Vahé Karamian

Python Programming

CS-110

CHAPTER 2

Writing Simple Programs

OBJECTIVES

- To know the steps in an orderly software development process.
- To understand programs following the input, process, output (IPO) patterns and be able to modify them in simple ways.
- To understand the rules for forming valid Python identifiers and expressions.
- Understand and write Python statements to output information to the screen, assign values to variables, get information entered from the keyboard, and perform 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

1. Analyze the Problem
2. Determine Specifications
3. Create a Design
4. Implement the Design
5. Test/Debug the Program
6. Maintain the Program

THE SOFTWARE DEVELOPMENT PROCESS

STEP 1: Analyze the Problem

Figure our what the problem to be solved is. Try to understand as much as possible about it. Until you really know what the problem is, you cannot begin to solve it.

THE SOFTWARE DEVELOPMENT PROCESS

STEP 2: Determine Specifications

Describe exactly what your program will do. At this point, you should not worry about how your program will work, but rather about deciding exactly what it will accomplish.

For simple programs this involves carefully describing what the inputs and outputs of the program will be and how they relate to each other.

THE SOFTWARE DEVELOPMENT PROCESS

STEP 3: Create a Design

Formulate the overall structure of the program. This is where the how of the program gets worked out. The main task is to design the algorithm(s) that will meet the specifications.

THE SOFTWARE DEVELOPMENT PROCESS

STEP 4: Implement the Design

Translate the design into a computer language and put it into the computer.

For this class, we will be using the Python language to implements our algorithms.

THE SOFTWARE DEVELOPMENT PROCESS

STEP 5: Test/Debug the Program

Try out your program and see if it works as expected. If there are any errors (bugs), then you should go back and fix them.

The process of locating and fixing errors is called debugging a program. During this phase, your goal is to find errors, so you should try everything you can think of that might break the program.

"Nothing is foolproof because fools are too ingenious."

THE SOFTWARE DEVELOPMENT PROCESS

STEP 6: Maintain the Program

Continue developing the program in response to the needs of your users.

Most programs are never really finished; they keep evolving over years of use.

TEMPERATURE CONVERTER

- **Analysis:** The temperature is given in Celsius, user wants its expressed in degrees Fahrenheit.

- **Specifications**
  - Input: Temperature in Celsius
  - Output: Temperature in Fahrenheit.
  - Equation: (9/5)(input) + 32

TEMPERATURE CONVERTER

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

TEMPERATURE CONVERTER

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

TEMPERATURE CONVERTER

- Pseudocode:

Input the temperature in degrees Celsius

*Calculate fahrenheit as (9/5) * celsius + 32*

Output fahrenheit

Let's convert it to Python!

TEMPERATURE CONVERTER

```
# convert.py
# A program to convert Celsius temps to Fahrenheit

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

main()
```

Elements of Programs

- **Names**
  - Names are given to variables (Celsius, Fahrenheit) , modules (main, convert), and etc…
  - These 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. This means they are not available for you to use as a name for a variable, etc… in your program.
- and, del, for, is, raise, assert, elif, in print, etc…
- For a complete list, see Table 2.1

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(spam)
Traceback (most recent call last):
  File "<pyshell#25>", line 1, in <module>
    print(spam)
NameError: name 'spam' 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.

ASSIGNMENT STATEMENTS

- Simple Assignment
  - <variable> = <expression>
  - <variable> is an identifier, and <expression> is an expression
  - The expression on the RHS is evaluated to produce a value which is then associated with the variable named on the LHS.

