A First Program

CSE 1310 – Introduction to Computers and Programming
Vassilis Athitsos
University of Texas at Arlington

Credits: a significant part of this material has been created by Dr. Darin Brezeale and Dr. Gian Luca Mariottini

Simplest Code: Numerical Calculations

- Start the Python shell. You see a welcoming message and the command prompt.

```
Python 3.3.2 (v3.3.2:d047928ae3f6, May 16 2013, 00:06:53)
   [MSC v.1600 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more
   information.
>>>
```

Terminology: we will call >>> “the command prompt”. This is Python’s way of telling you “I am waiting for your input”.

Simplest Code: Numerical Calculations

- Let’s type in a single number, and press ENTER.

```
Python 3.3.2 (v3.3.2:d047928ae3f6, May 16 2013, 00:06:53)
   [MSC v.1600 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more
   information.
>>> 14
```

Simplest Code: Numerical Calculations

- After we press ENTER, the computer evaluates what we just typed, and prints the result.

```
Python 3.3.2 (v3.3.2:d047928ae3f6, May 16 2013, 00:06:53)
   [MSC v.1600 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more
   information.
>>> 14
14
>>>
```

Simplest Code: Numerical Calculations

- After we press ENTER, the computer evaluates what we just typed, and prints the result.

```
Python 3.3.2 (v3.3.2:d047928ae3f6, May 16 2013, 00:06:53)
   [MSC v.1600 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more
   information.
>>> 14
14
>>>
```

- This is not very exciting yet, the computer did not tell us anything we did not know.

Python as a Calculator.

```
>>> (23*3) + 12/4.5
71.66666666666667
```

- We can type in arbitrary numerical expressions, and Python evaluates them.

- This is still not that exciting.
- However, such calculations are a useful building block for real programs.

Operators

- Here is a list of operators used in Python:

```
+      -      *      **     /      //     %
<<     >>     &      |      ^      ~
<      >      <=     >=     ==     !=     <>
+=     -=     *=     /=     //=    %=
&=     |=     ^=     >>=    <<=    **=
```

- Do not try to memorize them, but learn to look them up in the book as needed (chapter 1).
  - Try them out to verify you understand what they do.

Some Operators

```
>>> 2**10
1024
```

** is the “exponentiation” operator

```
>>> 13%4
1
```

% is the “remainder” operator

Order of Operators

- What does this do?

```
>>> 2**20/2
```

- Do we first do 2**20 and then divide by 2, or do we first do 20/2 and then 2**10?

Order of Operators

- What does this do?

```
>>> 2**20/2
```

- Do we first do 2**20 and then divide by 2, or do we first do 20/2 and then 2**10?
- The book defines the order of operators in chapter 1.
- Suggestion: USE THESE RULES MINIMALLY.

Order of Operators

- Instead of

```
>>> 2**20/2
```

You should type

```
>>> (2**20)/2
```

or

```
>>> 2**(20/2)
```

Circumference and Area of Circle

- Computing the circumference of a circle with radius = 20.231234:
  - Circumference = radius * pi * 2

```
>>> 20.231234 * 3.14159 * 2
127.1164848412
```

- Computing the area of the same circle:
  - area = (radius ** 2) * pi

```
>>> (20.231234 ** 2) * 3.14159
1285.8616750694227
```

Using Variables

```
>>> 20.231234 * 3.14159 * 2
>>> (20.231234 ** 2) * 3.14159
```

- Tedious to type in long numbers repeatedly.
- The above lines are hard to read.
- Instead, we can do:

```
>>> radius = 20.231234
>>> pi = 3.14159
>>> circumference = radius * pi * 2
>>> area = (radius ** 2) * pi
```

Using Variables

- When we type in these four lines, Python prints nothing back.

```
>>> radius = 20.231234
>>> pi = 3.14159
>>> circumference = radius * pi * 2
>>> area = (radius ** 2) * pi
```

- How can we see the actual results?

Using Variables

- When we type in these four lines, Python prints nothing back.

```
>>> radius = 20.231234
>>> pi = 3.14159
>>> circumference = radius * pi * 2
>>> area = (radius ** 2) * pi
```

- How can we see the actual results?

```
>>> circumference
127.1164848412
>>> area
1285.8616750694227
```

Doing Repeated Calculations

- What if we want to calculate the area and circumference of circles many times per day (or many times per hour)?
- We can just type in the formulas (as we did in the previous slides) again and again.
  - Any shortcomings of that approach?

The Need for a Program

- What if we want to calculate the area and circumference of circles many times per day (or many times per hour)?
- Typing in the formulas again and again is **tedious**, and **error prone**.
- Here is where we can use our first ***PROGRAM***.

Creating a Program

- Create a text file, called “circles_1.py”.
- Easy way, from Python shell:
  - File -> New Window
    - Creates a new text window
  - File -> Save
    - Allows you to save the file using a name of your choice.
    - IMPORTANT: Make sure you understand what a folder is, and that you know where your file is saved.
    - Talk to the class TA to learn how to do that.

