PYTHON: CONTROL STRUCTURES I

Objectives

- To understand the programming pattern simple/two-way/multiway decision and its implementation using a Python `if/if-else/if-elif-else` statement.
- To understand the concept of Boolean expressions and the `bool` data type
- To be able to read, write, and implement algorithms that employ decision structures, including those that employ sequences of decisions and nested control structures
- To understand the concepts of definite and indefinite loops as they are realized in the Python `for` and `while` statements.

Control structures

- Sequential control vs Selection control vs Iterative control
- A **control statement** is a statement that determines the control flow of a set of statements
- A **control structure** is a set of statements and the control statements controlling their execution
  - Three fundamental forms of control in programming are
    - Sequential
    - Selection
    - Iteration

Simple decisions

- For example,

```
if n < 1:
  print("Your input number is too low.")
if n > 10000:
  print("Your input number is too large.")
```

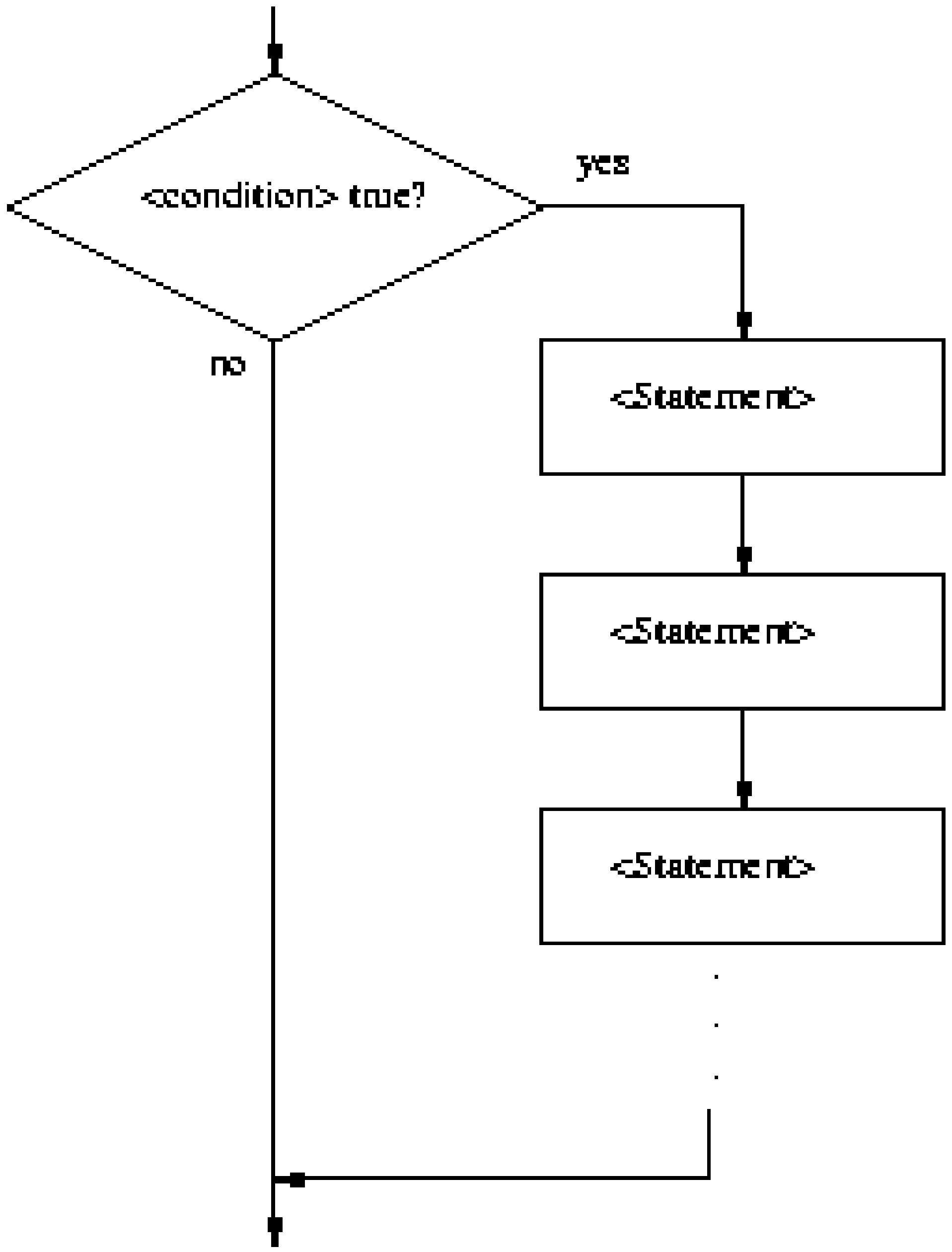
- A relational statement gives either `True` or `False` (Python's keywords)
- Try
  - `print(int(True))`
  - `print(int(False))`

