ENGG1811 Computing for Engineers

Week 5A
While

© UNSW, CRICOS Provider No: 00098G1

Iteration (Repetition)

- You have learnt about for-loops. You've used them to:
  - Say G'day to many students
  - Sum a list of numbers, etc.
- Python has another kind of loops called while
  - For is a definite loop, which means fixed number of iterations
  - While is an indefinite or conditional loop. Number of iterations can vary.
- I'd like to have a volunteer to help me to demonstrate these two kinds of loops

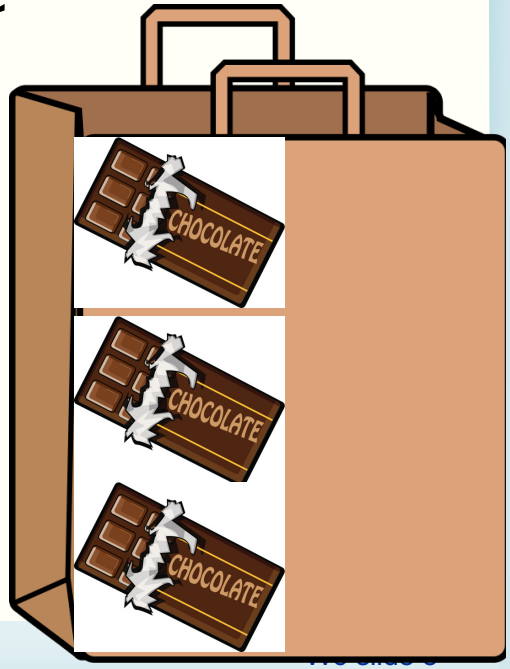
© UNSW, CRICOS Provider No: 00098G

Using “for” to give away chocolates

- The volunteer has 3 chocolates to give away

```
for k in range(3):
        Pick a student
        Give that student a chocolate
```

- The number of iterations is fixed to 3 at the beginning of the for-loop



© UNSW, CRICOS Provider No: 00098G

Giving away an unknown number of chocolates

Note the repetitions!

- The volunteer is given a bag of chocolates. The volunteer needs to give all of them away one by one but **doesn't know** how many there are in the bag

The while-loop

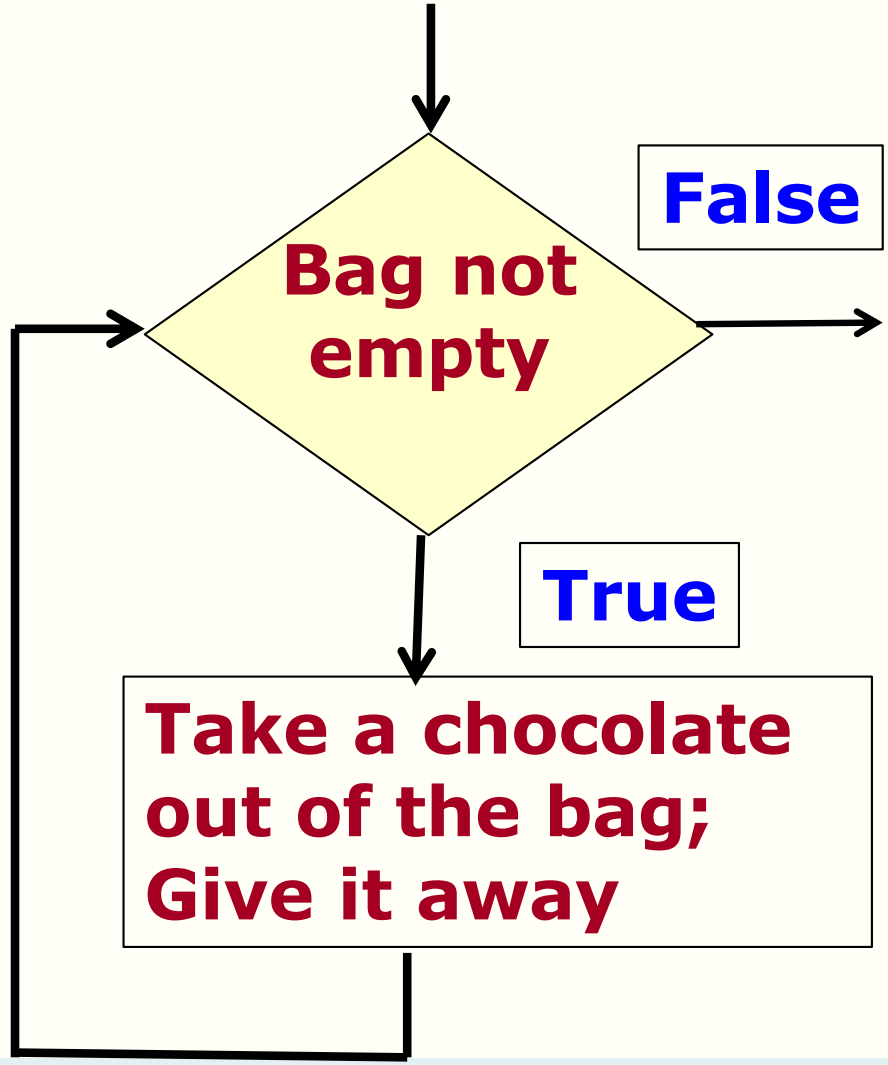
while (the bag is not empty):
    Take a chocolate out of the bag
    Give it away

