Python Programming: An Introduction to Computer Science

Chapter 2

Dan Fleck

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.

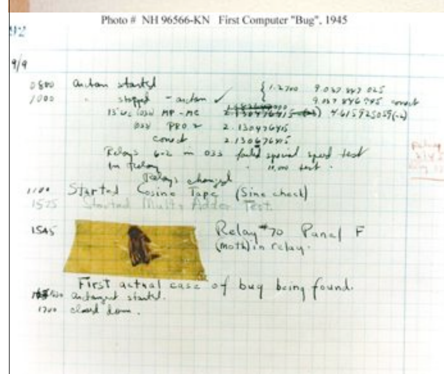
The Software Development Process

- **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!

Why is it called debugging?



The First "Computer Bug"
Moth found trapped between points at Relay # 70, Panel F, of the Mark II Aiken Relay Calculator while it was being tested at Harvard University, 9 September 1945. The operators affixed the moth to the computer log, with the entry: **"First actual case of bug being found"**. They put out the word that they had **"debugged"** the machine, thus introducing the term "debugging a computer program".

Courtesy of the Naval Surface Warfare Center, Dahlgren, VA., 1988.*U.S. Naval Historical Center Photograph.*

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.

Coming up: Example : Temperature Converter Analysis 9

Example : Temperature Converter Analysis

- 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

Coming up: Example : Temperature Converter Design 10

Example : Temperature Converter Design

- 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

Coming up: Example : Temperature Converter 11

Example : 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. However, There is no "official" syntax for pseudocode
- Using pseudocode, we can concentrate on the algorithm rather than the programming language.

Coming up: Temperature Converter Pseudocode 12

Temperature Converter Pseudocode

- 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!

Temperature Converter Python Code

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

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

main()
```


Temperature Converter Testing

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

```
>>>
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 : Identifiers

- Names of values: celsius, fahrenheit
- Names of functions: range, main, input
- Names of modules: convert

These names are called *identifiers*

- Every identifier must begin with a letter or underscore (“_”), followed by any sequence of letters, digits, or underscores.
- Good programmers use meaningful names
- Identifiers are case sensitive.

Elements of Programs : Identifiers

Identifiers are case sensitive.

- In Python, identifiers:
  - myVar
  - MYVAR
  - myvar
- Are all *DIFFERENT* because Python is case-sensitive

Reserved Words

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	from	lambda	return
break	else	global	not	try
class	except	if	or	while
continue	exec	import	pass	yield
def	finally	in	print	

Table 2.1: Python Reserved Words.

Using identifiers in expressions

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

Traceback (most recent call last):
  File "<pyshell#15>", line 1, in -toplevel-
    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.

Math Operators

- 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)

Precedence is:
PEMDAS - (), **, *, /, +, -

```
>>>
>>>
>>> x = 4
>>> y = 2
>>> x + y
6
>>> x - y
2
>>> x * y
8
>>> x / y
2
>>> x ** y
16
>>>
>>> x * 3
12
>>> d = x * 3
>>> d
12
>>>
```

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.

Coming up: Elements of Programs 21

Elements of Programs

```
print 3+4
print 3, 4, 3+4
print
print 3, 4,
print 3+      4
print “The answer is”,
   3+4
```

Coming up: Assignment Statements 22

Assignment Statements

- <variable> = <expr>
  variable is an identifier, expr 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.
- x = 3.9 * x * (1-x)
- fahrenheit = 9.0/5.0 * celsius + 32
- x = 5

Coming up: Assignment Statements 23

Assignment Statements

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

```
>>> myVar = 0
>>> myVar
0
>>> myVar = 7
>>> myVar
7
>>> myVar = myVar + 1
>>> myVar
8
>>>
```

Coming up: Assigning Input 24

Assigning Input

- Input: gets input from the user and stores it into a variable.
- <variable> = input(<prompt>)

```
>>> x = input("Enter a temp in farenheit >")
Enter a temp in farenheit >30
>>> x
30
>>>
```


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.

```
>>>
>>> def inp():
        yourInput = input("Input something >")
        print yourInput

>>> inp()
Input something >3 + 8
11
>>>
```


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.

Loop Index

Definite Loops: Example 1

```
# simple_for_loops_1.py
# Example of simple "for" loop operation
#
# general form:  for <var> in <sequence>:
#                        <body>

def main():

    for i in range(3):
        print i

    print

    for i in range(3):
        print 77

    print

    x = 3

    for i in range(x):
        print 88

    print

    for i in range(x):
        x = x + 1
        print (x)

    print

    for i in range(x):
        x+=1
        print x

main()
```

```
IDLE 1.2.1
>>> ================================
>>>
0
1
2

77
77
77
```

