



University of
South Australia

Problem Solving and Programming

Introduction to Programming



University of
South Australia

Copyright Notice

Do not remove this notice.

COMMONWEALTH OF AUSTRALIA

Copyright Regulations 1969

WARNING

This material has been produced and communicated to you by or on behalf of the University of South Australia pursuant to Part VB of the *Copyright Act 1968* (**the Act**).

The material in this communication may be subject to copyright under the Act. Any further reproduction or communication of this material by you may be the subject of copyright protection under the Act.

Do not remove this notice.

Introduction to Programming

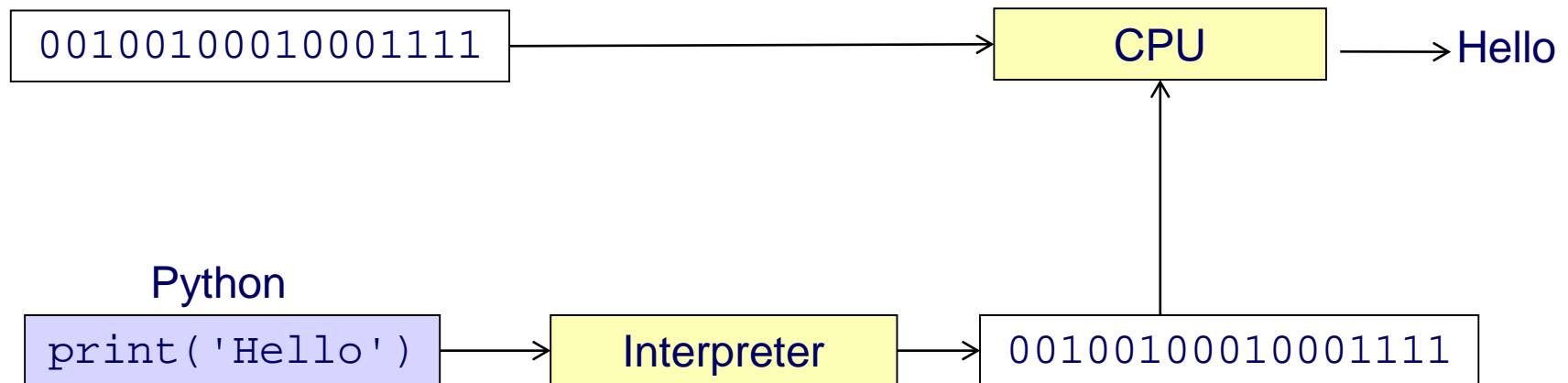
- Learning to write programs provides you with:
 - Practice in logical thinking.
 - Practice in problem solving.

Introduction to Programming

- What is a Programming Language?
 - A computer is an electronic machine; it does not understand, it just responds.
 - The heart of a computer is the Central Processing Unit (CPU).
 - For example: Pentium, Itanium, i3, i5, i7, etc.
 - The CPU is the part of the computer that actually runs programs.
 - The CPU responds to certain electrical signals in the form of binary numbers.
 - For example: 00101101
 - The CPU is designed to perform simple operations on pieces of data.
 - To carry out meaningful calculations, the CPU must perform many operations.
 - The CPU understands instructions written in machine language (binary).

Introduction to Programming

- What is a Programming Language?
 - A program is a set of instructions that a computer follows to perform a task.
 - Instructions could be written in machine language (binary), but that is way too hard!




- A programming language makes more sense to people, but it must be translated to binary for the CPU.

Introduction to Programming

- What is a Programming Language?
 - The CPU understands only machine language instructions, therefore, programs written in high-level languages must be translated into machine language in order to be executed.
 - Depending on the language, the programmer will use either a compiler or interpreter to make the translation.
- **Compiler:** translates a high-level language program into a separate machine language program.
 - Machine language program can be executed at any time, without using the compiler.
- **Interpreter:** translates each high-level instruction to its equivalent machine language instructions, and immediately executes them. This process is repeated for each high-level instruction.
 - Used by Python language.
 - Interprets one instruction at a time.
 - No separate machine language program.

Introduction to Programming

- A program is a set of instructions that a computer follows to perform a task.
- We write a program using a programming language.
- There are many programming languages each with their different strengths and weaknesses.
- In this course we use **Python**  :)

Why Python?

