Lesson 5: Round and round

Computing

Introduction to Python programming

Rebecca Franks

Countdown to lift off!

Task 1 Countdown

This program seems to be **repeating** the same actions over and over again.

```
1 count = 3
2 print(count)
3 count = count-1
4 print(count)
5 count = count-1
6 print(count)
7 count = count-1
```

Task 1 Countdown

Step 1

In Repl.it, type the **incomplete** program below, which will use `while` to **repeat** the block of actions.

Note: You will not be able to run the program successfully until you fill in the missing condition in the next step.

```
1 count = 3
2 while                :
3     print(count)
4     count = count-1
```

Task 1 Countdown

Step 2

The value of `count` is initialised to 3 and **decreased** by one in every **iteration**.

Fill in the missing **condition** in the `while` loop using one of the options below, so that the last value printed by the program is 1.

A. `while count > 1 :`
B. `while count >= 1 :`
C. `while count < 1 :`
D. `while count == 1 :`

Task 1 Countdown

Syntax checklist

If you encounter an error message, read it and try to fix the problem. Use the list below to check for common errors.

- misspelt `while`
- forgot the colon `:` after the condition in `while`
- forgot to **indent** the statements in the `while` block

Task 2 Lift off!

Step 1

Modify a single line in your current program so that the countdown starts from `10` instead of starting from 3.

Step 2

Insert a single line in your current program so that the message `Lift off!` is displayed when the countdown reaches zero.

Tip

This action needs to be executed after the iteration, so it should not be part of the `while` block. Be careful with indentation.

Task 3 Skip counting upwards

Modify your current program so that it starts from `1` and skip counts over every `3` numbers, until it exceeds `19` (i.e. the program should print `1, 4, 7, 10, 13, 16`, and `19`).

Tip

There are three statements that you will need to modify in your program:

- The assignment that determines where the counting starts
- The assignment that determines how `count` is modified in each iteration
- The condition that determines whether or not the iteration should continue

Tip

If your changes are incorrect, your program may keep displaying values forever! In that case, **terminate** your program (look for a ‘Stop’ button or try pressing Ctrl+C).

Times Table Practice

Worked Example Countdown

The program below displays a sequence of numbers, starting from `10` and counting down to `1`.

```
count = 10
while count >= 1:
  print(count)
  count = count-1
print("Lift off")
```

Worked Example Ten sixes

The program below simulates an experiment in which a dice is rolled repeatedly, until the number six has been rolled **ten** times.

Two counter variables are used: `rolls` keeps track of the total number of dice rolls and `sixes` keeps track of the number of sixes rolled. `rolls` is increased in every iteration, whereas the `sixes` is only increased when a **six** is rolled.

```
1  from random import randint
2  rolls = 0
3  sixes = 0
4  while sixes < 10:
5    dice = randint(1,6)
6    print(dice)
7    if dice == 6:
8      sixes = sixes + 1
9    rolls = rolls + 1
10 print("Ten sixes in", rolls, "dice rolls")
```

Task 1 A practice question

Open the Python program **oaknat.uk/comp-times-tables** in Repl.it using the link. It generates a single random times tables question and checks the user's answer to provide appropriate f

```
1  from random import randint
2  a = randint(2,12)
3  b = randint(2,12)
4  print(a, "times", b, "=")
5  answer = int(input())
6  product = a * b
7  if answer == product:
8    print("That is correct")
9  else:
10   print("I am sorry")
11   print(a, "times", b, "is", product)
```


Task 1 A practice question

Step 1

In order to generate multiple questions, insert all of the statements in the rectangle (lines 2 to 11) into a while statement, so that they are repeated.

Use `True` as the condition in the while statement. This means ‘repeat forever’.

```
while True:
    code for a single question
```


Task 1 A practice question

Step 2

Run your program. It will never stop asking questions, so you will need to terminate it (look for a ‘Stop’ button or try pressing Ctrl+C).

Syntax checklist

If you encounter an error message, read it and try to fix the problem. Use the list below to check for common errors

- misspelt `while` or `True`
- forgot the colon `:` after the condition in `while`
- forgot to **indent** the statements in the `while` block

Task 1 A practice question

Step 3

Introduce a variable called `questions` to keep track of the number of questions that have been posed to the user.

There are two modifications that you will need to make to your program:

- Assign an initial value to questions.
- Increase the value of questions by 1 in each iteration.

To make sure that `questions` is initialised and modified properly, use `print` to display the value of `questions`, anywhere within the `while` block.

```
    print("Question", questions)
```

Task 1 A practice question

Tip

The value of `questions` must be increased in every iteration, so the corresponding statement must be inside the `while` block. **Be careful with indentation**.

Step 4

Modify your program so that it asks exactly **three** questions.

There is only one modification that you will need to make to your program:

- Replace the `True` condition with a condition that checks the value of `questions`. The iteration should only continue if the number of questions posed **does not exceed** three.