

COMP1730/COMP6730

Programming for Scientists

Control, part 2: Iteration



Outline

- * Iteration: The `while` statement
- * Simulations.
- * Common problems with loops.



Iteration

Program control flow

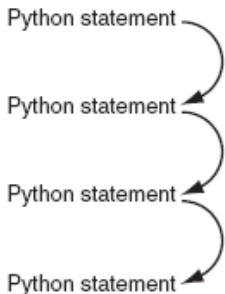


FIGURE 2.1 Sequential program flow.

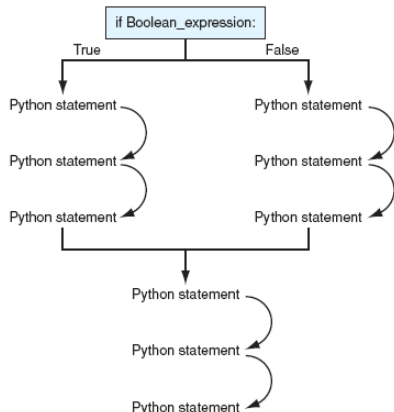
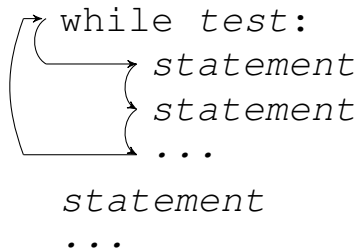


FIGURE 2.2 Decision making flow of control.

Images from Punch & Enbody

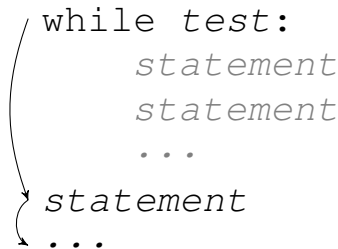
Iteration

```
while test:  
    statement  
    statement  
    ...  
statement  
...
```

A diagram illustrating a while loop iteration. A large curved arrow on the left side of the code block points from the bottom back to the top, indicating the repetition of the loop body.

UNTIL

```
while test:  
    statement  
    statement  
    ...  
statement  
...
```

A diagram illustrating an until loop iteration. A large curved arrow on the left side of the code block points from the bottom back to the top, indicating the repetition of the loop body.

- * Iteration *repeats* a suite of statements.
- * A test is evaluated before each iteration, and the suite executed (again) if it is true.

Iteration statements in python

- * The `while` loop repeats a suite of statements as long as a condition is true.
- * The `for` loop iterates through the elements of a collection or sequence (data structure) and executes a suite once for each element.
 - We'll come back to the `for` loop later in the course.

The `while` loop statement

```
while test_expression :  
    suite  
statement(s)
```

1. Evaluate the test expression (converting the value to type `bool` if necessary).
2. If the value is `True`, execute the suite once, then go back to **1**.
3. If the value is `False`, skip the suite and go on to the following statements (if any).

Suites (reminder)

- * A *suite* is a (sub-)sequence of statements.
- * A suite must contain at least one statement!
- * In python, a suite is delimited by indentation.
 - All statements in the suite **must be preceded by the same number of spaces/tabs** (standard is 4 spaces).
 - The indentation depth of the suite following `if / else / while` : must be greater than that of the statement.
- * A suite can include nested suites (`if`'s, etc).

Variable assignment (reminder)

- * A variable is a name that is associated with a value in the program.
- * Variable assignment is a statement:

var_name = expression

- Note: Equality is written `==` (two `=`'s).
- * A name–value association is created by the *first* assignment to the name;
- * *subsequent* assignments to the same name *change* the associated value.

```
→ 1 an_int = 3 + 2  
→ 2 an_int = an_int * 5
```

```
1 an_int = 3 + 2  
→ 2 an_int = an_int * 5
```

Global frame

an_int 5

Global frame

an_int 25

* For example,

```
an_int = 2 + 3
```

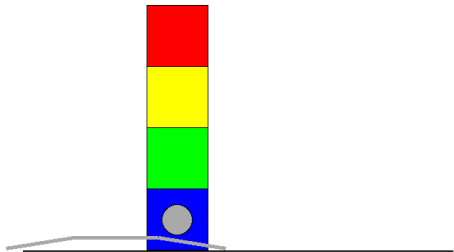
(From pythontutor.com)

```
an_int = an_int * 5
```

1. Evaluate expression `2 + 3` to 5.
2. Set value of `an_int` to 5.
3. Evaluate expression `an_int * 5` to 25.
4. Set value of `an_int` to 25.