- Minimises obstacles when learning how to program.
- Allows you to focus on how to think rather than how to construct the instructions.
- Focus on how to solve the problem rather than focusing on the language.
- Useful in the real world – i.e. used in industry
 - List of organisations using Python:
<https://wiki.python.org/moin/OrganizationsUsingPython>
- It's free! :)

Python Files

Example of file called `welcome.py`

```
#
# File:          welcome.py
# Author:        your name
# Student Id:    your student id
# Email Id:      your email id
# Date:          today's date
# Description:   Displays a welcome message to the screen.
#   This is my own work as defined by the University's
#   Academic Misconduct policy.
#

# Display welcome message to the screen
print('Welcome to Problem Solving and Programming!')
```

← Notice comment operator (#)

This produces the following output (Python Shell window):

```
Welcome to Problem Solving and Programming!
```

```
>>>
```

Solving Problems with Python

- To solve problems, we need to further understand how Python works...
- Comments
 - The comment operator is the hash sign #.
 - Span the entire line.
 - Comments are ignored by Python.
 - Important! Use comments to describe each variable definition and to highlight the major parts of a program.
 - There should be a space between the # and the comment text. Comments should start with an uppercase letter and end with a period.

Solving Problems with Python

■ Variables

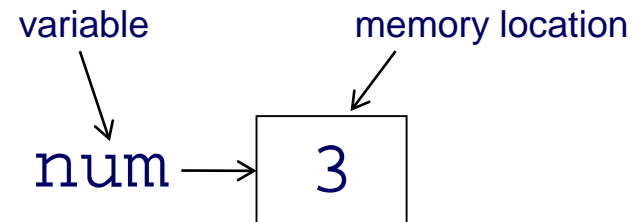
- A variable is a name that represents a value stored in the computer's memory.
- As you have seen, you can use Python like a calculator.
- However, it is usually more convenient to give names to the values you are using.
- Data that is input, calculated or output is stored in memory.
- To make it easy for us to refer to these memory locations, we can use names.
- We often want the values in these memory locations to be able to change (as a result of calculation, for example).
- We refer to these named memory locations as variables.
- Python variables are automatically created when they are initialized (first used).
 - Variables are defined when they are assigned a value by using an assignment statement. When a variable represents a value stored in memory, we say that the variable *references* the value.
- A variable is a location in memory used to store a value.

Solving Problems with Python

■ Creating Variables

- Python allows you to assign a value to a variable.
- A variable is created when it is first used.
- Use an assignment statement to create a variable and make it reference a piece of data.
- The following is an example of an assignment statement (used to define/create a variable).

```
num = 3
```



- A variable named `num` will be created and it will reference the value 3. We say that the `num` variable *references* the value 3.
- This is not read as 'equals'.
- Read as `num` is assigned a value of 3.
- May now use the variables in subsequent calculations.

Solving Problems with Python

- Variables and Constants

- Variable

- *adjective.* able or liable to change
 - *noun.* a name given to a memory location designed to store a data value (*Comp.Sci.*)
 - Called 'variable' as the value stored in it is likely to change during program execution

- Constant

- *noun.* a quantity that has a fixed value throughout a set of calculations
 - Called 'constant' as once set, the value stored does not/cannot change during program execution

Solving Problems with Python

- Rules for naming variables
 - All variable names must start with a letter (a–z, A–Z, or an underscore character '_').
 - A variable name cannot contain spaces.
 - They may contain letters (a–z, A–Z), numbers (0–9) and the underscore character (_).
 - Names are case sensitive.
`count` is different from `Count`.
 - You cannot use Python's keywords as a variable name.

Solving Problems with Python

■ Conventions:

- Use lowercase letters for variable names.
- Use UPPERCASE letters for constants.
- Names should be short enough to remember and **MUST** be self descriptive.
- Always choose names for your variables that give an indication of what they are used for.
- To represent names made up of multiple words, use one of the following approaches to make the variable names easy to read:
 1. Use an underscore to separate the words:

`game_status`

2. Use camelCase:

`gameStatus`

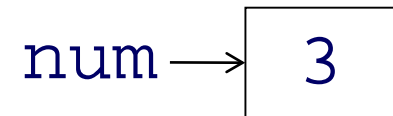
Either is okay as long as you are consistent in your approach.

Solving Problems with Python

- Variable Reassignment

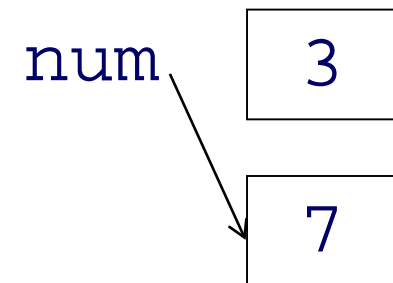
- Variables are called 'variable' because they can reference different values while a program is running.
- When you assign a value to a variable, the variable will reference that value until you assign it a different value.
- For example:

`num = 3`



This creates variable `num` and assigns it the value 3 (as seen above).

`num = 7`



This assigns the value of 7 to the `num` variable (as seen above).

The old value of 3 is still in the computer's memory, but it can no longer be used because it isn't referenced by a variable.