Comment: num_choc is a variable holding the number of chocolates in the bag

Is num_choc > 0?

    num_choc = num_choc - 1

    Give it away



© UNSW, CRICOS Provider No: 00098G

You use while all the time

while certain condition is true
Do a list of actions

- From a cheesecake recipe: while the mixture is not smooth, keep beating
- Navigator
  - While the destination has not been reached, continue to navigate
- Share on the forum of other real-life examples of ”while” that you can think of

© UNSW, CRICOS Provider No: 00098G

Iteration – While

- `while` statement continues to execute statements as long as a Boolean expression is True

```
while boolean-expression:
    statements
```

Loop guard

Loop body

- Loop guard is evaluated
  - If it is `True` execute the loop body and go back to start of loop to re-test the guard
  - Otherwise (i.e., it is `False`) exit loop and continue with the statement following the loop
- Loop body must change state so that loop guard can eventually become False (else infinite loop)
  - Will discuss this point later

© UNSW, CRICOS Provider No: 00098G

while: example 1

```
LIMIT = 3

x = 1

while x < LIMIT:
    print('x = ',x)
    x = 2 * x
```

Set x = 1 (first power of 2)

Is x less than LIMIT?

Print x

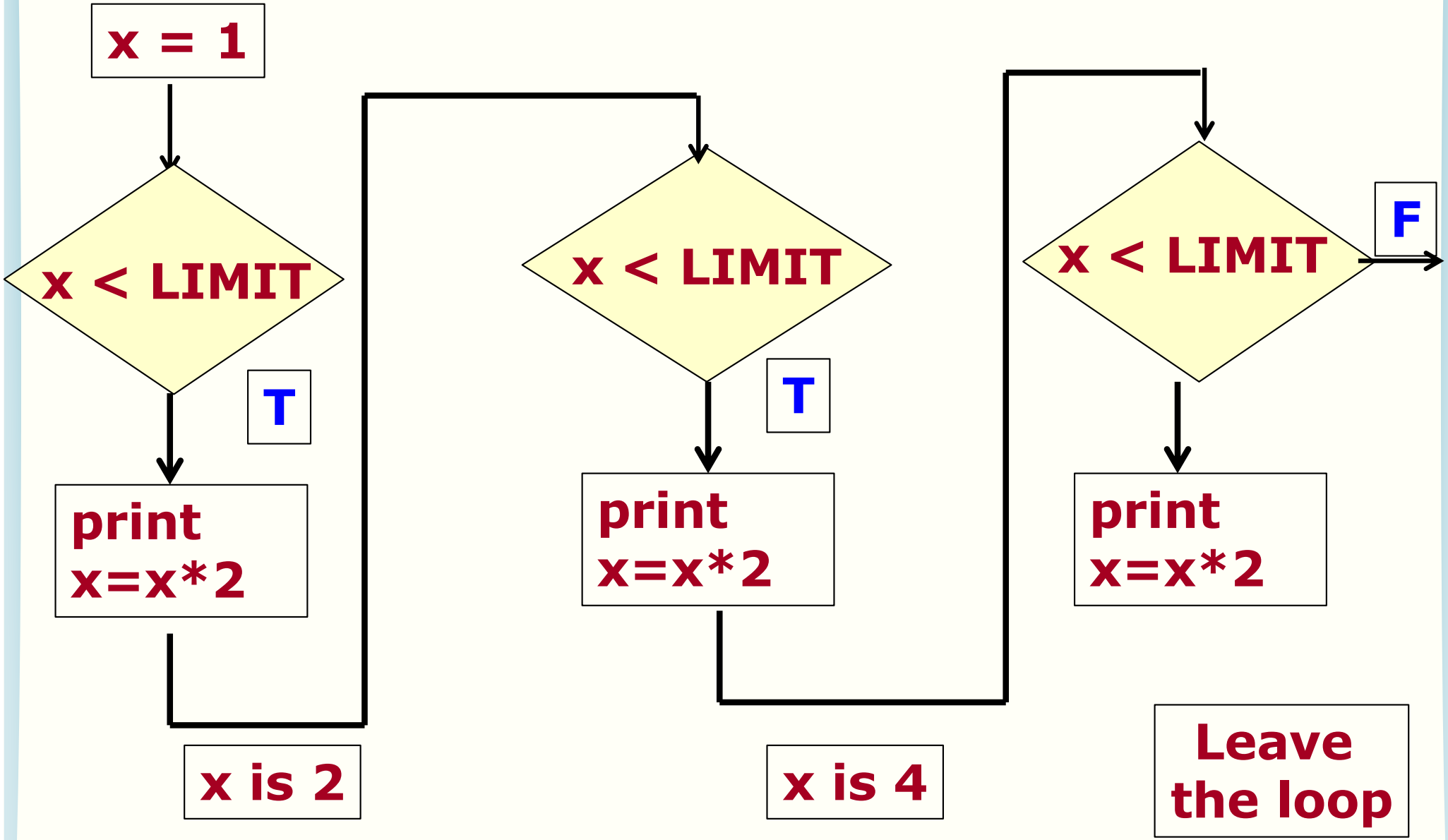
Set x to be 2 * x

Code in while_demo.py

Visualise the code at
http://pythontutor.com

© UNSW, CRICOS Provider No: 00098G

Walking through example 1



© UNSW, CRICOS Provider No: 00098G

Quiz

```
# This is the
# example earlier,
# the outputs are
# 1, 2

LIMIT = 3

x = 1

while x < LIMIT:
    print('x = ',x)
    x = 2 * x
```

Question: What are the outputs of the following program?

```
LIMIT = 8

x = 1

while x < LIMIT:
    x = 2 * x
    print('x = ',x)
```

(a) 1 2 4
(b) 2 4
(c) 2 4 8
(d) 2 4 8 16

© UNSW, CRICOS Provider No: 00098G

Quiz (Answer)

```
x = 1

Check x < 8?
   x = 2
   print 2

Check x < 8?
   x = 4
   print 4

Check x < 8?
   x = 8
   print 8

Check x < 8?
```

Question: What are the outputs of the following program?

```
LIMIT = 8

x = 1

while x < LIMIT:
    x = 2 * x
    print('x = ',x)
```

(a) 1 2 4
(b) 2 4
(c) 2 4 8
(d) 2 4 8 16

© UNSW, CRICOS Provider No: 00098G

How did you do it?

