L#10. Python Sequences: Lists, Tuples

Ago 2017

https://docs.python.org/2.5/lib/typesseq.html

Lists

	L[0]	L[1]	L[2]	L[3]	L[4]
L =	[3	3.0	‘Hola’	3+5j	True]

- Elements in a list can be of any type, e.g.: int, float, complex, Boolean, strings

```
L=[3, 3.0, 'Hola', 3+5j, True]


print(L)
print(type(L))

print(L[0])
print(L[2])
print(L[2][3])
```

```
# [int, float, string, complex, bool]


[3, 3.0, 'Hola', (3+5j), True]
<class 'list’>

3
Hola
a
# L[2] is the string ‘HOLA’; L[2][3] is the 4th element (i.e., index=3) of that string
```

	L[2][0]	L[2][1]	L[2][2]	L[2][3]
L [2]=	[‘H’	**‘o’**	**‘l’**	**‘a’]**

Reference: Python Lists

Elements of a list may be another sequence, e.g., string

Lists

index	0	1	2	3	4	5	6	7	8	9
value	1	2	3	4	5	6	7	8	9	10

Example a Lists of numbers constructed with square brackets and separating items with commas: [1, 2, 3,….].

```
L=[1,2,3,4,5,6,7,8,9,10]        # list of numbers

print(L)                        [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]


print(L[1:4])                   [2, 3, 4]

print(L[:6])                    [1, 2, 3, 4, 5, 6]

print(L[0:9:2])                 [1, 3, 5, 7, 9]
print(L[::2])                   [1, 3, 5, 7, 9]
print(L[:9:2])                  [1, 3, 5, 7, 9]
print(L[0::2])                  [1, 3, 5, 7, 9]

print(L[0])                     1
```

Slicing works the same as in strings

Reference: Python Lists

Lists generators: range and list functions

- Combine range() and list() to create list of integers.
- Use numpy.arange() for lists of float

```
start=1
stop=11
step=1
L=list(range(start,stop,step)))
print(L)

# range is iterable, which produces the
# numbers 1, 2, 3,..,10, one at a time but
# to save memory it does not store them.
# List() stores them
```

OUTPUT

Stop is not included, list stops at a previous element, i.e., 10

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Reference: Python Lists

Creation of Lists by Iteration

P1. Create a new list as the square of natural numbers

$$[\ ] + [1^2, 2^2, 3^2, 4^2, 5^2] \Longrightarrow [\ 1, 4, 9, 16, 25\ ]$$

```
x = [ ]                    # empty list
for i in [1, 2, 3, 4, 5]:
   x.append(i**2)          # append elements

print('x=',x)          # x=[1, 4, 9, 16, 25]
```

P2. Add more elements to previous x-list

$$[\ 1, 4, 9, 16, 25\ ] + [\ 6^2, 7^2, 8^2, 9^2] \Longrightarrow [\ 1, 4, 9, 16, 25, 36, 49, 64, 81]$$

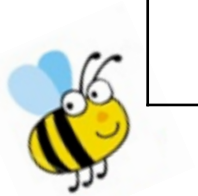
```
for i in [6, 7, 8, 9]:
   x.append(i**2)    # append additional elements

print('x=',x)        # x= [1, 4, 9, 16, 25, 36, 49, 64, 81]
```

FILE: list10.py

Python Membership Operators [PMO]

- Membership operators test for membership in a sequence, such as strings, lists, or tuples. There are two membership operators:

Operator	Description	Examples
in	Evaluates to true if it finds a variable in the specified sequence and false otherwise.	(1) a=[1,2,3,4,5] (2) b= 3 in a (3) print(b) # True **1** (4) c= 7 in a (5) print(c) # False **2** (6) print(7 not in a) # True **3**
not in	Evaluates to true if it does not finds a variable in the specified sequence and false otherwise.	



PMO Example: Curvebraker

- Searches for a particular grade in a grade list.

