

CHAPTER 6

Lab Assignment 06—Programming using Python

Title: Programming using Python

Due: Before the start of your Lab 07 session

Aims: Use the Python programming language to write simple programs

6.1 Preparation

Before doing this lab, you should:


- Read the chapter in the online course reference manual on Python.
- Go through your lecture overheads on Python.

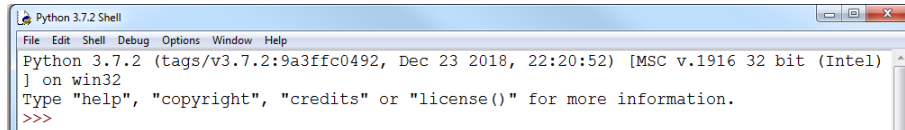
6.2 Introduction

Python is one of many programming languages that are commonly used. It is similar to programming languages such as Java, C, C++, Visual Basic and other languages used to write professional computer software. Although Python is a reasonably new computer language, it has become very popular.

We will use Python to write some simple computer programs. Before we get started, create a folder on your USB drive called Lab06. This is where we will save all our Python programs for this lab.

6.3 Getting started

Start the Python interpreter. This program can be found by clicking on the IDLE icon -  - pinned to the taskbar. You should see the Python Shell window as follows:

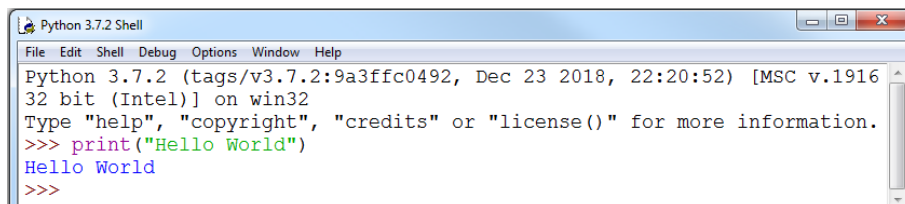


```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit (Intel)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
```

You can type any Python command at the prompt and it will be executed immediately. This is an excellent way to test out commands if you are not sure what they do, or if you have the correct instruction. For example, enter the following text at the prompt and hit `Enter`

```
Python Source Code
1 print("Hello World")
```

You should see the following output:



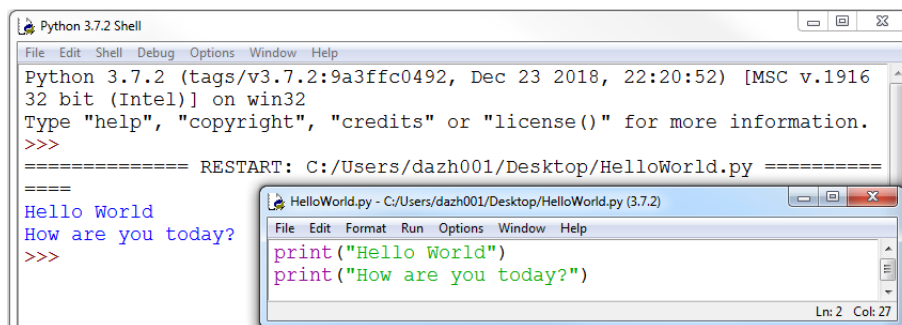
```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit (Intel)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("Hello World")
Hello World
>>>
```

We will write our Python programs using IDLE. Choose `File` → `New File`. You will see a new blank window appear. This is where we will type in our Python programs. Type the following text into the empty window:

```
Python Source Code
1 print("Hello World")
2 print("How are you today?")
```

Save the file as `HelloWorld.py` in your `Lab06` folder. Run your program by choosing `Run` → `Run Module` (alternatively, you could just press `F5` as a short-cut).

The program you have written will be executed, one line at a time, and the results will be displayed in the Python Shell window. You should see something like the following:



```

Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916
32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/dazh001/Desktop/HelloWorld.py =====
====
Hello World
How are you today?
>>>

HelloWorld.py - C:/Users/dazh001/Desktop/HelloWorld.py (3.7.2)
File Edit Format Run Options Window Help
print("Hello World")
print("How are you today?")
Ln: 2 Col: 27

```

Note that the program is stored in a plain text file, and when we run the program, the output is shown in the Python Shell.

6.4 Using variables

A variable is a name that we can use to refer to some information that we have stored. We use the equals sign = to store information to a named variable. This process is known as *assigning a value to a variable*. Start a new window and enter the following program:

```

Python Source Code
1 #Author: Insert your name here
2 #Date: January 2020
3
4 #Define the value of our variables
5 pairs_per_crate = 15
6 number_of_crates = 17
7
8 #Calculate the results
9 total_pairs = pairs_per_crate * number_of_crates
10
11
12 #Display the output
13 print(number_of_crates,"crates contain", total_pairs,
14       "pairs of shoes.")