To do or not to do

- The Python `if` statement is used to implement the decision.

```
    if <condition>:
        <body>
```

- The body is a sequence of one or more statements indented under the `if` heading
- The body is executed if `condition` is evaluated to `True`
- The body is skipped if `condition` is evaluated to `False`


<condition> true?
yes
no
<Statement>
<Statement>
<Statement>
.
.
.

Boolean Expressions (Conditions)

- The **Boolean data type** contains two Boolean values, denoted as True and False in Python
- A **Boolean expression** is an expression that evaluates to a Boolean value
- Need a **relational operator** to evaluate a boolean expression
- The relational operators on the next slide can be applied to any set of values that has an ordering
  - Number comparison
  - **Lexicographical ordering** for string comparison

Relational Operators

Python	Mathematics	Meaning
<	<	Less than
<=	≤	Less than or equal to
==	=	Equal to
>=	≥	Greater than or equal to
>	>	Greater than
!=	≠	Not equal to

EXERCISE 3.1

Try

```
x, y ,z = 1, 2, 3
x == y; x > y; x + y + z > 10
False = True; False == True; False > True;
True > False
```

EXERCISE 3.2

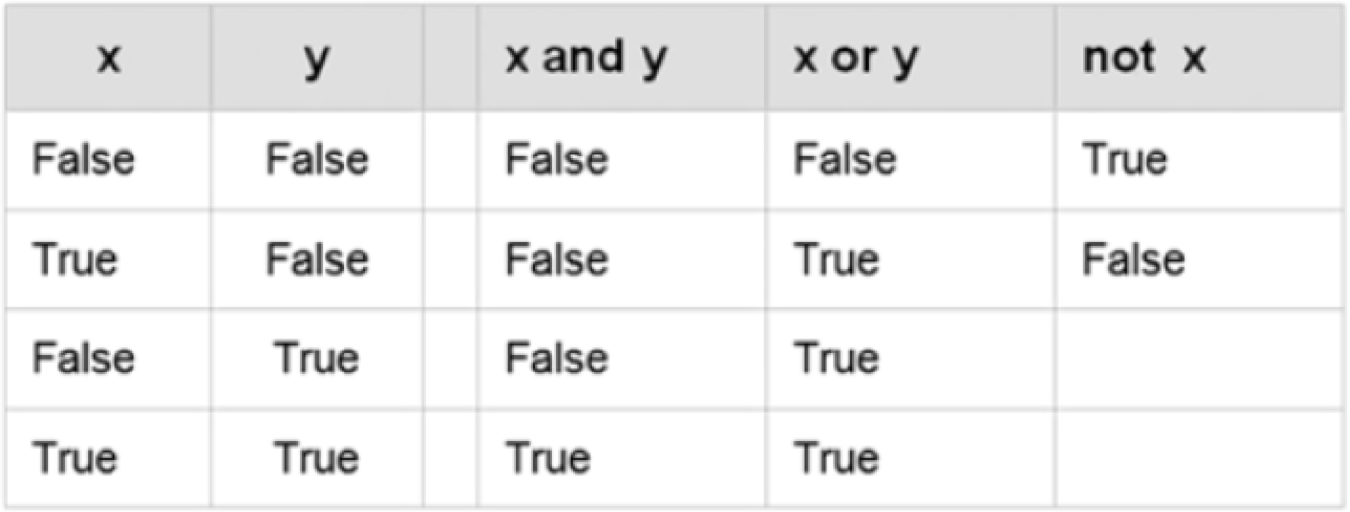
Try

```
x, y ,z = "Benjamin", "Ben", "ben"
x > y; y > z; y + z > x
"abc" > "123"; "abc" > ">>>"
```