Creating a Program

- Within the file, we put in this text:

```
# specify the radius value
radius = 25.12

# compute and print the circumference
pi = 3.14159
circumference = radius * 2 * pi
print("Circumference = ", circumference)

# compute and print the area
area = (radius ** 2) * pi
print("area = ", area)
```

Running the program

- From the text file window, choose
  Run -> Run Module

  (or simply press F5).

Running the program

- From the text file window, choose Run -> Run Module

  (or simply press F5).

```
Circumference =  157.8334816
area =  1982.388528896
```

Problem: Radius is Hardcoded

- Why is this a problem?

Problem: Radius is Hardcoded

- Why is this a problem?
- Biggest reason: the user needs to be a programmer.
  - You cannot use this program without changing the program.

Solution

- Allow the user to enter the radius value as input.

Revised Program

```
# get the radius from the user as a string
radius_string = input("Enter the radius of your circle: ")

# convert the radius string to a real number.
radius = float(radius_string)

# compute and print the circumference
pi = 3.14159
circumference = radius * 2 * pi
print("Circumference = ", circumference)

# compute and print the area
area = (radius ** 2) * pi
print("area = ", area)
```

Running the program

- From the text file window, choose Run -> Run Module

  (or simply press F5).

```
Enter the radius of your circle:
```

Running the program

- From the text file window, choose Run -> Run Module

  (or simply press F5).

```
Enter the radius of your circle: 2
```

Running the program

- From the text file window, choose Run -> Run Module

  (or simply press F5).

```
Enter the radius of your circle: 2
Circumference =  12.56636
area =  12.56636
```

Understanding the Program

```
# get the radius from the user as a string
radius_string = input("Enter the radius of your circle: ")

# convert the radius string to a real number.
radius = float(radius_string)

# compute and print the circumference
pi = 3.14159
circumference = radius * 2 * pi
print("Circumference = ", circumference)

# compute and print the area
area = (radius ** 2) * pi
print("area = ", area)
```

Understanding the Program

```
# get the radius from the user as a string
radius_string = input("Enter the radius of your circle: ")

# convert the radius string to a real number.
radius = float(radius_string)

# compute and print the circumference
pi = 3.14159
circumference = radius * 2 * pi
print("Circumference = ", circumference)

# compute and print the area
area = (radius ** 2) * pi
print("area = ", area)
```

Comment lines:

Are notes to ourselves or other people, the computer ignores them.

Understanding the Program

```
# get the radius from the user as a string
radius_string = input("Enter the radius of your circle: ")

# convert the radius string to a real number.
radius = float(radius_string)

# compute and print the circumference
pi = 3.14159
circumference = radius * 2 * pi
print("Circumference = ", circumference)

# compute and print the area
area = (radius ** 2) * pi
print("area = ", area)
```

Getting user input:

input is a PREDEFINED function in Python. Its job is to print out a message, receive input from the user, and store that input into a string.

Understanding the Program

```
# get the radius from the user as a string
radius_string = input("Enter the radius of your circle: ")

# convert the radius string to a real number.
radius = float(radius_string)

# compute and print the circumference
pi = 3.14159
circumference = radius * 2 * pi
print("Circumference = ", circumference)

# compute and print the area
area = (radius ** 2) * pi
print("area = ", area)
```

Type conversion:

radius_string is a string, meaning that it is a variable that stores text. Instead, we are interested in the contents of radius_string as a number. The int function is a PREDEFINED Python function, its job is to convert a string into a number.

Understanding the Program

```
# get the radius from the user as a string
radius_string = input("Enter the radius of your circle: ")

# convert the radius string to a real number.
radius = float(radius_string)

# compute and print the circumference
pi = 3.14159
circumference = radius * 2 * pi
print("Circumference = ", circumference)

# compute and print the area
area = (radius ** 2) * pi
print("area = ", area)
```

Assignments:

These lines perform numerical calculations, and store the results of those calculations in variables.

Understanding the Program

```
# get the radius from the user as a string
radius_string = input("Enter the radius of your circle: ")

# convert the radius string to a real number.
radius = float(radius_string)

# compute and print the circumference
pi = 3.14159
circumference = radius * 2 * pi
print("Circumference = ", circumference)

# compute and print the area
area = (radius ** 2) * pi
print("area = ", area)
```

Printing results:

These lines print out results. print is a predefined Python function. It prints out strings that we text, as well as values of variables.

Note separation by comma.

Modifications

```
# get the radius from the user as a string
radius_string = input("Enter the radius of your circle: ")

# convert the radius string to a real number.
radius = float(radius_string)

# compute and print the circumference
pi = 3.14159
circumference = radius * 2 * pi
print("Circumference = ", circumference)

# compute and print the area
area = (radius ** 2) * pi
print("area = ", area)
```

How would you modify this program to print “The area of the circle is “ instead of “area = “?

