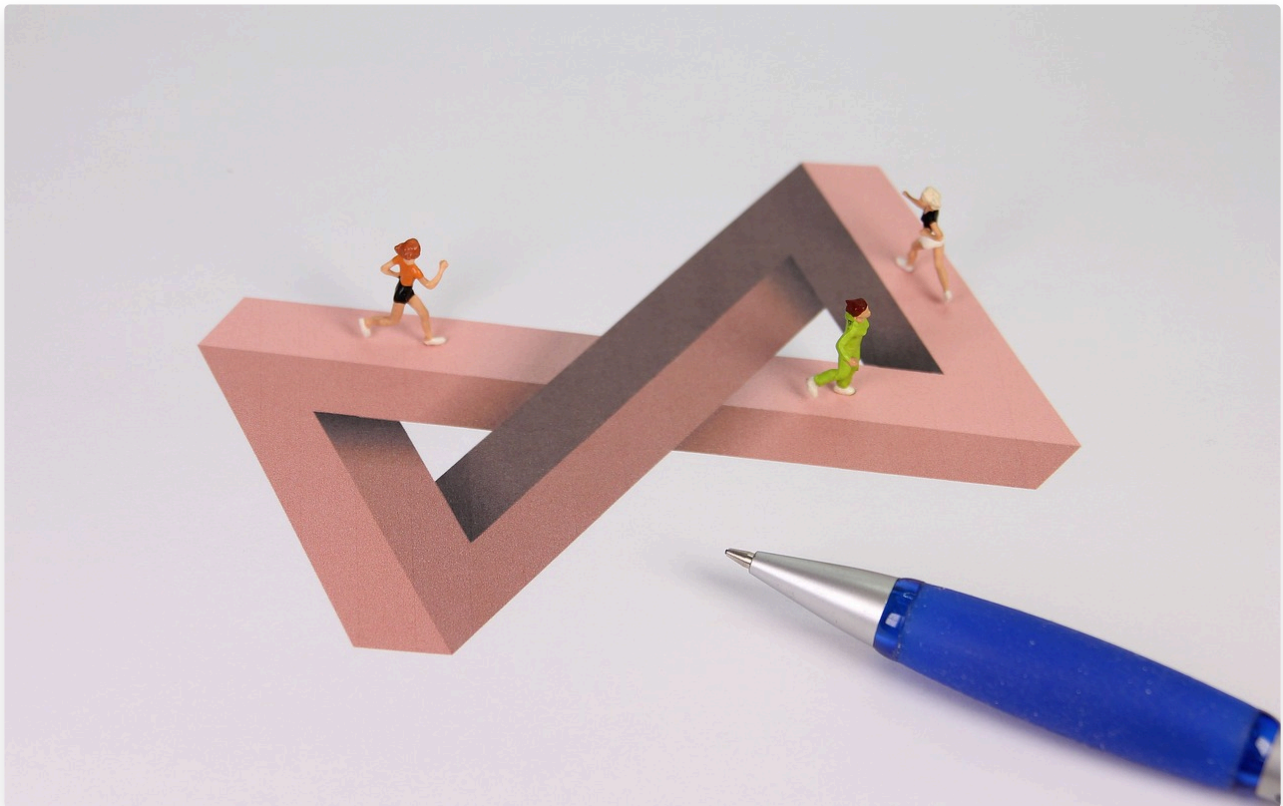


## Multi-dimensional Lists (Nested)

### Textbook

## Multi-dimensional Lists (Nested)



Multi-dimensional lists are lists with more than one dimension. This lesson covers how to create them, and when they might be useful.

### Overview

So far you've only worked with one dimensional lists, essentially a list that looks like this:

```
list = [1, 1, 1, 1, 1]
```

If we were to view this list visually on a grid, it would look like this:

1	1	1	1	1
---	---	---	---	---

This lesson introduces 2 dimensional lists, which can look like this:

```
list = [[1, 1, 1, 1, 1], [2, 2, 2, 2, 2], [3, 3, 3, 3, 3], [4, 4, 4, 4, 4]]
```

This list would have four rows and five columns, like the grid below:

1	1	1	1	1
2	2	2	2	2
3	3	3	3	3
4	4	4	4	4

As you can imagine, it's also possible to have 3D, 4D, or lists with any number of dimensions. For this lesson, the most advanced we'll get is a 2D list.

## How For Loops Create Lists

First, let's review how to create a one-dimensional list. An easy way to do this is shown in the code below:

```
1 n = 5
2 list = []
3 for i in range(n):
4     list.append(0)
5 print(list)
6
```

Try it!

This code uses a for loop and creates a list 5 elements in size, and puts the value "0" in each element.

```
[ 0, 0, 0, 0, 0 ]
```

With a 2D list it's helpful to view it as having rows and columns. A 1D list only has columns, when we add a second dimension we are adding in additional rows. The code to create this can be seen below:

```
1 rows = [1, 2, 3]
2 cols = ["red", "orange", "yellow", "green"]
3 list = []
4 for i in rows:
5     col = []
6     for j in cols:
7         col.append(j)
8     list.append(col)
9 print(list)
```

Try it!