ASSIGNMENT STATEMENTS

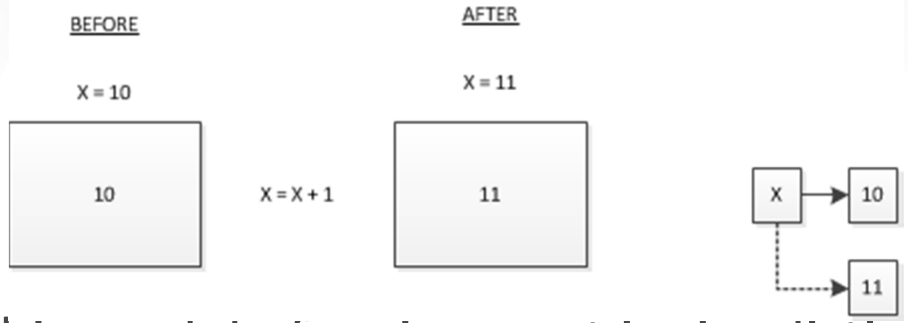
- X = 3.9 * X * (1 – X)
- Fahrenheit = 9.0 / 5.0 * Celsius + 32
- X = 5

- Variables can be reassigned as many times as you want!

```
>>> x = 7
>>> x
7
>>> x = 5
>>> x
5
>>> x = x + 1
>>> x
6
```

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.



- 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>), for string
- <variable> = eval(input(<prompt>)), for numerical values

```
>>> name = input("Enter your name: ")
Enter your name: Vahe Karamian
>>> name
'Vahe Karamian'
>>>
```

ASSIGNING INPUT

- 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.

```
>>> ans = eval(input("Enter an expression: "))
Enter an expression: 3+4*3
>>> print(ans)
15
>>>
```

SIMULTANEOUS ASSIGNMENT

- Several values can be calculated at the same time
- <var>, <var>, … = <exp>, <exp>, …
- Evaluate the expression in the RHS and assign them to the variables on the LHS.
- A, B = 1, 2
- How could you use this to swap the values of X and Y?
  - Why doesn't this work?
  - X = Y
  - Y = X
- We need to 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
>>> x, y = y, x
>>> print(x,y)
4 3
>>>
```

SIMULTANEOUS ASSIGNMENT

- We can use this same idea to input multiple variables from a single input statement!
- Use commas to separate the inputs

```
>>> def spameggs():
 spam, eggs = eval(input("Enter # of spam followed by # of eggs: "))
 print("You ordered ", eggs, "eggs and ", spam, " slices of spam!")

>>> spameggs()
Enter # of spam followed by # of eggs: 3, 4
You ordered  4 eggs and  3  slices of spam!
>>>
```

DEFINITE LOOPS

- A definite loop executes a definite number of times, i.e., at the time Python start 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.
- The variable after the *for* is called the *loop index*. It takes on each successive value in sequence.

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

DEFINITE LOOPS

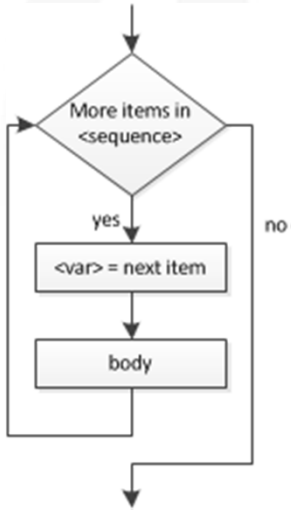
- In chaos.py, what did range(10) do?

```
>>> range(10)
range(0, 10)
```

- Range is a built-in Python function that returns a list of numbers, starting with 0.
- Range(10) will make the body of the loop execute 10 times.

DEFINITE LOOPS

- **for** loops alter the flow of program execution, so they are referred to as *control structures*.



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

FUTURE VALUE

- **Program:** Future Value
- **Inputs:**
  - Principal, amount of money invested in dollars
  - Apr, the annual percentage rate expressed as a decimal number.
- **Outputs:**
  - 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!

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

FUTURE VALUE

- **Implementation:** Each line translates to one line of Python.

```
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()
```

CHAPTER 2 HOMEWORK

Review Questions
TRUE/FALSE (ALL)
Multiple Choice (ALL)

Discussion Questions (ALL)

Programming Exercises
1, 2, 3, 4, 5, 6, 8, 9, 11

Extra Credit: Programming Exercise 7

Due Date: 02/01/2011 – Turn in Hard Copy + Submit Electronic Version