LISTS IN PYTHON

CREATE A LIST

- A LIST IS A COLLECTION WHICH IS ORDERED AND CHANGEABLE. IN PYTHON LISTS ARE WRITTEN WITH SQUARE BRACKETS.
- EXAMPLE
- THISLIST = ["APPLE", "BANANA", "CHERRY"]
  PRINT(THISLIST)

INTRODUCTION TO LISTS (CONT'D.)

Figure 7-1 A list of integers

Figure 7-2 A list of strings

Figure 7-3 A list holding different types

THE REPETITION OPERATOR AND ITERATING OVER A LIST

- REPETITION OPERATOR: MAKES MULTIPLE COPIES OF A LIST AND JOINS THEM TOGETHER
  - THE ***** SYMBOL IS A REPETITION OPERATOR WHEN APPLIED TO A SEQUENCE AND AN INTEGER
    - SEQUENCE IS LEFT OPERAND, NUMBER IS RIGHT
  - GENERAL FORMAT: `LIST * N`
- YOU CAN ITERATE OVER A LIST USING A `FOR` LOOP
  - FORMAT: `FOR X IN LIST:`

ACCESS ITEMS

- ACCESS ITEMS
- YOU ACCESS THE LIST ITEMS BY REFERRING TO THE INDEX NUMBER:
- EXAMPLE
- PRINT THE SECOND ITEM OF THE LIST:
- THISLIST = ["APPLE", "BANANA", "CHERRY"]
  PRINT(THISLIST[1])

NEGATIVE INDEXING

- NEGATIVE INDEXING MEANS BEGINNING FROM THE END, -1 REFERS TO THE LAST ITEM, -2 REFERS TO THE SECOND LAST ITEM ETC.

- EXAMPLE
- PRINT THE LAST ITEM OF THE LIST:

- THISLIST = ["APPLE", "BANANA", "CHERRY"]
- PRINT(THISLIST[-1])

RANGE OF INDEXES

- YOU CAN SPECIFY A RANGE OF INDEXES BY SPECIFYING WHERE TO START AND WHERE TO END THE RANGE.WHEN SPECIFYING A RANGE, THE RETURN VALUE WILL BE A NEW LIST WITH THE SPECIFIED ITEMS.
- EXAMPLE
- RETURN THE THIRD, FOURTH, AND FIFTH ITEM:
- THISLIST =["APPLE", "BANANA", "CHERRY", "ORANGE", "KIWI", "MELON", "MANGO"] PRINT(THISLIST[2:5])
- PRINT(THISLIST[:4])
- PRINT(THISLIST[2:])
- PRINT(THISLIST[-4:-1])

CHANGE ITEM VALUE

- TO CHANGE THE VALUE OF A SPECIFIC ITEM, REFER TO THE INDEX NUMBER:EXAMPLE
- CHANGE THE SECOND ITEM:
- THISLIST = ["APPLE", "BANANA", "CHERRY"]
  THISLIST[1] = "BLACKCURRANT"
  PRINT(THISLIST)

LOOP THROUGH A LIST

- YOU CAN LOOP THROUGH THE LIST ITEMS BY USING A FOR LOOP:

- EXAMPLE
- PRINT ALL ITEMS IN THE LIST, ONE BY ONE:

- THISLIST = ["APPLE", "BANANA", "CHERRY"]
- FOR X IN THISLIST:
- PRINT(X)

CHECK IF ITEM EXISTS

- TO DETERMINE IF A SPECIFIED ITEM IS PRESENT IN A LIST USE THE IN KEYWORD:

- EXAMPLE
- CHECK IF "APPLE" IS PRESENT IN THE LIST:

```
THISLIST = ["APPLE", "BANANA", "CHERRY"]
 IF "APPLE" IN THISLIST:
      PRINT("YES, 'APPLE' IS IN THE FRUITS LIST")
```

LIST LENGTH

- TO DETERMINE HOW MANY ITEMS A LIST HAS, USE THE LEN() FUNCTION:

- EXAMPLE
- PRINT THE NUMBER OF ITEMS IN THE LIST:

- THISLIST = ["APPLE", "BANANA", "CHERRY"]
- PRINT(LEN(THISLIST))

