Chapter 7 — In, If, Random, and While

Introduction

In the previous chapter we learned about what a Boolean value is. In this chapter we will see how to make the program more dynamic by skipping blocks of statements depending on Boolean values, and random conditions. Creating optional code in a program is known as selection. Also, this chapter will show how to handle run time errors and introduce random numbers.

Objectives

Upon completion of this chapter's exercises, you should be able to:
- Use the in operator to see if a list or tuple contains a value.
- Include the 'Random' module in your Python programs.
- Generate random integers, random floating-point numbers, and shuffle lists using the Random module.
- Create complex if statements that use Boolean expressions to conditionally perform suites of code.
- Develop exception handling suites to capture run-time errors and perform special suites if code when they are thrown.
- Construct loops using the while structure.
- Include the 'sys' module in your programs and use the exit method to terminate a program early.

Prerequisites

This chapter

Finding an item on a list or a key in a dictionary

The `in` operator returns true if an item is in a list (anywhere) or if the key is in a dictionary. This gives us a quick way to test if an item exists.

Examples

`"x" in ['a','z','x','b']` — is true ('x' is in the list)
`"f" in ['a','z','x','b']` — is false ('f' is not in the list)

`"b" in {'a':9, 'b':3}` — is true ('b' is a key)
`"x" in {'a':9, 'b':3}` — is false ('x' is not a key)
`9 in {'a':9, 'b':3}` — is false (9 is not a key, it is a value)

`value in sequence`	Operator
Returns true if the value is found in a sequence or other iterable object. For dictionaries tests if the value is a key. https://docs.python.org/3/reference/expressions.html#membership-test-operations http://bit.ly/2Qv9q4E	

Try this code with several state abbreviations that you know.

```
states = {"AL":"ALABAMA", "AK":"ALASKA", "AZ":"ARIZONA",
   "AR":"ARKANSAS", "CA":"CALIFORNIA ", "CO":"COLORADO",
   "CT":"CONNECTICUT" }
code = input("Enter State Code").upper()
print(code, "exists in dictionary:", code in states)
```

Random Numbers and Shuffle

To play an game or do a simulation we need a source of random numbers. A random number is a number that is selected from a range or group of numbers that has no apparent correlation to the previous numbers or future numbers selected.[2] You can think of a random number generated by Python to be like throwing a coin or dice, you just don't know what will happen next.

Importing the Random Module

To use random numbers in your programs you will need to include the following line of code BEFORE you want to use a random number method. It is recommended that your import statements, of which we will show a few more in this introduction, be included in the beginning of your program.

```
import random
```

2 http://mathworld.wolfram.com/RandomNumber.html

`module`	Concept
A module is an organizational unit if python code that is typically imported into other Python programs to add functionality. https://docs.python.org/3/glossary.html#term-module http://bit.ly/2Vuzxwq	

`import` `module`	Statement
The import statement will cause an external module to be loaded and initialized as part of your program. https://docs.python.org/3/reference/simple_stmts.html#import http://bit.ly/2FdIhlF	

`random`	Module
The random module adds a pseudo-random generator to your Python program. It has many methods to generate random integers, floating-points, and to shuffle lists. https://docs.python.org/3/library/random.html#module-random http://bit.ly/2VDwGBE	

Random Floating-point Numbers

The first method of the random object, you need to know is called `.random()`. It will return a non-negative random floating-point number less than one.

random.random()	Method of Random
The .random() method of random will return the next random floating-point number less than one. The range can be defined as 0 >= N < 1. https://docs.python.org/3/library/random.html#random.random	

```
import random
a = random.random()
print(a,"is a random number less than 1.")
print("true/false half of the time", a < .5)
```

```
0.2895374543350906 is a random number less than 1.
true/false half of the time True
```

Random Integers

The `random.randrange()` method will return a random integer. It works the same way as the range() function. There are three different ways to specify a range for the random number to be drawn from: 1) specifying the length of the range (starting at 0); 2) specifying the start and end values of a range; and 3) specifying the start, end, and step size.

random.randrange(*length_expr*)
random.randrange(*start_expr, stop_expr*)
random.randrange(*start_expr, stop_expr, step_expr*)
	Method of Random

