Computing

Lesson 3: In a `While`, Crocodile

Python Programming with Sequences of Data

Rebecca Franks

[1] *Materials from the Teach Computing Curriculum created by the National Centre for Computing Education*

Worked Example Form a band, version 1

This program repeatedly reads the names of musical instruments and adds them to a list. The program stops when the length of the list reaches three.

```
1  print("Let's form a band")
2  band = []
3  while len(band) < 3:
4      print("Pick an instrument:")
5      instrument = input()
6      band.append(instrument)
7  print(band)
```

Add instrument

Worked Example Form a band, version 2

This program repeatedly reads the names of musical instruments and adds them to a list. The program stops when a guitar is added to the list.

```
1  print("Let's form a band")
2  band = []
3  while "guitar" not in band:
4      print("Pick an instrument:")
5      instrument = input()
6      band.append(instrument)
7  print(band)
```

Add instrument

Task 1

You will develop a program that plans a European city-hopping trip, by randomly selecting cities out of a given list.

Step 1 - Open this program **oaknat.uk/comp-py-hopping-1** in Repl.it

```
1 from ncce.data import cities
2 from random import choice
3 city = choice(cities)
4 print(city)
```

Line 1 imports the list of cities that the program will use. **Note** that this is **not** a standard Python component. The list has been created specifically to allow you to perform this activity.

The `choice` function (imported from the random module in line 2) is used to randomly select an item out of the list of `cities` (line 3).

Task 1

Step 2

Run the program at least three or four times to see how a different European city is selected every time.

Step 3

Remove the last line from your program. You won't need it.

```
- print(city)
```

Task 1

Step 4

Add the lines on the **next slide** into your program. They are provided in **no particular order**.

Rearrange these lines so that your program repeatedly selects a random `city` out of the `cities` list, and adds it to the `trip` list.

Your program should stop when the length of the `trip` list reaches five items.

At the end, the program should display the itinerary, i.e. the `trip` list.

Tip: You must indent the lines that will form the `while`-block and will be repeated.

Task 1

```
+ print("City hopping random planner")
+ print("Itinerary:", trip)
+ trip = []
+ trip.append(city)
+ while len(trip) < 5:
```

Remember that these lines are presented in **no particular order**. You will need to decide the order of the lines of code.

Task 1

Here is some example input and output for how this program is **expected** to run.

Example

Note: Use this example to check your program. The actual cities contained in the itinerary are randomly selected and will be different every time the program is executed.

The program displays an initial message.	`City hopping random planner`
After compiling a list of five random cities, the program displays the itinerary:	`Itinerary: ['London', 'Stockholm', 'Moscow', 'Athens', 'Budapest']`

Task 2 No duplicates please

Extend the program so that every `city` selected is promptly **removed** from the list of `cities`. This way, no `city` can be selected **twice** for your trip.

Task 3 Last stop: London

Modify the program so that the process of random `city` selection continues until **London** is selected and added to the `trip`.

Before displaying the itinerary, consider also inserting London at the *beginning* of the trip list, so that the trip appears to be circular.

Task

You will start with an existing program that randomly picks a European capital out of a list and asks the user to guess it. You will extend the program so that it provides the user with hints.

Task

Step 1 - Open this program **oaknat.uk/comp-py-cities-1** in Repl.it

```
1  from ncce.data import cities
2  from random import choice

3  city = choice(cities)                    Pick a city

4  done = False
5  while not done:
6      print("Guess the capital:")          Ask the user
7      guess = input()

8      if guess == city:                    Give feedback
9        print("You've got it!")
10       done = True
11     elif guess == "":
12       print("It was", city)
13       done = True
14     else:
15       print("Try again")
```


Task

Take a look at the program that you have just opened:

Line 1 imports the list of `cities` that the program will use. **Note that this is not a standard Python component.** The list has been created specifically to allow you to perform this activity.

The `choice` function (imported from the `random` module in line 2) is used to randomly select an item out of the list of `cities` (line 3).

Task

Step 2

Run the program a couple of times to see how it currently works. You will find it is very hard for the user to guess the `city`, without any additional information.

Tip: Look at lines 11 to 13. While testing the program you can simply press Enter (without entering a city) and the city will be revealed, terminating the game.

Task

Step 3 - Add the incomplete lines below to your program, right above the `else`-block.

Complete the gaps so that the program reveals the first letter of the `city` to the user, in the case where the `city`'s first letter is different from the first letter of the user's `guess`.

Tip: You can access a specific character in a string just like you can access a specific item in a list, by using the string's name followed by an index in square brackets.

Tip: You must indent these lines properly, as they are part of the `while`-block.

```
+ elif                    :
+   print("The first letter is",              )
  else:
      print("Try again")
```


Task

Here is some example input and output to show how the program **should** run.

Example

Note: This example illustrates how your program should work. The output of your program will depend on the randomly selected city and the user's input, so it will be different each time you execute it.

The program displays a prompt and waits for keyboard input.	`Guess my European capital:`
The user types a reply.	`Prague`
The program displays a hint. The selected city is actually Moscow (different first letter).	`The first letter is M`

Task

Step 4

Add the incomplete lines below to your program, right above the `else`-block.

Complete the gaps so that the program reveals the length (number of characters) in city to the user, in the case where city's length is different from the length of the user's guess.

Tip: Use the `len` function to retrieve the number of characters in `city` and `guess`.

Tip: You must indent these lines properly, as they are part of the `while`-block.

```
+ elif                      :
+     print("It has",              , "letters")
  else:
      print("Try again")
```

Task

Here is some example input and output to show you how your program **should** run:

Example

Note: This example illustrates how your program should work. The output of your program will depend on the randomly selected city and the user's input, so it will be different each time you execute it.

The program displays a prompt and waits for keyboard input.	`Guess my European capital:`
The user types a reply.	`Prague`
The program displays a hint. The selected `city` is actually Paris (same first letter but different length).	`It has 5 letters`

Task

Step 5 - Add the incomplete lines below to your program, right above the `else`-block.

Complete the gaps so that the program reveals the second letter of `city` to the user, in the case where `city`'s second letter is not contained in the user's `guess`.

Tip: Use the in operator to check if a character is contained in a string.

Tip: You must indent these lines properly, as they are part of the while-block.

```
+ elif                        :
+     print("It contains letter",              )
  else:
      print("Try again")
```


Task

Here is some example input and output so that you can see how your program **should** run:

Example

Note: This example illustrates how your program should work. The output of your program will depend on the randomly selected city and the user's input, so it will be different each time you execute it.

The program displays a prompt and waits for keyboard input.	`Guess my European capital:`
The user types a reply.	`London`
The program displays a hint. The selected `city` is actually Lisbon (same first letter, same length, but second letter of `city` is not contained in `guess`).	`It contains letter i`

Explorer Task (Optional)

Extend the program so that it counts the number of attempts the user makes and displays them after the game is over.