ADD ITEMS

- TO ADD AN ITEM TO THE END OF THE LIST, USE THE APPEND() METHOD:
- EXAMPLE
- USING THE APPEND() METHOD TO APPEND AN ITEM:

```
THISLIST = ["APPLE", "BANANA", "CHERRY"]

THISLIST.APPEND("ORANGE")

PRINT(THISLIST)
```

- INSERT AN ITEM AS THE SECOND POSITION:

```
THISLIST = ["APPLE", "BANANA", "CHERRY"]
THISLIST.INSERT(1, "ORANGE")
PRINT(THISLIST)
```

REMOVE ITEM

HERE ARE SEVERAL METHODS TO REMOVE ITEMS FROM A LIST:

- THE REMOVE() METHOD REMOVES THE SPECIFIED ITEM:

```
THISLIST = ["APPLE", "BANANA", "CHERRY"]

THISLIST.REMOVE("BANANA")

PRINT(THISLIST)
```

- THE POP() METHOD REMOVES THE SPECIFIED INDEX, (OR THE LAST ITEM IF INDEX IS NOT SPECIFIED):

```
THISLIST = ["APPLE", "BANANA", "CHERRY"]

THISLIST.POP()

PRINT(THISLIST)
```

- THE DEL KEYWORD REMOVES THE SPECIFIED INDEX:

```
THISLIST = ["APPLE", "BANANA", "CHERRY"]
DEL THISLIST[0]
PRINT(THISLIST)
```

- THE DEL KEYWORD CAN ALSO DELETE THE LIST COMPLETELY:

```
THISLIST = ["APPLE", "BANANA", "CHERRY"]
DEL THISLIST
```

- THE CLEAR() METHOD EMPTIES THE LIST:

```
THISLIST = ["APPLE", "BANANA", "CHERRY"]
THISLIST.CLEAR()
PRINT(THISLIST)
```

COPY A LIST

- YOU CANNOT COPY A LIST SIMPLY BY TYPING LIST2 = LIST1, BECAUSE: LIST2 WILL ONLY BE A REFERENCE TO LIST1, AND CHANGES MADE IN LIST1 WILL AUTOMATICALLY ALSO BE MADE IN LIST2.
- THERE ARE WAYS TO MAKE A COPY, ONE WAY IS TO USE THE BUILT-IN LIST METHOD COPY().

MAKE A COPY OF A LIST WITH THE COPY() METHOD:

```
THISLIST = ["APPLE", "BANANA", "CHERRY"]
MYLIST = THISLIST.COPY()
PRINT(MYLIST)
```

- ANOTHER WAY TO MAKE A COPY IS TO USE THE BUILT-IN METHOD LIST().

MAKE A COPY OF A LIST WITH THE LIST() METHOD:

```
THISLIST = ["APPLE", "BANANA", "CHERRY"]
MYLIST = LIST(THISLIST)
PRINT(MYLIST
```

JOIN TWO LISTS

- THERE ARE SEVERAL WAYS TO JOIN, OR CONCATENATE, TWO OR MORE LISTS IN PYTHON.

```
LIST1 = ["A", "B" , "C"]

LIST2 = [1, 2, 3]

LIST3 = LIST1 + LIST2

PRINT(LIST3)
```

- APPEND LIST2 INTO LIST1:

```
LIST1 = ["A", "B" , "C"]
LIST2 = [1, 2, 3]

FOR X IN LIST2:
  LIST1.APPEND(X)

PRINT(LIST1)
```

- USE THE EXTEND() METHOD TO ADD LIST2 AT THE END OF LIST1:

```
LIST1 = ["A", "B" , "C"]

LIST2 = [1, 2, 3]

LIST1.EXTEND(LIST2)

PRINT(LIST1)
```

THE LIST() CONSTRUCTOR

- IT IS ALSO POSSIBLE TO USE THE LIST() CONSTRUCTOR TO MAKE A NEW LIST.