- How did you get the answer from the last question, did you do:
  - Let x be 1
  - Test the while loop guard, double x, print
  - Test the while loop guard, double x, print
  - …
  - Until loop guard fails

- Answer: Yes or No

© UNSW, CRICOS Provider No: 00098G

Different approach to the problem

Question: What are the outputs of the following program?

```
LIMIT = 8

x = 1

while x < LIMIT:
    x = 2 * x
    print('x = ',x)
```

(a) 1 2 4
(b) 2 4
(c) 2 4 8
(d) 2 4 8 16

© UNSW, CRICOS Provider No: 00098G

Quiz

Question: What are the outputs of the following program?

```
LIMIT = 64

x = 1

while x <= LIMIT:
    x = 2 * x
    print('x = ',x)
```

```
(a) 2 4 8 16 32
(b) 2 4 8 16 32 64
(c) 2 4 8 16 32 64 128
```

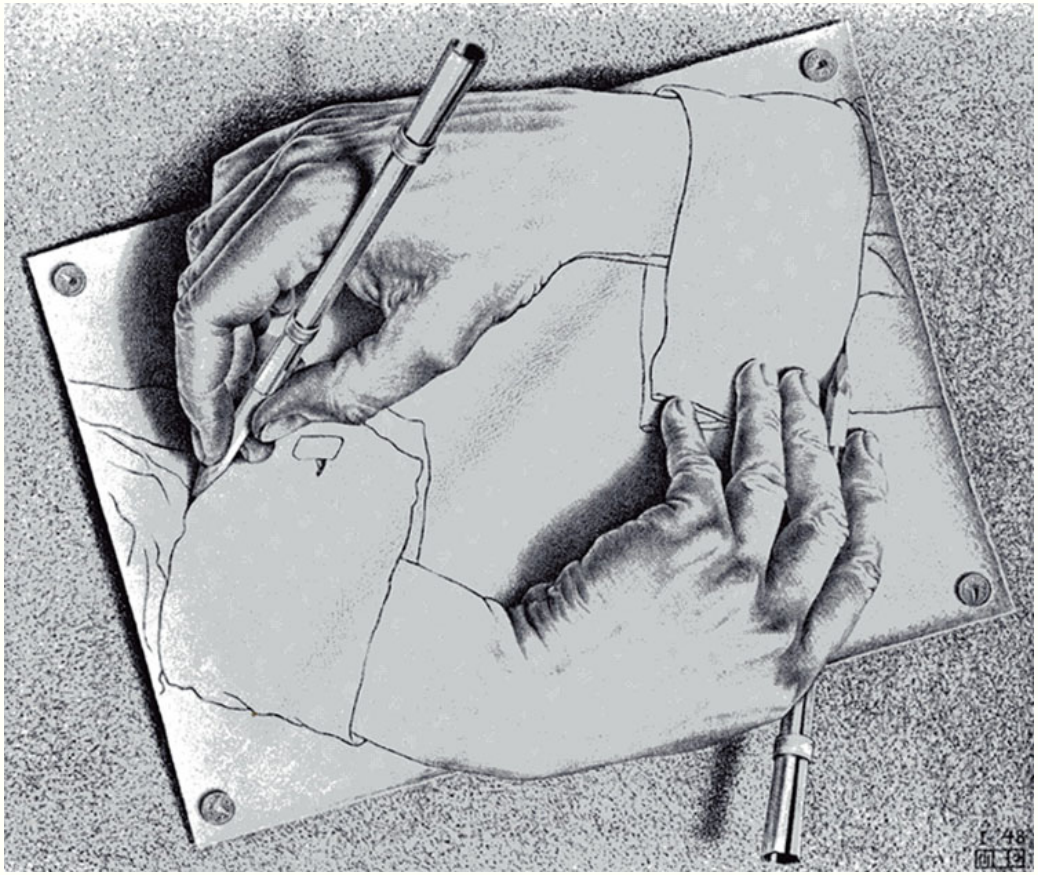
- Think backward: Determine the condition under which the loop guard is valid for the last time
- Is the answer (a), (b) or (c)? Do it without forward tracing.

