A list is an ordered and mutable Python container, being one of the most common data structures in Python.

WHAT DOES THE FOLLOWING DO?

```python
names = ["ada", 'bob', 'charlie', 'erin']
print(names[0])
print(names[3])
print(names[-1])
print(names)
for name in names:
  print(name)
```

WHAT DOES THE FOLLOWING DO?

```python
scores = [1, 2, 3, 4, 5]
print(max(scores))
scores.reverse()
print(scores)
```

WHAT DOES THE FOLLOWING DO?

```python
scores = [1, 2, 3, 4, 5]
max_value = max(scores)
print(scores.index(max_value))
```

WHAT DOES THE FOLLOWING DO?

```python
scores = [1, 2, 3, 4, 5]
max_value = max(scores)
print(scores[scores.index(max_value)])
```

WHAT HAPPENS IF YOU NEED TO REMEMBER AN UNKNOWN NUMBER OF THINGS?

As soon as the top score appears, people start to wonder what the second and third highest scores are:

1. Bob: 9.12
2. ???: ???
3. ???: ???

It seems that the organizers didn’t tell you everything you needed to know. The contest doesn’t just award a prize for the winner, but also honors those surfers in second and third place.
Our program currently iterates through each of the lines in the round2.txt file and works out the highest score. But what it actually needs to do is keep track of the top three scores, perhaps in three separate variables:

KEEPING TRACK OF 3 SCORES MAKES THE CODE MORE COMPLEX

```python
highest_score = 0
second_highest = 0
third_highest = 0
highest_name = ""
second_name = ""
third_name = ""

with open("round2.txt") as f:
  for line in f:
    name, score = line.split(" ")
    if score > highest_score:
      highest_score = score
      highest_name = name
    elif score > second_highest:
      second_highest = score
      second_name = name
    elif score > third_highest:
      third_highest = score
      third_name = name

print(f"1st place was {highest_name} with {highest_score}")
print(f"2nd place was {second_name} with {second_highest}")
print(f"3rd place was {third_name} with {third_highest}")
```

LET'S MAKE SOME LISTS!

```python
scores = []
names = []

with open(".foo/foo.txt") as f:
  for line in f:
    name, score = line.strip().split(" ")
    names.append(name)
    scores.append(float(score))
print(names)
print(scores)
```

[] is how you tell python that you want to make a list
list_name.append(item) is how you tell python you want to add something to that list

GREAT, SO WE HAVE TWO LISTS FILLED WITH STUFF

How can we ask python which number is the largest?

```
print(names, scores)

>>> ['ada', 'bob', 'charles', 'erin', 'fred', 'georgia', 'harry']
>>> [8.65, 9.12, 8.45, 7.81, 8.05, 7.21, 8.31]
```

HOW CAN WE FIND THE LARGEST NUMBER IN A LIST?