Modifications

```
# get the radius from the user as a string
radius_string = input("Enter the radius of your circle: ")

# convert the radius string to a real number.
radius = float(radius_string)

# compute and print the circumference
pi = 3.14159
circumference = radius * 2 * pi
print("Circumference = ", circumference)

# compute and print the area
area = (radius ** 2) * pi
print(“The area of the circle is ", area)
```

How would you modify this program to print “The area of the circle is “ instead of “area = “?

Changing Variable Names

```
# get the radius from the user as a string
radius_string = input("Enter the radius of your circle: ")

# convert the radius string to a real number.
radius = float(radius_string)

# compute and print the circumference
pi = 3.14159
circumference = radius * 2 * pi
print("Circumference = ", circumference)

# compute and print the area
area = (radius ** 2) * pi
print("area = ", area)
```

What if I want to change the name of variable radius_string to radius_text?

Changing Variable Names

```
# get the radius from the user as a string
radius_text = input("Enter the radius of your circle: ")

# convert the radius string to a real number.
radius = float(radius_text)

# compute and print the circumference
pi = 3.14159
circumference = radius * 2 * pi
print("Circumference = ", circumference)

# compute and print the area
area = (radius ** 2) * pi
print("area = ", area)
```

What if I want to change the name of variable radius_string to radius_text?

I have to simply replace all occurrences of radius_string with radius_text

The Importance of Syntax

```
# get the radius from the user as a string
radius_string = input("Enter the radius of your circle: ")

# convert the radius string to a real number.
radius = float(radius_string)

# compute and print the circumference
pi = 3.14159
circumference = radius * 2 * pi
print("Circumference = ", circumference)

# compute and print the area
area = (radius ** 2) * pi
print("area = ", area)
```

Python (like all programming languages) is very picky about syntax.

A single misplaced character can make a program not work.

Note the syntax used in this program, and make sure you use the SAME syntax in your code.

Syntax in this Program

```
# get the radius from the user as a string
radius_string = input("Enter the radius of your circle: ")

# convert the radius string to a real number.
radius = float(radius_string)

# compute and print the circumference
pi = 3.14159
circumference = radius * 2 * pi
print("Circumference = ", circumference)

# compute and print the area
area = (radius ** 2) * pi
print("area = ", area)
```

# in the beginning of comment lines.

parentheses and quotes when we use the input function

Quotes and commas when we use the print function

The Importance of Style

- Original program:

```
# get the radius from the user as a string
radius_string = input("Enter the radius of your circle: ")

# convert the radius string to a real number.
radius = float(radius_string)

# compute and print the circumference
pi = 3.14159
circumference = radius * 2 * pi
print("Circumference = ", circumference)

# compute and print the area
area = (radius ** 2) * pi
print("area = ", area)
```

The Importance of Style

- Alternative version of the same program:

```
a = input("Enter the radius of your circle: ")
b = float(a)
c = 3.14159
d = b * 2 * c
print("Circumference = ", d)
e = b ** 2 * c
print("area = ", d)
```

- Both versions will run EXACTLY the same.
- What makes the previous version preferable?

The Importance of Style

- Alternative version of the same program:

```
a = input("Enter the radius of your circle: ")
b = float(a)
c = 3.14159
d = b * 2 * c
print("Circumference = ", d)
e = b ** 2 * c
print("area = ", d)
```

- Both versions will run EXACTLY the same.
- What makes the previous version preferable?
- Readability. Makes code easier to verify and correct.

The Importance of Style

- Alternative version of the same program:

```
a = input("Enter the radius of your circle: ")
b = float(a)
c = 3.14159
d = b * 2 * c
print("Circumference = ", d)
e = b ** 2 * c
print("area = ", d)
```

- Specific differences:
  - Lack of comments
  - Non-descriptive variable names
  - Lack of empty lines to separate “blocks” of code

Some Guidelines

- To learn how to code, you need PRACTICE.
  - What will usually not work:
    - Listen to the lectures.
    - Go and try to do the assignments.
  - What will usually work:
    - Listen to the lectures and KEEP NOTES.
    - Actually run every piece of code that we do in class.
    - Understand every line of every piece of code we do in class.
    - Think of variations of what we do in class, and try them.
      - Predict what the variation will do, and verify by running it.
    - Then try the assignments.

Some Guidelines

- You need to understand the terminology:
  - Statements, expressions, tokens, literals, functions, strings, variables, operators, …
- You will encounter many terms in this course. YOU NEED TO LEARN EXACTLY WHAT THEY MEAN.
- DO NOT RELY ON ENGLISH. These terms have meanings in conversational English that are only vaguely related with their meaning in programming.

Terms We Have Seen So Far:

- Command prompt
- Text file
- Filename
- Folder
- Operator
- Variable
- Function
- Running a program