This code produces the following 2D list.

```
[ ["red", "orange", "yellow", "green"], ["red", "orange", "yellow", "green"], ["red", "orange", "yellow", "green"] ]
```

**Let's walk through this code.**

We have two lists—one named `rows` and one named `cols`.

We also have an empty list named `list`.

The code `for i in rows:` says that we are going to go through the list named `rows`, one at a time. So we start with the `1`. But what do we do with the `1`? Let's look inside the first for loop.

Inside the first for loop we have an empty list named `col = []`. And then inside the first for loop we have another loop! This is called a **nested for loop** and we will talk more about this in the next lesson. It means for each item in the list named `rows`, we will do another loop!

So remember that we are still working on our first loop for the `1` in the list named `rows`. For just that `1`, we are looping through all five values in the list named `cols`. What are we doing when we loop through the list named `cols`? We are appending a value of `j`. The value of `j` is whichever value in the list named `cols` that we are currently on.

That sounds complex, but for that first value in the list named `rows`, we end up with another list:

```
["red", "orange", "yellow", "green"]
```

This list then gets appended to the empty list named `list`.

Then we do the whole loop again starting with the value of `2` in the list named `rows`. So by the end we end up with three rows of values.

```
["red", "orange", "yellow", "green"]
```

```
["red", "orange", "yellow", "green"]
```

```
["red", "orange", "yellow", "green"]
```

On your screen, each of these lists is in a bigger list. So it looks like this:

```
[["red", "orange", "yellow", "green"], ["red", "orange", "yellow", "green"], ["red", "orange", "yellow", "green"]]
```

## Using Range to Create 2D Lists

```
1 rows = 5
2 cols = 5
3 list = []
4 for i in range(cols):
5     col = []
6     for j in range(rows):
7         col.append(i)
8     list.append(col)
9 print(list)
10
```

Try it!

This code uses one for loop to create the columns, and another for loop to create the rows in those columns. This is called nested for loops, and you'll learn more about these in the next lesson.

## Checkpoint

---

### Multi-dimensional Lists

Write a program that creates a 2D list with **5 rows and 3 columns**.

Put the value `5` in all of the indexes, and print the result.

Use a nested for loop to create your 2D list. Use `range` for this checkpoint.

### Requirements:

- Create a variable named `rows`
- Create a variable named `cols`
- Create a variable named `list` and assign it to an empty list
- Create a for loop that loops through the range of the variable named `rows`
- Inside the for loop, create a variable and assign it to an empty list
- Inside the for loop, create another for loop that loops through the range of the variable named `cols`.
- Inside the for loop, append the number 5 to the appropriate list
- After running the inner-loop, add the smaller list to the `list` variable.
- After running the inner-loop, add the smaller list to the `list` variable.

## Questions (5)

### 1. What kind of list is this an example of?

MULTIPLE CHOICE

`mylist = [1, 2, 3, 4]`

Choose the correct answer:

- A. Two Dimensional List
- B. One Dimensional List
- C. An empty list
- D. Three Dimensional List

### 2. What kind of list is this an example of?

MULTIPLE CHOICE

`[[1, 2, 3], [1, 2, 3], [1, 2, 3]]`

Choose the correct answer:

- A. One Dimensional List
- B. Two Dimensional List
- C. Three Dimensional List
- D. An Empty List

### 3. True or False: A for loop can create a list.

MULTIPLE CHOICE

Choose the correct answer:

- A. True
- B. False

**4. What is the output of the code provided?**

```
n = 5 list = [] for i in range(n): list.append(0) print(list)
```

**Choose the correct answer:**

- A. [0, 0, 0, 0, 0]
- B. [1, 1, 1, 1, 1]
- C. [5, 5, 5, 5, 5]
- D. [1, 2, 3, 4, 5]

**5. What is the purpose of using the range function in the code for creating 2D lists?**

```
rows = 5 cols = 5 list = [] for i in range(cols): col = [] for j in range(rows): col.append(i) list.append(col) print(list)
```

**Choose the correct answer:**

- A. To determine the size of each row and column
- B. To add negative numbers to the list
- C. To create a list with only one dimension
- D. To remove elements from the list

## Challenges (3)

### 1. Secret Agent Name Generator

You are picking out a secret agent name for yourself. You want to make sure it sounds cool and that there are a lot of options! You know that many secret agent names have a first and a last name.

You have some ideas of good first names and some cool last names, but aren't sure which ones sound the best when put together. Create a Python program that will put together these names in every combination so you can see which you like best!

1. Create a variable and assign it to a list of **4** cool first names.
2. Create another variable and assign it to a list of **4** cool last names.
3. Create an empty list.
4. Create a **for loop** that loops through the first names.
5. Inside the first **for loop**, create an **empty list**.
6. Inside the **for loop**, create another **for loop** that loops through the last names.
7. Inside the **for loop**, combine the first and last names and **append** them to the appropriate empty list.
8. Print the completed 2D list.