The randrange method will return an integer in the range specified. There are three general ways to defile the range: 1) with a single integer that will create a random integer that will be in the range of 0 up to length-1. 2) with start and stop that will create a random integer in the range of start to stop -1. 3) with start, stop, and step that will create a random integer from the range of integers incremented by step from start to stop-step. https://docs.python.org/3/library/random.html#functions-for-integers http://bit.ly/2C5j2yb	

The next random method is called `.randint()`. It will return a random integer between the two values passed to it. Be careful, randint's arguments do not work like slices or ranges do, they are inclusive (the number returned may include both the starting and ending integer).

random.randint(*start_expr, stop_expr*)	Method of Random
The randint method will return an integer in the range specified. It is different from randrange because the integer number is inclusively in the range on both ends. https://docs.python.org/3/library/random.html#functions-for-integers http://bit.ly/2C5j2yb	

```python
import random
# this program rolls a 20 sided die numbered 1-20
# and a 6 sided die numbered 1-6 and adds them together
d20 = random.randint(1,20)
d6 = random.randint(1,6)
print("your d20+d6 roll was", d20+d6)
```

```
your d20+d6 roll was 22
```

Shuffling Lists

Another really useful method is `random.shuffle()`. It does what you think it would do, it will shuffle the elements in a list into random order. You pass it a variable containing a list and the list will be returned to you in a random order.

random.shuffle(*list*)	Method of Random
The shuffle method will mix up a list. The list needs to be stored in a variable because this method modifies the list directly. https://docs.python.org/3/library/random.html#random.shuffle http://bit.ly/2RfIQC3	

```
import random
kids = ["bob","june","sam","alex","pat","gab"]
random.shuffle(kids)
print("team 1 is", kids[0:3])
print("team 2 is", kids[3:])
```

```
team 1 is ['gab', 'bob', 'sam']
team 2 is ['alex', 'june', 'pat']
```

Selection with If

Now that we have covered Boolean operators, comparisons, the in operator, and random numbers; it is time to put them all to use. We call statements that conditionally execute code, based on a value, the process of selection.[3] In Python the `if` statement is the main selection structure.[4]

If

The `if` statement is the simplest of the Python selection statements. It will allow for conditional execution of a single block of code. If the expression is not false (true) then the suite (indented code) will be executed.

```
a = int(input("enter an integer"))
if a <= 10:
    print("you entered a number less than or equal to 10")
```

Pay special attention to the colon(:) after the then and be sure to indent the code that you want executed when the condition is true.

3 https://en.wikibooks.org/wiki/A-level_Computing/AQA/Paper_1/Fundamentals_of_programming/Selection
4 https://docs.python.org/3/tutorial/controlflow.html

```
if condition:
    suite
```

Statement

The if statement will execute the suite of code if the condition expression is True.

https://docs.python.org/3/reference/compound_stmts.html#the-if-statement
http://bit.ly/2AzPg4F

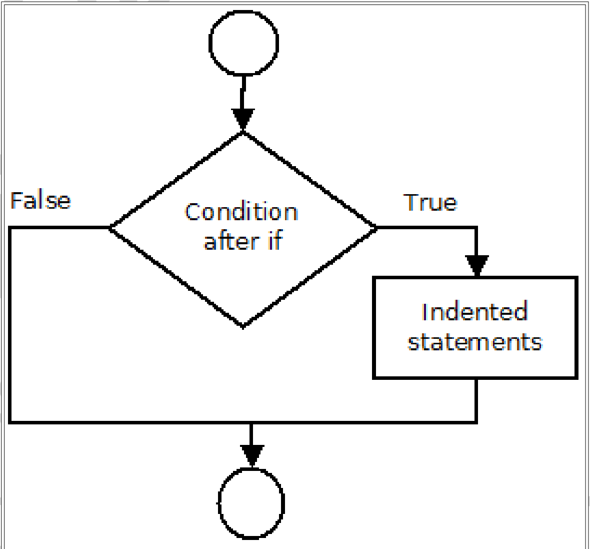

Illustration 14: Flowchart of If

Else

The suite after an `if` statement may be followed by the `else` statement. Else will allow you to have a second suite that is executed when the condition is false. This way you can have different statements executed for either condition.

```
if condition:
    suite
else:
    suite
```

Statement

The `if` statement followed by the `else` statement will execute the first suite of code if the condition expression is true and the second suite of code if the expression is false.