Two other membership operators

- The `in` operator is used to determine if a specific value is in a given list, returning True if found, and False otherwise.
- The `not in` operator returns the opposite result.
- Try
    - `10 in [10, 20, 30, 40]`
    - `10 not in [10, 20, 30, 40]`
    - `"blue" in ["red", "yellow", "black"]`
    - `"blue" not in ["red", "yellow", "black"]`
    - `"o" in "peter paul and mary"`

Boolean Operators



x	y		x and y	x or y	not x
False	False		False	False	True
True	False		False	True	False
False	True		False	True	
True	True		True	True	

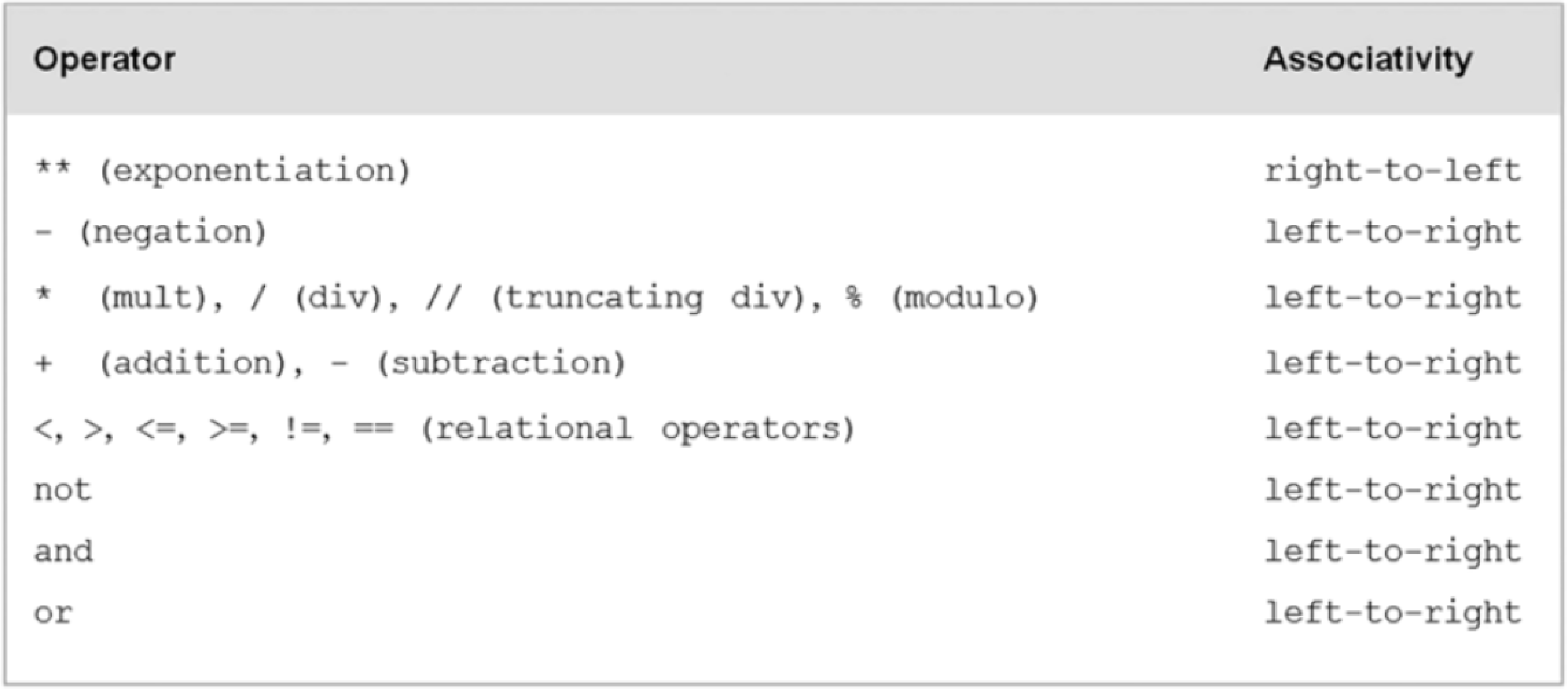
Source: Charles Dierbach. 2013. Introduction to Computer Science Using Python. Wiley.

