

File I/O and 2D Lists

October 21, 2020

Administrative Notes

Project 1 is out! It's on the GitHub repo.

- You have to submit a design by next Wednesday, October 28
- The project is due by Monday, November 2
- We'll talk about it at the end of lecture because today's lecture will tell you a lot about what you need to know to do the project

Expect Exam #2 on November 11 - three weeks from today

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 recent 2019 IAAF World Championships

RANK	COUNTRY				TOTAL
1	 UNITED STATES	14	11	4	29
2	 KENYA	5	2	4	11
3	 JAMAICA	3	5	4	12
4	 PR OF CHINA	3	3	3	9
5	 ETHIOPIA	2	5	1	8
6	 GREAT BRITAIN & N.I.	2	3	0	5
7	 GERMANY	2	0	4	6

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, 14, 11, 4, 29],  
    [2, 5, 2, 4, 11],  
    [3, 3, 5, 4, 12],  
    [4, 3, 3, 3, 9],  
    [5, 2, 5, 1, 8],  
    [6, 2, 3, 0, 5],  
    [7, 2, 0, 4, 6],  
]
```

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, "Kenya", 5, 2, 4, 11] 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, 14, 11, 4, 29]`,

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

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 7 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 31.

- 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

```
medal_table = [  
    [1, 14, 11, 4, 29],  
    [2, "Kenya", 5, 2, 4, 11],  
    [3, 3, 5, 4, 12],  
    [4, 3, 3, 3, 9],  
    [5, 2, 5, 1, 8],  
    [6, 2, 3, 0, 5],  
    [7, 2, 0, 4, 6],  
]
```

```
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", "Kenya", "Jamaica", "China", "Ethiopia", "Great Britain", "Germany"]
- **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)
```

Now, a code example reading this data from a file

Notes: reading in and throwing away header
lines that you don't need

- This is where “readline” comes into play