Problem: Counting boxes

- * How many boxes are in the stack from the box in front of the sensor and up?

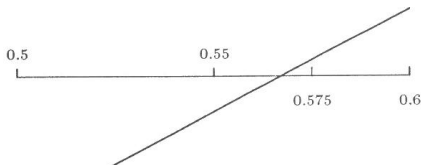
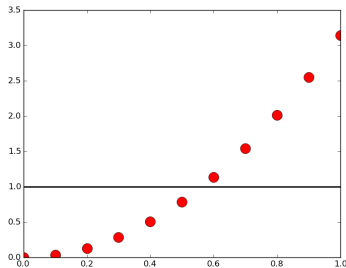


- * While `robot.sense_color() == ' '`, move the lift up, *and count how many times*; then move the lift down that many times.

```
def count_boxes():
    num_boxes = 0
    while robot.sense_color() != '':
        num_boxes = num_boxes + 1
        robot.lift_up()
    steps_to_go = num_boxes
    while steps_to_go > 0:
        robot.lift_down()
        steps_to_go = steps_to_go - 1
    return num_boxes
```

Problem: Solving an equation

- * Solve $f(x) = 0$.
- * The interval-halving algorithm:
 - if $f(m) \approx 0$, return m ;
 - if $f(m) < 0$, set l to m ;
 - if $f(m) > 0$, set u to m .



return from a loop

- * A loop (`while` or `for`) can appear in a function suite, and a `return` statement can appear in the suite of the loop.

```
def find_box(color):  
    while robot.sense_color() != '':  
        if robot.sense_color() == color:  
            return True  
        robot.lift_up()  
    return False
```

- * Executing the `return` statement ends the function call, and therefore also exits the loop.

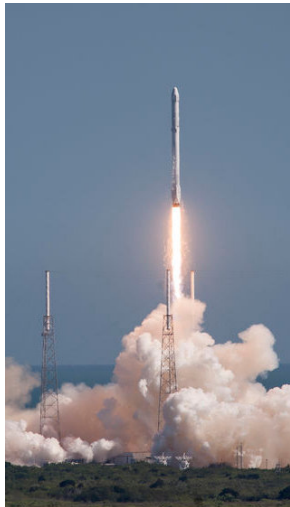


Simulation

Problem: How high does the Falcon 9 fly?

- * Acceleration is thrust (force) divided by mass.
- * 90%–96% of mass is fuel.
- * Rocket's engines have about 7.5% more thrust in vacuum than at sea level.

Image by SPACEX



Simulation

- * Approximate the evolution of a complex of coupled processes.
- * Simulate time by small steps (δt):
 - At each step, compute the change in each variable over δt using the current values of other variables.

Example: Rocket simulation

- * Altitude (a): $\delta a = v \cdot \delta t$
- * Velocity (v): $\delta v = \text{acceleration} \cdot \delta t$
- * acceleration = $(\text{thrust}(a)/m) - g$
 - assuming thrust(a) grows linearly between sea level pressure and vacuum (probably wrong).
- * Mass (m):
 - at time 0, $m = \text{take-off weight}$.
 - $\delta m = -B \cdot \delta t$.
 - burn rate $B = \text{take-off fuel weight} / \text{burn time}$.

Example: The Competitive Lotka-Volterra model of ecology

- * The change in the population of species i is

$$\delta x_i / \delta t = r_i x_i \left(1 - \left(\frac{x_i + \sum_{j \neq i} a_{ij} x_j}{K_i} \right) \right)$$

where

- r_i is the inherent growth rate of species i ;
- a_{ij} is the (negative) effect of species j on species i ;
- K_i is the population of species i that the environment can support (“carrying capacity”).



Writing and debugging loops

Repeat while condition is true

- * A `while` loop repeats as long as the condition (test expression) evaluates to `True`.
- * If the condition is initially `False`, the loop executes zero times.
- * If no variable involved in the condition is changed during execution of the suite, the value of the condition will not change, and the loop will continue forever.

Common problems with `while` loops

- * Loop never starts: the control variable is not initialised correctly.

```
# find smallest divisor of num:  
i = 1  
while num % i != 0:  
    i = i + 1
```

- `num % 1` is always 0!

Common problems with `while` loops

- * Loop never ends: the control variable is not updated in the loop suite, or not updated in a way that can make the condition false.

```
i = 0
while i != stop_num:
    i = i + step_size
```

- What if `stop_num < 0`?
- or `step_size < 0`?
- or `step_size` **does not divide** `stop_num`?