

# Python 11: Using for loops

## Teaching Resource

### Resources

- Slides **Python 11: Using for loops**
- You will need to either have Python IDLE installed or have access to an online Python IDE. We have used <https://editor.raspberrypi.org/en>
- Activity worksheets are included in this lesson. You will need to distribute these to your pupils
- We have added a walk-through video below

### Prior Knowledge

Printing, mathematical operators, data types and input using Python

- Data types in Python
- counted `while` loops in Python
- Selection using `if..elif..else` in a `while` loop
- Iteration over lists and strings using a `while` loop



2MB

Python 11 - For loops.pptx

## Lesson walk-through

You need to log in to access videos



## Learning Objectives

To teach pupils to:

- be able to count using a `for` loop
- be able to use a `for` loop to iterate over a string
- be able to use a `for` loop to iterate over a list
- be able to use selection in a `for` loop with a string
- be able to use selection in a `for` loop with a list

## Teacher Note

It is a deliberate decision to avoid the use of the `in` keyword to check if a value exists in a string or list. E.g. `if my_name in students:`

Iteration and selection are fundamental concepts in programming, and it is important that pupils can write the code that underlies this keyword.

# Slide Notes


## Starter

**Slide 2:** Show the code and ask pupils to prepare to explain each line of code. The annotated code follows on slide 3.

## Activity 1: Introducing `for` loops in Python Den

**Slide 5:** Show this slide and ask pupils for the solution. They should find this easy.

**Slide 6:** Show the solution and focus on the generated Python. This is a `for` loop which is a shortcut for a counted `while` loop. **Slide 7** explains how this `for` loop works.

 It is important to emphasise all the things that a `for` loop does automatically. It is not necessary to set `count` to 0 before the loop, and it would be incorrect to increase `count` yourself, the `for` loop does these things for you.

**Slide 8:** Once your pupils are ready, ask them to do levels 41 - 49 in Python Den. There are instructions and hints built into Python Den but some additional hints have been provided on this slide.

## Activity 2: `for` loops in Python

**Slide 10:** The `for` loop in Python is a useful and efficient shortcut to a `while` loop. However, much of what it does is hidden, so it is important to focus on how it works. Show the two loops and give pupils a moment to read the new loop.

**Slide 11:** This slide breaks down how the `for` loop works. Emphasise what is hidden, e.g. the creation of the `count` variable. Pupils should type in the code and try it, and then change it to output 0 to 20. The answer is shown on slide 7.

**Note:** If pupils find the use of the `for` loop confusing, you might like to relate it to the `while` loop.

I.e. these two loops output the same values

```
count = 5
while count < 20:
    print(count)
    count = count + 2

for count in range(5, 20, 2):
    print(count)
```

**Slide 12:** This slide shows how you can specify which value to start from. By default, it starts from 0. Give pupils time to try this.

**Slide 13:** This slide shows how you can use the optional step parameter to specify how much the loop control variable should be increase on each iteration. Give pupils time to try this.

**Slide 14:** This slide shows how you can count from a high number to a low number. The step value must be negative because the `for` loop always adds it. Give pupils time to try this. They should then do the exercises.

**Slide 15:** The exercise answers are shown here. Give pupils time for questions and to correct their answers, if necessary.

## Activity 3: Using for loops with strings

**Slide 18:** This slide shows how you can use the loop control variable as an index for string iteration. Give pupils time to try this. A common error is missing the `range` keyword. For example, this code does not work:

```
phrase = input("Enter a phrase ")
length = len(phrase)
for index in length: # range(length) is correct
    print(phrase[index])
```

**Slide 19:** This slide shows a for-each loop, which is an advanced use of the `for` loop. Because a string is an iterable sequence of characters, the `for` loop will automatically use indexes to get the next character in the string on each iteration. Once pupils have tried this, ask them to do the activities on the worksheet.

**Slide 20:** This slide shows the answer to the exercises. Go through each exercise and allow pupils time to correct their answers, if necessary.

## Activity 4: Using for loops with lists


**Slide 22:** This prediction exercises encourages pupils to engage with the code and use what they have learned so far to make a sensible prediction about some new code. Encourage pupils to come up with an answer; it doesn't matter if they are wrong!

**Slide 23:** Once pupils have made their predictions, allow them to type in the code and run it. They should make a note of the actual output and correct their prediction, if necessary. They should then go on to the exercises.

**Slide 24:** This slide shows the answer to the exercises. Go through each exercise and allow pupils time to correct their answers, if necessary.

## Activity 4: Selection in a for loop

**Slide 26:** Show the code on the screen and ask pupils to suggest how it can be extended to count the number of vowels in an input phrase.

 Depending on your class, you might want to simplify this and just ask them how they could extend the code to count a single letter, e.g. how many times does the letter "a" appear in the input phrase.

Here is the solution to that task:

```

phrase = input("Enter a phrase ")
count = 0 # Create a variable and initialise it to 0
for letter in phrase:
    print(letter)
    if letter == "a": # If the current letter is an a...
        count = count + 1 # add 1 to our count
# Output the count after the loop has finished
print(f"I found {count} occurrences of the letter a")

```

**Slide 27:** This slide shows a loop that iterates over a string "aeiou" and checks to see if the current letter matches each vowel. Ensure that pupils understand this new loop. Then ask them where it should be placed in the code. It must be inside the loop, because that is the point at which we are looking at an individual character. Allow discussion time for this and explore incorrect answers.

**Slide 28:** This slide shows where the code should go. You might like to ask pupils to try this before proceeding to adding counting.

**Slide 29:** Use this slide to explore how counting can be added into the previous code. Allow pupils to try this, perhaps in pairs, before showing the solution.

**Slide 30:** The solution is shown here. Discuss this and allow pupils to complete or correct their code before moving on to the exercises.

**Slides 31–32:** The answers are shown on these slides. Discuss these and allow pupils to complete or correct their code, as necessary.

## Plenary

**Slide 33:** Show this slide and give pupils time to consider the exit ticket.

```

1 phrase = input("Enter a phrase ")
2 count = 0
3 for letter in phrase:
4     for a_vowel in "aeiou":
5         if letter == a_vowel:
6             count = count + 1
7 print(f"I found {count} vowels")

```

There are several ways of doing this:

1. You could change vowels on line 4 to "aeiouAEIOU". However, this is somewhat wasteful as we iterate twice as many times.
2. It would be better to change line 5 to: `if letter.upper() == a_vowel.upper()`  
OR `if letter.lower() == a_vowel.lower()`. In this way, we convert the letter we are looking at and the vowel we are comparing it to the same case.