Multi-Dimensional Lists

October 18, 2021

Administrative notes

Test 1 results

- Very good, all things considered

Homework 5: DUE MONDAY, OCTOBER 25 AT MIDNIGHT

- It is NOT due Friday.
- It requires understanding of both multilevel lists and functions. Today's lecture is on multilevel lists; Wednesday we introduce functions.

Lists of lists

Now, back to lists. You can create a list of pretty much anything.

- A list of ints - a=[1,2,3,4]
- A list of floats - b = [1.0, 2.354, 3.67, -9.14]
- A list of strings - c = [“Verlander”, “Scherzer”, “Sanchez”, “Price”]
- A list of booleans - d = [True, False, True, True]

Can you create a list of lists?

Yes, you certainly can

2D List - aka, Matrix; aka, Table

Medal Table from the Track and Field (Athletics) Competition at the 2021 Tokyo Olympics

Rank	Country	Gold	Silver	Bronze	Total
1	USA	7	12	7	26
2	Italy	5	0	0	5
3	Kenya	4	4	2	10
4	Poland	4	2	3	9
5	Jamaica	4	1	4	9

How do we recreate that in python?

Each row will be a list with five entries: rank, gold medals won, silver medals won, bronze medals won, total medals won.

(We could have country name as a list element, too, but we'll leave that out for now.)

Then we'll create a list where each element is one of those lists

Creating a medal table

```
medal_table = [
   [1,7,12,7, 26],
   [2,5,0,0,5],
   [3,4, 4, 2, 10],
   [4,4,2,3, 9],
   [5,4,1,4, 9]
]
```

Some notes on this:

- Each row has the same number of elements, and they are all the same type. That is not required
  - Rows don’t have to have the same number of elements, elements can be of different type - we could have made the second row be [2,”Italy”, 5, 0, 0, 5] and it would be legal
  - But you’re getting into really bad coding habits if you do that.
- Separate each list by a comma!!!

Accessing list elements

Treat this as a table or matrix. Rows are the outer elements; columns are inside. Row and column indices both start at 0!!

len(medal_table) tells you how many ROWS are in the 2D-list

medal_table[0] is the list [1,7,12,7, 26],

medal_table[3][2] is 2 - the number of silver medals won by Poland

Using constants can help us keep track of which column means what

Constants to use with the medal_table

RANK = 0 #the first column is the country's rank

GOLDS = 1 # column 1 tells us how many gold medals the country won

SILVERS = 2 # column 2 tells us how many silver medals the country won

BRONZES = 3 # column 3 tells us how many bronze medals the country won

TOTAL = 4 # the last column tells us how many total medals the country won

medal_table [3][SILVERS] tells us how many silver medals the 4th place country won

So how many Gold medals did the top 5 countries win, combined?

```
golds_won = 0
for i in range(len(medal_table)):
    golds_won += medal_table[i][GOLDS]
print(golds_won)
```

The answer is 24.

- If you allow rows to have different numbers of elements, with different meanings, this type of calculation becomes meaningless.

Make sure you understand the table structure when doing column operations

```
medal_table   = [
[1,7,12,7, 26],
   [2,"Italy", 5,0,0,5],
   [3,4, 4, 2, 10],
   [4,4,2,3, 9],
   [5,4,1,4, 9]

]
```

```
golds_won = 0
for i in range(len(medal_table)):
   golds_won += medal_table[i][GOLDS]
print(golds_won)
```

Will fail, because the element in medal_table[1][GOLDS] isn't an integer

```
silvers_won = 0
for i in range(len(medal_table)):
      Silvers_won += medal_table[i][SILVERS]
print(silvers_won)
```

Won't fail, but it will give you the wrong answer

Creating a 2D list without entering the data

```
#write a routine that fills a 2D table with the
#successive squares - 1, 4, 9, 16, 25,...
ROWS = 5
COLUMNS = 10
square_table = []  #create the initial blank table
num_to_be_squared = 1
for i in range(ROWS):
   row = []
   for j in range(COLUMNS):
      row.append(num_to_be_squared**2)
      num_to_be_squared += 1
   square_table.append(row)
print(square_table)
```

Improving your output

```
# How do I make that output look prettier?
# print out each row on a separate line
for k in range(ROWS):
    print(square_table[k])
```

How do you add a column to a 2D list?

- Adding a row is easy - either “insert” or “append” a list
- Adding a row must be done one element at a time
- *# adding a column to our medal_table*
- *# to put the "country" in*
- countries =[**"United States"**, **“Italy”, “Kenya”, “Poland”, “Jamaica”**]
- **for** i **in** range(len(medal_table)):
- medal_table[i].insert(1, countries[i])
-
- **for** k **in** range(len(medal_table)):
- print(medal_table[k])
-

Adding a column (continued)

```
# Now we need to update the constant
definitions
# so that our previous code will still work
RANK = 0
COUNTRY = 1
GOLDS = 2
SILVERS = 3
BRONZES = 4
TOTALS = 5
```

```
golds_won = 0
for i in range(len(medal_table)):
    golds_won += medal_table[i][GOLDS]
print(golds_won)
```