = valores
```

```
a, b
```

```
rest
```

- Como convención, las variables no deseadas se marcan con barra al piso (_)

```
a, b, *_ = valores
```

Tuplas

Algunos métodos

Como son inmutables tienen pocos

count() cuenta el número de ocurrencias de un valor

```
a = 1,2,2,2,3,4,2

a.count(2)
```

¿Cuántas veces está el 2 en la tupla?

Universidad del Rosario | Educación Continua y Consultoría

Contenido

- Tuplas
- **Listas**
- Funciones de secuencia
- Diccionarios
- Conjuntos
- Listas, conjuntos y diccionarios por comprensión
- Algunos retos de programación

Universidad del Rosario | Educación Continua y Consultoría

Listas

Definición

Son de longitud variable

Su contenido puede ser modificado

Se definen con corchete cuadrado [] o usando la función list

```
lista_a = [2, 3, 7, None]

tupla = ('uno', 'two', 'tres')
lista_b = list(tupla)
lista_b[1] = 'dos'
```

Se usa para convertir un iterador en lista

```
gen = range(10)
```

→

```
list(gen)
```

Operaciones

Se agregan elementos con append()

insert() permite insertar un elemento dado un índice

```
lista_b
lista_b.append('cuatro')
lista_b
lista_b.insert(1, 'uno y medio')
```

El inverso de insert() es pop(). Se quitan elementos con pop() usando el índice

```
lista_b.pop(3)
```

remove() busca el primer elemento cuyo valor es igual al pasado por parámetro uy lo borra

```
lista_b.remove('uno')
```

Listas

Operaciones

- Para revisar si un elemento está en una lista

```
'dos' in lista_b
```

```
'dos' not in lista_b
```

- Para concatenar y combinar listas, se hace de forma similar a las tuplas (operador +)

```
[4, None, 'foo'] + [7, 8, (2, 3)]
```

- Si ya existe el arreglo, se pueden agregar elementos múltiples con el comando extent

```
x = [4, None, 'foo']
x.extend([7, 8, (2, 3)])
```

Listas

Ejercicio

Modelar un tablero de ajedrez con arreglos, de tal forma que podamos recuperar el color de la casilla dada su posición en x y y

tablero[7][7] es negra

tablero[4][3] es blanca

tablero[0][0] es negra

Universidad del Rosario | Educación Continua y Consultoría

Listas

Ejercicio (solución)

```
tablero = []

for i in range(8):
    tablero.append([])
    for j in range(8):
        tablero[i].append('Negro' if (i + j) % 2 == 0 else 'Blanco')

tablero[0][0]
tablero[7][7]
tablero[4][3]
```

Listas

Ordenamiento

Ordenando

Se puede llamar la función sort() para ordenar elementos del arreglo

```
a = [7, 2, 5, 1, 3]
a.sort()
```

Se le puede pasas a sort() una llave de ordenamiento. P.ej. Ordenar un arreglo de strings por longitud de cada string

