PONTIFICIA
UNIVERSIDAD
CATÓLICA
DE CHILE

INTRODUCCIÓN A LA PROGRAMACIÓN

Objetivos

- Recordar clase pasada
- Concluir exposición sobre **listas** y **tuplas**
- Comenzar con **listas de listas** (listas anidadas)
- Pequeño simulacro

Operaciones básicas

```
A = [1, 2.0, "iii", "cuatro"]      # lista de 4 elementos
B = []                             # lista sin  elementos
C = [1]                            # lista de un elemento
print( len(A) )                  # imprime el largo de A, 4

D = A + [5, 6.0, 'vii']          #  concatenacion de listas
print( len(D) )                  # imprime el largo de D, 7

print( 'iii' in A )               # True,  es elemento de A
print( 1 not in A )              # False, es elemento de A

print( D.index('vii') )          # 6, ya que D[6] es 'vii'
print( D.count('xii') )          # 0,  pues 'xii' not in D
```

¿¿Concatenar listas con tuplas??

- Conversión de tipos: así como `a=int(b)` convierte `b` en un `int` (sea `b` un `int`, `float`, `str`), podemos convertir una secuencia de un tipo en otra secuencia de otro tipo:

```
A = [1, 2.0, "iii", "cuatro"]                    # una lista
B = (5, 6.0, 'vii', 'eight')                     # una tupla

L = A + list(B)                  # concatena A con B-como-lista
M = tuple(A) + B                 # concatena A-como-tupla con B

X = list(A) + list(A)                # A ya era lista, se clona
```

Acceder a elementos según su índice

True	False	10	20	-5.5	'yup'	0	0	'nope'
0	1	2	3	4	5	6	7	8
-9	-8	-7	-6	-5	-4	-3	-2	-1

```
x = [True, False, 10, 20, -5.5, 'yup', 0, 0, 'nope']
```

Python nos deja acceder a los elementos de una lista de manera sencilla

- Basta indicar su índice o posición en la lista
- Funciona igual que con strings y tuplas

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Acceder a elementos según su índice

True	False	10	20	-5.5	'yup'	0	0	'nope'
0	1	2	3	4	5	6	7	8
-9	-8	-7	-6	-5	-4	-3	-2	-1

El uso de índices es como en ***strings***. Sabiendo esto, qué entrega:

- `len(x)` -->
- `x[ 1 ]` -->
- `x[ 6 ]` -->

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Acceder a elementos según su índice

True	False	10	20	-5.5	'yup'	0	0	'nope'
0	1	2	3	4	5	6	7	8
-9	-8	-7	-6	-5	-4	-3	-2	-1

El uso de índices es como en ***strings***. Sabiendo esto, qué entrega:

- `len(x)` --> **`9`**
- `x[ 1 ]` --> **`False`**
- `x[ 6 ]` --> **`0`**

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Acceder a elementos según su índice

True	False	10	20	-5.5	'yup'	0	0	'nope'
0	1	2	3	4	5	6	7	8
-9	-8	-7	-6	-5	-4	-3	-2	-1

El uso de índices es como en ***strings***. Sabiendo esto, qué entrega:

- `x[ -4 ]` -->
- `x[ -1 ]` -->
- `x[ -len(x) ]` -->

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Acceder a elementos según su índice

True	False	10	20	-5.5	'yup'	0	0	'nope'
0	1	2	3	4	5	6	7	8
-9	-8	-7	-6	-5	-4	-3	-2	-1

El uso de índices es como en ***strings***. Sabiendo esto, qué entrega:

- `x[ -4 ]` --> **`'yup'`**
- `x[ -1 ]` --> **`'nope'`**
- `x[ -len(x) ]` --> **`True`**

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Rebanadas (*slices*) de listas

True	False	10	20	-5.5	'yup'	0	0	'nope'
0	1	2	3	4	5	6	7	8
-9	-8	-7	-6	-5	-4	-3	-2	-1

El uso de *slices* es como en ***strings***. Sabiendo esto, qué entrega:

- `x[ 0 : 3 ]` -->
- `x[ 5 : 9 ]` -->
- `x[ -7:-2 ]` -->

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Rebanadas (*slices*) de listas

