# Exercise 4

Things get more interesting with lists. You'll apply your new knowledge to solve the questions below. Remember to run the following cell first.

In [ ]:
```
# Run this cell to setup the exercise

import sys
from pathlib import Path
learntools_dir = Path().absolute().parents[1]
sys.path.append(str(learntools_dir))
from learntools.core import binder; binder.bind(globals())
from learntools.python.ex4 import *
print('Setup complete.')
```

## 1.

Complete the function below according to its docstring.

In [ ]:
```
def select_second(L):
    """Return the second element of the given list. If the list has no secor
    element, return None.
    """
    pass

# Check your answer
q1.check()
```

In [ ]:
```
# q1.hint()
# q1.solution()
```

## 2.

You are analyzing sports teams. Members of each team are stored in a list. The Coach is the first name in the list, the captain is the second name in the list, and other players are listed after that. These lists are stored in another list, which starts with the best team and proceeds through the list to the worst team last. Complete the function below to select the **captain** of the worst team.

In [ ]:
```
def losing_team_captain(teams):
    """Given a list of teams, where each team is a list of names, return the
    from the last listed team
    """
```

```
# Check your answer
q2.check()
```

In [ ]:
```
# q2.hint()
# q2.solution()
```

# 3.

The next iteration of Mario Kart will feature an extra-infuriating new item, the *Purple Shell*. When used, it warps the last place racer into first place and the first place racer into last place. Complete the function below to implement the Purple Shell's effect.

In [ ]:
```
def purple_shell(racers):
    """Given a list of racers, set the first place racer (at the front of th
    place and vice versa.

    >>> r = ["Mario", "Bowser", "Luigi"]
    >>> purple_shell(r)
    >>> r
    ["Luigi", "Bowser", "Mario"]
    """
    pass

# Check your answer
q3.check()
```

In [ ]:
```
# q3.hint()
# q3.solution()
```

# 4.

What are the lengths of the following lists? Fill in the variable `lengths` with your predictions. (Try to make a prediction for each list *without* just calling `len()` on it.)

In [ ]:
```
a = [1, 2, 3]
b = [1, [2, 3]]
c = []
d = [1, 2, 3][1:]

# Put your predictions in the list below. Lengths should contain 4 numbers,
# first being the length of a, the second being the length of b and so on.
lengths = []

# Check your answer
q4.check()
```

In [ ]:
```
# line below provides some explanation
```

```
# q4.solution()
```

# 5. 🌶️

We're using lists to record people who attended our party and what order they arrived in. For example, the following list represents a party with 7 guests, in which Adela showed up first and Ford was the last to arrive:

```
party_attendees = ['Adela', 'Fleda', 'Owen', 'May', 'Mona',
'Gilbert', 'Ford']
```

A guest is considered 'fashionably late' if they arrived after at least half of the party's guests. However, they must not be the very last guest (that's taking it too far). In the above example, Mona and Gilbert are the only guests who were fashionably late.

Complete the function below which takes a list of party attendees as well as a person, and tells us whether that person is fashionably late.

In [ ]:
```
def fashionably_late(arrivals, name):
    """Given an ordered list of arrivals to the party and a name, return whe
    name was fashionably late.
    """
    pass

# Check your answer
q5.check()
```

In [ ]:
```
# q5.hint()
# q5.solution()
```

# Keep Going

That's it for lists and tuples! Now you have the baseline knowledge to **learn about loops**, which is where lists and tuples get really interesting.

---