

Python Programming Handbook

This is an informal introduction to Python programming. It introduces the beginner to some fundamental programming concepts such as: input/output, variables and conditional statements.

Overview

One point about programming must be clarified immediately: **Anyone can learn to program computers.** However, you must be willing to **spend some time studying and practising.**

There are two aspects to programming that must be mastered. One concerns **problem solving** and the other concerns the **programming language, in this case Python**, that is to be used.

You must learn how to solve problems. This is the core of programming. But you also must learn how to express your problem solution in Python, so that it can be carried out on a computer. These are two separate skills. You must try not to confuse them. It is difficult, however, to explain one without reference to the other.

Problem Solving Skills

Computer programming is about problem solving. Every computer program solves some particular problem, even programs to play games. It is impossible to write a computer program unless you understand the problem you are being asked to solve. In programming, **you solve** the problem, not the computer. A computer program describes to the computer **what** it must do and **how** it is to be done. You give the computer instructions in the form of a program, telling it what to do and how to do it. **The set of instructions required to solve a problem is a computer program**

The term **algorithm** is used to describe the **set of steps that solve a problem**. An algorithm may be expressed in English or in a programming language. A **computer program must be expressed in a programming language**. In programming, we first develop an algorithm for the problem at hand, and then we **translate** this algorithm to a Python program, so that it can be executed (carried out, ran) on a computer.

Sometimes we make mistakes in telling the computer what to do. We overlook part of the problem or do not understand what to do ourselves. In these cases, the computer program will not produce the “right answer”. It is, however, still solving a problem. It is simply not the problem we wanted to solve. For this reason, it is important to thoroughly check that your programs do indeed solve the problem you intended.

Lesson 1: Output

In this lesson we learn how to perform output – to display messages on your screen.

All program code will be displayed using the `Courier` font in this text.

Output is the term used to describe information that the computer displays (**writes**) on your screen or stores (writes) on a disk drive.

The commonest form of output is that of displaying a message on your screen. In Python, we use the `print` statement to display output on the screen.

The following `print` statement will display the message:

```
My name is Beth. This is my first program
```

on the screen.

```
print( "My name is Beth. This is my first program" )
```

This is a single Python program **statement**. A statement is a command to the computer to carry out an action. So our first Python program is composed of this single statement. To execute this program, it is stored in a file which we call `print1.py`. which contains one line:

```
print( "My name is Beth. This is my first program" )
```

To execute the program we use the command `python3` which runs the program:

```
% python3 print1.py  
My name is Beth. This is my first program
```

You may also run Python programs using an IDE which will be discussed later.

We call the message that `print` displayed a **string**.

A string is a list of characters in quotes. You may put any characters that you wish in a string. Strings can also be represented by characters inside either single or double quotes:

```
"My name is Beth."  
'My name is Beth. '
```

You can display strings using a `print`:

```
print( "Hello ! Goodbye!" )  
print( "rubbish 123 rubbish xyz @!$%^&*" )
```

Statements

When people communicate in any language, they use **sentences**. When you write down a sentence in English, you have a full stop at the end. This tells us where the sentence ends. You call the full stop a sentence **terminator** i.e. it indicates the end of a sentence.

Similarly, when you write programs you also use sentences to communicate with the computer. In programs, sentences are called **statements**. They also have a terminator. Python statements use the **end of line** to terminate a statement. The 2 statements below are terminated by the **newline character**.

```
print( "Hello ! Goodbye!" )
print( "rubbish 123 rubbish xyz @!$%^&* " )
```

After you entered `)` on the first line, you pressed the Return key on the keyboard to start a new line – we call this the **newline character**. Similarly after you entered `)` on the second line, you also pressed Return thus starting a new line.

If you make a mistake entering a statement, a computer will not understand the statement and it will display an error message. This often caused by a **spelling error** or a **missing bracket** or **quotation** mark. For example, in a programming, just as in English, you must always have the correct number of quotation marks and brackets.

Left brackets such as `(`, `{` and `[` are called **opening** brackets.

