Programming with Python

Section 1

Updated and modified by: Siamak Sarmady, UUT (www.uut.ac.ir)

Sept, 2016, v1.0

Why Python

- Clean, Simple and compact syntax.
- Similar to MATLAB and a good language for doing scientific as well as general computing (web, system, data processing etc.)
- Easy to combine Python with compiled languages like C, C++ and Fortran (as external modules)

- We use python 2.XX (last revision)
- You can use Python 3.XX if you want. There are slight changes

Simple Calculations

Our first example program: position of a ball thrown up at any moment y

$$y(t) = v_0 t - \frac{1}{2} g t^2$$

$$y = 5 \cdot 0.6 - 0.5 \cdot 9.81 \cdot 0.6^2$$

Method 1: Type python on command line to bring up the python shell. Then enter the statement:

```
>> print 5*0.6 - 0.5*9.81*0.6**2
```

Method 2: We can put the above statement in a file (e.g. ball1.py) and then run it:

```
E:\> python ball1.py
```

The program prints out `1.2342` in the terminal window.

Variables

We can use variables to store values for each parameter:

```
v0 = 5
g = 9.81
t = 0.6
y = v0*t - 0.5*g*t**2
print y
```

Note: In python, variables are weakly typed; you don't specify a type for the variables. C/C++ are strongly typed. You specify a type during definition of variable.

Note:

- The name of a variable in a program can contain the letters a-z, A-Z, underscore _ and the digits 0-9, but cannot start with a digit
- Variable names are case-sensitive (e.g., `a` is different from `A`)
- In this book variable names are lower case and words are separated with _
- You may use all uppercase letters for Constants e.g.: MY_CONSTANT

```
initial_velocity = 5
GRAVITY_ACCELERATION = 9.8
```

Comments

Program with comments:

```
# program for computing the height of a ball
# in vertical motion
v0 = 5     # initial velocity
g = 9.81   # acceleration of gravity
t = 0.6    # time
y = v0*t - 0.5*g*t**2   # vertical position
print y
```

Note:

- Everything after `#` on a line is a comment and ignored by Python
- Comments are used to explain what the computer instructions mean, what variables mean, how the programmer reasoned when she wrote the program, etc.

Note: If you want to use special or non-English characters, enter this comment line as the first line in your program or stick to English everywhere in a program.

```
# -*- coding: utf-8 -*-
```

Printf syntax for formatting text with numbers

```
t = 0.6; y = 1.2342
print 'At t=%g s, y is %.2f m.' % (t, y)
```

Inside the string we specify the slots. The slots will later be replaced with variable values (following a % sign).

```
%s       a string
%d       an integer
%0xd     an integer padded with x leading zeros
%f       decimal notation with six decimals
%e       compact scientific notation, e in the exponent
%E       compact scientific notation, E in the exponent
%g       compact decimal or scientific notation (with e)
%G       compact decimal or scientific notation (with E)
%xz      format z right-adjusted in a field of width x
%-xz     format z left-adjusted in a field of width x
%.yz     format z with y decimals
%x.yz    format z with y decimals in a field of width x
%%       the percentage sign % itself
```

Modules and packages

1. **Library:** a collection of useful program pieces.
2. **Libraries composed of:** packages and modules.
3. **Packages:** collection of modules.
4. **math module:** contains standard mathematical functions like sin x, ln x, e^x etc.

Program Development Flow

1. **High level pseudo code:** Write an algorithm using very high level pseudo code (i.e. only 4-10 lines)
2. **Low level pseudo code:** Write the algorithm using more detailed pseudo code (use flowchart for very complicated parts)
3. **Code:** Start converting the algorithm to programming language code
4. **Module testing:** Check individual functions and modules. Make sure they produce intended results.
5. **Validation:** Verify and validate the overall results of your program.

A program consists of statements

A program consists of statements of different types:

```
a = 1      # 1st statement (assignment statement)
b = 2      # 2nd statement (assignment statement)
c = a + b  # 3rd statement (assignment statement)
print c    # 4th statement (print statement)
```

multiple statements per line is possible with a semicolon in between the statements:

```
a = 1; b = 2; c = a + b; print c
```

Indentation blanks are important in Python programs. They mark a block of code:

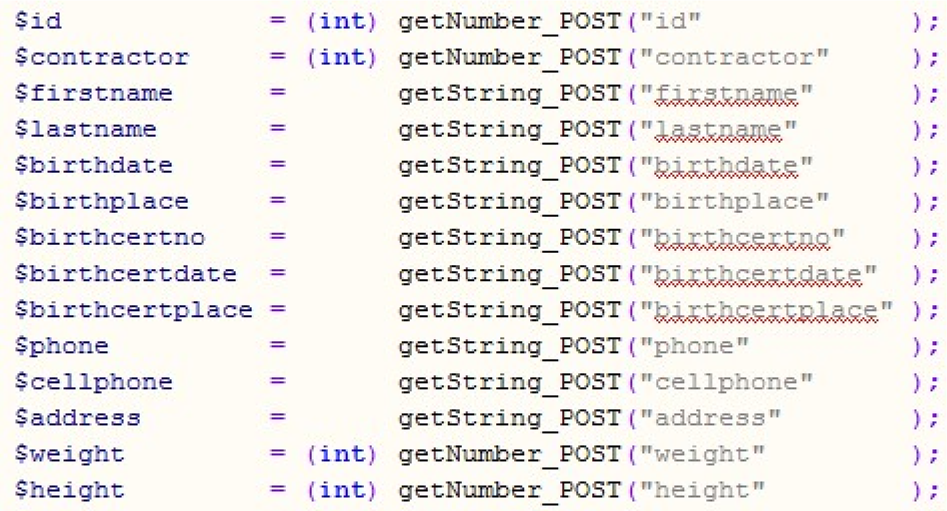
```
counter = 1
while counter <= 4:
    counter = counter + 1   # correct (4 leading blanks)

while counter <= 4:
counter = counter + 1       # invalid syntax
```

Spacing and orderly coding

Except the indentation space, spaces could be used freely to write clean code.