True	False	10	20	-5.5	'yup'	0	0	'nope'
0	1	2	3	4	5	6	7	8
-9	-8	-7	-6	-5	-4	-3	-2	-1

El uso de *slices* es como en ***strings***. Sabiendo esto, qué entrega:

- `x[ 0 : 3 ]` --> **`[True, False, 10]`**
- `x[ 5 : 9 ]` --> **`['yup', 0, 0, 'nope']`**
- `x[ -7:-2 ]` --> **`[10, 20, -5.5, 'yup', 0]`**

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Rebanadas (*slices*) de listas

True	False	10	20	-5.5	'yup'	0	0	'nope'
0	1	2	3	4	5	6	7	8
-9	-8	-7	-6	-5	-4	-3	-2	-1

El uso de *slices* es como en ***strings***. Sabiendo esto, qué entrega:

- `x[ 4 :-4 ]` -->
- `x[ -3:   ]` -->
- `x[   : 3 ]` -->

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Rebanadas (*slices*) de listas

True	False	10	20	-5.5	'yup'	0	0	'nope'
0	1	2	3	4	5	6	7	8
-9	-8	-7	-6	-5	-4	-3	-2	-1

El uso de *slices* es como en ***strings***. Sabiendo esto, qué entrega:

- `x[ 4 :-4 ]` --> **`[-5.5]`**
- `x[ -3:   ]` --> **`[0, 0, 'nope']`**
- `x[   : 3 ]` --> **`[True, False, 10]`**

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Recorrer con y sin índices

- Lo siguiente está escrito para tuplas, pero es igual para índices

```
T = (5, 10, 15, 20, 25)

# podemos recorrer directamente
for x in T:
    print( 'ahora x vale', x )

# podemos recorrer usando indices
for i in range(len(T)):
    print( i, '--->', T[i] )
```

```
ahora x vale 5
ahora x vale 10
ahora x vale 15
ahora x vale 20
ahora x vale 25
0 ---> 5
1 ---> 10
2 ---> 15
3 ---> 20
4 ---> 25
```

Chequeando el contenido

`in`, `not in`, `count`, `index`,
`==`, `!=`, `<`, `<=`, `>`, `>=`

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Podemos detectar elementos (in)

- La instrucción **in** detecta pertenencia
- **in** entrega **True**/**False** (booleano)
- Similarmente, está la instrucción **not in**, que es la negación de **in**

```
>>> L = [10,0.5,'txt']
>>> 10 in L
True
>>> 'txt' in L
True
>>> 0 in L
False
>>> 10 not in L
False
>>> 'txt' not in L
False
>>> 0 not in L
True
```

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

No podemos detectar sublistas (in)

- Una sublista es una porción (rebanada, slice) de una lista
  - Ej. [10,5] es sublista de [1,10,5,8]

- La instrucción **in** *no detecta* sublistas
  - ...ni subtuplas

```
>>> L = [10,0.5,'txt']

>>> [10,0.5] in L
False
>>> [10,0.5] == L[:2]
True

>>> [10] in L
False
>>> [10] == L[:1]
True
```

Contar elementos (count)

x.count(u)

- Entrega la cantidad de veces que aparece el elemento *u* dentro de *x*
- Si el elemento no fue encontrado, entrega 0

```
>>> x = [5,-5,0,0.5,"python",0,0]
>>> x.count(0)
3
>>> x.count(5)
1
>>> x.count(-5)
1
>>> x.count("python")
1
>>> x.count(10)
0
>>> x.count("PYTHON")
0
```

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Encontrar elementos (index)

x.index(u)

- Encuentra el *primer* índice en el cual se encuentra el elemento ***u***, o **falla**
- **No hay find** en list, tuple

```
>>> x = [5,-5,0,0.5,"python",0,0]
>>> x.index(0)
2
>>> x.index(5)
0
>>> x.index(-5)
1
>>> x.index("python")
4

>>> x.index(10)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: list.index(x): x not in list
>>> x.index("PYTHON")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: list.index(x): x not in list
```