```
b = ['saw', 'small', 'He', 'foxes', 'six']
b.sort(key=len)
```

Listas

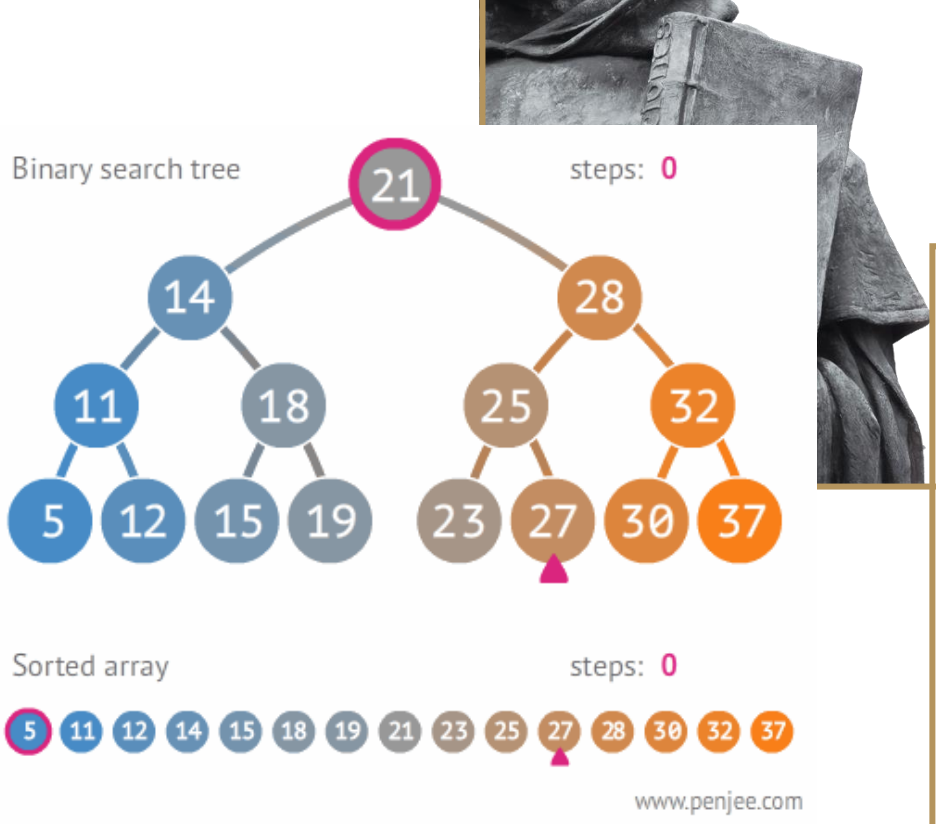
Ordenamiento

El módulo bisect implementa búsqueda e inserción binaria

bisect() busca el espacio donde debería ir el element

insort() adiciona el elemento en la ubicación

NOTA: usar solo con arreglos ordenados




```
import bisect

c = [1, 2, 2, 2, 3, 4, 7]

bisect.bisect(c, 2)
bisect.bisect(c, 5)
bisect.insort(c, 6)
```

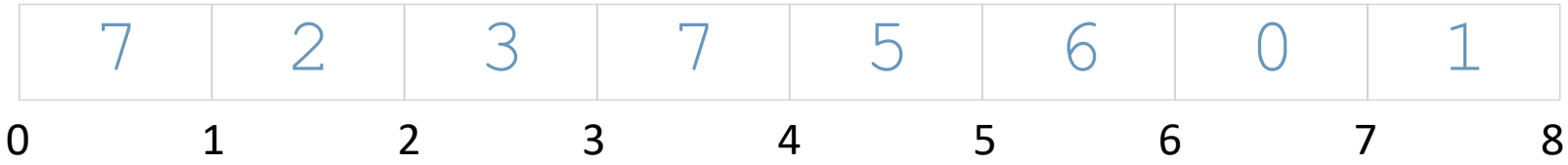
Listas

Slicing

Sacar tajadas

En el operador de índice, se introducen valores de inicio:fin

```
seq = [7, 2, 3, 7, 5, 6, 0, 1]
seq[1:5]
```



¿Qué hace la asignación `seq[3:4] = [6, 3]` ?

Listas

Slicing

Se puede omitir el índice de inicio o de fin

```
seq[3:]
```

```
seq[:5]
```

Se pueden usar también índices negativos

7	2	3	7	5	6	0	1
0	1	2	3	4	5	6	7
-8	-7	-6	-5	-4	-3	-2	-1

```
seq = [7, 2, 3, 7, 5, 6, 0, 1]
seq[-6:-3]
```

Listas

Slicing: Sacar tajadas

Se puede agregar un tercer valor en el índice para indicar saltos

```
seq[::2]
```

Saltos de 2 en 2

```
seq[::-1]
```

Reversando la lista

Universidad del Rosario | Educación Continua y Consultoría

Contenido

- Tuplas
- Listas
- **Funciones de secuencia**
- Diccionarios
- Conjuntos
- Listas, conjuntos y diccionarios por comprensión
- Algunos retos de programación

Universidad del Rosario | Educación Continua y Consultoría

Funciones de secuencia

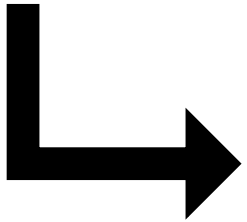
Enumerar

enumerate()

Agrega información del índice de la iteración