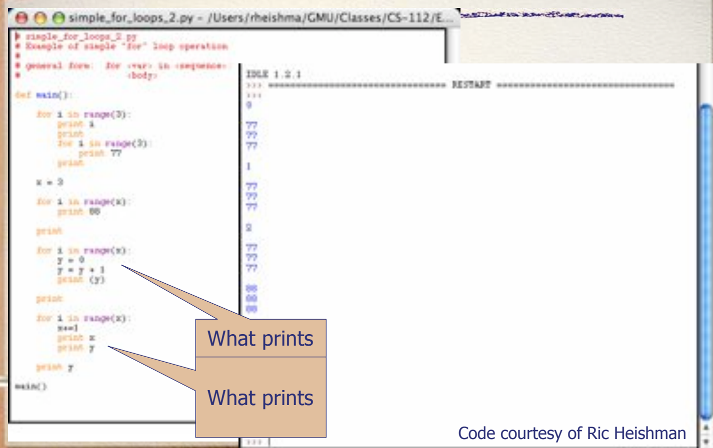
What prints

What prints

What prints

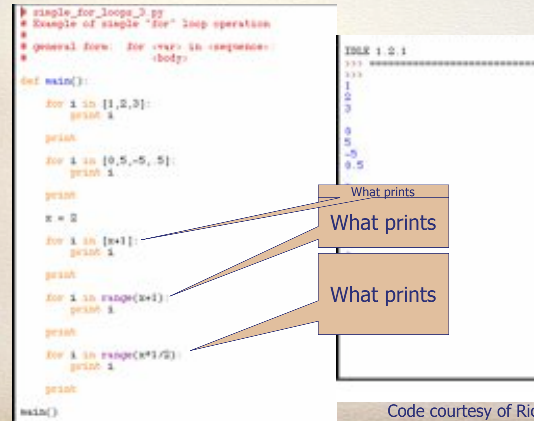
Code courtesy of Ric Heishman

Definite Loops: Example 2



Code courtesy of Ric Heishman

Definite Loops: Example 3



Code courtesy of Ric Heishman

Built-In Function: Range

range([start], stop [,step])

Hint: things in [] are optional!

This is a versatile function to create lists containing arithmetic progressions. It is most often used in for loops. The arguments must be plain integers. If the step argument is omitted, it defaults to 1. If the start argument is omitted, it defaults to 0. The full form returns a list of plain integers [start, start + step, start + 2 * step, ...]. If step is positive, the last element is the largest start + i * step less than stop; if step is negative, the last element is the smallest start + i * step greater than stop. step must not be zero (or else ValueError is raised). Example:

```
>>> range(10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> range(1, 11)
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> range(0, 30, 5)
[0, 5, 10, 15, 20, 25]
>>> range(0, -10, -1)
[0, -1, -2, -3, -4, -5, -6, -7, -8, -9]
```


How to exit a loop early

- *break* statement - terminates the nearest enclosing loop.

```
>>>
>>> for myVar in range(10):
    print myVar
    if myVar == 3:
        break
0
1
2
3
>>>
```

Lets see another example in breaktest.py

How to skip a loop if needed

- *continue* - continues with the next iteration of the loop.

```
def continueEx1():
  for i in range(10):
    if i > 5:
      continue
    print i,

>>> 0 1 2 3 4 5
```

Lets see example in continueExamples.py

Else in a for loop

Loop statements may have an `else` clause; it is executed when the loop terminates through exhaustion of the list (with `for`) or when the condition becomes false (with `while`), but not when the loop is terminated by a `break` statement. [Python Docs]

For loop else example

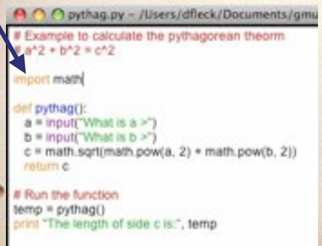
```
def elseExample2(input):
  for i in range(input):
    if i > 10:
      print 'i was greater than 10 in the loop'
      break
  else:
    print 'i was never greater than 10 in the loop!'

>>> elseExample2(9)
i was never greater than 10 in the loop
>>> elseExample(12)
i was greater than 10 in the loop
```

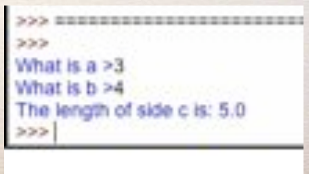

How to find information yourself

- To the Python docs!
  - http://docs.python.org/
  - Library Reference
  - Index
  - range

Using Python's Modules

- Python has a lot of code available in modules for you to use
- Using modules, you must "import" them.

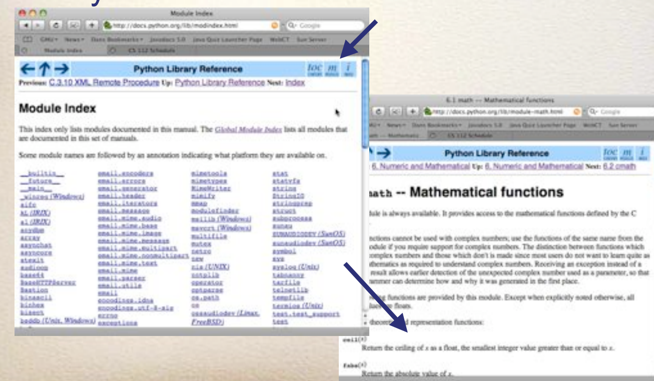




Coming up: What modules are available? 37

What modules are available?

- Many! Find info in the module index



Coming up: Another way to print 38

Another way to print

```
>>> numCats=5
>>> numDogs=7
>>> print "There were %d cats and %d dogs" %(numCats, numDogs)
There were 5 cats and 7 dogs
>>>
```

Printing is the SAME. You are formatting the String. This also works:

```
>>> myString = "There were %d cats and %d dogs" %(numCats,
    numDogs)
>>> print myString
There were 5 cats and 7 dogs
```

Inside the String you can put placeholders for other values. The placeholders specify a type.

%d = Signed integer decimal
%f = Floating point (decimal format)
%s = String

Coming up: String formats 39

String formats

You can also specify a minimum field width like this: "%20d" . This will force the number to take up 20 spaces.

```
>>> print "Num 1 = %10f" %(123.456)
Num 1 = 123.456000
```

To print in columns:

```
print "Col1          Col2"
print "%10f          %10f" %(12.23, 222.45)
print "%10f          %10f" %(444.55, 777)
Col1           Col2
 12.230000     222.450000
444.550000     777.000000
```

Coming up: Example Program: Future Value 40

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

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
  - Each line translates to one line of Python (in this case)
  - Print an introduction

    print "This program calculates the future"
    print "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

```
# 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

```
>>> 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
```