- When a value in memory is no longer referenced by a variable, the Python interpreter automatically removes it from memory (this process is called garbage collection).

Solving Problems with Python

- Keywords

- Keywords may not be used as variable names.
- The following is a complete list of Python keywords:

False	None	True	and
as	assert	break	class
continue	def	del	elif
else	except	finally	for
from	global	if	import
in	is	lambda	nonlocal
not	or	pass	raise
return	try	while	with
yield			

Solving Problems with Python

- Numeric Types

- int

- Integer – whole number with no decimal point.*

- bool

- Boolean – reference one of two values: True or False.*

- float

- Real number – number with a decimal point (floating point number).*

...more types later...

Solving Problems with Python

■ Performing Calculations

- Math expressions are used to perform a calculation.
- Most real world algorithms require calculations to be performed.
- An expression is a combination of values, variables, and operators.
- An expression performs a calculation and gives a value.
 - Math operator: is a tool for performing a calculation.
 - Operands: the values surrounding an operator.
 - Variables may be used as operands.
 - Resulting value is typically assigned to a variable (so we can use it again in the program).

Solving Problems with Python

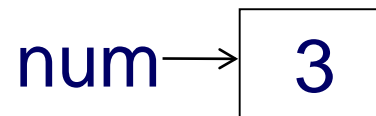
- Python Math Operators
 - Used to perform a calculation.
 - Python uses +, -, *, / for the add, subtract, multiply and divide operators which can be applied to all the basic data types.

Operator	Meaning
+	Addition - as you expect it to operate, except if mix floating point and integer, result will be floating point
-	Subtraction - as for +
*	Multiplication - as for +
/	Division for integer and floating point will always give floating point result.
//	Is a 'floor' operator - gives the largest integer value less than the result if used the division operator (but will be floating point if use it with floating point numbers). NOT what you may first expect if the result is negative, so make sure you try this one out for some negative values.
%	The modulo operator - for positive values it is just the remainder after division. Again, more complicated for negative numbers. Virutally always used in practice with non-negative values. Try out for floating point and for negative values.
**	Raise to a power - works with both integer and floating point values. What you expect for integer powers.

Solving Problems with Python

- Assignment Statement
 - Assigns a value to a variable.
 - A single equals sign (=) is called an assignment operator.
 - The assignment operator causes the result of the calculation to be stored in a computer memory location.
 - Example (arithmetic operations between two numbers):

`num = 1 + 2`



Read as `num` is assigned the value of 1 plus 2

Solving Problems with Python

- Math Operations

- Example (arithmetic operations between two int variables):

`num1 = 3`

`num2 = 5`

`total = num1 + num2`

`total` → 8

`num1` → 3

`num2` → 5

Read as add the current value of variable `num1` to the current value of variable `num2`, and assign the result to the variable `total`.

- The expression (right hand side) is evaluated and its result is assigned to the left hand side memory location.
- Assignment statement is NOT an algebraic equation.

Solving Problems with Python

- Math Operations

- What is the result of the following statements?

```
num = 6
```

num →

```
num = num + 1
```

Solving Problems with Python

- When you create expressions with operators, how is the expression evaluated?
 - Precedence and Associativity
- Order of Operations
 - Python follows the standard algebraic rules for the order of operation (operator precedence):
 - Parentheses (working from innermost to outermost).
 - Exponentiation operations (**).
 - Multiplication (*), division (/ and //) and remainder (%) operations (left to right).
 - Addition (+) and subtraction (-) operations (left to right).
 - Higher precedence is performed first.

Solving Problems with Python

- Order of Operations
 - Operators on the same precedence level are evaluated left to right (associativity).
- Example:

$$5 * (2 + 2) = \qquad 4 * 2 / 2 * 4 =$$

$$5 * 2 + 2 = \qquad 4 * 2 / (2 * 4) =$$

Solving Problems with Python

- More about Math Operations...
 - Two types of division:
 - / operator performs floating point division.
 - // operator performs integer division.
 - Positive results are truncated, negative are rounded away from zero.
 - Exponent operator (**):
 - Raises a number to a power.
 - Remainder operator (%):
 - Performs division and returns the remainder (modulus operator).

$$x \text{ ** } y = x^y$$

$$4 \% 2 = 0$$

$$5 \% 2 = 1$$

Solving Problems with Python

- Strings

- Python indicates strings through the use of single or double quotes.
- In addition to the `int` and `float` data types, Python also has a `str` data type, which is used for storing strings in memory.

- Strings can be assigned to variables.
- Example using double quotes:

```
str_example = "hello"
```

- Single quotes may also be used to indicate a string.
- For example:

```
str_example = 'hello'
```

End

Introduction to Programming