Solution-1

grades=['A','B','A','C','B','D','F','W',’A’] # grade list
if 'A' in grades:
print("There is a curvebreaker in {}".format(grades))
else:
print("There is not an 'A' in {}".format(grades))

There is a curvebreaker in ['A', 'B', 'A', 'C', 'B', 'D', 'F', 'W', 'A’] # note how grades list is inserted in the placeholder {}

PMO Example: List Overlap

- Take two lists and write a program that returns a new list containing only elements that are common between the lists (without duplicates). Make sure your program works on two lists of different sizes.

Solution-1

```
a=[1,2,3,4,5,6,7,7]
b=[5,6,7,8,9,10,11]
c=[]
for item in a:
    if item in b:
        if item not in c:
            c.append(item)
print (c)          # [5,6,7]
```

Solution-2

```
a=[1,2,3,4,5,6,7,7]
b=[5,6,7,8,9,10,11]
c=[]
for item1 in a:
    for item2 in b:
        if item1==item2:
            c.append(item1)

new=list(set(c))
print(new)
# repeated elements, if any, are removed by set() function
```

Soln 2 doesn't use PMO

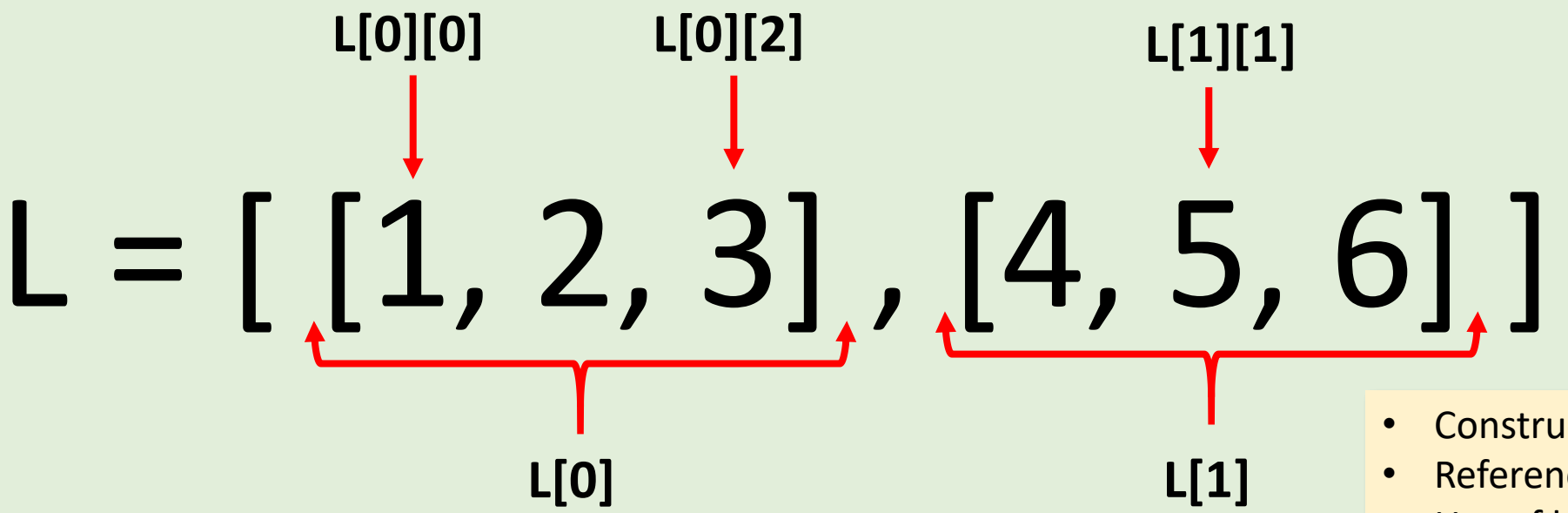
PLAY with the code: (Soln 1) exchange a and b lists and erase "**if item not in c"** statement and rerun the code

2D Lists: nested lists

Construct a 2D list as:

1	2	3
4	5	6

Combine two 1D lists of same size (row-by-row):



- Construct the list [in->out]
- Reference [out->in]
- Use of brackets

2D Lists: nested lists

Construct a 2D list as:

1	2	3
4	5	6

Reference elements on the List.

