

# Computer Science Programming Fundamentals

## Lists



# 💡 Lists Intro

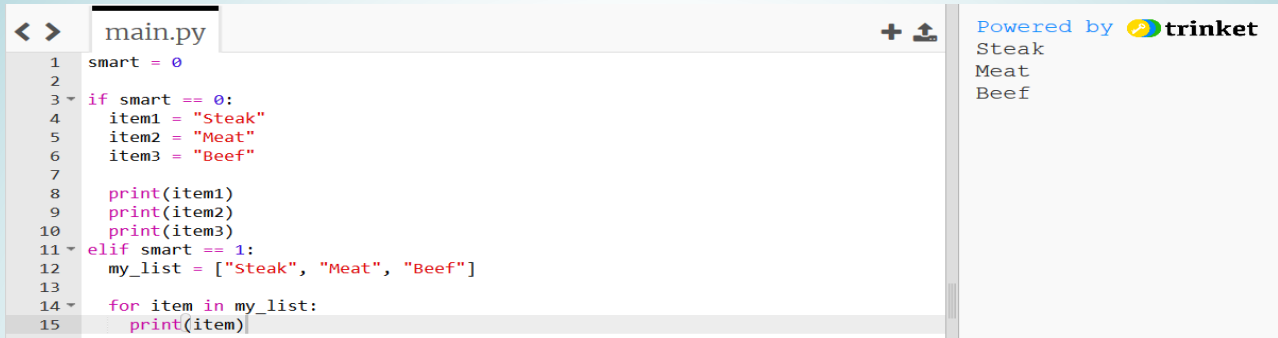
When we make a shopping list, we don't write each thing we need on a different piece of paper. We write it all on one. Why? Because this makes it easier to use!

When we make a program that works with many separate but similar items, it is possible to make a bunch of variables. However, it's often far easier to make a list!


In this lesson, we'll learn exactly how lists can make our lives easier.

# Lists pt 1

- Run the example:



```
1 smart = 0
2
3 if smart == 0:
4     item1 = "Steak"
5     item2 = "Meat"
6     item3 = "Beef"
7
8     print(item1)
9     print(item2)
10    print(item3)
11 elif smart == 1:
12    my_list = ["Steak", "Meat", "Beef"]
13
14    for item in my_list:
15        print(item)
```

Powered by  trinket

Steak  
Meat  
Beef

- Change the value of `smart` to 1, then run it again.

Just like in the last lesson, the output is the same, but the code making that output is different.

When `smart` is set to 0, Python makes three variables, then prints them out.

When `smart` is set to 1, Python makes one **list**, then uses a for loop to print all the items out.

# Lists pt 2

After we have created a list, we can access any item in that list like this:

- Run:

```
scary_things = ["clowns", "snakes"]  
print(scary_things[1])
```

Q: Why did the above code print "snakes" and not "clowns"?

A: In Python, item 1 is not the first time. The first item is item 0. To print "clowns":

- Run:

```
scary_things = ["clowns", "snakes"]  
print(scary_things[0])
```

The **number** we put in square brackets is called the **index**.

# Lists pt 3

Changing the value of an item in a list is the same as changing the value of a variable.

- Run:

```
scary_things = ["clowns", "snakes"]  
scary_things[1] = "bats"  
# Print the whole list this time  
print(scary_things)
```

Add an item to the end of the list like this:

- Run:

```
scary_things = ["clowns", "snakes"]  
scary_things.append("spiders")  
print(scary_things)
```

# Lists pt 4

There are two ways to remove items from the list.

- Remove by value:

```
scary_things = ["clowns", "snakes"]  
# I'm not afraid of clowns anymore!  
scary_things.remove("clowns")  
print(scary_things)
```

- Remove by index:

```
scary_things = ["clowns", "snakes"]  
scary_things.pop(1)  
print(scary_things)
```

We can use a function called `len()` to get the number of items in that list.

- Run:

```
scary_things = ["clowns", "snakes"]  
print(len(scary_things))
```



# Lists pt 5

Lists get even cooler when we combine them with for loops, as seen in the first example in this lesson.

- Run:

```
scary_things = ["clowns", "snakes"]  
for thing in scary_things:  
    print(thing, "are scary!")
```

**Q: What is the value of thing?**

**A:** It changes each time the loop runs. On the first run, its value is "clowns." On the second run, its value is "snakes."

The loop automatically ends when it's done going through all the items in the list.

# Lists pt 6

You can use the `in` keyword to test if something is in a list.

- Run:

```
scary_things = ["clowns", "snakes"]  
if "clowns" in scary_things:  
    print("I'm afraid of clowns.")
```

You can make lists of strings, ints, floats, and even other lists! A list holding other lists is essentially a **2D array**.

- Run:

```
floats_ints_strings = [  
    [1.5, 2.3, 6.4],  
    [6, 5, 4],  
    ["mother", "father", "sister"]  
]  
print(floats_ints_strings[2][1])
```

The **first index** tells Python which list you want to access. The **second index** tells Python which item in that list you want to access.



# Lists pt 7



## Activity:

- Your annoying roommate mixed up all the food in the pantry and now you don't know what belongs to whom. You decide to split up the items according to their expiration dates.
  - All even dates go to you.
  - All odd dates go to your roommate.
  - Things expiring before 2018 should probably be thrown away...
- Make a program that sorts the food in the pantry , putting your items in one list, your roommate's items in another list, and putting items that are too old in the trash. Don't let any expired food mix in with the good food!

# Lists pt 8

## A possible answer:

```
food_dates_in_pantry = [  
    2018, 2019, 2019, 2020,  
    2019, 2022, 2025, 1932,  
    2020, 2017, 2018, 2012  
]
```

```
trash = []
```

```
my_food = []
```

```
his_food = []
```

```
for date in food_dates_in_pantry:
```

```
    if date < 2018:
```

```
        trash.append(date)
```

```
    elif date % 2 == 0 and date not in trash:
```

```
        my_food.append(date)
```

```
    elif date % 2 != 0 and date not in trash:
```

```
        his_food.append(date)
```

```
print("My food:", my_food)
```

```
print("His food:", his_food)
```

# Lists pt 8



## Explanation:

For simplicity, have students use the same starting list.

Since there are three expired items, their programs should keep 9 items.