```
coleccion = [2,3,6,44,12,9]
```

```
i = 0
for valor in coleccion:
    print('Este es el ciclo {0}'.format(i))
    i += 1
```



```
for i, valor in enumerate(coleccion):
    print('Este es el ciclo {0}'.format(i))
```

Funciones de secuencia

Mapeo a diccionario

Cuando está indexando datos, un patrón útil que utiliza enumerate es calcular un **dict** que asigna los valores de una secuencia (que se supone que son únicos) a sus ubicaciones en la secuencia

```
una_lista = ['Andres', 'Beatriz', 'Carlos']

mapping = {}

for indice, valor in enumerate(una_lista):
    mapping[valor] = indice
```

```
mapping['Andres']
```

Universidad del Rosario | Educación Continua y Consultoría

Funciones de secuencia

Otras operaciones

`sorted()` retorna una nueva lista ordenada con los elementos de cualquier secuencia

```
sorted((1,4,8,2))
sorted([2,3,4,7,4,1,2,8,9])
sorted('buenas tardes')
```

`zip()` empareja elementos de listas o tuplas para crear una lista de tuplas

```
seq1 = ['uno', 'dos', 'tres']
seq2 = ['a' , 'be' , 'ce' ]

lista_de_tuplas = zip(seq1, seq2)
list(lista_de_tuplas)
```

Funciones de secuencia

Otras operaciones

zip() puede tomar muchas secuencias, y la lista que produce está deteminada por la secuencia más corta

```
seq1 = ['uno', 'dos', 'tres']
seq2 = ['a', 'be', 'ce']
seq3 = [False, True]

lista_de_tuplas = zip(seq1, seq2, seq3)
list(lista_de_tuplas)
```

Un uso muy común de zip() es iterar simultáneamente en múltiples secuencias, posiblemente también combinado con enumerate

```
seq1 = ['uno', 'dos', 'tres']
seq2 = ['a', 'be', 'ce']
for i, (a, b) in enumerate(zip(seq1, seq2)):
    print('i={0} a={1} b={2}'.format(i, a, b))
```

Funciones de secuencia

Otras operaciones

Para hacer unzip de una secuencia

```
nombres_completos = [('Andres', 'Agudelo'),
                     ('Brayan', 'Beltran'),
                     ('Camila','Cruz')]

nombres, apellidos = zip(*nombres_completos)
```

reverse se usa para invertir el orden de una lista

```
list(reversed(range(10)))
```

Universidad del Rosario | Educación Continua y Consultoría

Contenido

- Tuplas
- Listas
- Funciones de secuencia
- **Diccionarios**
- Conjuntos
- Listas, conjuntos y diccionarios por comprensión
- Algunos retos de programación

Diccionarios

Definición

```
diccionario_vacio = {}

d1 = {'a' : 'algun valor', 'b' : [1, 2, 3, 4]}
```

- Diccionario, *hash map* o array asociativo
- Estructura de datos más importante
- Colección de tamaño flexible de parejas llave-valor, donde la llave y el valor son objetos de Python
- Se crean con corchetes {}
- Las llaves se separan de los valores con dos puntos

Diccionarios

Operaciones

Se pueden acceder o insertar elementos usando la misma síntaxis con la que se acceden elementos de un arreglo o tupla:

```
d1 = {'a' : 'algun valor', 'b' : [1, 2, 3, 4]}

d1[7] = 'un entero'
d1['b']
```

Se pueden preguntar si una llave está en un diccionario:

```
'a' in d1
```

Diccionarios

Borrado de elementos

Se borran valores con la palabra del o con el método pop()

```
d1 = {'a' : 'algun valor', 'b' : [1, 2, 3, 4]}

d1[7] = 'un entero'
d1[5] = 'algun valor'
d1['dummy'] = 'otro valor'
```

```
d1
del d1[5]
d1
```

```
a_borrar = d1.pop('dummy')
a_borrar
d1
```

Diccionarios

Llaves y valores

Es posible pedir una lista con las llaves y otra lista con los valores del diccionario

- Las listas retornadas por ambos tiene el mismo orden

```
list(d1.keys())
list(d1.values())
```

- Se pueden unir dos diccionarios con el método update

