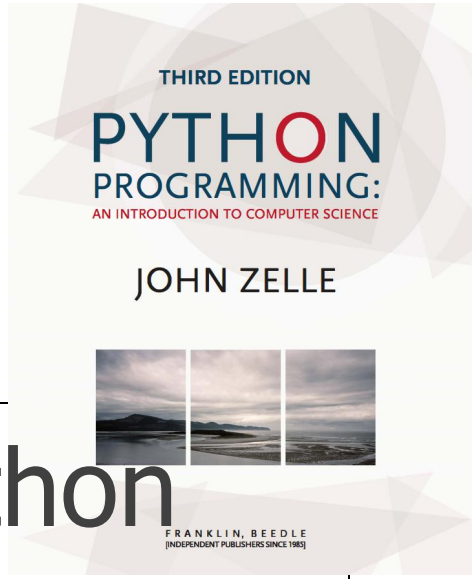


An Introduction to Python Programming

Chapter 2:Writing Simple Programs

Objectives

- To be able to understand and write Python statements to output information to the screen, assign values to variables, get numeric information entered from the keyboard, and perform a counted loop

The Software Development Process

- The process of creating a program is often broken down into stages according to the information that is produced in each phase.

The Software Development Process

- **Analyze the Problem**

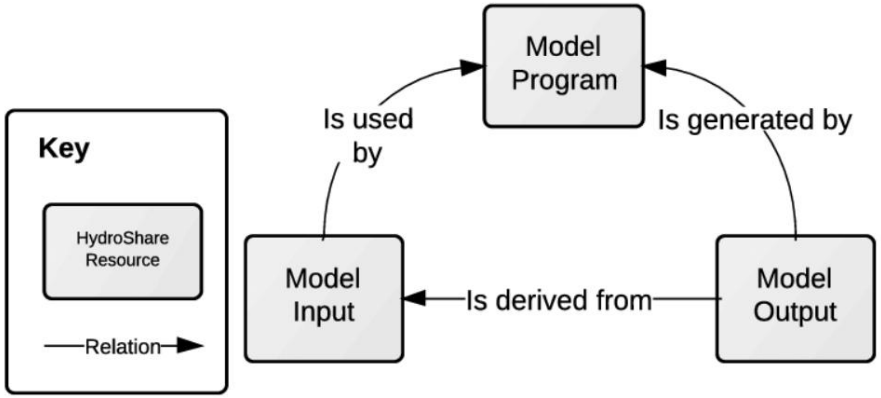
  Figure out exactly the problem to be solved. Try to understand it as much as possible.

The Software Development Process

- **Determine Specifications**

  Describe exactly what your program will do.

  - Don’t worry about *how* the program will work, but *what* it will do.
  - Includes describing the inputs, outputs, and how they relate to one another.



The Software Development Process

- **Create a Design**
  - Formulate the overall structure of the program.
  - This is where the *how* of the program gets worked out.
  - You choose or develop your own algorithm that meets the specifications.

The Software Development Process

A manufacturing firm produces two types of products: A & B. The units profit from product A is Rs. 100 and that of product B is Rs. 50. The goal of the firm is to earn a total profit of exactly Rs. 700 in the next week. Also, wants to achieve a sales volume for product A and B close to 5 and 4, respectively. Formulate this problem as a Goal programming model.

MODEL FORMULATION:

Let X1 and X2 be the number of units of products 'A' and 'B' produced, respectively.
The constraints of the problem can be stated as:

$100X_1 + 50X_2 = 700$ (Profit target goal)
$X_1 \leq 5$ (Sales target Goal)
$X_2 \leq 4$ (Sales target Goal)

GP MODEL FORMULATION: The problem can now be formulated as GP model as follows:

Minimization $Z = d_1^- + d_1^+ + d_2^- + d_3^-$
Subject to:

$100X_1 + 50X_2 + d_1^- - d_1^+ = 700$ (Profit target goal)
$X_1 + d_2^- = 5$ (Sales target Goal)
$X_2 + d_3^- = 4$ (Sales target Goal)
$X_1, X_2, d_1^+, d_2^-, d_3^- \geq 0$

The Software Development Process

- **Implement the Design**
  - Translate the design into a computer language.
  - We will use Python.

The Software Development Process

- **Test/Debug the Program**
  - Try out your program to see if it worked.
  - If there are any errors (*bugs*), they need to be located and fixed. This process is called *debugging*.
  - Your goal is to find errors, so try everything that might “break” your program!