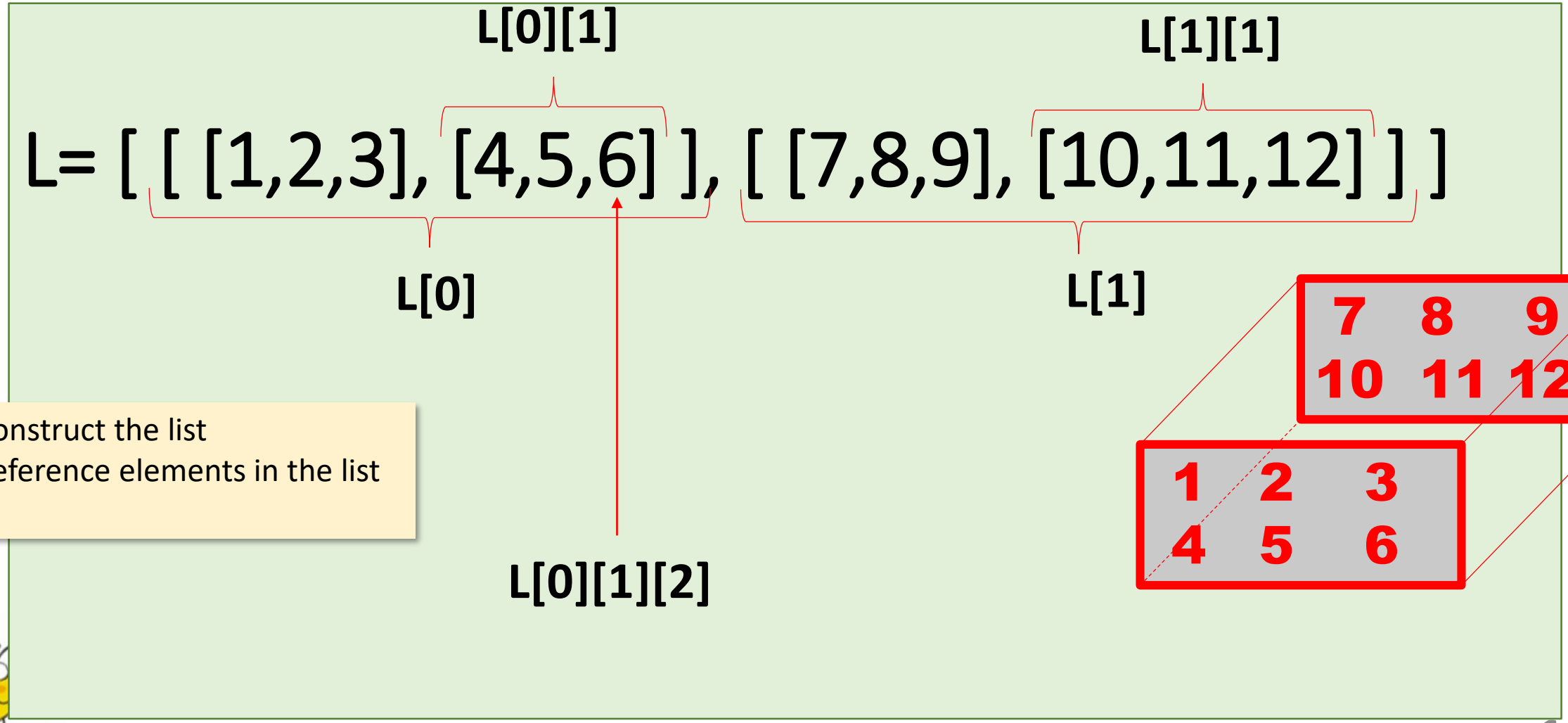
```
(1) L=[[1,2,3],[4,5,6]]
(2) print(L)
(3) print(L[0][0])

# Math Operations
(4) print(L[1][2]+L[0][0])
(5) print(L[0][2]*L[1][2])
(6) print(L[0][2]**L[0][1])
```

```
[[1, 2, 3], [4, 5, 6]]
1

7
18
9
```

