

Introduction to Programming

Dr. John Stavrakakis

School of Computer Science, University of Sydney



COMMONWEALTH OF AUSTRALIA

Copyright Regulations 1969

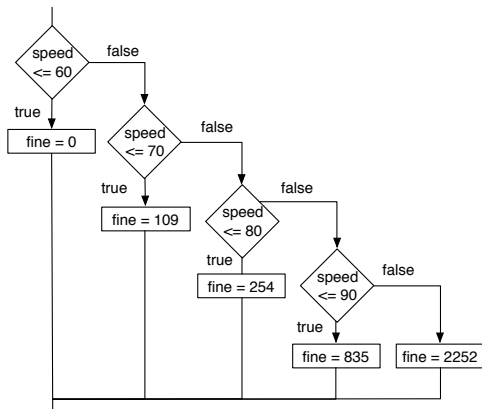
WARNING

This material has been reproduced and communicated to you by or on behalf of the University of Sydney pursuant to Part VB of the Copyright Act 1968 (**the Act**).

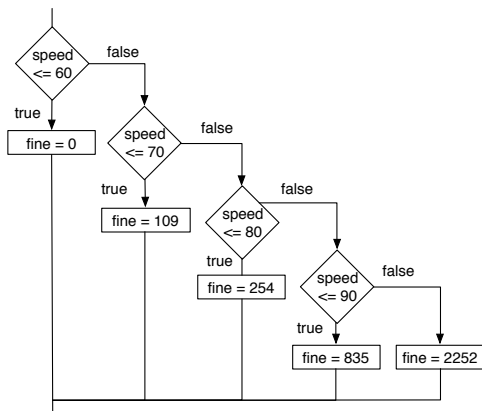
The material in this communication may be subject to copyright under the Act. Any further copying or communication of this material by you may be the subject of copyright protection under the Act.

Do not remove this notice.

Previously in INFO1110...if...else if



How many possible paths of code?



Lecture 6: While loops

Using a while loop; break

“While the *condition* is *true*, repeat what’s in the *statement*.”

Steps of while:

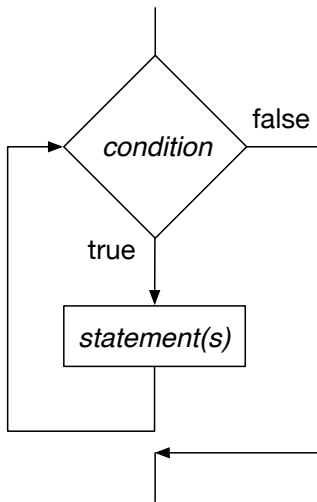
```
1  while (condition) do
2      ·  statement
```

① *condition* is checked: if it is *true* then go on to Step 2

② *statement* is executed, then go on to Step 1

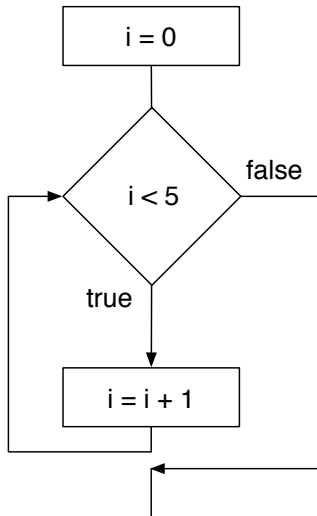
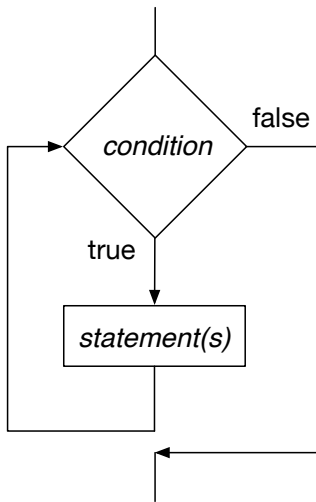
The while loop continues until *condition* is checked at Step 1 is *false*.

while loop diagram



- 1 begin with the *condition*
- 2 check the *condition*
- 3 if it's true, execute the *statement*; otherwise, exit the loop
- 4 go back to the *condition*

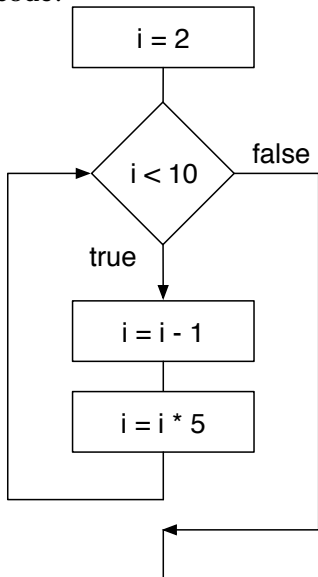
while loop example



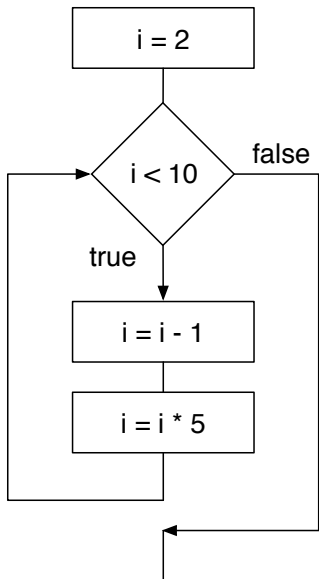
i

while loop diagram to code

code?



while loop desk checking

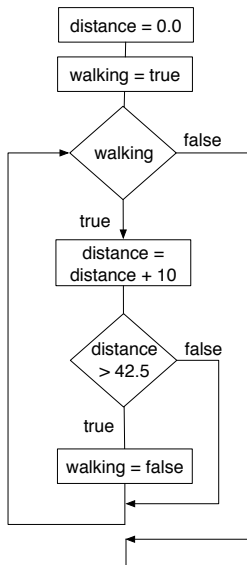


while loop desk checking (cont.)