EXERCISE 3.3

Try

- `True and False; True or False`
- `not (True) and False; not (True and False)`
- `(10 < 0) and (10 > 2); (10 < 0) or (10 > 2)`
- `not (10 < 0) or (10 > 2); not (10 < 0 or 10 > 2)`

Operator Precedence



Operator	Associativity
`**` (exponentiation)	right-to-left
`-` (negation)	left-to-right
`*` (mult), `/` (div), `//` (truncating div), `%` (modulo)	left-to-right
`+` (addition), `-` (subtraction)	left-to-right
`<`, `>`, `<=`, `>=`, `!=`, `==` (relational operators)	left-to-right
not	left-to-right
and	left-to-right
or	left-to-right

Source: Charles Dierbach. 2013. Introduction to Computer Science Using Python. Wiley.

EXERCISE 3.4

Try to figure out the answers before running them:

- `True or False and True and True`
- `True or not False and True and not True`
- `((True or not False) and True) and not True`

Two-Way Decisions

- In Python, a two-way decision can be implemented by attaching an `else` clause onto an `if` clause.
- This is called an `if-else` statement:

```
if <condition>:
    <statements>
else:
    <statements>
```

- E.g.,

```
if 1 <= n and n <= 10000:
    print("Your input number is " + str(n) + ".")
else:
    print("Your input must be between 1 and 10000.")
```

Multi-Way Decisions

- Use nested `if-else` statement to implement multi-way decisions.

-
  ```
  if <condition1>:
      <case1 statements>
  ```

-
  ```
  else:
      if <condition2>:
          <case2 statements>
      else:
          if <condition3>:
              <case3 statements>
          else:
              <default statements>
  ```

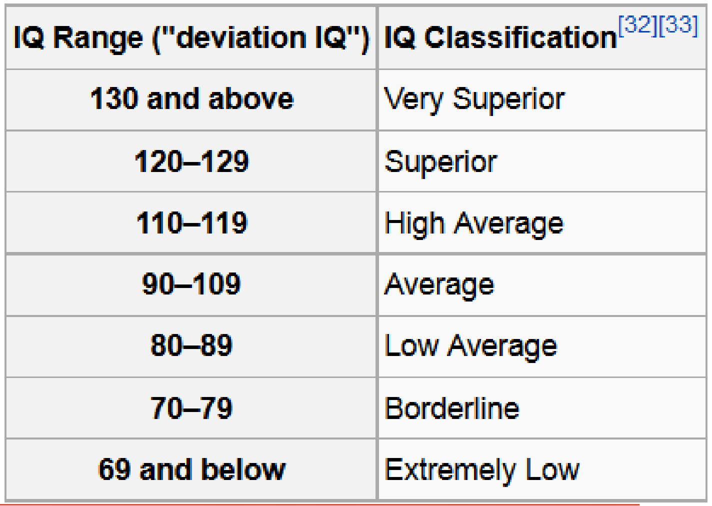
Multi-Way Decisions using `elif`

- In Python, `else-if` can be combined into `elif`:

```
if <condition1>:
   <case1 statements>
elif <condition2>:
   <case2 statements>
elif <condition3>:
   <case3 statements>
else:
   <default statements>
```

- The `else` is optional. If there is no `else`, it is possible that no indented block would be executed.