https://docs.python.org/3/reference/compound_stmts.html#the-if-statement
http://bit.ly/2AzPg4F

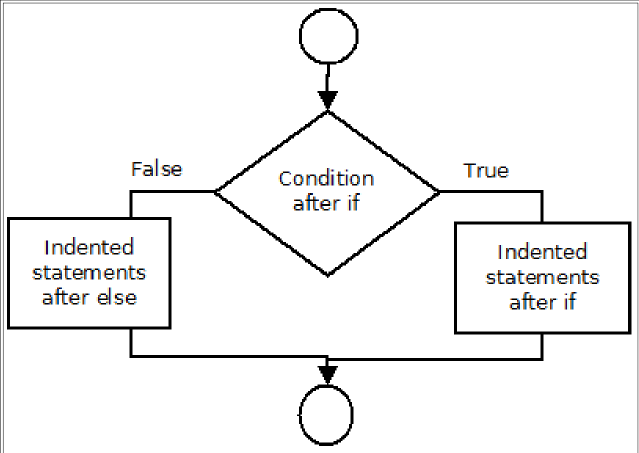

Illustration 15: Flowchart of If Else

```
import random
toss = random.random()
if toss < .10:
    print("really small")
    if toss < .05:
        print("actually really really small")
else:
    print("bigger than really small")
print("it was", toss)
```

Elif (Advanced and Optional)

One or more `elif` statements may follow an `if` statement. One or zero suites will be executed by testing each condition. Once a true condition's suite is complete all of the additional conditions and suites will be skipped. An `else` statement may follow the last `elif`.

```
if condition:
    suite
elif condition2:
    suite
else:
    suite
```

Statement

The `if` may be followed by one or more `elif` statements and an optional `else` statement. The program will execute the first suite that has a true condition and will skip all of the remaining conditions and suites. The optional `else` will execute if all of the conditions are evaluated to false.

https://docs.python.org/3/reference/compound_stmts.html#the-if-statement
http://bit.ly/2AzPg4F

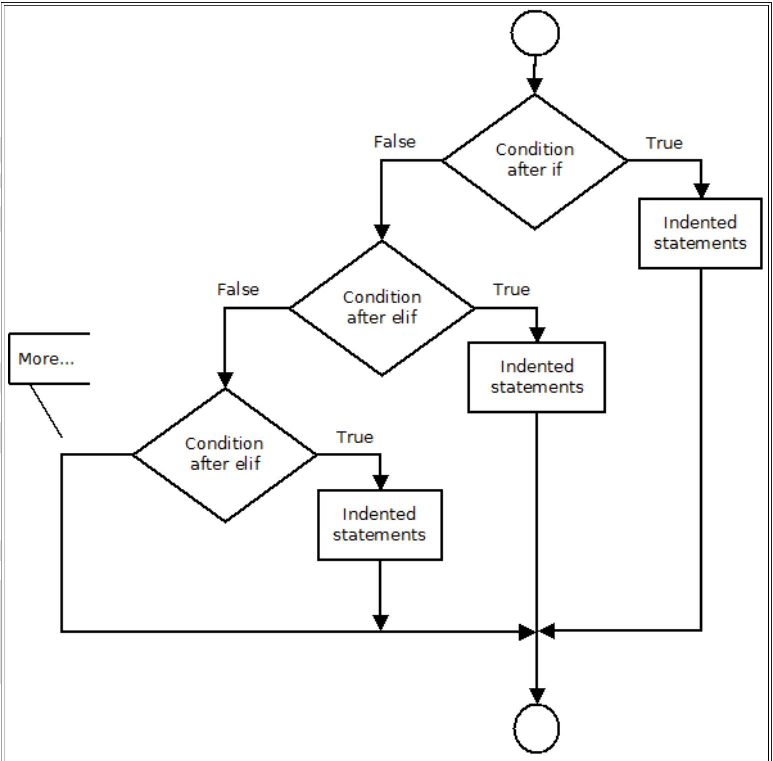

Illustration 16: Flowchart of If Elif

Three ways to do the same thing

The three programs, below, actually do the exact same thing. They each illustrate a different way to use `if` statements. This program will generate random grades for teacher when they don't want to grade papers. They have decided that 10% of students will get an A, 15% will get a "B", 50% will get a "C", 15% a "D", and the remaining 10% will get an "F" and that the grade will be assigned randomly.