```
d1
d1.update({'b': 'un nuevo valor', 'c': 12})
d1
```

Diccionarios

Creación

```
lista_llaves = ['llave1','llave2','llave3']
lista_valores = ['valor1','valor2','valor3']
```

Creando diccionarios de secuencias

```
mapping = {}
for llave, valor in zip(lista_llaves, lista_valores):
    mapping[llave] = valor
```

Un diccionario es una colección de 2-tuplas, por eso la función dict acepta como parámetro una lista de 2-tuplas

```
mapping_v2 = dict(zip(lista_llaves, lista_valores))
```

Dicionarios

Valores por defecto

Lógica

```
if llave in algun_diccionario:
    valor = algun_diccionario[llave]
else:
    valor = valor_default
```

Implementación

```
valor = algun_diccionario.get(llave, valor_default)
```

NOTA: Si el get no tiene valor por defecto, retorna un None

Diccionarios

Valores por defecto

```
palabras = ['amarillo', 'bate', 'bosque', 'ahorro', 'burro']
```

Lógica

```
por_letra = {}
for palabra in palabras:
    inicial = palabra[0]
    if inicial not in por_letra:
        por_letra[inicial] = [palabra]
    else:
        por_letra[inicial].append(palabra)
```

Implementación

```
por_letra_v2 = {}
for palabra in palabras:
    inicial = palabra[0]
    por_letra_v2.setdefault(inicial, []).append(palabra)
```

Diccionarios

Librería defaultdic

```
palabras = ['amarillo', 'bate', 'bosque', 'ahorro', 'burro']
```

Lógica

```
por_letra = {}
for palabra in palabras:
    inicial = palabra[0]
    if inicial not in por_letra:
        por_letra[inicial] = [palabra]
    else:
        por_letra[inicial].append(palabra)
```

Implementación

```
from collections import defaultdict
por_letra_v3 = defaultdict(list)
for palabra in palabras:
    por_letra_v3[palabra[0]].append(palabra)
```

Diccionarios

Hashabilidad

Hashabilidad: Las llaves del diccionario tienen que ser inmutables de tipo escalar (int, float, string) o tuplas (los objetos dentro de la tupla deben ser inmutables también)

```
hash('string')
hash((1,2,(2,3)))
hash((1,2,[2,3]))
```

¿cuál de estos tres no funciona y por qué?

Para usar una lista como llave, hay que convertirla en tupla, la cual será hashable si sus elementos lo son

```
d = {}
d[tuple([1, 2, 3])] = 5
```

Universidad del Rosario | Educación Continua y Consultoría

Contenido

- Tuplas
- Listas
- Funciones de secuencia
- Diccionarios
- **Conjuntos**
- Listas, conjuntos y diccionarios por comprensión
- Algunos retos de programación

Conjuntos

Definición

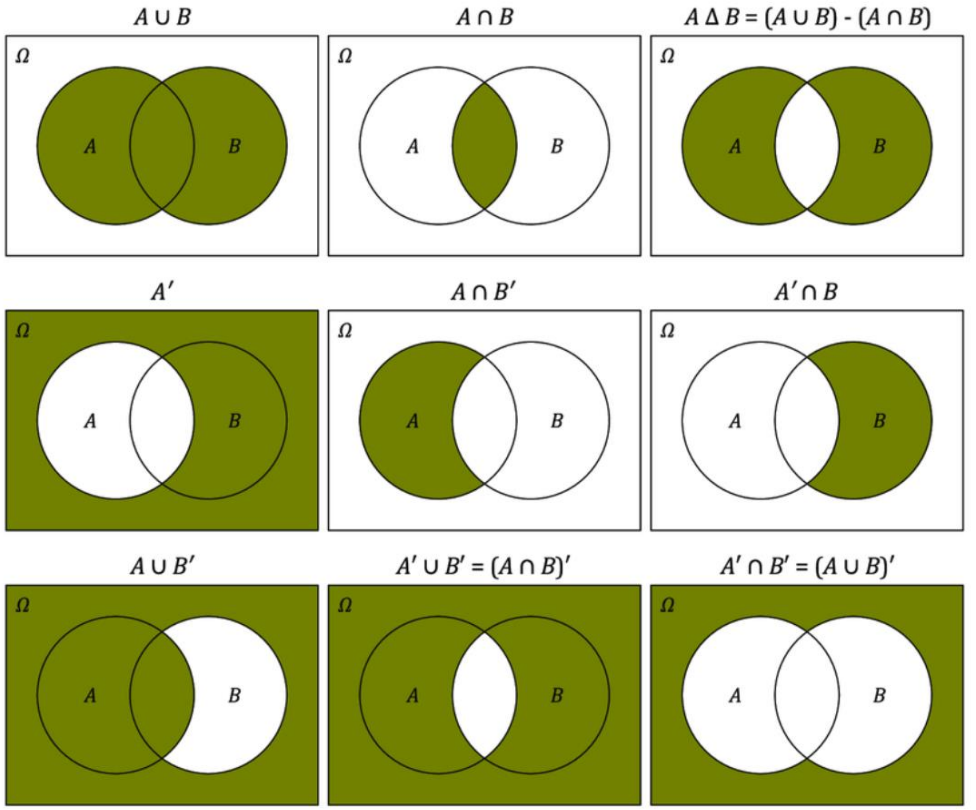
Colección sin orden de elementos únicos

- Son como dicts pero solo llaves, sin valores