Comparación entre listas/tuplas

- Comparación se hace de izquierda a derecha.

```
A = (1, 2, 3)
B = (1, 2, 3, 0)
C = (1, 2, 3, -1)
D = (3, 0, 0, 0)

print( A == B )  # False,   B tiene un elemento extra por sobre A
print( A < B )   # True, A es como B, pero sin el último elemento
print( B < C )   # False, aunque B[:3]==C[:3], ocurre B[3] > C[3]
print( C <= D )  # True,   C[0] < D[0],  luego se resuelve el  <=
```

Mutación de listas

Lo siguiente no aplica a tuplas

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Asignar elementos (=)

L[i] = u

- Asigna el valor u a la posición (índice) *i* de *L*

```
>>> a = [1,2,3]

>>> a[0] = -1
>>> print(a)
[-1, 2, 3]

>>> a[0] = 1
>>> print(a)
[1, 2, 3]

>>> b = a
>>> b[-1] = 1000
>>> print(b)
[1, 2, 1000]
>>> print(a)
[1, 2, 1000]
```

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Anexar elementos (append)

L.append(u)

- Anexa el valor ***u*** al final de la lista *L*, como un elemento más

```
>>> a = [1,2,3]

>>> a.append(-1)
>>> print(a)
[1, 2, 3, -1]

>>> a.append([0,5])
>>> print(a)
[1, 2, 3, -1, [0, 5]]

>>> b = a
>>> b.append("!!!")
>>> print(b)
[1, 2, 3, -1, [0, 5], '!!!']
>>> print(a)
[1, 2, 3, -1, [0, 5], '!!!']
```

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Insertar elementos (insert)

L.insert(i, u)

- Inserta el valor *u* en el índice *i* de la lista *L*, desplazando los índices de los elementos posteriores en la lista en **+1** lugar

```
>>> L = [0,1,2,3,4]
>>> print(L)
[0, 1, 2, 3, 4]

>>> L.insert(0,"ari")
>>> print(L)
['ari', 0, 1, 2, 3, 4]

>>> L.insert(3,"gato")
>>> print(L)
['ari', 0, 1, 'gato', 2, 3, 4]

>>> L.insert(-1,"miau")
>>> print(L)
['ari',0, 1,'gato',2, 3,'miau',
4]
```

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Remover elementos (remove)

L.remove(u)

- Remueve la primera aparición del valor ***u*** en la lista ***L***
- Si ***u*** no está en ***L***, arroja una excepción alegando que el valor no está en la lista

```
>>> a = [1,1,2,2,3,3]

>>> a.remove(1)
>>> print(a)
[1, 2, 2, 3, 3]

>>> a.remove(3)
>>> print(a)
[1, 2, 2, 3]

>>> a.remove(100)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: list.remove(x): x not in list
```

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Remover elementos (pop)

L.pop(i)

- Remueve el elemento en el índice *i* y lo retorna
- Si *i* no se entrega, se asume *i* = -1 (remover el último elemento)
- Los índices de los elementos posteriores cambian en -1

```
>>> L = ['cat', 'dog', 'bird']
>>> print( L )
['cat', 'dog', 'bird']

>>> a = L.pop( 0 )
>>> print( a )
cat
>>> print( L )
['dog', 'bird']

>>> a = L.pop( -1 )
>>> print( a )
bird
>>> print( L )
['dog']
```

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Extender una lista con otra (extend)

L.extend(M)

- Toma los elementos de *M* y los anexa al final de *L*
- *M* puede ser lista, tupla, string, u otra secuencia de valores
  - Si *M* es lista o tupla: se anexan los elementos a *L*
  - Si *M* es string: se anexa cada caracter como elemento a *L*

```
>>> L = [1,2]
>>> print(L)
[1, 2]

>>> L.extend([3,4])
>>> print(L)
[1, 2, 3, 4]

>>> L.extend((5,6))
>>> print(L)
[1, 2, 3, 4, 5, 6]

>>> L.extend("78")
>>> print(L)
[1, 2, 3, 4, 5, 6, '7', '8' ]
```

Ejemplo

Listas de Listas

Clase 31

Lista de listas

```
L = [   [ 1, 5,  2  ],
        [ 0, 0, 7,  1 ],
        [ 3, 2,  1  ]  ]
```

- El código anterior define una lista de listas

¿Cuánto vale?

