Python Programming

John Dunnion

School of Computer Science
University College Dublin

Variables

- A variable is one of the most important elements of a programming language
- A variable can be thought of as a named/labelled storage location for data in memory
- It's called a *variable* because its contents can change during the execution of the program
- It is a **storage location**, ie Python will reserve some memory to store the data.
- The *value taken by the variable* will be stored at that location

Using variables (1)

- Running the following program:

```
# Greeting program, v1.0
# Demonstrates the use of a variable

print('Good␣morning!')

firstname = 'John'
print('Hi␣' + firstname)
print('How␣are␣you?')
```

produces

```
Good morning!
Hi John
How are you?
```

Assignment

- An assignment statement gives a variable a value

  firstname = ' John '

- In Python, the assignment operator is denoted by the “=” character
- A variable is just a name
- The assignment statement associates the name on the left of the assignment symbol with the value on the right of the assignment symbol
- We say that the variable is assigned a value
  - `firstname` is assigned the value `"John"`
  - `firstname` is given the value `"John"`
  - `firstname` becomes the value `"John"`
- NB The “=” character is not the equals we use in mathematics!

Using variables (2)

- In our program, the statement

  firstname = ' John '

  creates a variable *firstname* containing the string“`John`”
- Note that when we use the variable in an expression or in a statement, the contents of the variable are used
- Recall that the output of our program has

  ```
  Hi John
  ```

  not

  ```
  Hi firstname
  ```

Using variables (3)

- The contents of a variable can be changed
- We simply have another assignment

```
# Greeting program, v2.0
# Demonstrates the use of a variable
name = 'John'
print('Hi␣' + name + '!')
print('How␣are␣you?')

# Get a new value of name
name = 'Mary'
print('Oh!␣You\'re␣' + name + 'now!')
```

produces

```
Hi John!
How are you?
Oh! You're Marynow!
```

Mind the gap!

- Correcting the output of our previous program:

```
# Greeting program, v2.1
# Demonstrates the use of a variable

name = 'John'
print('Hi␣' + name + '!')
print('How␣are␣you?')

# Get a new value of name
name = 'Mary'
print('Oh!␣You\'re␣' + name + '␣now!')
```

produces

```
Hi John!
How are you?
Oh! You’re Mary now!
```

Naming variables (1)

- A variable name can only contain the following:
  - letters (lowercase and uppercase, ie a–z and A–Z)
  - digits (0–9)
  - the “_” character
- A variable name cannot start with a digit
- Variable names in Python are case-sensitive
- `name` and `Name` are different variables
- There are a small number of reserved words or keywords that have built-in meanings in Python and cannot be used as variable names
- The different versions of Python have slightly different lists of reserved words

Naming variables (2)

- Choose *descriptive* names
- When you re-read your program in two weeks' time, or in a year's time, you will be grateful!
- When your team colleague reads your program in two years' time, after you've moved to a new section in the company, they (and you) will be extra grateful!
- For example, `tax_due` is a better name than `name` or `var3` or `x1234` or even `td`

Naming variables (3)

- Consider the following two programs:

```
# Greeting program, v3.0
# Demonstrates the use of variable names

name = 'John'
print('Hello ␣' + name + '!')
```

and

```
# Greeting program, v3.1
# Demonstrates the (bad) use of variable names

x = 'John'
print('Hello ␣' + x + '!')
```

- What is the difference in the output?
- None!

Don’t rely on variable names. . . (1)

- The fact that a variable is called a particular name does not confer on it any particular properties
- For example, a variable called `name` does not necessarily hold names (although clearly that would be a good idea)
- If the name of a variable called `name` is changed everywhere in the program to `abcxyz`, the program will run in exactly the same way
- Recall that the Python interpreter (and the compilers/interpreters for other languages) translates the source code into code that the machine can execute
- So the variable names are for the **benefit/convenience** of the programmer or (human) reader of the program, not the computer

Don’t rely on variable names. . . (2)

- Consider the following two programs:

```
# Greeting program , v4 . 0
# Demonstrates the further use of variable names

greeting = ' Hello '
name = ' John '
print ( greeting , name)
```

and

```
# Greeting program , v4 . 1
# Demonstrates the further (bad) use of variable

name = ' Hello '
greeting = 'John '
print ( greeting , name)
```