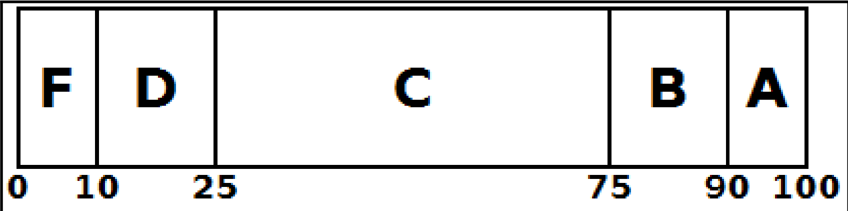

Illustration 17: Lazy Professor Problem

Way One: Using complex `if` statements so that only one grade will be displayed. The conditions in the if statements need to cover the full range of values returned by `random.random()` but have no overlapping values.

```
import random
g = random.random()
print(g)
if g >= .9:
    print ("A")
if g >= .75 and g < .9:
    print("B")
if g >= .25 and g < .75:
    print("C")
if g >= .10 and g < .25:
    print("D")
if g < .1:
    print("F")
```

Way Two: Using `else` statements with suites of nested `if/else` statements. Once a grade is found, the program does not execute any other of the conditions. The type of nesting may become confusing for very complex problems.

```

import random
g = random.random()
print(g)
if g >= .9:
    print ("A")
else:
    if g >= .75:
        print("B")
    else:
        if g >= .25:
            print("C")
        else:
            if g >= .10:
                print("D")
            else:
                print("F")
```

Way Three: The third solution uses the `elif` and `else` clauses to test at each grade point. When the first condition is found to be true, the other conditions are skipped.

```
import random
g = random.random()
```

```
print(g)
if g >= .9:
    print ("A")
elif g >= .75:
    print("B")
elif g >= .25:
    print("C")
elif g >= .10:
    print("D")
else:
    print("F")
```

Trapping Errors

Sometimes a program causes an error while it is running (usually the result of something that the human user has done. We catch errors with the try statement. There are many different ways to use it, but in this introduction we will keep the examples simple.

```
try:
    suite
except:
    suite
```

Statement

The `try/except` statements will catch an error in a suite of code and pass control to a special suite to handle the exception.

https://docs.python.org/3/reference/compound_stmts.html#the-try-statement
http://bit.ly/2RCXOkO

The first example using `except:` will catch all errors regardless of what they are.

```
n = input("enter a floating-point number?")
try:
    n = float(n)
except:
    print("You did not enter a valid number, zero used.")
    n = 0.0
print("you entered the number " + n)
```

Sometimes you may want to trap an error (keep it from terminating your program) but there is nothing special to be done for the error. You MUST have an except block but you may use the statement `pass` to tell Python that there is nothing to do here.

What Error was Thrown

You can add an optional Exception as variable after the except statement. This will set the specified variable to an object that contains details about the specific exception that was thrown.

```
try:
    a = int(input('Enter an integer?'))
    print('the reciprocal of', a , 'is', 1/a )
except Exception as e:
    print("exception of type",type(e), "with message", e)
```

```
Enter an integer?seven
exception of type <class 'ValueError'> with message invalid literal
for int() with base 10: 'seven'
```

Trapping Specific Errors

In addition to catching the exception. You can create excepts for different errors, and handle each of them differently. The following example catches the exception if the string the user enters can't be converted into an integer, the exception if they enter zero, and a generic except to catch any other errors.

```
try:
    a = int(input('Enter an integer?'))
    print('the reciprocal of', a , 'is', 1/a )
except ValueError:
    print('you did not enter an integer')
except ZeroDivisionError:
    print('the reciprocal of zero is undefined.')
except Exception as e:
    print(type(e))
    print("other exception of type",type(e), e)
```

You can find a list of Python exceptions at https://docs.python.org/3/library/exceptions.html .