```
set([2, 2, 2, 1, 3, 3])
```

```
{2, 2, 2, 1, 3, 3}
```

- Se pueden declarar con la función set o con corchetes {}



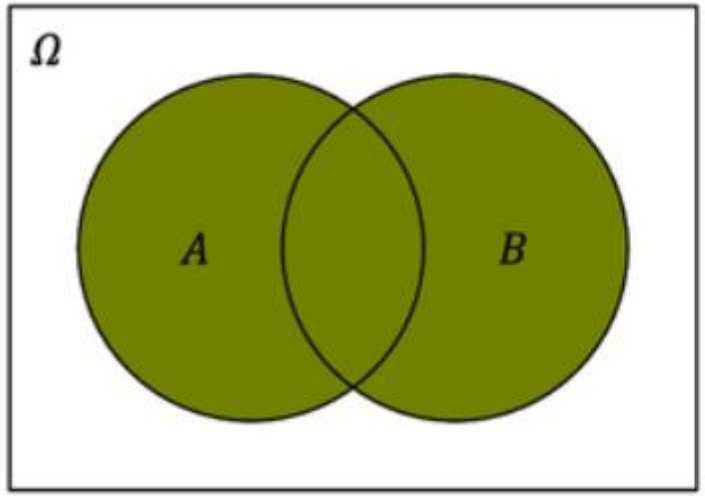
Soportan operaciones matemáticas de conjuntos como union, intersección, diferencia y diferencia simétrica

Conjuntos

Operaciones

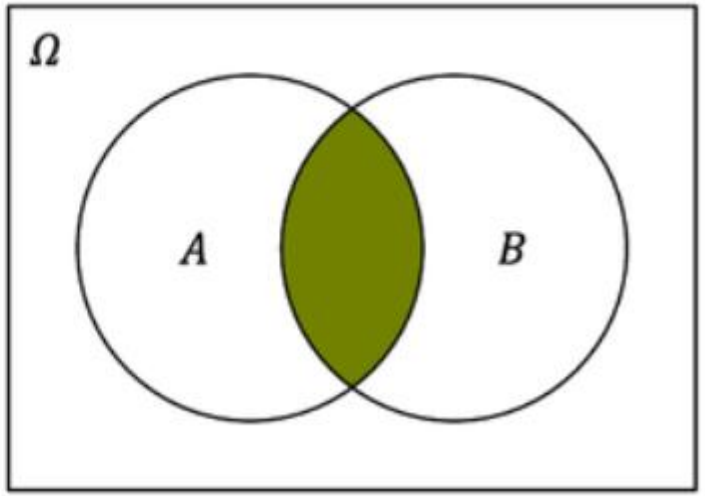
```
a = {1, 2, 3, 4, 5}
b = {3, 4, 5, 6, 7, 8}
```

$A \cup B$