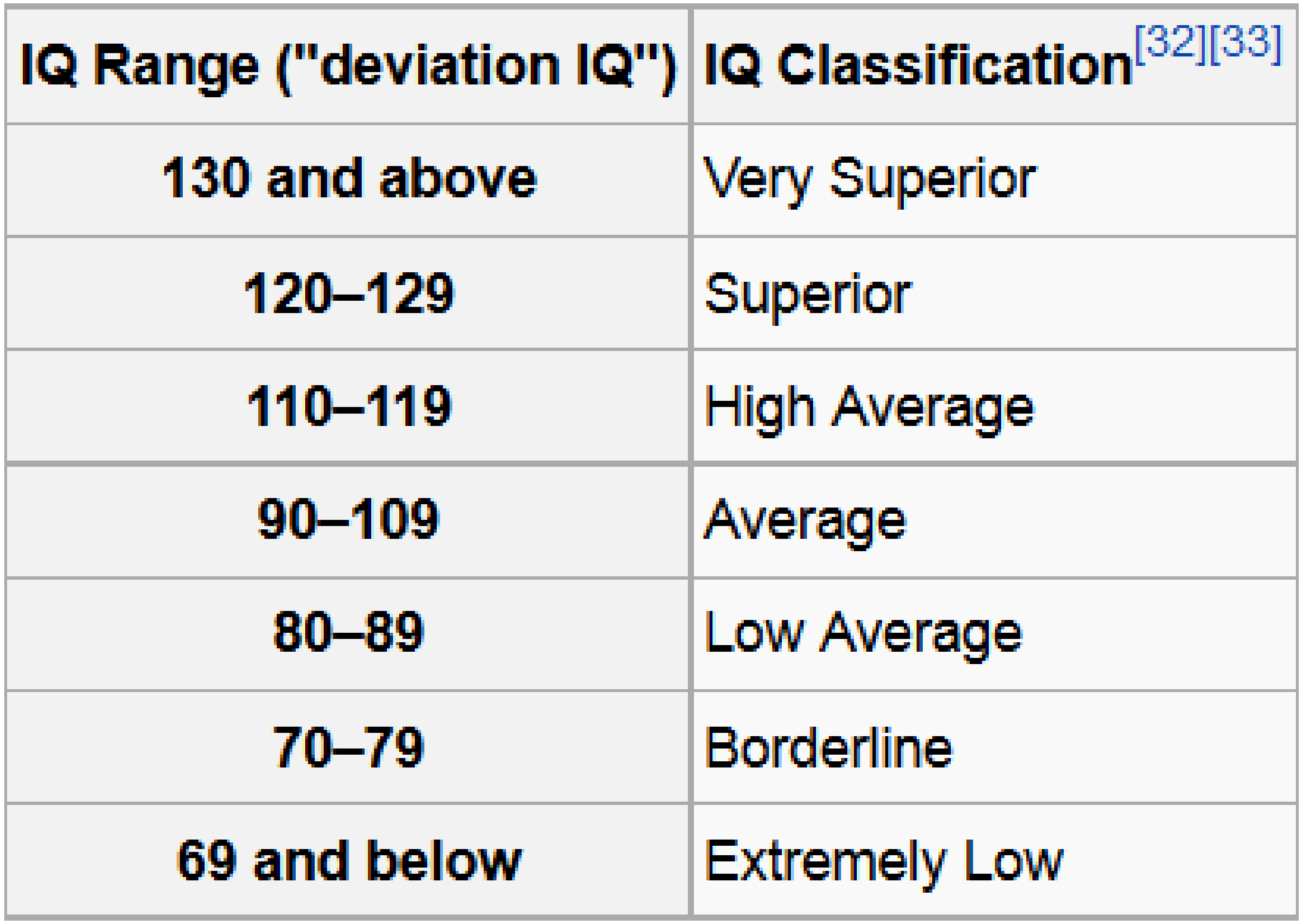
EXERCISE 3.5



IQ Range ("deviation IQ")	IQ Classification[32][33]
130 and above	Very Superior
120–129	Superior
110–119	High Average
90–109	Average
80–89	Low Average
70–79	Borderline
69 and below	Extremely Low

Write a program to ask for a user's IQ and use `if-elif` statements to print out the IQ classification according to the table on the next slide.

```
if <condition1>:
   <case1 statements>
elif <condition2>:
   <case2 statements>
elif <condition3>:
   <case3 statements>
else:
   <default statements>
```



IQ Range ("deviation IQ")	IQ Classification[32][33]
130 and above	Very Superior
120–129	Superior
110–119	High Average
90–109	Average
80–89	Low Average
70–79	Borderline
69 and below	Extremely Low

Source: http://en.wikipedia.org/wiki/IQ_classification

EXERCISE 3.6

Ask user to enter 3 numbers. Your program will output the numbers in descending order. Write down the steps (pseudocode to solve the problem) before coding.

Note that no sorting algorithm is required.