While

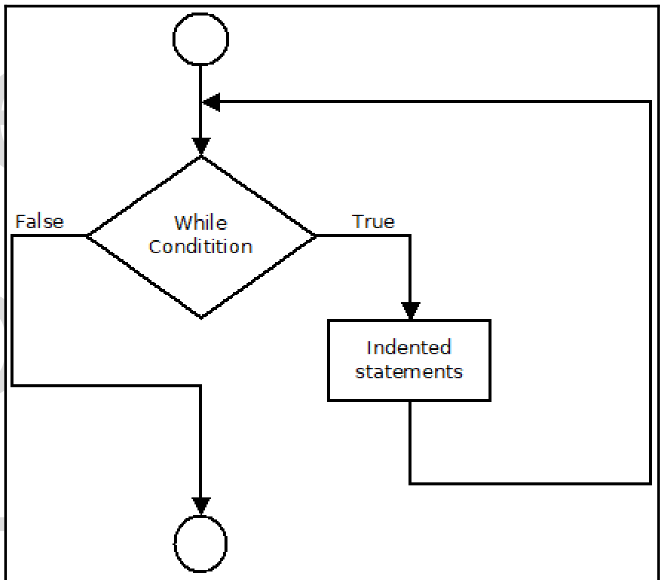

Illustration 18: While statement

The while loop will continue if the condition is true. This type of loop is usually used where we need to wait for something to change.

`while condition:`
 `    suite`
	Statement

The `while` loop will continue executing the suite of code until the condition becomes false. The loop test is done before each time through the loop, and the suite may be executed zero or more times. Be careful to make sure that the loop does not continue forever. https://docs.python.org/3/reference/compound_stmts.html#the-while-statement http://bit.ly/2Rc2heQ	

```
# flipping coins — Keep flipping heads (0) until I get a tails
  (1)
# display the coin and the count of heads
import random
heads = 0
tails = 0
while tails==0:
    if random.randint(0,1) == 0:
        heads = heads + 1
        print("head")
    else:
 print("tails")
        tails = tails + 1
print("i had", heads, "heads before i had a tail")
```

```
head
head
tails
i had 2 heads before i had a tail
```

Python also gives us a special statement to end a loop at any time. It is the `break` statement. When a break occurs the program immediately jumps to the first statement after the indented code of the loop.

`break` Statement

The `break` statement causes the program to jump out of a `for` or `while` loop.

https://docs.python.org/3/reference/simple_stmts.html#the-break-statement
http://bit.ly/2GZwF7z

An example of a while loop with a break can be seen below. It will continue looping until a valid floating-point number is entered. You could add an if statement before the break if you wanted the number to be within a certain range.

```
while True:
    try:
        a = float(input("Enter a float number?"))
        break
    except:
        pass
print("you entered", a)
```

Terminating Programs Early

Most of the programs you will see in this book, will terminate when we get to the end of the code. With more complex programs or ones who catch exceptions, it may become necessary to stop a program while it is executing. To do this we use the `sys.exit()` method.

`sys`	Module
The `sys` module allows programs to see many values and interact with the interpreter. To use an external module in your program you first must import it. https://docs.python.org/3/library/sys.html http://bit.ly/2Ayx7Ek	

`sys.exit()`	Method of sys
Tell the Python interpreter to stop the current program. https://docs.python.org/3/library/sys.html?highlight=sys%20module#sys.exit http://bit.ly/2RgNdge	

The `sys.exit()` method will cause a system level exception and will cause your program to terminate prematurely. To use it you must first `import sys` into your program. You also may pass an argument to `sys.exit("message")` with some sort of message to tell the user why the program trrminated.

```python
# loop 10 times by stopping the program
import sys

for t in range(1000000):
    if t == 10:
        sys.exit()
    print(t)
#
print("normal exit never happens")
```

```
1
2
3
```

```
4
5
6
7
8
9
10
An exception has occurred, use %tb to see the full traceback.

SystemExit
```

Summary

Goes here

Important Terms

- break
- elif
- else
- except
- exit
- if
- import
- in
- randint
- random
- randrange
- shuffle
- suite
- sys
- try
- while

Exercises

Here

Word Search

```
· s x b r e a k d t r e
l y z l i s x x e a a j
s s q h n b z v x i n t
i m p o r t j f i x d b
w d f i r s u i t e r e
h i s h u f f l e o a l
i · e x c e p t d c n i
l n i p t r j k t · g f
e i f t o g e l s e e s
n t r y i u w i u a o o
u o g r a n d i n t k w
c m r a n d o m l z w z
```

break, elif, else, except, exit, if, import, in, randint, random, randrange, shuffle, suite, sys, try, while

References