It is impossible to make anything foolproof because fools are so ingenious.

The Software Development Process

- **Maintain the Program**
  - Continue developing the program in response to the needs of your users.
  - In the real world, most programs are never completely finished – they evolve over time.

Example Program: Temperature Converter

- Analysis – the temperature is given in Celsius, user wants it expressed in degrees Fahrenheit.
- Specification
  - Input – temperature in Celsius
  - Output – temperature in Fahrenheit
  - Output = 9/5(input) + 32

Example Program: Temperature Converter

- Design
  - Input, Process, Output (IPO)
  - Prompt the user for input (Celsius temperature)
  - Process it to convert it to Fahrenheit using F = 9/5(C) + 32
  - Output the result by displaying it on the screen

Example Program: Temperature Converter

- Before we start coding, let's write a rough draft of the program in *pseudocode*
- Pseudocode is precise English that describes what a program does, step by step.
- Using pseudocode, we can concentrate on the algorithm rather than the programming language.

Example Program: Temperature Converter

- Pseudocode:
  - Input the temperature in degrees Celsius (call it celsius)
  - Calculate fahrenheit as (9/5)*celsius+32
  - Output fahrenheit
- Now we need to convert this to Python!

Example Program: Temperature Converter

```
#convert.py
# A program to convert Celsius temps to Fahrenheit
# by: Susan Computewell

def main():
    celsius = eval(input("What is the Celsius temperature? "))
    fahrenheit = (9.0/5.0) * celsius + 32
    print ("The temperature is ",fahrenheit," degrees Fahrenheit.")

main()
```

Example Program: Temperature Converter

- Once we write a program, we should test it!

```
>>> main()
What is the Celsius temperature? 0
The temperature is  32.0  degrees Fahrenheit.
>>> main()
What is the Celsius temperature? 100
The temperature is  212.0  degrees Fahrenheit.
```

Elements of Programs

- Names
  - Names are given to variables (celsius, fahrenheit), modules (main, convert), etc.
  - These names are called *identifiers*
  - Every identifier must begin with a letter or underscore (“_”), followed by any sequence of letters, digits, or underscores.
  - Identifiers are case sensitive.

Elements of Programs

- These are all different, valid names
  - X
  - Celsius
  - Spam
  - spam
  - spAm
  - Spam_and_Eggs
  - Spam_And_Eggs

Elements of Programs

- Some identifiers are part of Python itself. These identifiers are known as *reserved words*. They are not available for you to use in your program.
- Python also includes quite a number of built-in fnctions
- For a complete list, see table 2.1

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

Table 2.1: Python keywords

Elements of Programs

- Expressions
  - The fragments of code that produce or calculate new data values are called *expressions*.
  - *Literals* are used to represent a specific value, e.g. 3.9, 1, 1.0
  - Simple identifiers can also be expressions.

Elements of Programs

```
>>> X = 5
>>> X
5
>>> print ( x)
5
>>> print (spa)
Traceback (most recent call last) :
File "<stdin>", line 1, in <module>
NameError: name 'spa' is not defined
```

- NameError is the error when you try to use a variable without a value assigned to it.

Elements of Programs

- Simpler expressions can be combined using *operators*.

  +, -, *, /, **

- Spaces are irrelevant within an expression.
- The normal mathematical precedence applies.

  ((x1 – x2) / 2*n) + (spam / k**3)

Elements of Programs

- Output Statements
  - A print statement can print any number of expressions.
  - Successive print statements will display on separate lines.
  - A bare print will print a blank line.
  - If a print statement ends with a “,”, the cursor is not advanced to the next line.

```
print(<expr>, <expr>, ..., <expr>)
print()
print(<expr>, <expr>, ..., <expr>, end="\n")
```