© UNSW, CRICOS Provider No: 00098G

Iteration – termination

- Loop body must change state so that loop guard can eventually become False

  (otherwise we have produced an *infinite loop*)



MC Escher (1948) *Handteckning* [Drawing Hands]; lithograph

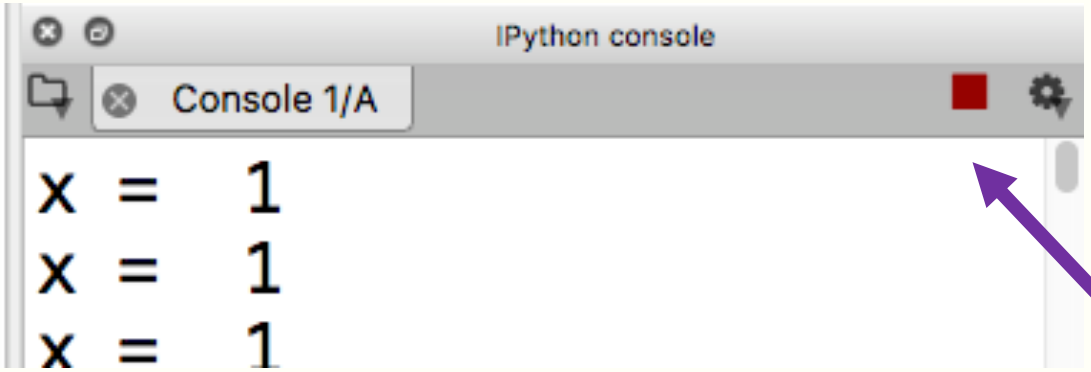
http://cs.nyu.edu/courses/spring04/V22.0002-001/Escher_hands_2.jpg

© UNSW, CRICOS Provider No: 00098G

Infinite loop: demo

```
12 LIMIT = 64
13
14 x = 1
15
16 while x <= LIMIT:
17     x = 2 * x
18     print('x = ',x)
```

- Program in infinite.py
- Let us check that it runs and terminates first
- After that you will comment out Line 17 and run the program
  - You will observe that the program keeps printing x = 1
  - Without line 17, the program won't end



To stop the program, press this button

© UNSW, CRICOS Provider No: 00098G

Infinite loop: demo

```
12 LIMIT = 64
13
14 x = 1
15
16 while x <= LIMIT:
17     x = 2 * x
18     print('x = ',x)
```

- You get an infinite loop if the loop-guard does not become False
- Line 17 in this example is very important because it changes the variable x so that in the end the loop-guard becomes False

© UNSW, CRICOS Provider No: 00098G

Game playing

- We use computers to emulate game playing

- The scenario:
  - Two players: A and B
  - The match is played in rounds
  - Each round can have three outcomes: Either one of the players wins or a tie
  - The winner of each round gets one point, otherwise 0
  - The first player to get 5 points wins the match

- We do not know the number rounds the match will have, that's why we need the while loop

- Code in game.py and play_rock_paper_scissors.py

© UNSW, CRICOS Provider No: 00098G

Determining the winner of a round

- The play() function
  - The code is inside play_rock_paper_scissors.py but let us have a high level look at what it does
  - Returns ’a’, ’b’ if there is a winner; 0 means a tie

```
In [107]: import play_rock_paper_scissors as play_rps

In [108]: winner = play_rps.play(); print(winner)
0

In [109]: winner = play_rps.play(); print(winner)
b

In [110]: winner = play_rps.play(); print(winner)
0

In [111]: winner = play_rps.play(); print(winner)
a
```