- Construct the list
- Reference elements in the list

Math operations with lists are not very efficient, You should use numpy module and arrays, to be seen soon. Reference: Python Lists

3D Lists (nested lists): combine 2D lists



- Construct the list
- Reference elements in the list

3D Lists: reference and print

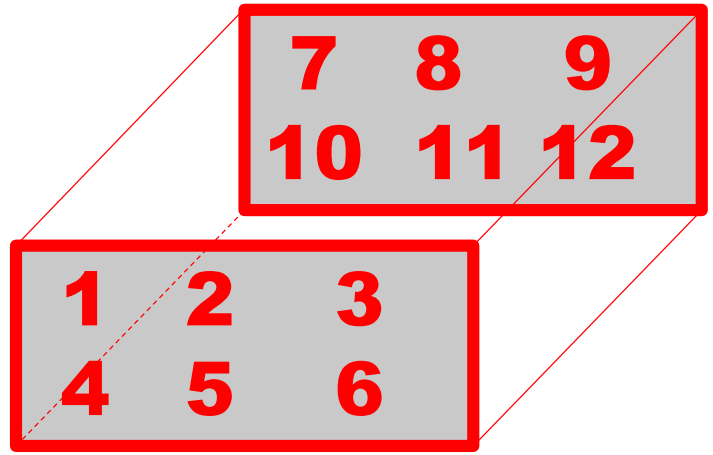
2D

```
L= [ [ [1,2,3], [4,5,6] ], [ [7,8,9], [10,11,12] ] ]
print(L)
```

2D

```
print(L[0])
print(L[1])

print(L[0][0])
print(L[0][1])

print(L[1][0])
print(L[1][1])

# Reference element with value 1
print(L[0][0][0])
# Reference element with value 12
print(L[1][1][2])
```

```
[[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]]

[[1, 2, 3], [4, 5, 6]]
[[7, 8, 9], [10, 11, 12]]

[1, 2, 3]
[4, 5, 6]
[7, 8, 9]
[10, 11, 12]

1
12
```



- Construct the list
- Reference elements in the list

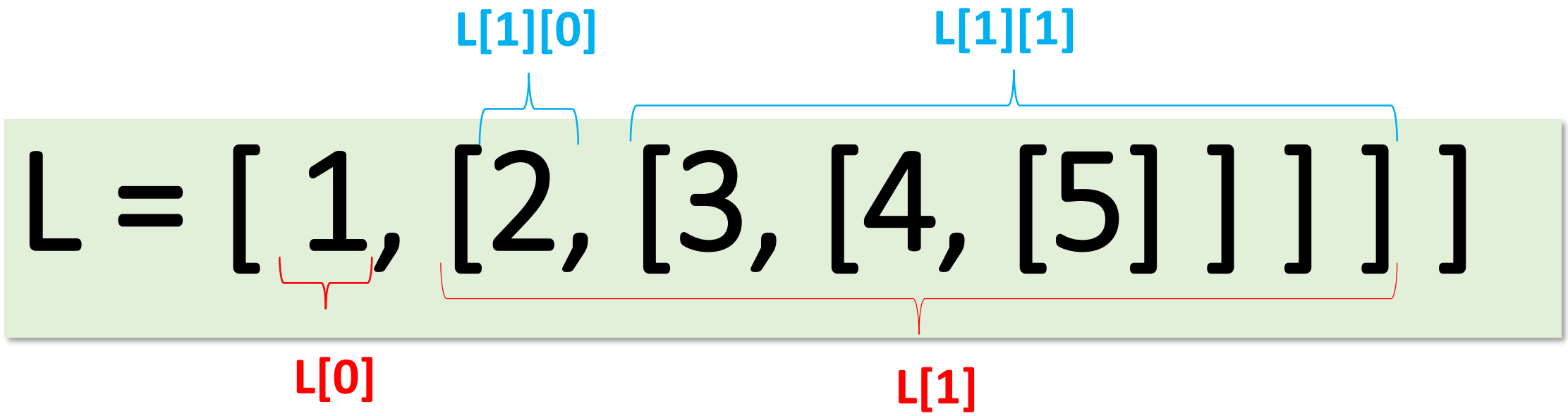
N-dimensional Lists: nested lists [optional]