What is the final value of i?

How many iterations will there be in total?

while loop desk check



distance	walking
0	T

“While the *condition* is *true*, repeat what’s in the *statement*.”

Steps of while:

```
1  while (condition) do
2      ·  statement
```

❶ *condition* is checked: if it is *true* then go on to Step 2

❷ *statement* is executed, then go on to Step 1

The while loop continues until *condition* is checked at Step 1 is *false*.

Python Syntax:

```
while condition :
    statement
```

while example

```
1 bucket_capacity = 5.0
2 bucket_filled = 0.0
3 sand_handful = 0.20
4
5 while bucket_filled < bucket_capacity :
6     bucket_filled = bucket_filled + sand_handful
7     print("bucket_filled = " + str(bucket_filled))
```

How many iterations *do you expect?*

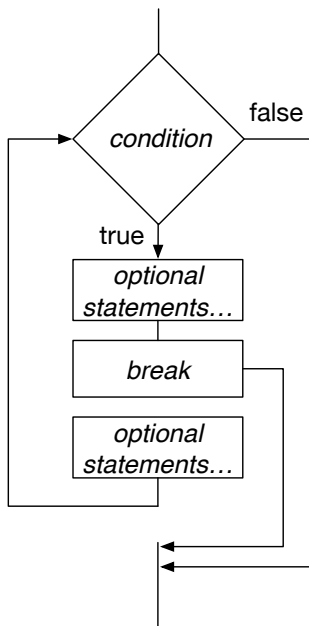
while (true)

while True would imply an infinite loop

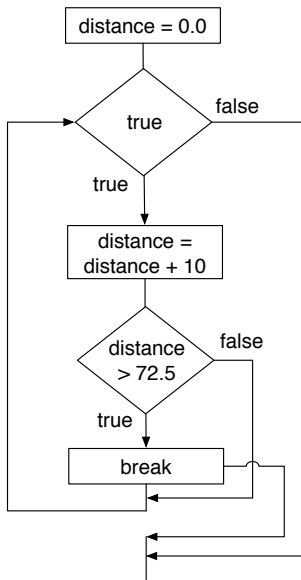
```
1 number = 1
2 while True:
3     print(number)
4     number = number + 1
```

break; is a statement used to leave the body of a loop

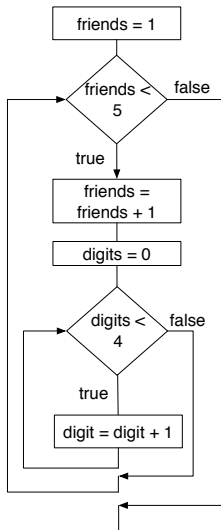
- `break` is a special reserved word
- When used in loops, it means “break from the current iteration of this loop”
- It isn’t allowed in an `if` statement.



break example

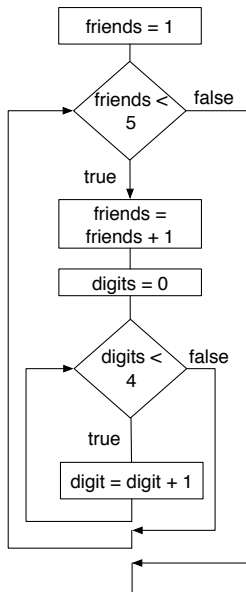


Nested loop example



friends	digits
1	0

Nested loop example



Our programs so far have relied upon the user to enter data manually.

It is very useful to work with files to read in large amounts of data as input

Our UNIX environment is all about files and this will also be important for the upcoming assignment.

Our sample.txt

```
This is my file, the first line  
I have written this on the second line  
last line!
```

Read from a file (cont.)

Our sample.txt

This is my file, the first line
I have written this on the second line
last line!

```
1 with open('sample.txt', 'r') as myfile:  
2     print ( myfile.readline() )  
3     print ( myfile.readline() )  
4     print ( myfile.readline() )
```

Will print each line in the file

Read from a file (cont.)

Take a closer look at **with**

```
1 with open('sample.txt', 'r') as myfile:  
2     # Perform the instructions XYZ
```

The instructions XYZ... will *only* execute if the file was opened *successfully*.

After the instructions XYZ... are completed, the file will be *automatically* closed

How do we read all lines in the file? Loops of course!

```
1 with open('sample.txt', 'r') as myfile:
2     while True:
3         line = myfile.readline()
4         if line == "":
5             break
6         print(line, end='')
```

First problem: we know we need a loop, but how many times?

Second problem: what is the activity that requires repetition? Can we solve this for one case?

Third problem: how do we support the activity that needs repetition?

Write a program to describe a fisherman's catch. Each time a fish is caught it is first weighed, then thrown in a bucket. Your program will keep track of how many fish are caught and the total weight of the bucket. If the bucket exceeds a certain heavy weight the fisherman will stop fishing. The program will display the number of fish caught and the total weight.

programming exercise (cont.)

Iteration in programming means *doing something again and again in a systematic way*.

We use iteration for such things as:

- Scanning through a list of command-line arguments
- Operating on everything in a set
- Processing everything in an array
- Printing shapes in ASCII art

Iteration enables us to set up *arbitrary patterns of repetition*: do something once, a million times, or until a condition is no longer true.