The quotation marks at the start of a phrase: `"` (double) and `'` (single) are called **opening** quotes.

Right brackets such as `)`, `}` and `]` are called **closing** brackets and quotation marks at the end of a phrase are called **closing** quotes.

*A simple rule is that **for every opening** bracket or quotation mark you must have a **corresponding closing** bracket or quotation mark.*

Example 1.1: Matching brackets and quotes:

```
"Hello"
( age > 10 )
'x'
list[10]
```

Breaking this rule causes a **syntax error**. The following are examples of syntax errors:

```
age > 10)
'x
list[10
prin( "Hello ! Goodbye!" )
```

The last error is due to misspelling `print`

What to do when an error occurs

When a syntax error occurs, you must work out what mistake(s) you have made. This means checking the statements of your program and seeing where the syntax is incorrect. You then **edit** your program to correct the mistake. When you have corrected your program, it can be executed. **To execute a program or to run a program means that the computer carries out the instructions making up the program.**

A Python program is made up of a group of one or more statements. These statements allow us to control the computer. Using them, we can **display information on the screen, read information from the keyboard** and **process information** in a variety of ways.

We can classify statements as:

Input/Output (I/O) statements (e.g. display information on screen)

Variable manipulation statements (e.g. to do arithmetic) and

Conditional statements (e.g. to make decisions)

Lesson 1 Exercises – try these in your own time

1. What is the output of the following print statement?

```
print( "Have a great day!")
```

- a. "Have a great day!"
- b. "Have a great day"
- c. Have a great day!
- d. 'Have a great day!'

2. What is the output of the following statements?

```
print("Hi there!")  
print("How are you doing?")
```

- a. Hi there! How are you doing?
- b. How are you doing? Hi there!
- c. "Hi there!"
How are you doing?"
- d. Hi there!
How are you doing?

3. Write a program that prints a message saying

I love Python!

4. Write a program that prints a message saying your name and your age, e.g.

My name is Colin. I am 20 years old!

5. Write a program to display the message "Welcome to Python" three times, on separate lines using three `print` statements.
6. Write a program to display the message "Python is awesome!" two times, on separate lines, using only one `print` statement and `\n`
7. What are the syntax errors in the following statements:

```
print( "Hello ! Goodbye! )  
print( Hello ! Goodbye!" )  
print( "Hello ! Goodbye!"  
print "Hello ! Goodbye!" )  
prnt( "Hello ! Goodbye!" )
```

Lesson 1 Assignments – you email these to UCD

1. Write a program that prints out your favourite food, followed by a blank line, followed by your favourite colour, followed by a blank line, followed by your favourite animal. Use either the `\n` character in one `print` statement or use separate `print` statements.
2. Print a square made up of 4 @ characters per line using a single `print` statement.

```
@@@@@  
@@@@@  
@@@@@  
@@@@@
```

Lesson 2: Input and Variables

Input is the term used to describe the transfer of information from the keyboard (or a disk) to the computer. We can use the word **read** for input e.g read information from the keyboard. The question arises – where do we store the information we read in. This introduces the concept of **variables**.

A variable is a container for information.

This means that we can store information in a variable. It is called a variable because at any time we can change (**vary**) the information it stores.

So when we input information, we store it in a one or more variables. We **give each variable a unique name**, which we use to identify it. The following are examples of variable names we could use in a Python program:

```
colour
my_age
pension_age
name
taxcode
tax_rate
temperature
name
hourly_pay
```

Fundamental principle of writing clear programs

Choose meaningful names for variables

Using meaningful variable names it makes your programs easier to understand. For example, if you are writing a program which deals with pension ages then you could use any of the following variable names to store the pension age but which one makes is easiest to understand:

```
pension_age
pa
p
x
pna
```

The variable name `pension_age` is the obvious choice. When you see this name you automatically know what it the variable is being used for. If you use a name like `p` or `x` then the name gives you no idea about what the variable is being used for.

We use the Python **input** statement to read information from the keyboard, into a variable.

