TEMASEK
JUNIOR COLLEGE

Temasek Junior College
2022 JC1 H2 Computing
Practical 8 – Iteration Control Structures in Python II

Session Objectives
By the end of this session, you will learn:
(i) infinite loops
(ii) the use of the `break` statements in loops.
(iii) the use of `continue` statements in loops.

For this practical session, we shall make use of a Jupyter Notebook to document the exercises.
- Download the Jupyter Notebook **Practical_8.ipynb** from Google Classroom.
- Submit your file through the Practical 8 Google Classroom link at the end of the session.

§8.1 Infinite Loops

From Practical 7, we have seen that a Python `while` loop functions on the basis that the loop will be executed repeatedly as long as the condition remains `True`.

Hence, it is important that the condition must eventually become `False`. This is so that the loop will end after the desired number of iterations.

What then happens if the condition remains `True` "eternally"?

Exercise 1
Type the following code in a new cell of your Jupyter Notebook.

```
i = 0

while i < 5:
  print(i)
  i = i - 1
```

Observe that in **Exercise 1**, the loop prints integers (starting from 0) in descending order infinitely. This is known as an **infinite loop**.

When coding any program, it is important that we define the conditions correctly when implementing loops to ensure that the program does not get "trapped" within an infinite loop.

§8.2 Breaking Out of a `while` Loop – Using `break` Statements

Can a `while` loop be exited during the midst of execution?

Definitely yes!

To exit a `while` loop during execution, we can use the `break` statement when a condition is met.

The general syntax structure is as follows:

```
while <condition_1>:
    statement_1
    if <condition_2>:
        break
    statement_2
```

Exercise 2
Type the following code in a new cell of your Jupyter Notebook.

```
i = 0

while True:
  print(i)
  if i >= 10:
      break  # Exit the loop if i >= 10.
  i = i + 1  # Otherwise increase the value of i by 1
```

Even though `break` statement can be used to exit the loop when the value of `i` has reached 10, it is not a good practice to do so as the condition of the `while` loop can be changed easily to exit the loop when `i >= 10`.

Exercise 3
Type your code in a new cell of your Jupyter Notebook.

Write a program that asks the user to input random messages (i.e. the program can receive any string input). However, the programme will stop asking the user if he or she makes an empty entry.

§8.3 Skipping an Iteration in a `while` Loop – Using `continue` Statement

Since it is possible to break out of a `while` loop during execution, is it then possible to skip an iteration of a `while` loop during its execution?

Again, definitely yes!

To skip an iteration, we can use the `continue` statement when a condition is met.

The general syntax structure is as follows:

```
while <condition_1>:
    statement_1
    if <condition_2>:
        continue
    statement_2
```

Exercise 4
Type the following code in a new cell of your Jupyter Notebook.

```
i = 0

while i < 10:
  i = i + 1
  if i % 2 == 0:
      continue  # If the remainder when i is divided by 2 is 0, skip
this iteration.
  print(i)  # Otherwise, print the value of i.
```

Exercise 5
Type your code in a new cell of your Jupyter Notebook.

Write a program that asks the user to input 10 positive numbers. Print out each number entered. If the user enters a negative number instead, do not print out the number.
*Note that we can accept positive numbers, not just positive integers.

§8.4 Using `while-else` (Self-Read)

A `while-else` structure can be implemented to execute some other statements at the end of the last iteration.

The general syntax structure is as follows:

```
while <condition>:
    statement_1
else:
    statement_2
```

Exercise 6
Type the following code in a new cell of your Jupyter Notebook.

```
i = 0
total = 0

while i < 10:
    total = total + i
    i = i + 1
else:
    print(total)  #Prints the total after the last iteration.
```

Exercise 7
Type the following code in a new cell of your Jupyter Notebook.

```
i = 0
total = 0

while i < 10:
    total = total + i
    i = i + 1
    print("Current i is", i)
    if i == 3:
        print("exit the loop for a while")
        break
else:
    print(total)  #Prints the sum after the last iteration.
```

For **Exercise 7**, what do you observe about the use of `break` and `else`?

When the value of `i` reaches 3, the program has exited the loop and it does not reach the last iteration, hence the statement within the `else` block, which is `print(total)` in this case, is not executed.

§8.5 Breaking Out of a `for` Loop – Using `break` Statements

`break` statements can also be used to break out of a `for` loop.

The general syntax structure is as follows:

```
for <item> in iterable:
    statement_1
    if <condition>:
        break
    statement_2
```

Exercise 8
Type your code in a new cell of your Jupyter Notebook.

Write a program using `for` loop with `break` statement, to find the lowest common multiple (LCM) of 2, 3 and 5.
* Note: There are other methods to write a more efficient program to find LCM, we are just practicing our usage of `for` loop with `break` statement here.

§8.6 Skipping an Iteration in a `for` Loop – Using `continue` Statements

`continue` statements can also be used to skip an iteration in a `for` loop.

The general syntax structure is as follows:

```
for <item> in iterable:
    statement_1
    if <condition>:
        continue
    statement_2
```

Exercise 9
Type your code in a new cell of your Jupyter Notebook.

Code a programme that can print all numbers between 1 and 99 except for those which can be divided by 2, 3 and 5, using `for` loop and `continue` statement.

§8.7 Using `for-else` (Self-read)

A `for-else` structure can be implemented to execute some other statements at the end of the last iteration.

The general syntax structure is as follows:

```
for <item> in iterable:
    statement_1
else:
    statement_2
```

Exercise 10
Type the following code in a new cell of your Jupyter Notebook.

```
total = 0

for i in range(1,11):
    total = total + i
else:
    print(total) #Prints the total after the last iteration.
```

Exercise 11
Type the following code in a new cell of your Jupyter Notebook.

```
total = 0

for i in range(1,11):
    total = total + i
    print(i)
    if i > 4:
        print("Break out of the loop")
        break
else:
    print(total)  # Prints the total after the last iteration
```