Elements of Programs

```
print (3+ 4)
print (3, 4, 3 + 4)
print ()
print ("The aswer is", 3 + 4)
```

produces this output:

```
7
3 4 7
The answer is 7
```

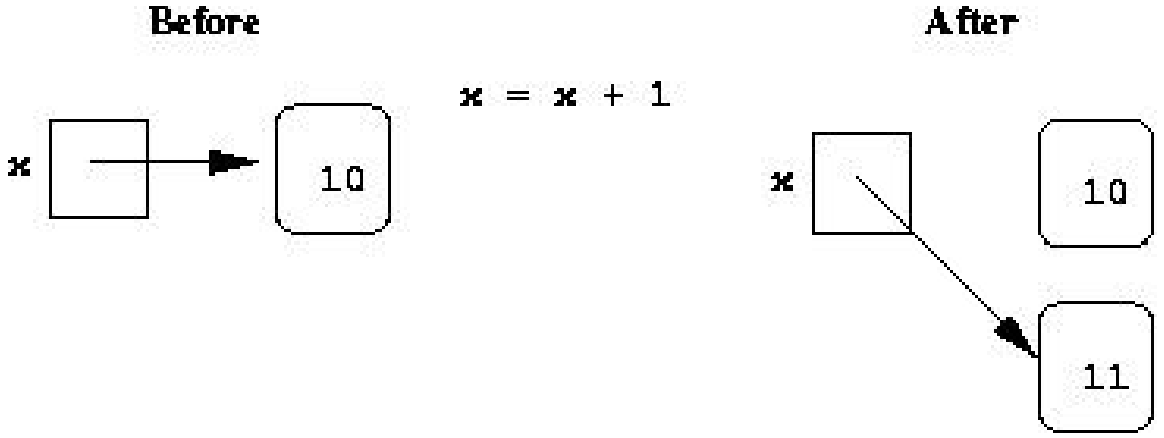
Assignment Statements

- The expression on the RHS is evaluated to produce a value which is then associated with the variable named on the LHS.
- Variables can be reassigned as many times as you want!

```
>>> myVar = 7
>>> myVar
7
>>> myVar = myVar + 1
>>> myVar
8
>>>
```

Assignment Statements

- Variables are like a box we can put values in.
- Technically, this model of assignment is simplistic for Python.
- Assigning a variable is more like putting a “sticky note” on a value and saying, “this is x”.



Assigning Input

- The purpose : get input from the user and store it into a variable.
- <variable> = input(<prompt>)
- E.g., x = input(“Enter a temperature in Celsius: ”)

Assigning Input

- The program waits for the user to enter a value and press <enter>
- The expression that was entered is evaluated and assigned to the input variable.
- You need to eval the input when you want a number instead of some raw text (a string).
- **Beware**: the eval fnction is very powerful and also potentially dangerous.

```
>>> ans = eval(input("Enter an expression: "))
Enter an expression: 3 + 4 * 5
>>> print(ans)
23
>>>
```

Simultaneous Assignment

- Several values can be calculated at the same time

  <var>, <var>, … = <expr>, <expr>, …

- Evaluate the expressions in the RHS and assign them to the variables on the LHS

  sum, diff = x+y, x-y

- How could you use this to swap the values for x and y?
- Quite easily in Python!

  x, y = y, x

Simultaneous Assignment

```
>>> x=3
>>> y=4
>>> x,y=y,x
>>> print(x,y)
4 3
>>>
```

```
>>> def main():
...     print("This program computes the average")
...     score1,score2=eval(input("Enter two scores: "))
...     aver=(score1+score2)/2
...     print("The average of the score is:",aver)
...
>>> main()
This program computes the average
Enter two scores: 40,60
The average of the score is: 50.0
>>>
```

- **Beware**:This trick will not work for string (non-evaled) input

Definite Loops

- A *definite* loop executes a definite number of times, i.e., Python starts the loop knowing exactly how many *iterations* to do.

  for <var> in <sequence>:

  <body>

- The beginning and end of the body are indicated by indentation.
- The variable after the *for* is called the *loop index*. It takes on each successive value in *sequence*.

Definite Loops

```
>>> for i in [0,1,2,3]:
...     print(i)
...
0
1
2
3
>>>
```

```
>>> for odd in [1,3,5,7,9]:
...     print(odd * odd)
...
1
9
25
49
81
>>>
```