```

Save the program as `ShoeRetail.py`. Run the program and examine the output. Make sure you understand how the output is generated using the program.

A shoe company ships shoes to a supplier in New Zealand in crates of 15 pairs of shoes. The supplier will send these shoes to its 11 shoe retailers around New Zealand. Modify the above program to calculate how many pairs of shoes each retailer receives (assuming they all receive the same number of shoes), and how many pairs of shoes remain with the supplier.

You **must** use variables to store **all** values. You will need to use the `//` mathematical operator to calculate the number of pairs of shoes each retailer receives. You

will also need to use the % mathematical operator to calculate the number of pairs of shoes that remain with the supplier. Your modified program should produce the following output:

```
17 crates contain 255 pairs of shoes.  
Each retailer receives 23 pairs of shoes.  
2 pairs of shoes remain with the supplier.
```

Q1: Take screenshots of your program's source code **and** output using the snipping tool and paste them into your lab report. The screenshots must be large enough for your code and output to be clearly legible. Make sure that you have inserted **your own name** as the author in the first line of the program.

6.5 Reading input from the user

Without asking the user for input, our programs are very limited. To read input from the user, we use the instruction:

```
user_input = input("Enter the data: ")
```

The `input` function reads user input as text. To convert this into a number we need to use either the `int` function to convert to an integer, or the `float` function to convert to a floating point number. For example to read input from the user as an integer, we would use the instruction:

```
user_int = int(input("Enter the data: "))
```

To read input from the user as a floating point number, we would use the instruction:

```
user_float = float(input("Enter the data: "))
```

The following program is used to calculate the area and perimeter of a rectangle. Read the program carefully and make sure that you understand what each instruction does.

```
Python Source Code  
1 #Author: Andrew Luxton-Reilly  
2 #Date: January 2020  
3  
4 print("This program calculates the area")  
5 print("and perimeter of a rectangle")  
6 print()  
7  
8 width = int(input("Please enter the width: "))  
9 height = int(input("Please enter the height: "))  
10  
11 area = width * height
```

```
12 perimeter = 2 * width + 2 * height
13
14 print("The area is", area)
15 print("The perimeter is", perimeter)
```

Write a new program called `HeightConversion.py` that converts a person's height in metres to a height in feet and inches. Your program must first ask the user to enter their name. The program should then print a greeting to the user followed by a prompt for the user to enter their height in metres. You can assume that the user will always enter a floating point value (a number with a fractional component like 1.78).

To find the equivalent height in feet and inches, your program should first convert the height in metres to a height in inches. There are 39.37 inches in a metre. Your program will then use this height in inches to calculate a height in feet and inches. There are 12 inches in a foot. You will need to use the `//` and the `%` operators, much like you did in the previous exercise to calculate this. Note, that you will need to use the `int()` function to ensure that the number of feet and inches are both integer values. Display the results as shown in the sample output below:

Example 1:

```
Please enter your name: Damir
Hello Damir welcome to the height converter.
We will convert your height in metres
to a height in feet and inches.

Please enter your height in metres: 1.84

You are 1.84 metres
or 6 feet and 0 inches tall.
```

Example 2:

```
Please enter your name: Bart
Hello Bart welcome to the height converter.
We will convert your height in metres
to a height in feet and inches.

Please enter your height in metres: 1.68

You are 1.68 metres
or 5 feet and 6 inches tall.
```

Q2: Take screenshots of your program's source code **and** output using the snipping tool and paste them into your lab report. The screenshots must be large enough for your code and output to be clearly legible. Make sure that you have inserted **your own name** as the author in the first line of the program.

6.6 Using a function from a library

Python comes with a number of library functions. We will be using a function that generates a random number. The library is called `random`. Before we are able to access any of the functions for a given library, we have to instruct Python that we want to use that library. To do that, we use the `import` command as follows:

```
import random
```

Once the library is imported, we can use the functions that are stored in that library. The function we are interested in is called `randint(a,b)`. It is used to generate a random integer value between the values of `a` and `b` (both inclusive). We can use this function anywhere that Python is expecting a number. To use this function in our program, we use the name of the library, then a dot, then the name of the function from that library. For example, to get a random number between 1 and 6 inclusive, we would use:

```
random.randint(1,6)
```

Random numbers are commonly used in video games. For example, random numbers can be used to calculate the damage inflicted when a video game character attacks.