- EXAMPLE
- USING THE LIST() CONSTRUCTOR TO MAKE A LIST:

- THISLIST = LIST(("APPLE", "BANANA", "CHERRY")) # NOTE THE DOUBLE ROUND-BRACKETS
- PRINT(THISLIST)

LIST METHODS

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

LIST METHODS

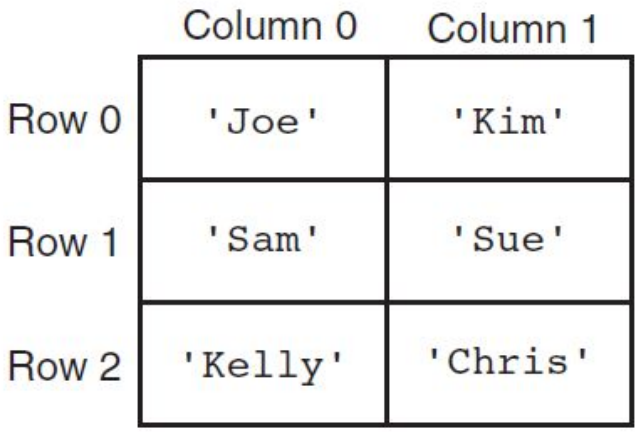
```
>>> x = list()
>>> type(x)<type 'list'>
>>> dir(x)['append', 'count', 'extend', 'index', 'insert', 'pop',
'remove', 'reverse', 'sort']
>>>
```

TWO-DIMENSIONAL LISTS

- TWO-DIMENSIONAL LIST: A LIST THAT CONTAINS OTHER LISTS AS ITS ELEMENTS
  - ALSO KNOWN AS NESTED LIST
  - COMMON TO THINK OF TWO-DIMENSIONAL LISTS AS HAVING ROWS AND COLUMNS
  - USEFUL FOR WORKING WITH MULTIPLE SETS OF DATA
- TO PROCESS DATA IN A TWO-DIMENSIONAL LIST NEED TO USE TWO INDEXES
- TYPICALLY USE NESTED LOOPS TO PROCESS

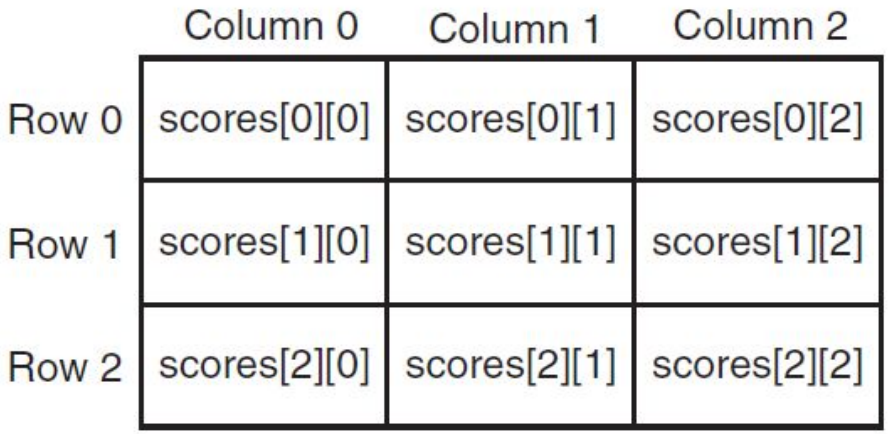
TWO-DIMENSIONAL LISTS (CONT'D.)

Figure 7-5 A two-dimensional list



TWO-DIMENSIONAL LISTS (CONT'D.)

Figure 7-7 Subscripts for each element of the `scores` list



- **ACCESSING A MULTIDIMENSIONAL LIST:**

```
a = [[2, 4, 6, 8, 10], [3, 6, 9, 12, 15], [4, 8, 12, 16, 20]]
print(a)
```

```
a = [[2, 4, 6, 8, 10], [3, 6, 9, 12, 15], [4, 8, 12, 16, 20]]
for record in a:
    print(record)
```

```
a = [ [2, 4, 6, 8 ],      [ 1, 3, 5, 7 ],       [ 8, 6, 4, 2 ],       [ 7, 5, 3, 1 ] ]
for i in range(len(a)) :
    for j in range(len(a[i])) :
        print(a[i][j], end=" ")
    print()
```

- **CREATING A MULTIDIMENSIONAL LIST WITH ALL ZEROS:**

```
m = 4
n = 5
 a = [[0 for x in range(n)] for x in range(m)]
print(a)
```

