1811/2807/7001ICT
Programming Principles

School of Information and Communication Technology
Griffith University

Trimester 1, 2024

12 Indefinite Loops With while

The first kind of loop we encounter is the `while` loop.

This is Python's indefinite loop.

12.1 Definite versus indefinite

In situations where repetition is required, there are two main cases:

definite loop: the programmer or the program can know when the loop starts how many repetitions (*iterations*) are required; or

indefinite loop: the programmer or the program can't know how many iterations will be required, because it must repeat until some condition is met.

In this section we will deal with the latter case, and in the next section, the former.

12.2 The while statement

In most modern languages, the keyword that introduces a *pre-tested* indefinite loop is `while`.

This template shows the structure of Python's `while` statement.

```
while condition:
    loop body statement(s)
```

The *condition* is an expression that must evaluate to `True` or `False`.

While the *condition* is `True` the *loop body statement(s)* will execute.

Pre-tested means that the *condition* is evaluated *before* the loop body ever executes. So the body may execute zero or more times.

12.3 Examples

Problem: Print the numbers from 1 to 10.

```
# file: while1.py
# Print the numbers from 1 to 10 on one line.

i = 1
while i <= 10:
   print(i, end = " ")
   i = i + 1
print()
```

This requires a variable which takes each of the values to print.

The first time it is given a value is referred to as its *initialisation*.

When its value is increased by one, it is *incremented*.

Problem: Print the numbers from 10 down to 1.

```
# file: while2.py
# Print the numbers from 10 down to 1 on one line.

i = 10
while i >= 1:
   print(i, end = " ")
   i = i - 1
print()
```

Decreasing a variable's value by one is called *decrementing*.

Problem: The magic number guessing game can be improved by looping until the magic number has been guessed, like this.

```
You win if you guess the magic number.
Enter a number: 1
Better luck next time.
Try higher.
Enter a number: 7
Some people think so, but not me.
Better luck next time.
Try lower.
Enter a number: 3
You win!
It is the first odd prime. Yay!
```

Let's start with a *skeleton* with the while loop in place.

```
# file: magic1.py
# The magic number guessing game.

while ????? :
   # body
```

We can copy some of the statements from our old program to make this one. As we do so, we must decide whether the statements are going to be:

- repeated, so they must go in the loop body; or
- executed only once, so they should go before or after the loop.

This has the statements copied into place.

```
print("You win if you guess the magic number.")
while ????? :
   n = int(input("Enter a number: "))
   if n == 7:
      print("Some people think so, but not me.")
   print("Better luck next time.")
   if n < 3:
      print("Try higher.")
   else:
      print("Try lower.")
print("You win!")
print("It is the first odd prime. Yay!")
```

What should the loop condition be?

We know that the program should end when the user enters 3.

So the loop should keep going while the user has entered anything else.

```
print("You win if you guess the magic number.")
while n != 3:
   n = int(input("Enter a number: "))
   if n == 7:
      print("Some people think so, but not me.")
   print("Better luck next time.")
   if n < 3:
      print("Try higher.")
   else:
      print("Try lower.")
print("You win!")
print("It is the first odd prime. Yay!")
```

Try it.

```
$ python3 magic3.py
You win if you guess the magic number.
Traceback (most recent call last):
  File "magic3.py", line 5, in <module>
    while n != 3:
NameError: name 'n' is not defined
$
```

Oops! We tested the value of `n` before it was assigned one.

What value should it have?

It should have the first number entered by the user.

Which means we need to prompt for and read `n` before the loop as well as inside it.

And the prompt and read statement inside the loop needs to be moved to the bottom of the loop.

```
print("You win if you guess the magic number.")
n = int(input("Enter a number: "))
while n != 3:
   if n == 7:
      print("Some people think so, but not me.")
   print("Better luck next time.")
   if n < 3:
      print("Try higher.")
   else:
      print("Try lower.")
   n = int(input("Enter a number: "))
print("You win!")
print("It is the first odd prime. Yay!")
```

```
$ python3 magic4.py
You win if you guess the magic number.
Enter a number: 1
Better luck next time.
Try higher.
Enter a number: 7
Some people think so, but not me.
Better luck next time.
Try lower.
Enter a number: 3
You win!
It is the first odd prime. Yay!
$
```

This is an example of the *sentinel pattern*.

Programs often read inputs until a special value is read that indicates the end of input has been reached, the *sentinel*.

In out case the sentinel is 3.

The pattern can be represented by this template.

```
read an input value
while the value is not the sentinel:
   process the value
   read an input value
```

12.4 Shorthand assignment operators

Python has these shorthand assignment operators that are used most often in loop bodies.

shorthand	*is equivalent to*
`a += b`	`a = a + b`
`a -= b`	`a = a - b`
`a *= b`	`a = a * b`
`a /= b`	`a = a / b`
`a //= b`	`a = a // b`
`a %= b`	`a = a % b`
`a **= b`	`a = a ** b`

Python does not have ++ or --.

12.5 In-class Examples

Solve these problems as a class.

Hint: a negative value makes a good sentinel.

Use shorthand assignment operators where appropriate.

> *Problem:* Prompt for and read marks for a test until a negative number is entered (which can not be a valid mark). Print the number of marks entered and the average (arithmetic mean) of the marks.
>
> *Problem:* Extend the problem so that it prints out the highest and lowest marks as well.

Section summary

This section covered:

- the difference between definite and indefinite loops;
- the `while` statement;
- the sentinel pattern for reading and processing input; and
- shorthand assignment operators.