QUIZ

```
L = [ 1, [2, [3, [4, [5] ] ] ] ]
```

How do you reference the element valued 3 in this list?

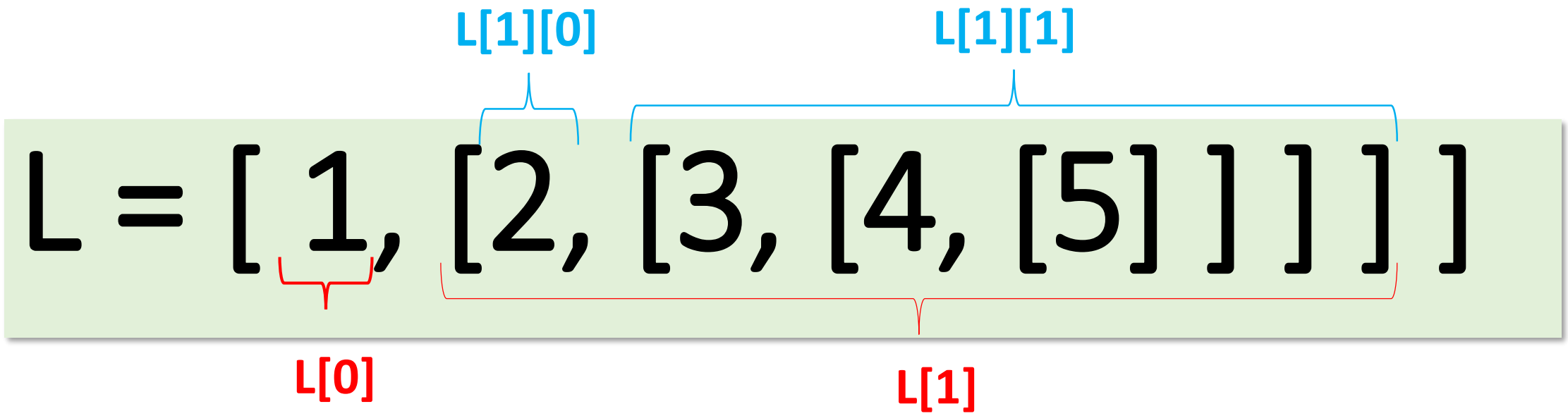
N-dimensional Lists: nested lists



HINT: **Note how commas separate elements**

N-dimensional Lists: nested lists

SOLUTION



How do you reference the element valued 3 in the L-list?

R= L[1][1][0]

Therefore, if you print print(L[1][1][0]) you get 3

N-dimensional Lists: nested lists

```
L = [ 1, [2, [3, [4, [5] ] ] ] ]
(1) print(L)
(2) print(L[0])
(3) print(L[1])
(4) print(L[1][0])
(5) print(L[1][1])
(6) print(L[1][1][0])
(7) print(L[1][1][1])
(8) print(L[1][1][1][0])
(9) print(L[1][1][1][1])
(10) print(L[1][1][1][1][0])
```

QUIZ

What is the output?