- `len(L)    ==`
- `len(L[0]) ==`
- `len(L[1]) ==`
- `L[0]      ==`
- `L[1][-1]  ==`

Lista de listas

```
L = [   [ 1, 5,  2  ],
        [ 0, 0, 7,  1 ],
        [ 3, 2,  1  ]  ]
```

- El código anterior define una lista de listas

¿Cuánto vale?

- `len(L)    == 3`
- `len(L[0]) ==`
- `len(L[1]) ==`
- `L[0]      ==`
- `L[1][-1]  ==`

Lista de listas

```
L = [   [ 1, 5,  2  ],
        [ 0, 0, 7,  1 ],
        [ 3, 2,  1  ]  ]
```

- El código anterior define una lista de listas

¿Cuánto vale?

- `len(L)    == 3`
- `len(L[0]) == 3`
- `len(L[1]) ==`
- `L[0]      ==`
- `L[1][-1]  ==`

Lista de listas

```
L = [   [ 1, 5,  2  ],
        [ 0, 0, 7,  1 ],
        [ 3, 2,  1  ]  ]
```

- El código anterior define una lista de listas

¿Cuánto vale?

- `len(L)    == 3`
- `len(L[0]) == 3`
- `len(L[1]) == 4`
- `L[0]      ==`
- `L[1][-1]  ==`

Lista de listas

```
L = [   [ 1, 5,  2 ],
        [ 0, 0, 7,  1 ],
        [ 3, 2,  1 ]  ]
```

- El código anterior define una lista de listas

¿Cuánto vale?

- `len(L)    == 3`
- `len(L[0]) == 3`
- `len(L[1]) == 4`
- `L[0]      == [1, 5, 2]`
- `L[1][-1]  ==`

Lista de listas

```
L = [   [ 1, 5,  2 ],
        [ 0, 0, 7,  1 ],
        [ 3, 2,  1 ]  ]
```

- El código anterior define una lista de listas

¿Cuánto vale?

- `len(L)    == 3`
- `len(L[0]) == 3`
- `len(L[1]) == 4`
- `L[0]      == [1, 5, 2]`
- `L[1][-1]  == 1`

Cambiando 7 por 9

```
L = [   [ 1, 5,  2  ],
        [ 0, 0, 7,  1 ],
        [ 3, 2,  1  ]  ]

L
```

- Consideremos la lista apuntada por la variable *L* (a la izquierda)
- ¿Cómo cambiamos ese 7 por un 9?

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Cambiando 7 por 9

```
L = [   [ 1, 5,  2  ],
        [ 0, 0, 7,  1 ],
        [ 3, 2,  1  ] ]

L
```

```
L = [   [ 1, 5,  2  ],
        [ 0, 0, 7,  1 ],
        [ 3, 2,  1  ] ]

L[1]
```

Cambiando 7 por 9

```
L = [   [ 1, 5,  2  ],
        [ 0, 0, 7,  1 ],
        [ 3, 2,  1  ]  ]


L
```

```
L = [   [ 1, 5,  2  ],
        [ 0, 0, 9,  1 ],
        [ 3, 2,  1  ]  ]


L
```

```
L = [   [ 1, 5,  2  ],
        [ 0, 0, 7,  1 ],
        [ 3, 2,  1  ]  ]


L[1][2]
```

```
L = [   [ 1, 5,  2  ],
        [ 0, 0, 9,  1 ],
        [ 3, 2,  1  ]  ]


L[1][2]
```

```
L = [   [ 1, 5,  2  ],
        [ 0, 0, 7,  1 ],
        [ 3, 2,  1  ]  ]


L[1]
```

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Cambiando 7 por 9

```
L = [   [ 1, 5,  2  ],
        [ 0, 0, 7,  1 ],
        [ 3, 2,  1  ]  ]


L
```

```
L = [   [ 1, 5,  2  ],
        [ 0, 0, 7,  1 ],
        [ 3, 2,  1  ]  ]


L[1]
```

```
L = [   [ 1, 5,  2  ],
        [ 0, 0, 7,  1 ],
        [ 3, 2,  1  ]  ]


L[1][2]
```

```
L = [   [ 1, 5,  2  ],
        [ 0, 0, 9,  1 ],
        [ 3, 2,  1  ]  ]


L[1][2] = 9
```