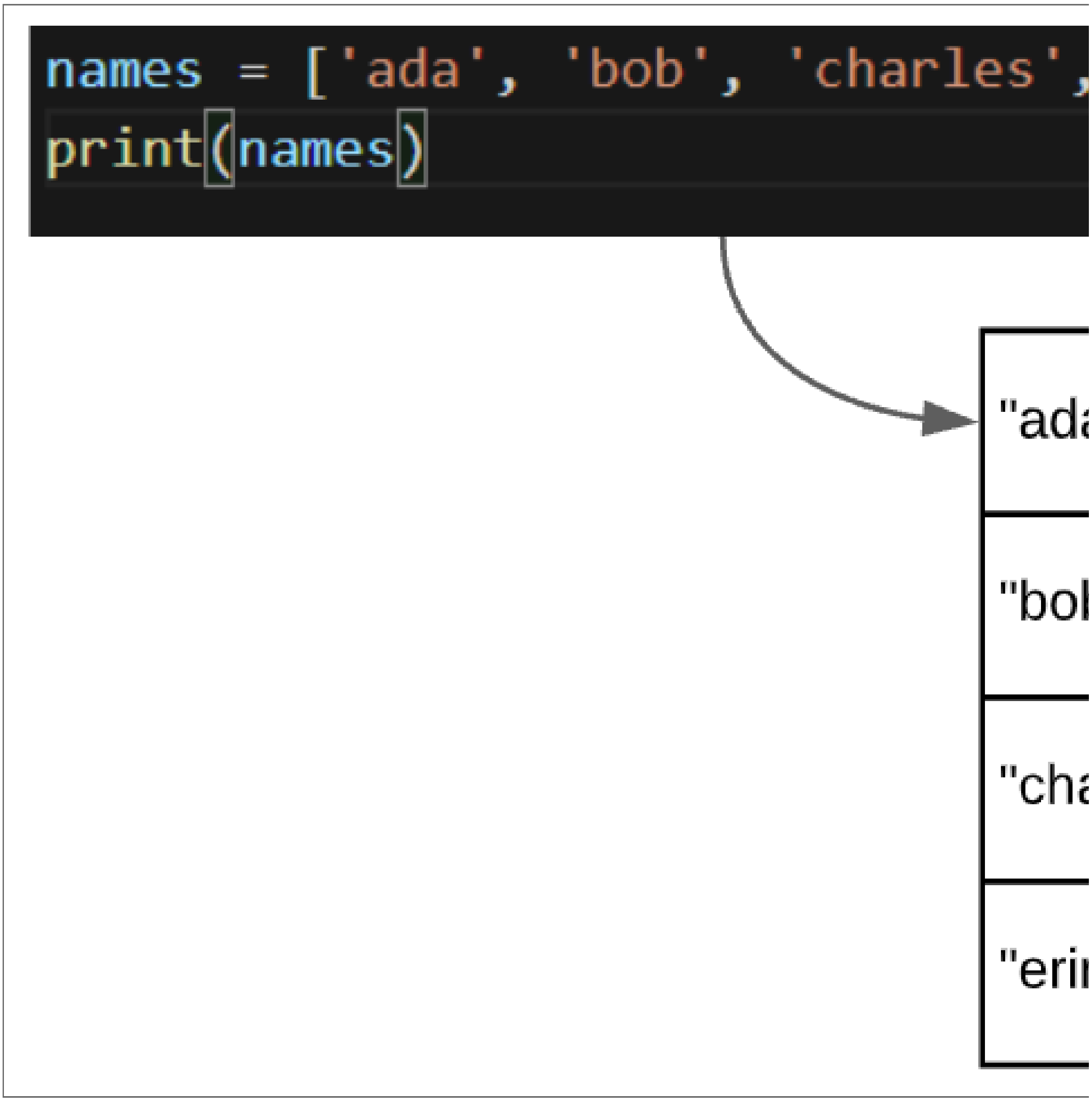
With the max function!
You can google: "how to find largest number in list python"

```
print(names)
print(max(scores))

>>> ['ada', 'bob', 'charles', 'erin', 'fred', 'georgia', 'harry']
>>> 9.12
```