```
a.union(b)
a | b
```

$A \cap B$




```
a.intersection(b)
a & b
```

Conjuntos

Operaciones

```
c= a.copy()
c |= b
c
```

```
d = a.copy()
d &= b
d
```

Function	Alternative syntax	Description
`a.add(x)`	N/A	Add element `x` to the set `a`
`a.clear()`	N/A	Reset the set `a` to an empty state, discarding all of its elements
`a.remove(x)`	N/A	Remove element `x` from the set `a`
`a.pop()`	N/A	Remove an arbitrary element from the set `a`, raising `KeyError` if the set is empty
`a.union(b)`	`a | b`	All of the unique elements in `a` and `b`
`a.update(b)`	`a |= b`	Set the contents of `a` to be the union of the elements in `a` and `b`
`a.intersection(b)`	`a & b`	All of the elements in *both* `a` and `b`
`a.intersection_update(b)`	`a &= b`	Set the contents of `a` to be the intersection of the elements in `a` and `b`
`a.difference(b)`	`a - b`	The elements in `a` that are not in `b`
`a.difference_update(b)`	`a -= b`	Set `a` to the elements in `a` that are not in `b`
`a.symmetric_difference(b)`	`a ^ b`	All of the elements in either `a` or `b` but *not both*
`a.symmetric_difference_update(b)`	`a ^= b`	Set `a` to contain the elements in either `a` or `b` but *not both*
`a.issubset(b)`	N/A	`True` if the elements of `a` are all contained in `b`
`a.issuperset(b)`	N/A	`True` if the elements of `b` are all contained in `a`
`a.isdisjoint(b)`	N/A	`True` if `a` and `b` have no elements in common

Conjuntos

Operaciones

```
a_set = {1, 2, 3, 4, 5}
```

Un conjunto está contenido en (es subset)

```
{1, 2, 3}.issubset(a_set)
```

Un conjunto contiene todos los elementos de (es superset)

```
a_set.issuperset({1, 2, 3})
```

Los conjuntos son iguales solo si el contenido de ambos es igual

```
{1, 2, 3} == {3, 2, 1}
```

Universidad del Rosario | Educación Continua y Consultoría

Contenido

- Tuplas
- Listas
- Funciones de secuencia
- Diccionarios
- Conjuntos
- **Listas, conjuntos y diccionarios por comprensión**
- Algunos retos de programación

Definición

Una de las características más queridas de Python

Permite formar una nueva lista de forma concisa al filtrar los elementos de una colección, transformando los elementos que pasan el filtro en una expresión concisa.

Lógica

```
result = []
for val in collection:
    if condition:
        result.append(expr)
```

Implementación

```
[expr for val in collection if condition]
```

Ejemplo

```
[expr for val in collection if condition]
```

Ejemplo:

```
strings = ['a', 'as', 'bat', 'car', 'dove', 'python']
```

¿Qué lista se produce con el siguiente comando?

```
[x.upper() for x in strings if len(x) > 2]
```

Universidad del Rosario | Educación Continua y Consultoría

Listas, conjuntos y diccionarios por comprensión

Creación

Listas

```
[expr for val in collection if condition]
```

Diccionarios

```
{key-expr:value-expr for val in collection if condition}
```

Conjuntos

```
{expr for val in collection if condition}
```

Ejemplo

```
strings = ['a', 'as', 'bat', 'car', 'dove', 'python']
```

Suponga que queremos conocer la longitud de los strings que se encuentran en la lista strings

```
long_unicas = {len(x) for x in strings}
```

Se puede expresar de forma más funcional con la función map()

```
set(map(len, strings))
```

Universidad del Rosario | Educación Continua y Consultoría

Listas, conjuntos y diccionarios por comprensión

Diccionario por comprensión

```
loc_mapping = {val : index for index, val in enumerate(strings)}
```

Actividad:

Construya una lista que contenga todos los números primos entre 1 y 100

- Usando ciclos for y declaraciones condicionales
- Usando listas por comprensión

Actividad

Construya una lista que contenga todos los números primos entre 1 y 100

- Usando ciclos for y declaraciones condicionales
- Usando listas por comprensión

Anidación

Listas anidadas por comprensión

Supongamos dos listas de nombres

```
all_data = [['John', 'Emily', 'Michael', 'Mary', 'Steven'],
            ['Maria', 'Juan', 'Javier', 'Natalia', 'Pilar']]
```

Se creará una lista con los nombres que tengan dos **e** o más

