```python
# "Parallel" lists store many facts about individual "things",
# each fact saved in a different list at the same corresponding index.

names = ["Mike Trout", "Albert Pujols", "Andrelton Simmons", "Martín Maldonaldo"]
hits = [123, 143, 164, 221]
homeRuns = [33, 23, 14, 14]

userChoice = input("Enter the name of a 2017 Angels player: ")

# Print the player's information, by locating their name
# in the list called "names".

playerIndex = -1
for i in range(len(names)):
    if names[i] == userChoice:
        playerIndex = i
        break


# A SECOND WAY:
playerIndex = names.find(userChoice)



# Print the stats.
if playerIndex >= 0:
    print("Statistics for", names[playerIndex])
    print("Hits:", hits[playerIndex])
    print("Home runs:", homeRuns[playerIndex])
else:
    print("I don't know that player!")

```

```python
# The way that lists are passed as parameters to functions is identical
# to other object types.

# To review:
def changeInt(x):
    x = 10

age = 34
changeInt(age)
print(age)     # Are you willing to make a bet about what gets printed?

# The variable age is not mutated, for two reasons:
# 1. There is no mutation in changeInt. There is assignment.
# 2. The variable that is assigned to (x) is in a different namespace than
#    age ... there's just no way that age could be mutated.


# These rules apply to lists as well
def changeList(y):
    y = [1, 2, 3]

lottery = [15, 7, 11]
changeList(lottery)
print(lottery)


# But we can actually mutate lists by passing them to functions.
def appendToList(y):
    y.append(99)
    # append is a mutation. It is a method on lists that doesn't
    # assign a new value; it mutates the existing value.

appendToList(lottery)
print(lottery)

# Draw a picture of what's going on!!!


```