APPEND(): ADDS AN ELEMENT AT THE END OF THE LIST

```
a = [[2, 4, 6, 8, 10], [3, 6, 9, 12, 15], [4, 8, 12, 16, 20]]
a.append([5, 10, 15, 20, 25])
print(a)
```

EXTEND(): ADD THE ELEMENTS OF A LIST (OR ANY ITERABLE), TO THE END OF THE CURRENT LIST.

```
a = [[2, 4, 6, 8, 10], [3, 6, 9, 12, 15], [4, 8, 12, 16, 20]]
a[0].extend([12, 14, 16, 18])
print(a)
```

REVERSE(): REVERSES THE ORDER OF THE LIST.

```
a = [[2, 4, 6, 8, 10], [3, 6, 9, 12, 15], [4, 8, 12, 16, 20]]
a[2].reverse()
print(a)
```

- # FIRST METHOD TO CREATE A 1 D LIST

```
n = 5
arr = [0]*n
print(arr)
```

SECOND METHOD TO CREATE A 1 D LIST

```
n = 5
arr = [0 for i in range(n)]
print(arr)
```

Using above first method to create a

2D LIST

```
rows, cols = (5, 5)
arr = [[0]*cols]*rows
print(arr)
```

- # USING ABOVE SECOND METHOD TO CREATE A 2D ARRAY

```
rows, cols = (5, 5)

arr = [[0 for i in range(cols)] for j in range(rows)]

print(arr)
```

TUPLES

- A TUPLE IS A COLLECTION WHICH IS ORDERED AND **UNCHANGEABLE.** IN PYTHON TUPLES ARE WRITTEN WITH ROUND BRACKETS.
- CREATE A TUPLE:

```
thistuple = ("apple", "banana", "cherry")
      print(thistuple)
```

- ACCESS TUPLE ITEMS

```
thistuple = ("apple", "banana", "cherry")
          print(thistuple[1])
```

- NEGATIVE INDEXING

```
thistuple = ("apple", "banana", "cherry")
        print(thistuple[-1])
```

RANGE OF INDEXES

```
thistuple = ("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")
        print(thistuple[2:5])
```

RANGE OF NEGATIVE INDEXES

```
thistuple = ("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")
        print(thistuple[-4:-1])
```

- CHANGE TUPLE VALUES

```
x = ("apple", "banana", "cherry")
y = list(x)
y[1] = "kiwi"
x = tuple(y)
print(x)
```

LOOP THROUGH A TUPLE

```
thistuple = ("apple", "banana", "cherry")
   for x in thistuple:
        print(x)
```

CHECK IF ITEM EXISTS

```
thistuple = ("apple", "banana", "cherry")
if "apple" in thistuple:
     print("yes, 'apple' is in the fruits tuple")
```

- TUPLE LENGTH

```
thistuple = ("apple", "banana", "cherry")
        print(len(thistuple))
```

```
thistuple = ("apple", "banana", "cherry")
thistuple[3] = "orange"
print(thistuple)
```

---------- ??????

CREATE TUPLE WITH ONE ITEM

```
thistuple = ("apple",)
print(type(thistuple))

#NOT A TUPLE
thistuple = ("apple")
print(type(thistuple))
```

- REMOVE ITEMS

```
thistuple = ("apple", "banana", "cherry")
del thistuple
print(thistuple) #this will raise an error because the tuple no longer exists
```

- JOIN TWO TUPLES

```
tuple1 = ("a", "b" , "c")
tuple2 = (1, 2, 3)
tuple3 = tuple1 + tuple2
print(tuple3)
```

- THE TUPLE() CONSTRUCTOR

```
thistuple = tuple(("apple", "banana", "cherry")) # note the double round-brackets
print(thistuple)
```

- TUPLES DO NOT SUPPORT THE METHODS:
  - `APPEND`
  - `REMOVE`
  - `INSERT`
  - `REVERSE`
  - `SORT`

- COUNT() METHOD

```
thistuple = (1, 3, 7, 8, 7, 5, 4, 6, 8, 5)
x = thistuple.count(5)
print(x)
```

INDEX() METHOD

```
thistuple = (1, 3, 7, 8, 7, 5, 4, 6, 8, 5)
x = thistuple.index(8)
print(x)
```