KNJ201 Engineering Design and Build A

Practical 1: Introduction to programming with Python

In this practical session you will write some basic programs using the Python programming language. Please read the following carefully:

* One program must be written for each part.
* Each part has a set of tests that must pass.
* All programs must be commented – explain what the program does – this includes at the start of the program and all functions, classes and methods.
* All programs must be submitted to CloudCampus.
* Once submitted each program will be tested.
* You will get one mark for each test passed.

# Before You Come to the Class

Before the practical class you must:

* Choose and install an IDE (integrated development environment) for Python; you will use this to edit your python source code.
* Install Python 3 on your computer (if you don’t already have it).

All programs must start with the following first line:

#!/usr/bin/env python3

Before you come to the practical class you must write a program called id.py. This program will be used to identify who you are. An example id.py file is given below:

def id():

id = "12345678" # Student ID

name = "Benjamin Millar" # Student name

cohort = "Electrical" # Must be Electrical or Communications

return (id, name, cohort)

# Part 1

## Print Text

The print() function prints text to the screen.

Write a program that prints “Hello World!”.

## Importing Modules, Command-line Arguments, String Formatting

The import command adds a module to your program. Modules provide additional functions for you to use.

The sys module incudes argv which is an array containing the command line arguments. To import the sys module simply add a line to your program as follows:

import sys

Now we can use sys.argv[1] to access the first command line argument. For example, if you program is called part1.py, and it is called as follows:

$ part1.py Ben

then the sys.argv[1] will equal the string “Ben”.

Strings can be formatted by adding text and numbers with a percent symbol: %d for numbers and %s for strings. For example:

print("This is the number %d and this is the string %s." % (123, "Hi!"))

This code will give the following output:

This is the number 123 and this is the string Hi!

Change your program to print “Hello <name>!” where <name> is taken from the command line argument.

## If, Else, Elif

The if statement allows you to execute code only if a condition is met. For example, the following will only print “None” if the number of arguments is zero (note that the len() function gives the number of arguments in sys.argv):

if len(sys.argv) == 0:

print("None")

The else statement will execute code only if the if condition was false. For example, the following will print “Some” if the number of arguments is not zero:

if len(sys.argv) == 0:

print("None")

else:

print("Some")

We can test more than one condition using the elif statement (this mean “else if”). For example, the following will print “Many” if the number of arguments is more than 10:

if len(sys.argv) == 0:

print("None")

elif len(sys.argv) > 10:

print("Many")

else:

print("Some")

Combine your programs from part 2.1 and 2.2 so that if no argument is given then “Hello world!” is printed and if an argument is given then print “Hello <name>!”.

## Loops

A while loop will execute a statement multiple times until its condition is false. For example, the following code will print the numbers from 1 to 5:

i = 1

while i < 6:

print(i)

i += 1

A for loop will execute a statement once for each item in a list. For example, the following code will print each fruit in the fruit list:

fruits = ["apple", "banana", "cherry"]

for x in fruits:

print(x)

Change your program so that if more than one argument is given then it will print “Hello <name>!” for each argument given. Note that sys.argv[0] is the name of your python script and shouldn’t be included as a name.

## Part 1 Tests

|  |  |  |
| --- | --- | --- |
|  | Arguments | Expected Result |
| Test 1 | No arguments. | Prints “Hello world!” |
| Test 2 | One argument <name>. | Prints “Hello <name>!” |
| Test 3 | Multiple arguments; e.g. <name 1> <name 2> <name 3> | Prints the following lines:  “Hello <name 1>!”  “Hello <name 2>!”  “Hello <name 3>!” |

# Part 2

## Variables, Types and Ranges

Variables are used to store values. You give a variable a name and assign a value to it. For example, the following code will create a variable named x with the value 5 and a variable named greeting with the value “Hello”:

x = 5

greeting = "Hello"

Variables can have different types. In the above example x is an integer and greeting is a string. If you need to convert a string to an int you can use the int() function. For example, if you need to change a command-line argument (which are always a string) to an integer you can use the following code:

count = int(sys.argv[1])

Write a program that takes one integer argument and prints the numbers from 1 up to the given number, one number per line.

## Range Function

The range() function produces a list of numbers. For example, range(6) will give a list starting at 0 and finishing at 5. Note that the given number is not included in the list.

You can also set the starting number, for example, range(2, 6) will give a list of numbers starting at 2 and finishing with 5.

Finally, you can set the increment (how much will be added each time), for example, range(2, 10, 2) will give the following list: [2, 4, 6, 8].

Write a program that takes two arguments, a number and a string. The number counts how many times to print the string. For example, given arguments “3” and “apple” it must print:

3 apples

2 apples

1 apple

Note that there is an “s” added to the end of “apple” when there is more than one.

## Defining a Function

We have used some functions already (e.g. range() and int()), you can also write your own. A function is defined with the def key word as follows:

def add(x, y):

return x + y

This example takes two parameters (the variables in brackets) and adds them together. Note that it returns the sum. We can use the function as follows:

x = 2

y = 3

z = add(x, y)

print("%d + %d = %d” % (x, y, z))

This will print:

2 + 3 = 5

Write a function called printThing(x, thing) and change your program to use printThing to print x things according to part 3.2 above.

## Part 2 Tests

Combine the programs you have written above to pass all of the below tests.

|  |  |  |
| --- | --- | --- |
|  | Arguments | Expected Result |
| Test 1 | No arguments. | Prints “Error: Need more arguments.” |
| Test 2 | One argument <count>. | Prints numbers from 1 to count on one line each number. |
| Test 3 | Two arguments: <count> <thing> | Prints “<number> <thing>” if number is singular and “<number> <thing>s” if number is greater than 1, with one line for each number and number ranging from count to 1. See part 3.2 above for an example. |

# Part 3

## Reading and Writing Files

In Python files can be read using the open() function. The open function takes a filename and gives a File object. You can use the File object’s read() function to get all of the file’s contents as a string. You can use the File object’s write() function to write a string to the file.

Write a program that reads a file named file.txt and counts how many times the word monkey is found in the file. Create a new file named monkeys.txt with the text “<count> monkeys found.” in it.

Hint: You will need to find a function to help you count how many monkeys are in file.txt. Also, you will need to look at the open() function and think about which file mode to use.

## Part 3 Tests

|  |  |  |
| --- | --- | --- |
|  | Arguments | Expected Result |
| Test 1 | No arguments. A file named file.txt will be present. | Creates a file named monkeys.txt with the text “<count> monkeys found.” in it, where <count> is the number of monkeys found in file.txt. |

# Submission