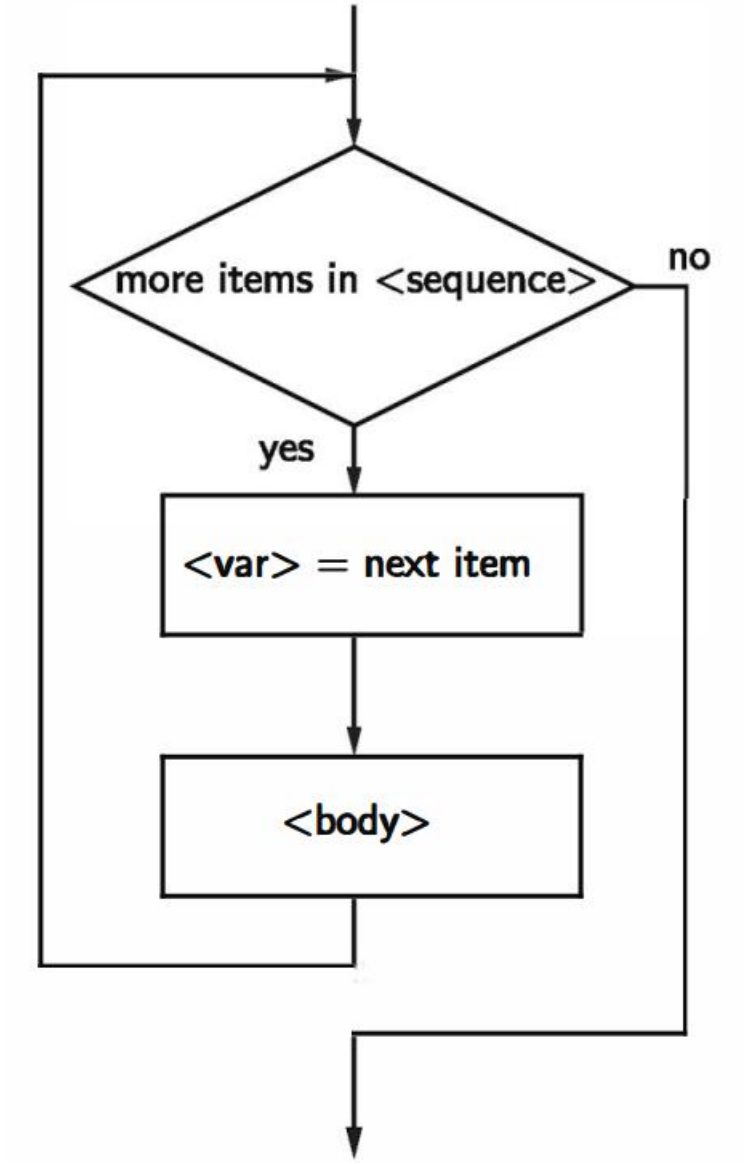
Definite Loops

- In chaos.py, what did *range(10)* do?
- Range(10) will make the body of the loop execute 10 times.

```
>>> list(range(10))    # turns range(10) into an explicit list
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Definite Loops

- **Flowchart** of for loops



Example Program: Future Value

- Analysis
  - Money deposited in a bank account earns interest.

  How much will the account be worth 10 years from now?
  - Inputs: principal, interest rate
  - Output: value of the investment in 10 years

Example Program: Future Value

- Specification
  - User enters the initial amount to invest, the principal
  - User enters an annual percentage rate, the interest
  - The specifications can be represent like this …

Example Program: Future Value

- **Program** Future Value
- **Inputs**
  principal The amount of money being invested, in dollars
  apr The annual percentage rate expressed as a decimal number.
- **Output** The value of the investment 10 years in the future
- **Relatonship** Value after one year is given by *principal* * (1 + *apr*). This needs to be done 10 times.

Example Program: Future Value

- Design
- Using pseudocode, We can formulate our ideas without worrying about all the rules of Python.

```
Print an introduction
Input the amount of the principal (principal)
Input the annual percentage rate (apr)
Repeat 10 times:
    principal = principal * (1 + apr)
Output the value of principal
```

Example Program: Future Value

- Implementation

```
# futval.py
#    A program to compute the value of an investment
#    carried 10 years into the future

def main():
    print("This program calculates the future value")
    print("of a 10-year investment.")

    principal = eval(input("Enter the initial principal: "))
    apr = eval(input("Enter the annual interest rate: "))

    for i in range(10):
        principal = principal * (1 + apr)

    print("The value in 10 years is:", principal)

main()
```

Example Program: Future Value

```
>>> main()
This program calculates the future value of a 10-year investment.
Enter the initial principal: 100
Enter the annual interest rate: .03
The value in 10 years is: 134.391637934
>>> main()
This program calculates the future value of a 10-year investment.
Enter the initial principal: 100
Enter the annual interest rate: .10
The value in 10 years is: 259.37424601
```