W04 Learning Activity (1 of 3): While Loops

Overview

Many complex problems can be solved by repeating simple tasks over and over again. For example, image or movie editing software might repeat the same procedure for each frame of the movie or for each pixel of an image. Similarly, programs that model a company's financial data or perform scientific analysis need to examine each data point, one after another.

In this week, you will learn the tools you need to make your programs perform steps many times, and you will apply this by creating an interactive word puzzle.

Loops

An important feature of programs is the ability for the computer to repeat certain steps over and over again. This concept of a *loop* is the topic of this week.

As you might expect, there are many variations of loops depending on how many times or under which conditions the program should repeat certain pieces. In Python, we tell the program to loop by using either the `for` or the `while` keyword, depending on our specific situation.

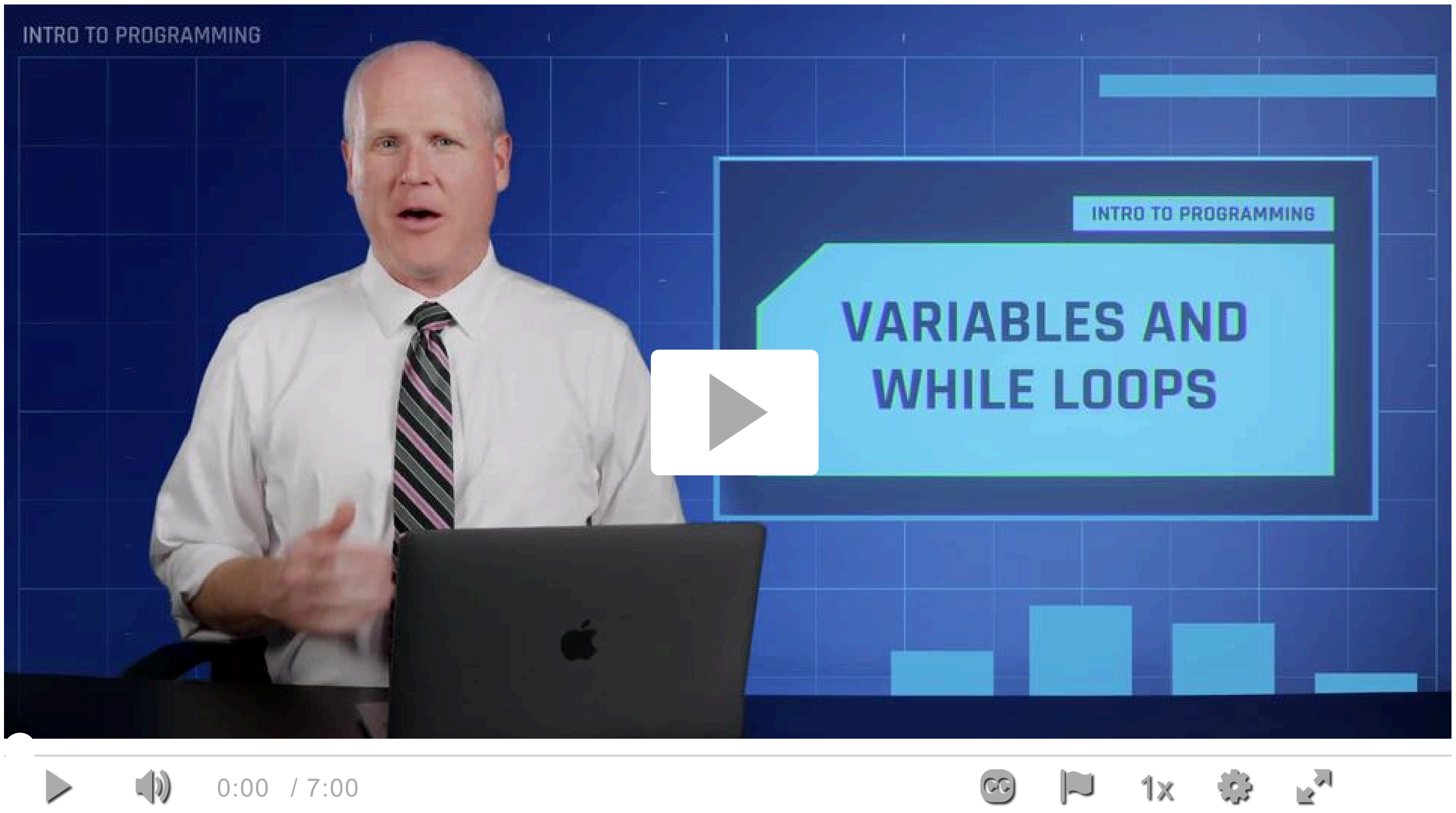
Preparation Material

There are two types of loops in Python, `while` loops and `for` loops. This learning activity will focus on `while` loops and the next learning activity will focus on `for` loops.

Watch the following videos:

Direct link: While Loops

Direct link: [Variables and While Loops]()



A **`while`** loop continues *while* something is still true, or *as long as* it's true, or stated another way, "until it's no longer true."