The damage an attack inflicts depends on the strength of the character (a number between 0–255 inclusive) and the strength of their weapon (a number between 0–127 inclusive). The damage inflicted will be a random number between: `character strength` and `(character strength+weapon strength)`. Therefore if a character has a strength of 125 and a weapon of strength 100, then their attack could inflict anywhere between 125–225 damage (inclusive).

Write a program named `DamageGenerator.py` that:

- Prints out the heading as in the screenshot on the next page.
- Prompts the user to enter the character's name.
- Generates a random number between 0–255 (inclusive) to represent the character's strength.
- Generates a random number between 0–127 (inclusive) to represent their weapon's strength.
- Generates a random number between `character strength` and `character strength+weapon strength`.
- Displays the information in the right format.

The following two examples show possible outputs when the program is run. Remember that your program needs to produce output in the exact same format as

these examples.

```
*****
*Damage Generator*
*****

Please enter the character's name: Sauron
Sauron has a strength of 216
Sauron has a weapon with a strength of 107
Sauron attacks and inflicts 323 points of damage
```

```
*****
*Damage Generator*
*****

Please enter the character's name: Aragorn
Aragorn has a strength of 182
Aragorn has a weapon with a strength of 113
Aragorn attacks and inflicts 198 points of damage
```

Q3: Take screenshots of your program's source code **and** output using the snipping tool and paste them into your lab report. The screenshots must be large enough for your code and output to be clearly legible. Make sure that you have inserted **your own name** as the author in the first line of the program.

6.7 Conditions

With an if statement we can control whether a block of code gets executed. Have a careful look at the code on the following page.

Three scenarios are being evaluated by this block of code. **If** both random marks are **greater than or equal to** 50 the text "You have passed both assessments!" will be printed. **Else if** only one of the random marks is **greater than or equal to** 50 the text "You have passed one of the assessments!" will be printed. **Else** both random marks must be less than 50 resulting in the text "You have failed both assessments!" being printed.

Python Source Code

```
1 #Author: Damir Azhar
2 #Date: July 2019
3
4 import random
5
6 random_mark1 = random.randint(0,100)
7 random_mark2 = random.randint(0,100)
8 print("Random mark 1: ",random_mark1)
9 print("Random mark 2: ",random_mark2)
10 if random_mark1 >= 50 and random_mark2 >= 50:
11     print("You have passed both assessments!")
12 elif random_mark1 >= 50 or random_mark2 >= 50:
13     print("You have passed one of the assessments!")
14 else:
15     print("You have failed both assessments!")
```

We refer to `random_mark1 >= 50` as the condition. Conditions usually involve the use of relational operators like less than (`<`), greater than (`>`), less than or equal to (`<=`), greater than or equal to (`>=`), equal to (`==`) and not equal to (`!=`). Two conditions can be joined together using an **and** or an **or**. If an **and** is used, both conditions need to be true for the overall condition to be true. If an **or** is used, at least one of the conditions needs to be true for the overall condition to be true.

You should refer to your online course reference manual and lecture notes for a more detailed discussion.

6.7.1 Damage Generator Revisited

We will now use `if/elif/else` statements to extend the `DamageGenerator.py` program discussed in the previous section. Copy the code you have in `DamageGenerator.py` and save it in a Python file called `DamageGeneratorUpdated.py`. Update the code so that your program now does the following:

- Prints out the heading.
- Prompts the user to enter the character's name.
- Generates a random number between 0–255 (inclusive) to represent the character's strength.
- Generates a random number between 0–127 (inclusive) to represent their weapon's strength.
- Generates a random number between 1–100 (inclusive) to represent their luck statistic.
- Generates a random number between `character strength` and `character strength + weapon strength` to represent the damage inflicted.

- If the character's luck statistic is greater than 90, they have landed a critical hit and receive a 10% damage bonus.
- Else if the character's luck statistic is greater than 30, they have landed a regular attack and there is no change to the damage inflicted.
- Else if the character's luck statistic is greater than 10, they have landed a sloppy attack and there is a 20% penalty to the damage inflicted.
- Else the character misses their attack and inflicts 0 damage.
- Displays the appropriate information in the right format, as in the examples below.