```
nombres_sel = []

for nombres in all_data:
    a_agregar = [nombre for nombre in nombres if nombre.count('e') >= 2]
    nombres_sel.extend(a_agregar)
```

```
nombres_sel_v2 = [nombre for nombres in all_data for nombre in nombres if nombre.count('e') >= 2]
```

Universidad del Rosario | Educación Continua y Consultoría

Contenido

- Tuplas
- Listas
- Funciones de secuencia
- Diccionarios
- Conjuntos
- Listas, conjuntos y diccionarios por comprensión
- **Algunos retos de programación**

Universidad del Rosario | Educación Continua y Consultoría

Algunos retos de programación

Reto 1

John trabaja en una tienda de ropa. Tiene una gran pila de medias que debe combinar por color para la venta. Dado un arreglo de enteros que representan el color de cada media, determine cuántos pares de medias con colores coincidentes hay.

Por ejemplo, hay $n = 7$ medias de colores $ar = [1,2,1,2,1,3,2]$. Hay un par de color 1 y uno de color 2. Quedan tres medias impares, uno de cada color. El número de pares es 2.

Actividad: Complete la función sockMerchant. Debe devolver un número entero que represente el número de pares de medias coincidentes que están disponibles.

sockMerchant tiene los siguientes **parámetros**:

- n: el número de medias en el arreglo ar
- ar: arreglo con los colores de cada media. Cada $ar[i]$ es un color

```
def sockMerchant(n, ar):
```

Formato de entrada

Arreglo Python con el listado de números enteros que representa cada color. Ej:

Restricciones

$1 \leq n \leq 100$

$1 \leq ar[i] \leq 100$ donde $0 \leq i \leq n$

```
medias = [10, 20, 20, 10, 10, 30, 50, 10, 20]
```

Formato de salida

Devuelve el número total de pares de medias que John puede vender.

https://www.hackerrank.com/

```
sockMerchant(9, medias)
```

Universidad del Rosario | Educación Continua y Consultoría

Algunos retos de programación

Reto 2

Gary es un ávido excursionista. Sigue sus caminatas meticulosamente, prestando mucha atención a pequeños detalles como la topografía. Durante su última caminata, tomó exactamente los pasos. Por cada paso que dio, notó si era un ascenso o un descenso. Las caminatas de Gary comienzan y terminan a nivel del mar y cada paso hacia arriba o hacia abajo representa un cambio de unidad en la altitud. Definimos los siguientes términos:

- Una montaña es una secuencia de pasos consecutivos sobre el nivel del mar, comenzando con un paso hacia arriba desde el nivel del mar y terminando con un paso hacia el nivel del mar.
- Un valle es una secuencia de pasos consecutivos debajo del nivel del mar, comenzando con un paso hacia abajo desde el nivel del mar y terminando con un paso hacia el nivel del mar.

Dada la secuencia de pasos hacia arriba y hacia abajo de Gary durante su última caminata, encuentre e imprima la cantidad de valles por los que caminó.

Por ejemplo, si el camino de Gary es $S = [DDUUUUDD]$, primero ingresa a un valle de profundidad 2. Luego se sube y sube a una montaña de 2 unidades de altura. Finalmente, regresa al nivel del mar y termina su caminata.

Actividad: Complete la función countingValleys. Debe devolver un número entero que denote el número de valles que Gary atravesó.sockMerchant tiene los siguientes **parámetros**:

- n: el número de pasos en la caminata de Gary
- ar: arreglo describiendo la travesía

Formato de entrada

Arreglo Python con el listado de pasos de la travesía. Ej:

Restricciones
$2 \leq n \leq 10^6$
$ar[i] \in \{U, D\}$

Formato de salida

Devuelve el número total de valles que Gary atravesó.

```
def countingValleys(n, ar):
```

```
recorrido = 'UDDDUDUU'
```

```
countingValleys(8, recorrido)
```

https://www.hackerrank.com/

Unidad de Educación Continua y Consultoría
construimos país desde

#URSolucionesInnovadoras
#URConsultoría

@RosarioContinua /EduContinuaURosario @RosarioContinua