The code

```
12 # Constant
13 NUM_POINTS_TO_WIN = 5 # Number of rounds needed to win
14
15 # Initialisation
16 num_rounds_played = 0 # To count the number of rounds played
17 points_a = 0 # To count A's points
18 points_b = 0 # To count B's points
19
20 # Use while to continue to play until
21 # one player has won
22 while points_a < NUM_POINTS_TO_WIN and \
23       points_b < NUM_POINTS_TO_WIN:
24     # play one round
25     winner = play_rps.play()
26     print('Outcome of round',num_rounds_played,':',winner)
27     # Increment the number of rounds played
28     num_rounds_played += 1
29     # Keep track of the score
30     if winner == 'a':
31         points_a += 1
32     elif winner == 'b':
33         points_b += 1
```


© UNSW, CRICOS Provider No: 00098G

Logical condition for the loop guard

- `While` statement continues to execute statements as long as a Boolean expression is True

```
while boolean-expression:
    statements
```

Loop guard

Loop body

- A situation that often arises is that you know the condition to stop the iteration, this means the loop guard that you need is:

  not (condition to stop the iteration)

© UNSW, CRICOS Provider No: 00098G

Example

- The game ends if

```
points_a >= 5 or points_b >= 5
```

- The game continues if

```
not(points_a >= 5 or points_b >= 5)
```

- By De Morgan's Law, this is equivalent to:

```
not(points_a >= 5) and not(points_b >= 5)
```

```
points_a < 5 and points_b < 5
```


© UNSW, CRICOS Provider No: 00098G

While exercise

- In this exercise, you will be using while with a list
- You can assume that there is always a negative number in the list
- Your code should print out all the numbers before the first negative number in the list
- For example, If you are given the list [1,3,6,-2,7,8,-9,4] then your code should print out 1,3,6
- You can use the code while_exercise_prelim.py to get started

© UNSW, CRICOS Provider No: 00098G

for versus while

- The program for_while.py contains the following code. They do the same job: print the numbers in a list one by one
- Which program is simpler?

```
8  # %% Using for
9  num_list = [5, 7, 9, 11]
10
11 for num in num_list:
12     print(num)
13
```

```
14 # %% using while
15 num_list = [5, 7, 9, 11]
16
17 index = 0
18 while index < len(num_list):
19     print('index =',index)
20     print(num_list[index])
21     index = index + 1
```

- Note: Line 19 is added so that we can see the variable index changing in each iteration. It's not needed for the program to work.

© UNSW, CRICOS Provider No: 00098G

For – known number of iterations

- Marge is going on holiday for 5 days
- The number of days is known in advance

Picture: http://cinema.theiapolis.com/movie-2UGY/the-simpsons-movie/gallery/homer-and-marge-simpson-1089622.html

While – unknown number of iterations

- Marge is going on holiday but doesn't know for how long

The break statement

- Instead of using the loop guard to break from a while loop, you can use the break statement

- When a break statement is encountered, the program will leave the loop and execute the first statement outside the while loop
- You can also use break with a for-loop

```
while test expression:
    # codes inside while loop
    if  condition:
        break
    # codes inside while loop

# codes outside while loop
```

```
for var in sequence:
    # codes inside for loop
    if  condition:
        break
    # codes inside for loop

# codes outside for loop
```

Picture from https://www.programiz.com/python-programming/break-continue

© UNSW, CRICOS Provider No: 00098G

The break statement (Example)

- Let us assume that you are given a list of numbers. Your aim is to write a piece of code which does the following:
  - It prints out all the numbers before the first negative number in the list.
  - If the list has no negative numbers, it prints all the numbers out.

- If the list is [1,3,6,-2,7,8,-9,4], the program should print put 1, 3, 6
- If the list is [7,5,6,4], the program should print out all the four numbers

- Code in break_and_loops.py

© UNSW, CRICOS Provider No: 00098G

Additional examples of while

- The continue statement
  - https://www.programiz.com/python-programming/break-continue
- You can use while with else, see:
  - https://www.python-course.eu/python3_loops.php
- An example of using break to check whether an integer is a prime number
  - https://www.programiz.com/python-programming/examples/prime-number

© UNSW, CRICOS Provider No: 00098G

Summary

- The while loop
  - Writing while loop
  - Reasoning with while loop
- Using break

© UNSW, CRICOS Provider No: 00098G