Example 1: Write a program to ask the user to enter their favourite colour. The program reads this colour that the user types on the keyboard and displays a message followed by the colour entered by the user.

```
favourite_colour = input("Enter your favourite colour: ")  
  
print( favourite_colour )
```

If we execute the program the following appears on the screen (the bolded text is that entered by the user on the keyboard. We will use this convention throughout the text).

```
Enter your favourite colour: blue  
blue
```

The variable `favourite_colour` is used to store the characters that the user types on the keyboard that is, it will store a list of characters.

When you use a variable name with `print` it will display the **value** of the variable i.e. the string `blue` in the example above.

The `input()` statement does two tasks: it displays the string in quotes and then reads text from the keyboard, (for example the word *blue* may be entered), and it places this text in the variable `favourite_colour`.

We have seen that the `print` statement is used to display output on the screen. It can be used to display strings and numbers in the same `print` statement.

Example 2: Write a program to ask the user to enter their favourite colour. The program reads this colour that the user types on the keyboard and displays a message followed by the colour entered by the user.

```
favourite_colour = input("Enter your favourite colour: ")  
  
print("Yuk ! I hate ", favourite_colour )
```

If we execute the program the following appears on the screen:

```
Enter your favourite colour: blue  
Yuk ! I hate blue
```

The statement

```
print("Yuk ! I hate ", favourite_colour )
```

instructs the computer to display the message *Yuk ! I hate* followed by the value of the variable `favourite_colour` i.e. *blue* in this example.

We use the expression “the value of a variable” to mean “the value contained in a variable”.

We take the phrase “the value of `favourite_colour` is blue” to mean “the value contained in the variable `favourite_colour` is blue”. We will use the shorter form from now on.

Make sure you understand the difference between:

```
print( "favourite_colour" )  
and  
print( favourite_colour )
```

In the first case, the word (string) `favourite_colour` appears on the screen.

In the second case, the **value** of a variable called `favourite_colour` is displayed which could be anything, for example the word *blue* or whatever value the user has given the variable like *red*, *pink* and *orange*. You can store **many words** in a string variable.

Rules for Variable Names

Python and every programming language, has rules on how you name variables:

- **A variable name can only contain the following:**
 - **letters** (lowercase and uppercase, ie a–z and A–Z)
 - **digits** (0–9)
 - the underscore character “_”
- A variable name **cannot start** with a digit
- Variable names in Python are **case-sensitive** – it distinguishes between uppercase and lower case letters so that `colour` and `Colour` are **different variables**
- There are a number of **reserved words** or **keywords** that have built-in meanings in Python and cannot be used as variable names (e.g. `if`, `return`, `def`, `del`, `break`, `for`, `in`, `else`, `while`, `import`)

The following are legal or valid variable names in Python:

```
Colour, name, firstname, surname, class1, class 602,  
first_name, second_name, address_line1.
```


The use of the underscore “_” character is very useful in creating meaningful names made up of 2 or more words.

Do not confuse the underscore character with the minus sign “-“. The minus sign (or any other symbol) cannot be used in a variable name. thus `first-name` is not a valid variable name.

Comments

In a Python program, any text after `#` is called a comment and **is ignored by Python**. This text is there to help explain to someone reading the program, what the program does and how the program works. **Comments are an important component of programs**. This is because when you read your programs some time after writing them, you may find them difficult to understand, if you have not included comments to explain what you were doing. They are even more important if someone else will have to read your programs e.g. your tutor who is going to grade them!

It is a useful idea to start all programs with comments which give the name of the file containing the program, the purpose of the program, the authors name and the date on which the program was written, as the first comments in any program. Example 2 could be written as:

```
# colour.py: Prompt user to enter colour and display a message
# Author: Joe Carthy
# Date: Oct 20 2022

favourite_colour = input("Enter your favourite colour: ")

print("Yuk ! I hate ", favourite_colour )
```

Lesson 2 Exercises

1. Valid or invalid variable names

- Is the variable name `TotalMarks` correct?
- Is the variable name `number-of-students` correct?
- Is the variable name `firstName` correct?
- Is the variable name `myVar1` correct?
- Is the variable name `customerName` correct?
- Is the variable name `productPrice` correct?
- Is the variable name `3rdStudent` correct?
- Is the variable name `isAvailable?` correct?
- Is the variable name `total-sales` correct?
- Is the variable name `customer_email` correct?