(1)
(2)
(3)
(4)
(5)
(6)
(7)
(8)
(9)
(10)

Reference elements in the list

N-dimensional Lists: nested lists [optional]

```
L = [ 1, [2, [3, [4, [5] ] ] ] ]
(1) print(L)
(2) print(L[0])
(3) print(L[1])
(4) print(L[1][0])
(5) print(L[1][1])
(6) print(L[1][1][0])
(7) print(L[1][1][1])
(8) print(L[1][1][1][0])
(9) print(L[1][1][1][1])
(10) print(L[1][1][1][1][0])
```

SOLUTION

(1) [1, [2, [3, [4, [5]]]]]
(2) 1
(3) [2, [3, [4, [5]]]]
(4) 2
(5) [3, [4, [5]]]
(6) 3
(7) [4, [5]]
(8) 4
(9) [5]
(10) 5

- Reference elements in the list
- Print the elements

✓ Syntax method to create Lists

✓ Recommended for:

- Performance (runs faster)
- Vectorization (processes all elements at once)
- Shorter codes (codes are simplified)

List Comprehension

Slide 18-25

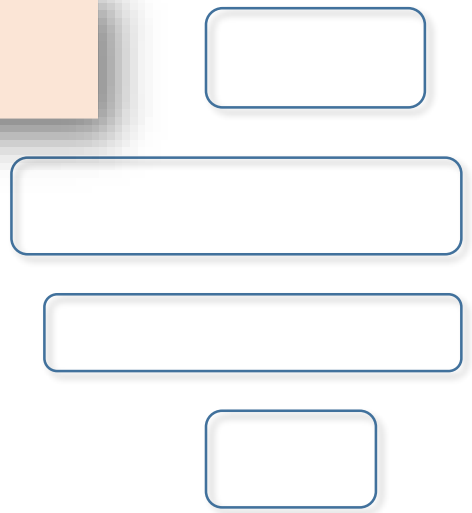
Comprensión de Listas (o agregación de listas)

List Comprehension Construction

impares

```
# Given list of integers in a sequence construct a new list with the square of odd values:
L = [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ]

SO=[]

for x in L:
    if x%2 != 0:
        SO.append( x**2 )
```

Identify the syntax elements to construct a List Comprehension



Esta seria nuestra solución clásica, sin usar Comprensión de Lista y la vamos a utilizar para construir la Compresión de Lista

List Comprehension Construction

```
# Given list of integers in a sequence construct a new list with the square of odd values:
L = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

SO = []

for x in L:
    if x%2 != 0:
        SO.append( x**2 )
```

ONLY the enclosed elements will be used to construct the List Comprehension

List Comprehension Construction

```
# Given list of integers in a sequence construct a new list with the square of its odd values:
L = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ]
```

```
SO = [ ]
for x in L:
    if x%2 != 0 :
        SO.append( x**2 )
```

Cuatro líneas de código se convierten en una línea

```
SO = [ x**2 for x in L If x%2 != 0 ]
```

List Comprehension Syntax Components

```
#  Given list of integers in a sequence construct a new list with only the square of odd
# values:
L = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

SO = [ x**2  for x in L  if x%2 != 0 ]
```

Output expression

Input sequence

Optional Conditional Statement

for loop is mandatory to generate the output list

A variable representing members of the input sequence

```
print(SO)     # Output:  [ 1, 9, 25, 49, 81 ]
```

List Comprehension(1)

A simple problem: Construct a new list by squaring the elements of a list

num	1	2	3	4	5
sqr	1^2	2^2	3^2	4^2	5^2

Create list using loops	Using List Comprehension
num = [1, 2, 3, 4, 5] sqr = [] for n in num: sqr.append(n**2) print(sqr)	num = [1, 2, 3, 4, 5] sqr = [n**2 for n in num] # no ":" in for print(sqr)
OUTPUT	OUTPUT
[1, 4, 9, 16, 25]	[1, 4, 9, 16, 25]

List Comprehension(2)

