Program Design and Objects

Wrestling with Python

Specifications

- Before you have the program, you have the specification of the problem
  - These are especially problematic
  - Mainly because they are written in English by humans ☺
- But we can use them to work out what the program needs to do

Unpicking the Golf Club Spec.

Each golfer who plays a round of golf will return with a scorecard which contains the following:

- *The name of the golfer*
- *A set of 18 pairs of numbers. The first number is the par for a given hole on the golf course (the number of shots it should take to get the ball into the hole). The second number is the number of shots the golfer actually took to get the ball into the hole*

- If we want to store information in our Golf Scoring program (which is required for the later tasks) we need to decide what is to be stored

Verb/Noun Analysis

Each golfer who plays a round of golf will return with a scorecard which contains the following:

- *The name of the golfer*
- *A set of 18 pairs of numbers. The first number is the par for a given hole on the golf course (the number of shots it should take to get the ball into the hole). The second number is the number of shots the golfer actually took to get the ball into the hole*

- We can get some idea of what the program needs to do by applying some textual analysis to the description
  - Verbs are actions
  - Nouns are things

Finding the Nouns

Each golfer who plays a round of golf will return with a scorecard which contains the following:

- *The name of the golfer*
- *A set of 18 pairs of numbers. The first number is the par for a given hole on the golf course (the number of shots it should take to get the ball into the hole). The second number is the number of shots the golfer actually took to get the ball into the hole*

- This piece of the specification is really telling us about the data design, and so we are looking for nouns
- We use the verbs to identify program actions
  - Next time.....

UNIVERSITY OF Hull

Finding the Nouns

Each golfer who plays a round of golf will return with a scorecard which contains the following:

- *The name of the golfer*
- *A set of 18 pairs of numbers. The first number is the par for a given hole on the golf course (the number of shots it should take to get the ball into the hole). The second number is the number of shots the golfer actually took to get the ball into the hole*

- These are all the noun type things that map onto the data in our program
- Now we can think of the kind of data that we are holding for each item

Verb/Noun Analysis

Each golfer who plays a round of golf will return with a scorecard which contains the following:

- *The name of the golfer*
- *A set of 18 pairs of numbers. The first number is the par for a given hole on the golf course (the number of shots it should take to get the ball into the hole). The second number is the number of shots the golfer actually took to get the ball into the hole*

- I've found three kinds of things:
  - string 🟦
  - integer 🟥
  - Some kind of container 🟪

What happened to the golfer?

- I've removed the golfer from my design as we don't actually store any information about him/her specifically
- If we wanted to store a bunch of scorecards from a particular golfer the object might come back as a container
- But we might need to change other parts of the design if we do this – why?

What happened to the golfer?

- I’ve removed the golfer from my design as we don’t actually store any information about him/her specifically
- If we wanted to store a bunch of scorecards from a particular golfer the object might come back as a container
- But we might need to change other parts of the design if we do this – why?
  - Because we probably wouldn’t store the name of the golfer on each scorecard if they are all assigned to the same person

Relationships between the data

- A scorecard contains
  - The name of the golfer
  - A collection of 18 holes
- A hole contains
  - The par value for the hole
  - The number of shots that the golfer took

Some names

```
scorecard     playerName
hole par shots roundOfHoles
```

- I could use these names for my Python variables

Purple Things

- A purple thing on my diagram is either a collection (a bunch of identical things) or a class (a bunch of related things)
- Which is which?

Purple Things

- A purple thing on my diagram is either a collection (a bunch of identical things) or an object (a bunch of related things)
- Which is which?
  - The `roundOfHoles` is a collection (everything in it is the same thing)
  - The other items are objects

Objects and Python

- Python lets us create objects that can hold information
- We did this a while back when we were storing cricket scores
- The object holds items that we can lump together
- They are described in classes

A hole class

```
class hole:
    def __init__(self, par, shots):
        self.par = par
        self.shots = shots
```

- This is a hole class
- It just contains a single method which is used to initialise it

Classes and Objects

- The class information tells the Python system how to make an instance of the class
  - This is called an *object*
- We have told the system how to create a `hole` instance
- However, we have not actually created any `hole` objects yet

Creating an object

```
h = hole(3, 4)
print(h.par)
```

- This creates an instance of the `hole` class and then prints out the par value held in it
- Note that we pass in the par and the shots when we create the instance
- This is passed into the `__init__` method

The __init__ method

```
class hole:
    def __init__(self, par, shots):
        self.par = par
        self.shots = shots
```

- The `__init__` method is called the *constructor* for the class
- It is called automatically when we make a new instance

self in the __init__ method

```
class hole:
    def __init__(self, par, shots):
        self.par = par
        self.shots = shots
```

- The `__init__` method needs a reference to the object that is being created
- Python sets this reference and passes it into the method as the first parameter

Creating object attributes

```
class hole:
    def __init__(self, par, shots):
        self.par = par
        self.shots = shots
```

- The `__init__` method creates `par` and `shots` attributes which are held in the object
- The input values are copied into these attributes

Fetching data from an object

```
h = hole(3, 4)
print(h.par)
```

- You can get data out of attributes in an object by referring to them by name
  - They are effectively just like regular Python variables, they just live in an object

Lists in Python

- We can now create a data item that will hold the results for a single hole
- But we need 18 of them
- We don’t want to create 18 separate variables, one for each hole
- What we want is a List of holes

Creating an empty list

- You can create an empty list

```
roundOfHoles = []
```

- At the moment this list holds nothing at all
- The program can add items into the list and the list will keep track of them for us

Appending to a list

```
roundOfHoles = []
roundOfHoles.append(hole(3,4))
roundOfHoles.append(hole(4,6))
```

- Once you have your list you can append things to it
- `roundOfHoles` now holds two values
- We can append as many items as we like

Appending to a list

```
roundOfHoles = []
roundOfHoles.append(hole(3,4))
roundOfHoles.append(hole(4,6))
```

- In the real program these values would be ones that the user has entered
- We would have a loop that goes around reading them in from the user

Working through a list

```
totalPar = 0
for h in roundOfHoles:
    totalPar = totalPar + h.par
```

- We can use for loops to work through the elements of a list
- This code works out the total par for all the holes

Making a scoreCard

```
class scoreCard:
    def __init__(self, name):
        self.name = name
        self.roundOfHoles = []
```

- The `scoreCard` object contains the name of the player and a list of the holes
- We can make the list in the constructor

UNIVERSITY OF Hull

Methods in classes

```
class scoreCard:
    def __init__(self, name):
        self.name = name
        self.roundOfHoles = []

    def getPar(self):
        totalPar = 0
        for h in self.roundOfHoles:
            totalPar = totalPar + h.par
        return totalPar
```

- We can add methods to classes to make them do things for us

Using class methods

```
s = scoreCard("Rob")
s.roundOfHoles.append(hole(3,4))
s.roundOfHoles.append(hole(3,4))
s.roundOfHoles.append(hole(3,4))
print(s.getPar())
```

- This code adds three holes to the scorecard for Rob and then prints out the par for the course

Things to do

- Improve your golf score program so that it holds the scorecard information as shown
- Add a method to the scoreCard class that will print out the score for a player
- Look at using this class to help with tasks 2 and 3