For Loops

- The `for` statement allows us to iterate through a sequence of values.

-
  ```
  for <var> in <sequence>:
     <body>
  ```
- The loop index variable `var` takes on each successive value in the sequence, and the statements in the body of the loop are executed once for each value.
- For example,

  -
    ```
    for num in range(0, 10):
        print(num)
    ```

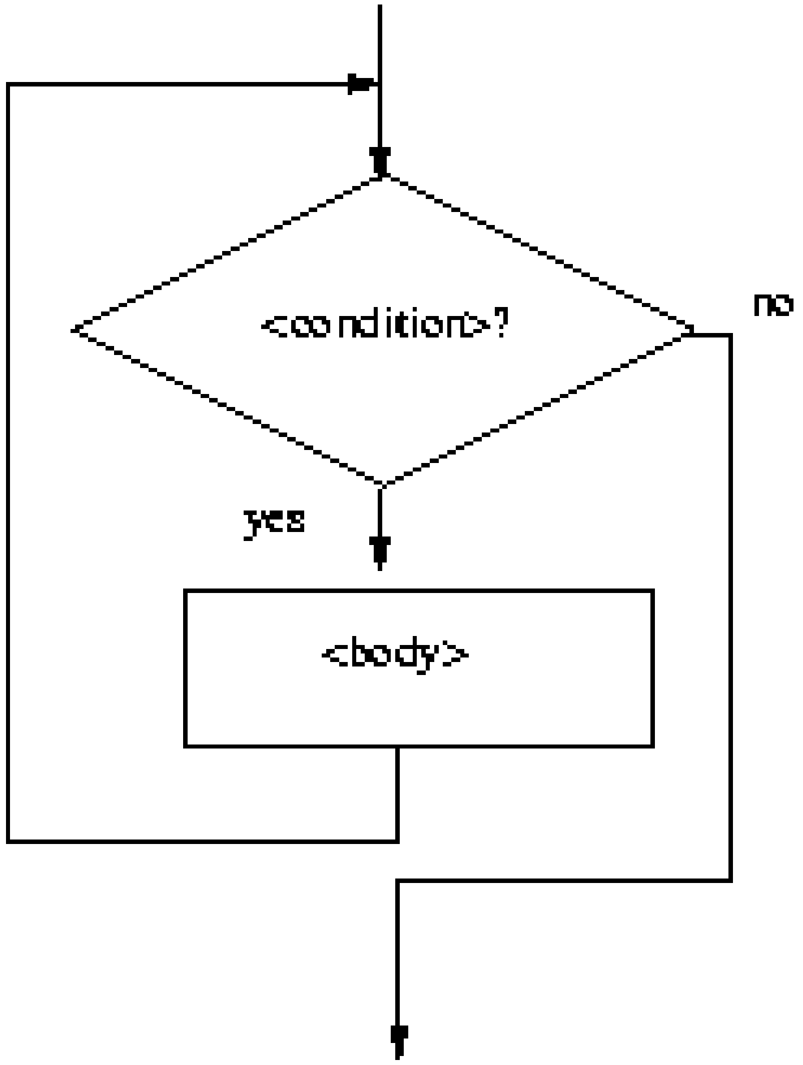
Indefinite Loops

- The `for` loop is a definite loop, meaning that the number of iterations is determined when the loop starts.
- We can't use a definite loop unless we know the number of iterations ahead of time.
- The *indefinite* or *conditional* loop keeps iterating until certain conditions are met.

Indefinite Loops

- `while <condition>:`
  ```
      <body>
  ```
- `condition` is a Boolean expression, just like in `if` statements. The body is a sequence of one or more statements.
- Semantically, the body of the loop executes repeatedly as long as the condition remains true. When the condition is false, the loop terminates.

A Pre-test Loop



An example

- Here's an example of using a `while` loop to print out range(10).

```
i = 0
while i <= 9:
    print(i)
    i = i + 1
```

- The code has the same output as this `for` loop:

```
for i in range(10):
    print i
```

EXERCISE 3.7

Use a `while` loop to print out range(2, 10, 2).

Note: The 3rd argument of range() here means each time the number is incremented by 2.

EXERCISE 3.8

Try

```
i = 0
while i <= 9:
    print(i)
```

Interactive Loops

```
# average2.py
# A program to average a set of numbers
# Illustrates interactive loop with two accumulators

moredata = "yes"
sum = 0.0
count = 0
while moredata[0] == 'y':
     x = int(input("Enter a number >> "))
     sum = sum + x
     count = count + 1
     moredata = input("Do you have more numbers (yes or no)? ")

print("\nThe average of the numbers is", sum / count)
```

EXERCISE 3.9

One problem with the program is that if you type too fast, you may enter a number as an answer to the yes/no question. What will the program respond to this "error"? How will you modify the program to rectify the response?

END