Create list using loops	**Using List Comprehension**

# Multiples of 2 below 20 	
list1 = [ ]
 	
for x in range(20):
  if x%2 ==0
  list1.append(x)
 	
print(list1)
	list1 = [x for x in range(20) if x % 2 == 0] # no ":" in if
	print(list1)
	
	OUTPUT
	[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]

# Multiples of 3 and 5 below 20	

list2=[]
for x in range(20):
  if x%3==0:
  if x%5==0:
  list2.append(x)
 	
print(list2) #output:
	list2 = [x for x in range(20) if x % 3 == 0 if x % 5 ==0]
	print(list2)
	
	OUTPUT
	[0, 15]

List Comprehension(3)

list1 = [1, 2, 3, 4]
list2 = [2, 3, 4, 5]

Nested Loops	**Using List Comprehension**

# Find common elements: 	
list1 = [1, 2, 3, 4]
list2 = [2, 3, 4, 5]
common=[]
for a in list1:
  for b in list2:
  if a==b:
  common.append(a)
print(common)
	# Find common elements:
	list1 = [1, 2, 3, 4]
	list2 = [2, 3, 4, 5]
	
	common = [a for a in list1 for b in list2 if a == b]
	
	print(common)

OUTPUT	**OUTPUT**
[2, 3, 4]	[2, 3, 4]

List Comprehension(4)

Create a 2D List (a matrix or table) in which the first column will be the square of 1, 2, 3 and the second column is the cubic of the same numbers.

$$\begin{bmatrix} 1^2 & 1^3 \\ 2^2 & 2^3 \\ 3^2 & 3^3 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 4 & 8 \\ 9 & 27 \end{bmatrix}$$

Loops and 2D	**Using List Comprehension**

# Create 2D list 	
list3 = [1, 2, 3]
list4 = [ ]
 	
for a in list3:
  list4.append([a**2, a**3])
	list3= [1, 2, 3]
	
	list4 = [[a**2, a**3] for a in list3]
	
	print(list4)

OUTPUT	**OUTPUT**
[[1, 1], [4, 8], [9, 27]]	[[1, 1], [4, 8], [9, 27]]

List Comprehension(5): The hardest example

Write code to report from a list of numbers which element is even or odd

1	2	3	4	5	6	7	8	9	10
‘Even’	‘Odd’	‘Even’	‘Odd’	‘Even’	‘Odd’	‘Even’	‘Odd’	‘Even’	‘Odd’

Create list using loops

```
list3=[]
for i in range(11):
    if i%2==0:
        list3.append("Even")
    else:
        list3.append("Odd")

print(list3)
```

Using List Comprehension

```
list3 = ["Even" if i%2==0 else "Odd" for i in range(11)]
print(list3)
```

OUTPUT

```
['Even', 'Odd', 'Even', 'Odd', 'Even', 'Odd', 'Even', 'Odd', 'Even', 'Odd’, 'Even']
```

Some more examples:

- online.upr.edu/Algorithms and computer Programming/python list comprehension
- https://medium.com/better-programming/list-comprehension-in-python-8895a785550b
- https://docs.python.org/3/tutorial/datastructures.html

Functions (or Methods) that accept Lists as arguments

- **append**(*x*)
- **extend**(*iterable*)
- **insert**(*i*, *x*)
- **remove**(*x*)
- **pop**([*i*])
- **clear**()
- **index**(*x*[, *start*[, *end*]])
- **count**(*x*)
- **sort**(*key=None*, *reverse=False*)
- **reverse**()
- **copy**()

CODE FILE: listFunctions.py

There are about 28 methods

Go to the reference to see them all with examples

https://www.programiz.com/python-programming/methods/list

https://www.w3schools.com/python/python_ref_list.asp

Tuples (Español: Tupla)

	Plural	Singular
English	Tuples	Tuple
Español	Tupla	Tuple

1. Tuples are like lists. Accept different type elements.
2. Tuples cannot be modified once created (immutable).
3. Elements: (a,b,c) or a,b,c
4. Empty tuple must have ().
5. Single-item tuples must have a trailing comma: (d,).
6. Sirve para agrupar, como si fueran un único valor, varios valores que, por su naturaleza, deben ir juntos.