BEFORE WE ASK PYTHON TO TELL US WHERE THAT NUMBER IS, LET'S TALK ABOUT LISTS

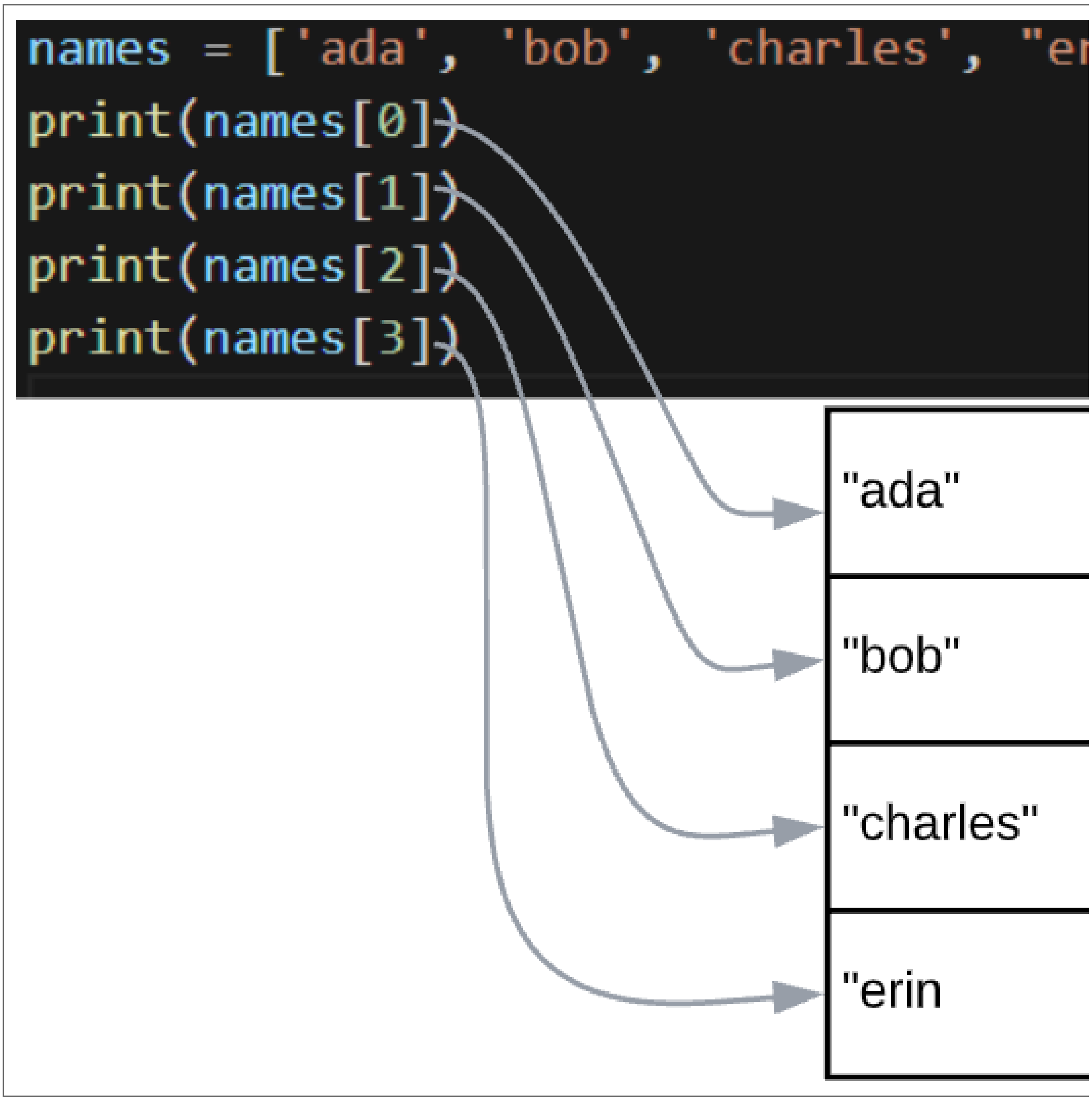
Lists are a collection of things



```
names = ['ada', 'bob', 'charles',
print(names)
```

LIST INDEXES

We can access individual elements through the list index



```
names = ['ada', 'bob', 'charles', "er
print(names[0])
print(names[1])
print(names[2])
print(names[3])
```

HOW DOES THAT HELP?

```python
names = ['ada', 'bob', 'charles', "erin"]
i = 0
while i < len(names):
  print(names[i])
  i += 1
```

Does anybody feel that while loop is a bit wonky?

FOR (EACH) LOOP

```python
names = ['ada', 'bob', 'charles', "erin"]
for name in names:
  print(name)
```

BACK AT IT

Our goals in our game

- Get python to find the highest score
- Get python to tell us where it is
- Use that information to print the name and score
- Do something with that data (maybe store it elsewhere)

FIND THE HIGHEST SCORE IN A LIST

```python
names = ['ada', 'bob', 'charles', 'erin', 'fred', 'georgia', 'harry']
scores = [8.65, 9.12, 8.45, 7.81, 8.05, 7.21, 8.31]
highest_score = max(scores)
print(highest_score)

>>> 9.12
```