**Note: Double check your spacing and indentation!**

#### Requirements:

- Create a variable with a list of 4 first names and a variable with a list of 4 last names.
- Create an empty list and assign it to a variable.
- Create a for loop that loops through the list of first names
- Inside the for loop, create a variable and assign it to an empty list
- Inside the for loop, create another for loop that loops through the list of last names.
- Inside the for loop, concatenate the first and last names together.
- Print the 2D list

## 2. Fruit Blender

You are working at a juice shop. Your manager has asked you to come up with a new, unique combination of fruits for an all new juice. You decide to create a Python program to help you come up with combinations you might have never thought of before.

Your manager gives you the following list of flavors to combine with.

**["apple", "grape", "peach", "cinnamon", "vanilla"]**

Your program will take in a user input of fruits and will combine each fruit with each flavor your manager gave you.

For example:

Input: `kumquat starfruit papaya`

Output: `[['kumquat apple', 'kumquat grape', 'kumquat peach', 'kumquat cinnamon', 'kumquat vanilla'],  
['starfruit apple', 'starfruit grape', 'starfruit peach', 'starfruit cinnamon', 'starfruit vanilla'],  
['papaya apple', 'papaya grape', 'papaya peach', 'papaya cinnamon', 'papaya vanilla']]`

Don't forget to add a space between the fruits!

Hint: This is how to create a list from an input

```
rows = input("Input a list of fruits").split()
```

### Requirements:

- Create a variable and set it equal to an input that generates a list of fruits.
- Create a variable and set it equal to the list of fruits your manager said to mix with.
- Create an empty list and set it equal to a variable.
- Create a for loop that loops through the inputted list of fruits.

## 3. Subtracting in a 2D List

Create a program that will generate **a 2D list**.

Consider the following list.

```
cols = [2, 5, 10, 100]
```

The user will input a list of numbers. The program will then subtract all the numbers in the list named `cols` from each number in the inputted list.

For example:

Input: `10 20 30 40`

Output: `[[8, 5, 0, -90], [18, 15, 10, -80], [28, 25, 20, -70], [38, 35, 30, -60]]`

Hint: This is how to create a list of integers from an input:

```
rows = [int(n) for n in input("Input a list of numbers").split()]
```

## Answer Keys & Solutions

### Checkpoint Solutions

#### Multi-dimensional Lists

```
1 rows = 5
2 cols = 3
3 list = []
4 for i in range(rows):
5     col = []
6     for j in range(cols):
7         col.append(5)
8     list.append(col)
9 print(list)
```

### Questions

#### 1. What kind of list is this an example of?

MULTIPLE CHOICE

Correct Answer:

- A. Two Dimensional List ✗ Incorrect
- B. One Dimensional List ✓ Correct
- C. An empty list ✗ Incorrect
- D. Three Dimensional List ✗ Incorrect

#### Explanation:

This is just a normal list.

#### 2. What kind of list is this an example of?

MULTIPLE CHOICE

Correct Answer:

- A. One Dimensional List ✗ Incorrect
- B. Two Dimensional List ✓ Correct
- C. Three Dimensional List ✗ Incorrect



D. An Empty List

✗ Incorrect

#### Explanation:

See how there's lists inside the list? This means it's a multidimensional list.

### 3. True or False: A for loop can create a list.

MULTIPLE CHOICE

#### Correct Answer:

A. True

✓ Correct

B. False

✗ Incorrect

#### Explanation:

We learned how to do this in this lesson.

### 4. What is the output of the code provided?

MULTIPLE CHOICE

#### Correct Answer:

A. [0, 0, 0, 0, 0]

✓ Correct

B. [1, 1, 1, 1, 1]

✗ Incorrect

C. [5, 5, 5, 5, 5]

✗ Incorrect

D. [1, 2, 3, 4, 5]

✗ Incorrect

#### Explanation:

The code is appending the value of 0 to the list

### 5. What is the purpose of using the range function in the code for creating 2D lists?

MULTIPLE CHOICE

#### Correct Answer:

A. To determine the size of each row and column

✓ Correct

B. To add negative numbers to the list

✗ Incorrect

C. To create a list with only one dimension

✗ Incorrect

D. To remove elements from the list

✗ Incorrect

### Explanation:

The range goes through each number

## Challenges

### 1. Secret Agent Name Generator

Solution:

```
1 rows = ["Doctor", "Quick", "Captain", "Fritz"]
2 cols = ["Silver", "Shadow", "Dynamo", "Canary"]
3 list = []
4 for i in rows:
5     col = []
6     for j in cols:
7         col.append(i + " " + j)
8     list.append(col)
9 print(list)
```

### 2. Fruit Blender

Solution:

```
1 rows = input("Input a list of fruits").split()
2
3 cols = ["apple", "grape", "peach", "cinnamon", "vanilla"]
4
5 list = []
6 for i in rows:
7     col = []
8     for j in cols:
9         col.append(i + " " + j)
10    list.append(col)
11 print(list)
```

### 3. Subtracting in a 2D List

Solution:

```
1 rows = [int(n) for n in input("Input a list of numbers").split()]
2
3 cols = [2, 5, 10, 100]
4
5 list = []
```

```
6 for i in rows:
7     col = []
8     for j in cols:
9         col.append(i - j)
10    list.append(col)
11 print(list)
```