COMP 10280
Programming I (Conversion)

John Dunnion

School of Computer Science
University College Dublin

COMP 10280 Programming I (Conversion)/Lecture 3

Outline

Primitive elements of a programming language

- Each programming language has a set of primitive constructs:
  - Syntax
  - Static Semantics
  - Semantics

Syntax

- The syntax of a language defines the strings of characters and symbols that are allowed in the language
- The syntax defines the structure of the clauses/sentences/statements in a language
- Natural language also has syntax:
  - The cat sat on the mat.
  - Tá mé fuar.
- Rules of syntax must be followed:
  - Cat sat mat on the the.
  - Mé tá fuar.
- 3 + 3 is syntactically valid in Python
- 3 3 is not syntactically valid

Static semantics

- The static semantics of a language defines which syntactically valid strings in a language have a meaning
- I am big [Valid]
- <noun> <verb> <adjective>
- I are big [Not valid]
- 3.2 * 3.2 [Valid]
- 3.2 * 'abcd' [Not valid]

Semantics

- The semantics of a language associates a meaning with each syntactically correct string that has no static semantic errors
- In natural language, semantics of a sentence can be ambiguous
- In programming languages, each legal statement is unambiguous (has exactly one meaning)

Syntax errors

- Most common for novice programmers in a new language
- Least dangerous type of error
- Compiler/interpreter will find them
- Programmer usually told where an error is (and sometimes how to fix it)
- Program will not be executed

Static semantic errors

- Some programming languages (eg Java) do a lot of static semantic checking
- Other languages do less
- C, Python. . .
- Python does do a lot of static semantic checking while the program is running
- However, if it does not catch an error, the behaviour of the program is unpredictable

Semantic errors

- If a program has no syntax errors and no static semantic errors, it will run
- We don't usually say that a program has a semantic error
- The program has a meaning, ie it has semantics
- However, the semantics it has is not necessarily the semantics the programmer/designer intended it to have
- If a program's semantics is not what was intended, it is usually bad news and it can be dangerous

What might happen if the program has an error?

- It might crash
- It might keep running and never stop
- It might run to completion and produce an answer that might, or might not, be correct

Categorising Python

- Python is a high-level, general-purpose, interpreted programming language
- *High-level* rather than *Low-level*
  - Deals with abstract operations (eg put a pop-up menu on the screen)
  - (Rather than operations at machine level (eg move the contents of a 64-bit data register to a particular memory location))

Categorising Python

- *General-purpose* rather than *Domain-specific*
  - Primitive operations of the language are widely applicable
  - (Rather than fine-tuned to a particular domain or type of application
  - (eg Adobe Flash for adding interactivity and animation to Web pages))
- *Interpreted* rather than *Compiled*
  - Instructions in the program are executed directly (by an interpreter)
  - (Rather than translated (by a compiler into assembly/machine code))

Describing Python

- Python is a general-purpose language
- It can be used to build almost any kind of program that does not need direct access to the computer's hardware
- It is relatively simple
- It is easy to learn
- The interpreter gives good feedback
- The interpreter can produce error messages that are easy to correlate with the source code
- It has a huge number of libraries that provide extended functionality

Describing Python

- Interpreted programs usually run more slowly than compiled code
- Interpreted programs usually take up more space than compiled code
- It is not great for programs that have high reliability constraints(?)
- It is not great for systems that are built and maintained by many people over a long period of time(?)
- There are two versions. . .

Basic Elements of Python

- A Python program, or Python script, is a sequence of *definitions* and *commands*
- The definitions are evaluated and the commands are executed by the Python interpreter in the Python shell
- A command, or statement, instructs the interpreter to do something

Starting Python

- Start a Python shell
- Double-click on an icon
- Enter `python` at a terminal window
- Can choose Python 2 or Python 3

My first program

```
print('Hello, world.')
```

This causes the interpreter to produce the following output on the screen:

```
Hello, world.
```

Comments

- Any text following the # character is ignored by the Python interpreter
- This is called a comment
- Comments are used to enhance the readability of code for human readers

```
# My first program
print('Hello, world.')
```

This causes the interpreter to produce the following output on the screen:

```
Hello, world.
```

Printing strings

- The `print` command takes a variable number of strings (text enclosed in single quotes) and prints them, separated by a space, in the order in which they appear

```
# My second program
print('Good morning!')
print('Vietnam!')
print('Good morning,', 'Vietnam!')
```

This causes the interpreter to produce the following output on the screen:

```
Good morning!
Vietnam!
Good morning, Vietnam!
```

IDLE

- Typing programs into the shell is highly inconvenient
- Most programmers prefer to use some form of *text editor*
- Most prefer to use an integrated development environment (IDE)
- The IDE that comes as part of the standard Python distribution is IDLE
- IDLE provides:
  - A text editor with syntax highlighting, auto-completion and smart indentation
  - A shell with syntax highlighting
  - An integrated debugger
- When IDLE starts, it opens a shell window

IDLE menus

- IDLE provides the user with a number of menus
- The file menu
  - Create a new editing window
  - Open a file containing an existing Python program
  - Save the contents of the current editing window into a file (with a `.py` file extension)
- The edit menu
  - Standard text-editing commands
  - Copy, paste, find, ...
  - Commands for editing Python code
  - Indent a region of code
  - Comment out a region of code
  - ...