Writing programs

COR 100.13 / Year 1 - Game programming with Python

Marcus Birkenkrahe

September 24, 2024

Using string values

- In Python, text values are called *strings*. They can be used just like integer or float values, and you can store them in variables.
- Python recognizes text values when they are enclosed in (single or double) quotation marks: ‘"spam"‘ is a string, ‘spam‘ is a variable.
- Store the string ‘hello‘ in the variable ‘spam‘, then ‘print‘ it:

```
spam = 'hello'
print(spam)
```

- Strings can have any keyboard character in them and they can be as long as you like:

```
print("ijdinfnns d    \n   ***&&^6///34/$$$\n\
once upon the time in a galaxy far, far away...")
```

- In the last example, Python recognized two special characters: “ (new line) and ‘\‘ (continue here).
- Strings can be *concatenated*: enter ‘'Hello' + 'World!"‘ in the next code block:

```
print('Hello'+'World!')
```

- What's behind this? All values have a *data type* (integer, float, string, or Boolean), and the ‘+‘ operator works differently on them:

```
print(2 + 2) # add two integers
print(2. + 2.) # add to floats
print('2' + '2') # add two strings
print(True + False) # add two Booleans (True is 1 and False is 0)
```

Creating the Hello World program

- It's traditional for programmers to make their first program display 'Hello world¡ on the screen
- This goes back to the first proper programming manual written by Kernighan and Ritchie for the C programming language, and it stuck (this seminal book also has its own Wikipedia page).
- There's also a free C programming course by Dr. Chuck where he essentially reads the book and runs its code (freeCodeCamp).
- Enter the following code in the code block:
  1. '# This program says hello and asks for my name'
  2. 'print('Hello world!')'
  3. 'print('What is your name?')
  4. 'name = input()'
  5. 'print('It is good to meet you, ' + name)

```
# This program says hello and asks for my name'
print('Hello world!')
print('What is your name?')
name = input()
print('It is good to meet you, ' + name)
```

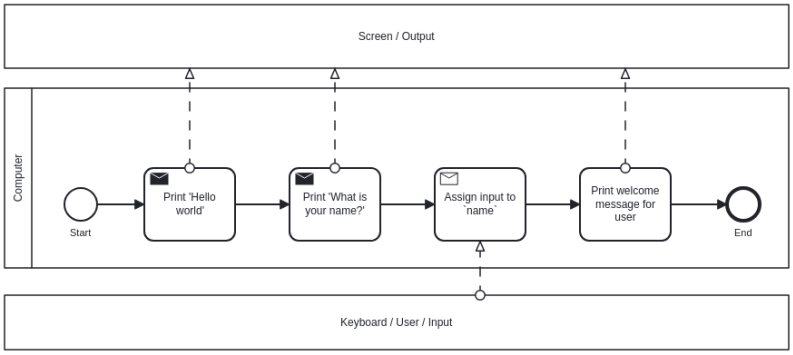
- Let's analyze the program:
  1. This is a comment - Python ignores everything after '#'
  2. Displays a string on the screen - the program title.
  3. Displays a string on the screen - a question for the user.
  4. Asks for keyboard input and assigns it to the variable 'name'.
  5. Concatenates the welcome and the value 'name' of name & prints them.

- When you write your own programs, it is useful to add this information to the code using comments:

```
# This program says hello and asks for my name‘
print(’Hello world!’)  # Print title
print(’What is your name?’)  # Ask user for input
name = input() # assign keyboard input to ‘name‘
print(’It is good to meet you, ’ + name) # print result as concatenation
```

- In practice, you would first write the comments as a form of pseudocode, and/or put them in a process model. The more complicated a program is, and the more people are working on it, the more important it is that you follow these development practices!

- BPMN Process model for the Hello world program (created at bpmn.io):



Getting input, printing output

- ‘print()‘ and ‘input()‘ are built-in functions.

- The value between the parentheses of a function is called its *argument*, like for mathematical functions $f(x)$.

- When the function is called, the argument is passed to the function for evaluation.

- Examples:

```
# Print the string argument "hello world"
print("hello world")
```

```
# Print the number argument 2
print(2)

# Print the value of the expression 2 + 2
print(2+2)
```

- You can get short help on any function (or keyword) with the ‘help‘ function. In the next code block, pass the name of the ‘print‘ function as an argument to the ‘help‘ function:

```
help(print)
```

- Do the same thing for ‘input‘: get ‘help‘ using the ‘help‘ function

```
help(input)
```

- Both ‘help‘ texts contain a lot of technical information that you may not understand (yet). Especially when you encounter a new function, it’s worth going down the rabbit hole of documentation to understand absolutely everything that the ‘help‘ can tell you.

- Unfortunately, the official Python documentation for input and output at python.org is not any clearer.

- Here is the ‘input‘ command from the program again:

```
name = input()
print('Hello, ' + name)
```

- What happens here? The function ‘input‘ is called without an argument. As the ‘help‘ explains, it reads "a string from standard input". Standard input (*stdin*) in this case means the keyboard.

- Standard input could also be passed to a Python script: after tangling the single command above as a Python file ‘input.py‘, it can be run on the command line if ‘input‘ is a file containing input:

```
echo 'Marcus' > inputFile
python3 input.py < inputFile
```

Forgetting and naming variables

- What happens to the variables when the program is finished?
- It depends:
    1. If you're working in an interactive notebook like an IPython shell, or in Emacs Org-mode, the variables are alive as long as the notebook session is running.
    2. If you run a program on the command line like 'python3 input.py' above, everything is gone when the program is finished.
- Your variables have to be named by you. There are a few rules and recommendations for that:
    1. Don't start a name with anything but a (lowercase) letter (underscores are reserved, numbers or operators are not allowed)
    2. Observe the fact that variable names are case-sensitive: 'SPAM' is not the same as 'spam'.
    3. You must not have whitespace (empty characters) within the name.
    4. Variable names are usual lower case. You can form longer names either by connecting them with underscore '_' or with *camelCase*: for example: '$\text{my}_{\text{number}}$' or 'myName'.

Exercise: Practicing Python Variable Naming Conventions

This exercise is designed to help you practice proper variable naming conventions in Python.

Instructions:

1. Identify and correct the invalid variable names in the list below.
2. For each corrected name, explain why the original name was invalid and how the corrected name follows the rules.

Invalid Variable Names:

```
1stNumber = 10
My Number = 5
```

```
sum-total$ = 50
_spam_eggs = 12
SPAM = "hello"
```

Task:

- Correct each of the variable names based on the rules provided.
- Make sure to follow these rules:
  - Variable names must start with a lowercase letter.
  - No special characters (like '$' or spaces) are allowed in variable names.
  - Variable names are case-sensitive.
  - Use either underscores '_' or camelCase for multi-word variable names.

Example Answer:

```
# 1stNumber is invalid because it starts with a number.
first_number = 10

# My Number is invalid because it contains a space.
my_number = 5

# sum-total$ is invalid because it contains a special character ($).
sum_total = 50

# _spam_eggs is technically valid but underscores at the beginning are reserved for sp
spam_eggs = 12

# SPAM is valid, but variable names are usually lowercase. A better name could be:
spam = "hello"
```

Feel free to apply similar corrections and explanations to the rest of the variables!

Summary

- All values have a data type (float, integer, string, or Boolean).
- Strings must be enclosed in single or double quotation marks.

- Strings can be concatenated with the ‘+‘ operator.
- Functions carry out complicated instructions, they are called with or without arguments, e.g. ‘print(2)‘ or ‘input()‘.
- Functions can be used anywhere a value is used: ‘name=input()‘.