The following shows an example of a PHP code. As you see the code seems clean and readable.



```
$id             = (int) getNumber_POST("id"             );
$contractor     = (int) getNumber_POST("contractor"     );
$firstname      =       getString_POST("firstname"      );
$lastname       =       getString_POST("lastname"       );
$birthdate      =       getString_POST("birthdate"      );
$birthplace     =       getString_POST("birthplace"     );
$birthcertno    =       getString_POST("birthcertno"    );
$birthcertdate  =       getString_POST("birthcertdate"  );
$birthcertplace =       getString_POST("birthcertplace" );
$phone          =       getString_POST("phone"          );
$cellphone      =       getString_POST("cellphone"      );
$address        =       getString_POST("address"        );
$weight         = (int) getNumber_POST("weight"         );
$height         = (int) getNumber_POST("height"         );
```

Error is caused by (unintended) integer division

Given *C* as a temperature in Celsius degrees, compute the corresponding Fahrenheit degrees F.

```
C = 21
F = (9/5)*C + 32
print F
```

Result:

```
53
```

Using a calculator: 9/5 times 21 plus 32 is 69.8, not 53. In python 2.XX dividing two integers will be performed as integer division. Corrected program (with correct output 69.8):

```
C = 21
F = (9.0/5)*C + 32
print F
```

In Python 3.X and MATLAB division of two integers will still produce a float.

Everything in Python is an object

Variables refer to objects (holding information about the variable):

```
a = 5       # a refers to an integer (int) object
b = 9       # b refers to an integer (int) object
c = 9.0     # c refers to a real number (float) object
d = b/a     # d refers to an int/int => int object
e = c/a     # e refers to float/int => float object
s = 'b/a=%g' % (b/a)  # s is a string/text (str) object
```

We can convert between object types:

```
a = 3              # a is int
b = float(a)       # b is float 3.0
c = 3.9            # c is float
d = int(c)         # d is int 3
d = round(c)       # d is float 4.0
d = int(round(c))  # d is int 4
d = str(c)         # d is str '3.9'
e = '-4.2'         # e is str
f = float(e)       # f is float -4.2
```

Precedence in Arithmetic expressions

- In python, terms are evaluated from left to right.
- Terms are the sections separated with + or –
- Parenthesis increases the precedence to the highest (they are evaluated from left to right, from inner to outer).
- In each term power ** has higher priority. Then multiplication * and division / are evaluated with equal precedence from left to right.

Precedence in Arithmetic expressions

- Example: $\frac{5}{9}+\frac{2a^4}{2}$, in Python can be written as 5.0/9 + 2*a**4/2.0

Evaluation is done as follows:

• r1 = 5.0/9	Terms from left to right, first term (0.55)
• r2 = a**4	Second term starting from power (16.0)
• r3 = 2*r2	Now mult. And div. from left to right (32.0)
• r4 = r3/2.0	(16.0)
• r5 = r1 + r4	(16.55)

Standard mathematical functions are in the `math` module

- What if we need to compute $\sin x$, $\cos x$, $\ln x$, etc. in a program? Such functions are available in Python's math module
- In general: lots of useful functionality in Python is available in modules - but modules must be *imported* in our programs

Compute 2√ using the sqrt function in the math module:

import the module (make it available):

```
import math
r = math.sqrt(2)
```

or import a function from the math module:

```
from math import sqrt
r = sqrt(2)
```

or import all functions from in the math module:

```
from math import *   # import everything in math
r = sqrt(2)
```

Another example with functions from math module

```
from math import sin, cos, log
x = 1.2
Q = sin(x)*cos(x) + 4*log(x)    # log is ln (base e)
print Q
```

Round-off errors

```
v1 = 1/49.0*49
v2 = 1/51.0*51
print '%.16f %.16f' % (v1, v2)
```

Output with 16 decimals becomes

```
 0.9999999999999999 1.0000000000000000
```

- Real numbers are represented using only 16 digits on a computer. As a result calculations will not be exact and will have very small error

Notice: Python has a module called "decimal" that allows real numbers to be represented with adjustable accuracy. So the round off error can be made even smaller.

Using Python interactively (like a calculator)

- So far we have performed calculations in Python *programs*
- Python can also be used interactively in what is known as a *shell*
- Type `python`, `ipython`, or `idle` in the terminal window
- A Python shell is entered where you can write statements after `>>>` (IPython has a different prompt)

```
Terminal> python
Python 2.7.6 (r25:409, Feb 27 2014, 19:35:40)
...
>>> C = 41
>>> F = (9.0/5)*C + 32
>>> print F
105.8
>>> F
105.8
```

Notice: Previous commands can be recalled and edited (with up and down buttons)

Notice: In interactive mode, a variable can be printed by just typing its name.