FIND WHERE THAT SCORE WAS

We can use the list.index(value) function!

```python
names = ['ada', 'bob', 'charles', 'erin', 'fred', 'georgia', 'harry']
scores = [8.65, 9.12, 8.45, 7.81, 8.05, 7.21, 8.31]
highest_score = max(scores)
highest_index = scores.index(highest_score)
print(highest_index)

>>> 1
```

USE THAT INDEX LOCATION FOR BOTH SCORE AND NAME

```python
names = ['ada', 'bob', 'charles', 'erin', 'fred', 'georgia', 'harry']
scores = [8.65, 9.12, 8.45, 7.81, 8.05, 7.21, 8.31]
highest_score = max(scores)
highest_index = scores.index(highest_score)
print(names[highest_index], scores[highest_index])

>>> Bob 9.12
```

I HEARD YOU LIKE LISTS IN YOUR LISTS

```python
names = ['ada', 'bob', 'charles', 'erin', 'fred', 'georgia', 'harry']
scores = [8.65, 9.12, 8.45, 7.81, 8.05, 7.21, 8.31]

high_scores = []

highest_score = max(scores)
highest_index = scores.index(highest_score)
highest_name = names[highest_index]
highest_score = scores[highest_index]

high_scores.append([highest_name, highest_score])

print(high_scores)

>>> [['bob', 9.12]]
```

CAN WE REMOVE STUFF FROM LISTS?

```python
names = ['ada', 'bob', 'charles', 'erin', 'fred', 'georgia', 'harry']
scores = [8.65, 9.12, 8.45, 7.81, 8.05, 7.21, 8.31]

high_scores = []

highest_score = max(scores)
highest_index = scores.index(highest_score)
highest_name = names[highest_index]
highest_score = scores[highest_index]

high_scores.append([highest_name, highest_score])

names.remove(highest_name)
scores.remove(highest_score)

print(high_scores)

>>> [['bob', 9.12]]
```

REFACTORING

```python
def get_high_score(scores, names):
  highest_score = max(scores)
  highest_index = scores.index(highest_score)
  highest_name = names[highest_index]
  highest_score = scores[highest_index]
  scores.remove(highest_score)
  names.remove(highest_name)
  return [highest_name, highest_score], scores, names

names = ['ada', 'bob', 'charles', 'erin', 'fred', 'georgia', 'harry']
scores = [8.65, 9.12, 8.45, 7.81, 8.05, 7.21, 8.31]

high_scores = []

for i in range(3):
  high_score, scores, names = get_high_score(scores, names)
  high_scores.append(high_score)

print(high_scores)

>>> [['bob', 9.12]]
```

REFACTORING 2

```python
def get_high_score(scores, names):
  highest_score = max(scores)
  highest_index = scores.index(highest_score)
  highest_name = names[highest_index]
  highest_score = scores[highest_index]
  scores.remove(highest_score)
  names.remove(highest_name)
  return [highest_name, highest_score]

names = ['ada', 'bob', 'charles', 'erin', 'fred', 'georgia', 'harry']
scores = [8.65, 9.12, 8.45, 7.81, 8.05, 7.21, 8.31]

high_scores = []

for i in range(3):
  high_score = get_high_score(scores, names)
  high_scores.append(high_score)

print(high_scores)

>>> [['bob', 9.12]]
```

REFACTORING 2

```python
def get_high_score(scores, names):
  highest_score = max(scores)
  highest_index = scores.index(highest_score)
  highest_name = names[highest_index]
  highest_score = scores[highest_index]
  scores.remove(highest_score)
  names.remove(highest_name)
  return [highest_name, highest_score] #, scores, names

names = ['ada', 'bob', 'charles', 'erin', 'fred', 'georgia', 'harry']
scores = [8.65, 9.12, 8.45, 7.81, 8.05, 7.21, 8.31]

high_scores = []

for i in range(3):
  high_score = get_high_score(scores, names)
  high_scores.append(high_score)

for score in high_scores:
  print(f"{score[1]} : {score[0].title()}")

>>> 9.12 : Bob
>>> 8.65 : Ada
>>> 8.45 : Charles
```

Speaker notes