For example, you might keep asking the user for a number as long as, or *while* they keep typing numbers under 10, and stop as soon as they enter one that is larger than that:

```
number = 0

# Keep looping as long as the number is less than 10
while number < 10:
    number = int(input("What is the number? "))

print("Finished with the loop")
```

The output of this program could be something like the following:

```
What is the number? 3
What is the number? 7
What is the number? 2
What is the number? 24
Finished with the loop
```

This also works when the computer is updating a value, rather than getting input from the user. The following program counts up to 5 and then stops:

```
# Start with the number 1
number = 1

# Keep looping as long as the number is less than or equal to 5
while number <= 5:
    # Display the number
    print(f"The number is: {number}")

    # Update the number to be one more than it was
    number = number + 1

print("Finished with the loop")
```

The output of this program could be something like the following:

```
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
Finished with the loop
```

Declaring variables before they are used

Just like with **`if`** statements, any variables that will be used in the **`while`** expression, need to be declared and assigned values **before** they are referenced in that condition.

In the following example, the program will cause an error because it tries to check of number is less than 10, but the variable does not exist yet.

```
# BAD EXAMPLE: This code does not define a value for the number before

while number < 10: # ERROR, number is not defined yet
    number = int(input("What is the number? "))

print("Finished with the loop")
```

The following example corrects the mistake by declaring the variable and setting it equal to a value prior to the loop.

```
# GOOD EXAMPLE: This code correctly sets the variable to a value before

number = 0

while number < 10:
    number = int(input("What is the number? "))

print("Finished with the loop")
```

In an example such as these, what value should you initialize the variable to? The answer is that you can pick any value that allows the program to enter the loop. In the case above, the loop will run as long as **`number`** is less than 10. So if we started it at something more than 10, it would never run:

```
# BAD EXAMPLE: This code sets the variable to a number that prevents
# the code from ever entering the loop.
```

```
number = 25 # ERROR: This number is greater than 10, so the loop will r

while number < 10: # This body of this loop will NEVER run
    number = int(input("What is the number? "))

print("Finished with the loop")
```

In the following two examples, the mistake is corrected by assigning the variable to a value that allows the loop to run.

```
# OK EXAMPLE: This code sets the variable to a number that allows the l
# But it is not great, because it sets it to a non-standard value of 6.

number = 6 # This number is less than 10, so the loop will run, but it

while number < 10: # This body of this loop will run just fine
    number = int(input("What is the number? "))

print("Finished with the loop")
```

```
# GOOD EXAMPLE: This code sets the variable to a number that allows the
# It uses a standard initialization value of 0.

number = 0 # This number is less than 10, and is a standard value

while number < 10: # This body of this loop will run just fine
    number = int(input("What is the number? "))

print("Finished with the loop")
```

Notice that in the previous two examples, the variable is set to a value less than 10, so the loop will run. Technically, any number less than 10 would work (including negative numbers), but it is common practice to initialize unused integer variables to `0` or `-1`, to initialize strings to `""`, and to initialize boolean variables to `False`.

Variable Scope

Just as you need to declare a variable before the loop if you want to use it in the condition statement, variables that are first declared inside the body of a loop (or an `if` statement) should not be used after the loop. Sometimes programming languages like Python will allow this to work, but it is not considered good practice, because it can cause bugs to arise in your code that are difficult to track down.

```
# BAD EXAMPLE: This code uses the variable "name" outside the loop wher
# it was declared

number = 0

while number < 10:
    number = int(input("What is the number? "))
    name = input("What is your name?")

print(f"Your name is: {name}")
```

```
# GOOD EXAMPLE: This code first declares the variable "name" before the
# that it can be used afterward.

number = 0
name = ""

while number < 10:
    number = int(input("What is the number? "))
    name = input("What is your name?")

print(f"Your name is: {name}")
```

Activity Instructions

Demonstrate your understanding of loops by completing the following individual checkpoint assignment.

Write a Python Program that does each of the following:

1. Use a while loop to ask the user for a positive number (>= 0). Continue asking as long as the number is negative, then display the number. For example:

```
Please type a positive number: -3
Sorry, that is a negative number. Please try again.
Please type a positive number: -8
Sorry, that is a negative number. Please try again.
Please type a positive number: -1
```

```
Sorry, that is a negative number. Please try again.
Please type a positive number: 12
The number is: 12
```

2. Use a **`while`** loop, to simulate a child asking their parent for a piece of candy. Have the program keep looping until the user answers "yes", then have the program output "Thank you." For example:

```
May I have a piece of candy? no
May I have a piece of candy? no
May I have a piece of candy? no
May I have a piece of candy? no
May I have a piece of candy? yes
Thank you.
```

Sample Solution

When your program is finished, please view a sample solution of this program to compare your approach to that one.

You should work to complete this checkpoint program first, without looking at the sample solution. However, if you have worked on it for at least an hour and are still having problems, you may feel free to use the sample solution to help you finish your program.

- Sample solution

Testing Procedure

Verify that your program works correctly by following each step in this testing procedure:

1. For the positive number program, enter several negative numbers, then a positive one.
2. Try entering a positive number first, and ensure that it doesn't ask again.
3. Try entering 0 and ensure that it correctly treats it as a positive number.
4. For the piece of candy program, test your last loop by entering different amounts of no's before finally saying yes.
5. Try saying "yes" the very first time, and ensure that it works as expected.

Submission

When you have completed all of the learning activities for this week, you will return to Canvas and submit the associated quiz there.

Up Next

- W04 Learning Activity (2 of 3): For Loops

Other Links:

- Return to: Week Overview | Course Home