Prelab2

We encourage you to work together on the Pre Lab. The Pre Lab is not graded but will help you prepare for your lab session. In the Pre Lab, you may find questions to answer. We do not require you to provide us the answers, but we do recommend you to try to answer these questions. If you have any questions on the material in the pre lab, first check the book and recitation slides, if you do not find your answer please email your recitation TA or the course instructors.

1. Variable and Expressions

A variable is used to give a name to a value so that you can refer to it at other points in a program. So, basically a variable is a name. Variables are called that way because their value can be changed over time. You can roughly think of a variable as the name put on on a mailbox. The name of the mailbox (that is the variable name) is fixed, but the content of the variable can change over time. Think of, by analogy, as the content of a real mailbox changes each time the postman put some new mail into it.

Variable names in Python MUST start with a character. Below are some examples of valid variable names:

- a
- x
- X
- y
- myvar
- myname

A variable name can also contain numbers. Examples:

- var1
- var2
- var10

A variable name cannot start with a number, and cannot contain a blank character. If you do that, Python will give you a syntax error:

```python
>>> 1myname
SyntaxError: invalid syntax
>>> my name
SyntaxError: invalid syntax
>>> 
```

Now, how to tell Python that a certain name, say `a` is a variable? Well, you need to **assign a value to it**:

```python
a = 10
```

Read the equal sign as a "arrow, meaning that the value at the right of the equal sign is assigned to the variable name in the left side:
a = 10 means a <- 10

If you want to see the effect of your assignment, that is, the value of the variable a, you can just ask Python to print it:

```python
>>> a = 10
>>> print(a)
10
```

print is a Python pre-defined function. Note that you can see the current value of a variable just by typing the variable name:

```python
>>> a = 10
>>> a
10
```

Variables can contain integer numbers, decimal numbers, string characters. Examples:

- `x = 100`
- `X = 33.33`
- `myname = 'John'`

**TRY IT:**

Just writing X in your Python IDLE Shell window results in an error!

```python
>>> X
Traceback (most recent call last):
  File "<pyshell#22>"`, line 1, in <module>
    X
NameError: name 'X' is not defined
```

But:

```python
>>> X = 10
>>> print(X)
10
```

is correct. The X = 10 is an example of an **assignment statement**. Note also that once you defined a variable, when you type the variable name in the Python Shell window, the (last) value of the variable is shown (see above).

Now, let's change the value of a variable.

```python
>>> X = 10
>>> print(X)
10
>>> X = 50
```
You can see that now the value of X is 50!

You can assign a variable not only a value, but also an expression. Look at the following:

```
>>> X = 10
>>> print(X)
10
>>> X = 50
>>> print(X)
50
>>> X = X + 1
>>> print(X)
51
```  

X + 1 is an expression. In this case, Python evaluates first the expression to the right of the equal sign and assign the result of the evaluation to the variable in the left side. So, read the assignment X = X+1 as:

- get the current value of variable X
- add 1 to it
- assign the new value to the variable X

**CAVEATS**

- The characters can be lower case or upper case. But be careful! Python is case sensitive! Hence for Python the variable x is different from the variable X! So, use meaningful variable names.

- Provided that your variable name starts with a character, can you use any name you want? Well, Python has some reserved words, that can not be used as variable names. You can find them here: http://docs.python.org/py3k/reference/lexical_analysis.html#keywords.

**TIPS & SUGGESTIONS**

TIP1: we do suggest you to use meaningful variable names, so that you, or another person reading your program can more easily understand what kind of values the variable will hold. Programs are executed by the computer, but they must be understood by other humans too!

**2. How to get the input from the user**

Often your program needs to get some data from the user. The data entered by the user then need to be stored in some variable, so that they can be used in the program. The program may need from the user a string of characters, or it may need some numeric data. To get a string of characters, Python provides the input statement.

Here is an example:

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

Executing the input statement causes Python to print out the prompt “Enter your name: ”. Then Python pauses expecting the user input. After the user entered John, the value John is assigned to the variable name.

However, your program may need to get numeric data from the user, rather than a string of characters. In this case you need to use a slightly different form of the input statement. For example, look at the program convert.py that converts a temperature from Celsius to Fahrenheit degrees.

    >>> celsius = eval(input("What is the Celsius temperature? "))
    What is the Celsius temperature? 100
    >>> print(celsius)
    100
    >>>

You can see that the variable celsius is assigned the value 100 after the execution of the statement above. You can now use the variable celsius in any arithmetic statement.

If you use the eval(input()) statement to input a string of character you will get an error:

    >>> myName = eval(input("Enter your name: "))
    Enter your name: John
    Traceback (most recent call last):
      File "<pyshell#32>", line 1, in <module>
        myName = eval(input("Enter your name: "))
      File "<string>", line 1, in <module>
    NameError: name 'John' is not defined
    >>>

TRY TO ANSWER:

Q1: Suppose you want to print the following string of characters: 'Hello Hello Hello'.

Find at least two different ways of doing that.

Q2: How can you print Hello n times, where n changes?

4. Python built-in functions

Python provides you a set of so-called built-in functions. Built-in functions are always available, whereas if you want to use a function provided by a Python library, you must first of all import the library. print() is a built-in function. You can find the complete list of Python built-in functions here: http://docs.python.org/library/functions.html.
From:
http://courses.cs.purdue.edu/ - Computer Science Courses

Permanent link:
http://courses.cs.purdue.edu/cs17700:prelab2

Last update: 2012/08/26 11:00