The following four examples show possible outputs when the program is run. Remember that your program needs to produce output in the exact same format as these examples.

```
*****
*Damage Generator*
*****

Please enter the character's name: Aragorn
Aragorn has a strength of 63
Aragorn has a weapon with a strength of 98
Aragorn lands a critical hit and inflicts
158.4 points of damage
```

```
*****
*Damage Generator*
*****

Please enter the character's name: Aragorn
Aragorn has a strength of 247
Aragorn has a weapon with a strength of 27
Aragorn attacks and inflicts 254 points of damage
```

```
*****
*Damage Generator*
*****

Please enter the character's name: Aragorn
Aragorn has a strength of 178
Aragorn has a weapon with a strength of 2
Aragorn lands a sloppy attack and inflicts
144.0 points of damage
```

```

*****
*Damage Generator*
*****

Please enter the character's name: Aragorn
Aragorn has a strength of 67
Aragorn has a weapon with a strength of 106
Aragorn has missed their attack
No damage is inflicted

```

Q4: Take screenshots of your program's source code **and** output using the snipping tool and paste them into your lab report. The screenshots must be large enough for your code and output to be clearly legible. Make sure that you have inserted **your own name** as the author in the first line of the program.

6.8 Loops

A while loop can also be used to control the flow of execution. A detailed discussion on while loops is provided in the lecture slides and online course reference manual. Once you are familiar with how a while loop works carefully **desk check** the code and answer the questions that follow. Do not type the code in to the computer.

Python Source Code

```

1 #Author: Damir Azhar
2 #Date: January 2020
3
4 prev = 1
5 curr = 3
6 count = 3
7 end = 5
8 print(prev)
9 print(curr)
10 while count < end:
11     temp = curr
12     curr = curr + (curr - prev) ** 2
13     prev = temp
14     count = count + 1
15     print(curr)
16 print("The End!")

```

Q5: What is the output of the program?

Q6: How many times is the body of the loop executed?

Q7: How many times is the condition `count < end` checked?

Q8: What is the value of the variable `count` when the while loop finishes?

6.9 Guessing game

Using the pseudocode below, write a program called `GuessingGame.py` that plays a simple guessing game. The program should:

- Generate a random number between 1 and 100. Assign this number to a variable called “goal”.
- Set a variable called “guess” to 0.
- Print a message that describes the aim of the game (as shown in the example below).
- Print a blank line.
- While guess is not correct (i.e. `guess != goal`), do the following steps:
 - Ask the user to enter a guess and store the value in the “guess” variable.
 - If the guess is greater than the goal, print the message “Too high, try again.”
 - If the guess is less than the goal, print the message “Too low, try again.”
 - If the guess is the same as the goal, print the message “Well done!”
- Print the message “See you later.”.

You will need to use a `while` loop and some `if` statements to complete this program. An example of the game in action can be seen below:

```
The object of this game is to
guess a number between 1 and 100

Please guess the number: 50
Too low, try again.
Please guess the number: 75
Too low, try again.
Please guess the number: 87
Too high, try again.
Please guess the number: 80
Too low, try again.
Please guess the number: 83
Too low, try again.
Please guess the number: 85
Well done!
See you later.
```

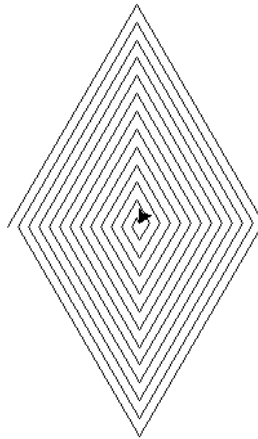
Q9: Take screenshots of your `GuessingGame.py` source code and output using the snipping tool and paste them into your report. The screenshot must be large enough for your code to be clearly legible. Make sure that your code includes a comment specifying **your name**.

6.10 Turtle Graphics

Using the material presented on Turtle Graphics in your lecture overheads and the online course reference manual have a look at the following Python code as well as the output it produces. Type the following program and save it to a file `DiamondSpiral.py`. Run the program. Make sure that you understand what each instruction does.

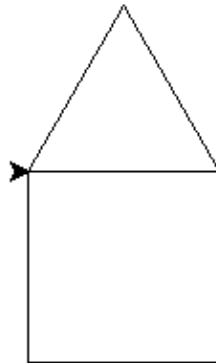
Python Source Code

```
1 #Author: Damir Azhar
2 #Date: January 2020
3
4 import turtle
5
6 step = 200
7 decrement = 4
8 angle1 = 60
9 angle2 = 120
10 count = 0
11
12 turtle.left(angle1)
13 while step > 0:
14     turtle.forward(step)
15     step = step - decrement
16     if count % 2 == 0:
17         turtle.right(angle2)
18     else:
19         turtle.right(angle1)
20     count = count + 1
```



Q10: Take screenshots of the program's source code and output and paste them in your lab report. The screenshots must be large enough for your code and output to be clearly legible.

Now write a Python program called `House.py` where you use the turtle to draw a house as shown below.



Q11: Take screenshots of your program's source code **and** output using the snipping tool and paste them into your lab report. The screenshots must be large enough for your code and output to be clearly legible. Make sure that you have inserted **your own name** as the author in the first line of the program.