- Write a program that asks the user for their name using `input`. Store the name in a variable and display a personalized greeting using the variable.
- What is the output, if any, of the following program:

```
# print('hello\n')  
# print('bye bye\n')
```

- Write a program that prompts the user to enter their favourite color and favourite animal using the `input`. Store these values in separate variables and display them in a sentence :

My favourite colour is ... and my favourite animal is ..

Lesson 2 Assignments

- Write a program that prompts the user to enter their favourite number using `input`. Store the number in a variable and print a message that includes the user's favourite number.
- Create a program that simulates a hospital registration system. Prompt the user to enter the following information:

Name
Surname
Age
Height (in cm)

Store each piece of information in a separate variable with an appropriate name.

Finally, print the information in the following format:

```
Name: Joe  
Surname: Carthy]  
Age: 100  
Height: 180
```

Appendix 1: Solutions

Lesson 1 Solutions

1. What is the output of the following print statement?

```
print( "Have a great day!")
```

c. Have a great day!

2. What is the output of the following statements?

```
print("Hi there!")  
print("How are you doing?")  
    Hi there!  
    How are you doing?
```

3. Write a program that prints a message saying

I love Python!

```
print("I love Python! ")
```

4. Write a program that prints a message saying your name and your age, e.g.

My name is Colin. I am 20 years old!

```
print("My name is Colin. I am 20 years old! ")
```

5. Write a program to display the message "Welcome to Python" three times, on separate lines using three `print` statements.

```
print("Welcome to Python! ")  
  
print("Welcome to Python! ")  
  
print("Welcome to Python! ")
```

6. Write a program to display the message "Python is awesome!" two times, on separate lines, using only one `print` statement and `\n`

```
print("Python is awesome!\n Python is awesome!\n ")
```

7. What are the syntax errors in the following statements:

```
print( "Hello ! Goodbye! )- missing closing "  
print( Hello ! Goodbye!" ) - missing opening "  
print( "Hello ! Goodbye!" - missing closing )  
print "Hello ! Goodbye!" ). - missing opening (  
prnt( "Hello ! Goodbye!" ) - misspelt print
```

Lesson 2 Solutions

1. Valid or invalid variable names

- a. Is the variable name `TotalMarks` correct - **Yes**
- b. Is the variable name `number-of-students` correct? **NO** – cannot use `-` in variable name
- c. Is the variable name `firstName` correct? **Yes**
- d. Is the variable name `myVar1` correct? **Yes**
- e. Is the variable name `customerName` correct? **Yes**
- f. Is the variable name `productPrice` correct? **Yes**
- g. Is the variable name `3rdStudent` correct? **NO** – cannot start with a digit
- h. Is the variable name `isAvailable?` correct? **Yes**
- i. Is the variable name `total-sales` correct? **NO** – cannot use `-` in variable name
- j. Is the variable name `customer_email` correct? **Yes**

2. Write a program that asks the user for their name using `input`. Store the name in a variable and display a personalized greeting using the variable.

```
name = input("Enter your name: ")  
  
print("Hello, how are you ", name )
```

3. What is the output, if any, of the following program:

```
# print('hello\n')  
# print('bye bye\n')
```

No output because any text following `#` is treated as a comment and ignored by Python

4. Write a program that prompts the user to enter their favourite colour and favourite animal using the `input`. Store these values in separate variables and display them in a sentence :

```
favourite_colour = input("Enter your favourite colour: ")
favourite_animal = input("Enter your favourite animal: ")

print("My favourite colour is ", favourite_colour, "and my
favourite animal is", favourite_animal )
```