Python Programming: An Introduction to Computer Science

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

- **Implement the Design**
  - Translate the design into a computer language.
  - In this course 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!

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 = \( \frac{9}{5} \text{input} + 32 \)

Example Program: Temperature Converter

- **Design**
  - Input, Process, Output (IPO)
  - Prompt the user for input (Celsius temperature)
  - Process it to convert it to Fahrenheit using \( F = \frac{9}{5}(C) + 32 \)
  - 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 \((9/5)\cdot celsius + 32\)
- Output fahrenheit
- Now we need to convert this to Python!

```
def main():
    celsius = 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!

```python
>>> 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.
>>> main()
What is the Celsius temperature? -40
The temperature is -40.0 degrees Fahrenheit.
```
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.

```python
>>> x = 5
>>> print x
5

>>> print spam
NameError: name 'spam' is not defined

>>> print 3+4
7
>>> print 3, 4, 3+4
3 4 7
>>> print 3+4, The answer is 7
The answer is 7
```

```
>>> a
5
>>> print a
5
>>> print spam
NameError: name 'spam' is not defined
```

```
>>> x = 5
>>> print x
5

>>> print spam
NameError: name 'spam' is not defined
```

Elements of Programs

- Simpler expressions can be combined using *operators*.
  - +, -, *, /, **
  - Spaces are irrelevant within an expression.
  - The normal mathematical precedence applies.
  - \((x1 - x2) / 2^n + (spam / k**3)\)
Assignment Statements

- $x = 3.9 \times x \times (1-x)$
- $\text{fahrenheit} = 9.0/5.0 \times \text{celsius} + 32$
- $x = 5$

Variables can be reassigned as many times as you want!

```python
>>> myVar = 5
>>> myVar
5
>>> myVar = 7
>>> myVar
7
>>> myVar = myVar + 1
>>> myVar
8
>>> myVar
8
```

Assignment Statements

- Variables are like a box we can put values in.
- When a variable changes, the old value is erased and a new one is written in.

<table>
<thead>
<tr>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$ = 10</td>
<td>$x$ = 11</td>
</tr>
</tbody>
</table>

Technically, this model of assignment is simplistic for Python.
- Python does not recycle these memory locations (boxes).
- Assigning a variable is more like putting a “sticky note” on a value and saying, “this is $x$”.

Assigning Input

- The purpose of an input statement is to get input from the user and store it into a variable.
- `<variable> = input(<prompt>)`
- E.g., `x = input("Enter a temperature in Celsius: ")`

First the prompt is evaluated
- 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.

```python
>>> def inp():
    x = input("Enter something ")
    print x

>>> inp()
Enter some
7
```
Simultaneous Assignment

- Several values can be calculated at the same time
- `<var1>, <var2>, ... = <expr1>, <expr2>, ...`
- Evaluate the expressions in the RHS and assign them to the variables on the LHS

Simultaneous Assignment

- `sum, diff = x+y, x-y`
- How could you use this to swap the values for `x` and `y`?
  - Why doesn’t this work?
    - `x = y`
    - `y = x`
  - We could use a temporary variable...

Simultaneous Assignment

- We can swap the values of two variables quite easily in Python!
  - `x, y = y, x`
  - `>>> x = 3` `>>> y = 4`
  - `>>> print x, y`
    - `3 4`

Simultaneous Assignment

- We can use this same idea to input multiple variables from a single input statement!
- Use commas to separate the inputs
  - `>>> def spamneggs(): spam, eggs = input("Enter the number of slices of spam followed by the number of eggs: ") print "You ordered", eggs, "eggs and", spam, "slices of spam. Yum!"
    - `>>> spamneggs()`
      - Enter the number of slices of spam followed by the number of eggs
      - Ordered 2 eggs and 3 slices of spam. Yum!

Definite Loops

- A `definite` loop executes a definite number of times, i.e., at the time Python starts the loop it knows exactly how many `iterations` to do.
- for `<var>` in `<sequence>`:
  - `<body>`
  - The beginning and end of the body are indicated by indentation.
Definite Loops

```python
>>> for i in [0,1,2,3]:
    print i
0
1
2
3
>>> for odd in [1, 3, 5, 7]:
    print odd*odd
1
9
25
49
```

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 ...

**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

**Relationship** Value after one year is given by `principal * (1 + apr)`. This needs to be done 10 times.
Example Program: Future Value

- Design
  - 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

implementation
- Each line translates to one line of Python
  - (in this case)
- Print an introduction
  - print "This program calculates the future value of a 10-year investment."
- Input the amount of the principal
  - principal = input("Enter the initial principal: ")

Example Program: Future Value

- Input the annual percentage rate
  - apr = input("Enter the annual interest rate: ")
- Repeat 10 times:
  - for i in range(10):
    - Calculate principal = principal * (1 + apr)
    - principal = principal * (1 + apr)
- Output the value of the principal at the end of 10 years
  - print "The value in 10 years is:", principal

Example Program: Future Value

```python
# 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 of a 10-year investment."
    principal = input("Enter the initial principal: ")
    apr = 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

```python
>>> 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
```