6.11 Lab Summary

Q12: Write a brief (paragraph) description of what you did in this lab.

You have now completed the compulsory part of this lab. The remaining sections are completely optional. No marks are allocated to the programs contained in

these sections. However, they are good practice and will help you to understand the Python programming language better, so it is recommended that you give some of them a try.

6.12 Optional programs

Leap years (challenging)

Due to the fact that it actually takes about $365 \frac{1}{4}$ days for the earth to circle the sun, every fourth year has an extra day added to the end of February and the year is called a leap year. Leap years have 366 days, non leap years have 365 days.

Unfortunately, this correction is not quite perfect, and so once every century (at the turn of the century) we do not have the normal leap year. However, even that isn't quite accurate enough, so on every 4th century, we do have the leap year (so 3 out of every 4 centuries we don't have the leap year, but on the 4th century we do).

The following paragraph describes how to determine if a given year is a leap year:

Every year that is evenly divisible by 4 is a leap year, except that a year which is also divisible by 100 is not a leap year, unless it is divisible by 400, in which case it is a leap year after all!

Examples:

- 1900 was not a leap year, because it is divisible by 100, but not by 400
- 1904 was a leap year, because it is divisible by 4, but not by 100
- ...
- 1996 was a leap year, because it is divisible by 4, but not by 100
- 2000 was a leap year because it is divisible by 400
- 2004 was a leap year because it is divisible by 4, but not by 100
- 2005 was not a leap year because it is not divisible by 4

Write a program that asks the user to enter a year. The program should decide if the year is a leap year or not and print out the decision to the user (i.e. display either "That is a leap year", or "That is not a leap year").

You will need to use `if` statements and the remainder operator `%` to write this program.

Prime numbers (challenging)

A prime number is any number greater than 1 that is only evenly divisible by itself and 1. In other words, if the number can be divided by any positive number apart from itself and 1 and leave no remainder after the division, then it is not a prime number.

Write a program that asks the user to enter a number. The program should decide whether the number is a prime number or not and inform the user.

You will need to use a `while` loop, `if` statements and the remainder operator `%` to write this program.

The Rock Paper Scissors Game (challenging)

Rock Paper Scissors is a hand game that can be played by two or more people, where the three items in question are represented with hand gestures. The objective of the game is to select a hand gesture which defeats that of the opponent. The rules that govern this are as follows:

- Rock beats scissors.
- Scissors beats paper.
- Paper beats rock.

Based on the above rules and using the following pseudocode, write a program called `RockPaperScissors.py` that plays a simple game of Rock Paper Scissors against the computer.

- Print the title "The Rock Paper Scissors Game".
- Set the variable `rock` to be 1, the variable `paper` to be 2 and the variable `scissors` to be 3.
- Set the user's input to be 0.
- While the user has not entered -1:
 - Ask to user to enter 1 to choose rock, 2 to choose paper, 3 to choose scissors or -1 to quit the game.
 - Randomly assign rock, paper or scissors as the computer's choice by generating a random number between 1 and 3 and assigning it to a variable named `computer_choice`.
 - If the user has not entered -1:
 - * If the user has chosen rock:
 - Print "You chose rock"
 - * If the user has chosen paper:
 - Print "You chose paper"
 - * If the user has chosen scissors:
 - Print "You chose scissors"
 - * Repeat the previous 3 steps for the computer printing "The computer chose" followed by the computer selection.
 - * If the user wins:
 - Print "Congratulations you win!"
 - * If the computer wins:
 - Print "Sorry you lose!"
 - * If the user and the computer make the same selection:
 - Print "The match was a tie!"
- Print "Goodbye"

An example of the program running is shown on the next page:

```
The Rock Paper Scissors Game
Please enter 1 for rock, 2 for paper,
3 for scissors or -1 to quit: 1
You chose rock
The computer chose scissors
Congratulations you win!

Please enter 1 for rock, 2 for paper,
3 for scissors or -1 to quit: 3
You chose scissors
The computer chose scissors
The match was a tie!

Please enter 1 for rock, 2 for paper,
3 for scissors or -1 to quit: -1
Goodbye
```

You will need to use a while loop and some if statements to complete this program.

6.13 What to Hand In

This lab is worth 10% of the marks for the practical component of the COMPSCI 111/111G course. You must hand in:

- A signed CompSci111 cover sheet/attendance sheet.
- The answers to all the questions asked in this lab (typed with correct spelling and grammar).
- All printouts required for this lab.

Your lab report should be stapled together and the entire report should be handed in to the appropriate box no later than 5 minutes before the beginning of your next lab session.