Introduction to PYTHON

Python 001 – Print I

The "print" command is used to print information to the output screen.

```
print "Hello World!"
```

...prints "Hello World" (without the quotes) to the screen.

Assignment

Write a program that prints your first name to the screen.

Save as "001.py".

Note that all programs should be saved in your "Name - Python Tutorial" folder on the network drive from now on. See your instructor if you are unsure how to do this.

Python 002 – Print II

Type in the following program, then identify and fix any errors:

```
print "Hi, I am a computer
"print Hi, I am a computer"
Print Hi, I am a computer
```

Assignment

Correct the errors and save as "002.py"

Python 003 – Program Header

All programs should include a “program header”. This header should, as a minimum, include the author’s (programmer’s) name, the date, the filename, and a program description. An example header is shown below:

```
# author: Jane Doe
# date: 2010-03-26
# filename: put2.py
# description: prints user’s name to the screen
```

Note that a "#” sign is used before each of the header lines. These are used to tell the program to ignore these lines and treat them like comments.

For most programs written in this class, you may copy and paste the program requirements into the "description" part of the header. If it is too wide to fit on the screen, break it up into two or more lines.

Assignment

Add an appropriate program header to 001.py. Save as "003.py".

Python 004 – Concatenation I

By appending a comma (",") to a print statement, the next print statement will print its output on the same line as the first statement. Example:

```
print "Nice",
print "picture!"
```

...prints "Nice picture!" on the screen. Note that when printed in this manner (using a comma), a space is inserted between the two printed objects. In the truest sense, using a comma to join two printed objects is not concatenation, but this following example is:

```
print "Nice" + "picture"
```

If you run this second example you will note that the output does not include a space between the two objects. This is true concatenation, i.e., joining of two strings end-to-end.

Assignment

Add a line to 003.py that prints your last name after your first name. Be sure it prints on the same line.

Save as "004.py".

Python 004a – Concatenation II

Assignment

As in the assignment above, print your first and last names, but this time print them using a single print statement and the "+" operator. What did you have to do to include a space between your first name and last name?

Save as "004a.py".

Python 005 – Addition I

The addition operator in Python is the plus sign ("+").

The following statement adds the numbers 3 and 4 and prints the sum to the screen:

```
print 3 + 4
```

Assignment

Write a program that prints the sum of 5, 6, and 7.

Save as "005.py".

Python 006 – Addition II

Assignment

Write a program that prints the sum of 5.123 and 6.29.

Save as "006.py".

Python 007 – Addition III

Assignment

Modify 006.py so that there are two print statements. The first prints the two values to be added, and the second does the math.

The output should look similar to "The sum of __ and __ is __. ".

Save as "007.py".

Python 008 – Subtraction

The subtraction operator in Python is a hyphen (“-”).

The following statement subtracts the number 3 from 4 and prints the result (1):

```
print 4 - 3
```

Assignment

Write a program that subtracts 5 from 6.

Save as "008.py".

Note: don't forget that each of your programs should have a proper program header!

Python 009 – Multiplication

The multiplication operator in Python is an asterisk ("*").

The following statement multiplies the numbers 3 and 4 and prints the result (12):

```
print 3 * 4
```

Assignment

Write a program that prints the product of 5 and 6.

Save as "009.py".

Python 010 – Division

The division operator in Python is a slash (“/”).

The following statement divides the number 3 by 4:

```
print 3 / 4
```

Assignment

Write a program that prints the quotient of 5 divided by 6. Save as "010.py".

Python 011 – Exponents

The exponent operator in Python is two asterisks (“**”).

The following statement calculates the square of 3 (3^2):

```
print 3 ** 2
```

Assignment

Write a program that prints the answer of 2 to the exponent 8.

Save as "011.py".

Python 012 – Order of Operations

Remember from math class that 3 + 2 X 5 = 13, not 25. The same holds true for Python.

Even though Python will calculate algebraic equations properly, it is good programming practice to group items into their intended order of operation by surrounding them with parentheses (round brackets). For example, the above equation, written with a Python print statement, is better written as:

```
print 3 + (2 * 5)
```

A more complex equation may be written as:

```
print 12 + ( ( 4 - 6 ) / 3 )
```

Things to remember:

- Python uses '*' for multiplication, not 'x'
- Python uses '**' for exponentiation
- Python only uses parentheses (round brackets), not square brackets
- Brackets can be nested (i.e., inside each other), and it is strongly encouraged if it makes the statement clearer

Assignment

Write a program that prints the equivalent of each of the following algebraic statements to the screen. Don't forget to use parentheses for clarity.

- 2 + 4 x 5
- 2 x 6 - 4/2
- 8 + 2
- (4+9/3)
- 3[(4+12)-2(3-1)]
- 23 + 42 - 6/3

For each output statement, copy and paste the above formulas exactly as shown, then append your calculated answer. For example, the 5th line would look something like this:

```
3[(4+12)-2(3-1)] = <your calculated answer here>
```

Save as "012.py".

Python 013 – Concatenation II

As learned previously, a comma can be used to separate individual items that you want to print. For example:

```
print "2 + 2 =", 2 + 2
```

...prints "2 + 2 = 4" on the screen.

Assignment

Write a program that prints the 12 times table, from 12 X 1 to 12 X 12.

Each line output to the screen must be created using a single put statement, and the product of each line must be calculated. For example, if the assignment were to add 5 to all numbers from 1 to 12, the first line would be:

put "1 + 5 =", 1 + 5

Save as "013.py".

Python 014 – Field Widths

A field width specification can be included to tell Python how much space you want something to use when printed on the screen. This is useful when you are printing columns of data or information, such as on a sales receipt. For example:

```
# prints the first line
print "%12s" % "Item",
print "%12s" % "Quantity",
print "%12s" % "Cost",
print "%12s" % "Total"

# prints the second line
print "%12s" % "Apples",
print "%12s" % "3",
print "%12s" % "0.75",
print "%12s" % "2.25"
```

...prints the following:

```
        Item    Quantity        Cost       Total
      Apples           3        0.75        2.25
```

Note that this example uses quoted "strings" for the printed text. If you were to print numbers (or calculated numbers) you would use "d" for integer numbers (such as 10, 998, or 10567) or "f" for real numbers such as 1.12, 10.98, or 987.176, as this example below demonstrates:

Also note that real numbers are padded with trailing zeroes. To designate the number of decimal places in a real number, add a decimal to the field width, such as "%12.2".

This example demonstrates the use of field width with numbers:

```
print "%10d" % 198
print "%10.2f" % 198
print "%10f" % 209.87
```

...produces:

```
       198
    198.00
209.870000
```

Assignment

Create a neatly-spaced "times table" from 12 x 1 to 1 x 12 that uses field widths. Use formulas to calculate the products.

```
12    X    1     =    12
12    X    2     =    24
12    X    3     =    36
12    X    4     =    48
12    X    5     =    60
12    X    6     =    72
12    X    7     =    84
12    X    8     =    96
12    X    9     =    108
12    X    10    =    120
12    X    11    =    132
12    X    12    =    144
```

Save as "014.t".