In mathematics, a **tuple** is a finite ordered list (sequence) of elements.
https://data-flair.training/blogs/python-tuple/

Tuples, Basic Operations

```
T=(1,2,3,4,5,6,7,8,9,10)                # Initialize
TT=1, 2.0, 'three', False,[1,2,3]       # different types

print(T)                                # Output
print(type(T))                          # Data type

# Referencing
print(T[0])                             # first element
print(T[-1])                            # last element

# Slicing (same as strings and lists)
print(T[ 1: 4 ] )
print( T[ : : -1] )

# a TUPLE canNOT be modified
T[0]=2
```

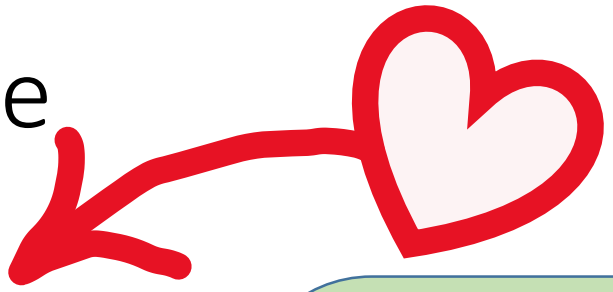
```
(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
<class 'tuple'>

1
10

(2, 3, 4)
(10, 9, 8, 7, 6, 5, 4, 3, 2, 1)

TypeError: 'tuple' object does
not support item assignment
```

Tuples, Basic Operations

```
# Packing & unpacking
scores=(100, 95,89, 96)  # packing
ex1,ex2,ex3,ex4=scores  # unpacking
print(ex4)          # output one element
print(type(ex1)

# Inmutable?
pesos= (1, 2, 3, [4, 5] )
pesos[0]=2        # Error

pesos[3][0]=5
print(pesos)

# Functions
print(len(pesos))
print(max(scores))
print(min(scores))
print(sum(scores))
```

```
96
<class 'int'>


(1, 2, 3, [5, 5])

4
100
89
380
```

Iteration over a sequence



sequence= AQUI VOY

```
for item in sequence:
    operation=item*2
    print(operation)
```

AQUI VOY:

[“sal”,”pepper”,”sazon”]

[1,2,3,4,5]

“Juan del Pueblo”

(89,93, 87, 97)

(“Jose”,89.99,”A”)

Exercise-1

Write a Python program to initialize and print a 2D tuple. Each tuple element contains rows with name, score, and letter grade of one student.

name	final score	letter grade
Tito	93	A
Pepe	99	A
Papo	81	B

Hint: one element should be:

(“Tito”, 93, “A”)

```
# Initialize List of tuples
grades=(("Tito",93,"A"),
        ("Pepe",99,"A"),
        ("Papo",81,"B"))
# Output
print(grades)

# Otherwise (better output):
for t in grades:
    print(t)
```

Exercise-1

OUTPUT


```
# Initialize List of
tuples
grades=(("Tito",93,"A"),
        ("Pepe",99,"A"),
        ("Papo",81,"B"))
# Output
print(grades)


# Otherwise (better
output):
for t in grades:
    print(t)
```

(('Tito', 93, 'A'), ('Pepe', 99, 'A'), ('Papo', 81, 'B'))

('Tito', 93, 'A')

('Pepe', 99, 'A')

('Papo', 81, 'B')

Exercises

2. Write a Python program to reverse the tuple T=(1,2,3,4,5).

```
T=(1,2,3,4,5)

# Alternative-1
print( T [ : : -1 ] )

# Alternative-2
# Use the reversed function:
print(tuple(reversed(T)))
# reversed is an iterator that doesn't
# store the values
```


ADIOS!