J. BASKARAN M.Sc., B.Ed. (C.S)
jbaskaran89@gmail.com
Puducherry.

J. ILAKKIA M.Sc., M.Phil., B.Ed. (C.S)
jilakkia@gmail.com
Puducherry.

14. IMPORTING C++ PROGRAMS IN PYTHON

Section – A

Choose the best answer **(1 Mark)**

1. Which of the following is not a scripting language?

   (A) JavaScript (B) PHP (C) Perl **(D) HTML**

2. Importing C++ program in a Python program is called

   (A) wrapping (B) Downloading (C) Interconnecting (D) Parsing

3. The expansion of API is

   (A) Application Programming Interpreter **(B) Application Programming Interface**

   (C) Application Performing Interface (D) Application Programming Interlink

4. A framework for interfacing Python and C++ is

   (A) Ctypes (B) SWIG (C) Cython **(D) Boost**

5. Which of the following is a software design technique to split your code into separate parts?

   (A) Object oriented Programming **(B) Modular programming**

   (C) Low Level Programming (D) Procedure oriented Programming

6. The module which allows you to interface with the Windows operating system is

   (A) OS module (B) sys module (C) csv module (D) getopt module

7. getopt() will return an empty array if there is no error in splitting strings to

   (A) argv variable (B) opt variable **(C)args variable** (D) ifile variable

8. Identify the function call statement in the following snippet.

```
if __name__ =='__main__':
main(sys.argv[1:])
```

   (A) main(sys.argv[1:]) **(B) __name__** (C) __main__ (D) argv

9. Which of the following can be used for processing text, numbers, images, and scientific data?

   (A) HTML (B) C (C) C++ **(D) PYTHON**

10. What does __name__ contains ?

    (A) c++ filename (B) main() name **(C) python filename** (D) os module name

Section-B

Answer the following questions **(2 Marks)**

1. What is the theoretical difference between Scripting language and other programming language?

Scripting Language	Programming Language
A scripting language requires an interpreter.	A programming language requires a compiler.
A scripting language need not be compiled.	A programming languages needs to be compiled before running .
Example: JavaScript, VBScript, PHP, Perl, Python, Ruby, ASP and Tcl.	**Example:** C, C++, Java, C# etc.

2. Differentiate compiler and interpreter.

Compiler	Interpreter
Compiler generates an Intermediate Code.	Interpreter generates Machine Code.
Compiler reads entire program for compilation.	Interpreter reads single statement at a time for interpretation.
Error deduction is difficult	Error deduction is easy
Comparatively faster	Slower
Example: gcc, g++, Borland TurboC	**Example:** Python, Basic, Java

3. Write the expansion of (i) SWIG (ii) MinGW

SWIG - Simplified Wrapper Interface Generator - Both C and C++

MinGW - Minimalist GNU for Windows

4. What is the use of modules?

- Modules are used to break down large programs into small manageable and organized files.
- Modules provide reusability of code.
- We can define our most used functions in a module and import it, instead of copying their definitions into different programs.

5. What is the use of cd command. Give an example.

- **Syntax:** cd <absolute path>

- **"cd"** command used to change directory and absolute path refers to the complete path where Python is installed.
- **Example:** c:\>cd c:\ program files \ openoffice 4 \ program

Section-C

Answer the following questions **(3 Marks)**

1. Differentiate PYTHON and C++.

PYTHON	C++
• Python is typically an "interpreted" language	• C++ is typically a "compiled" language
• Python is a dynamic-typed language	• C++ is compiled statically typed language
• Data type is not required while declaring variable	• Data type is required while declaring variable
• It can act both as scripting and general purpose language	• It is a general purpose language

2. What are the applications of scripting language?

- To automate certain tasks in a program
- Extracting information from a data set
- Less code intensive as compared to traditional programming language
- can bring new functions to applications and glue complex systems together

3. What is MinGW? What is its use?

- MinGW refers to a set of runtime header files.
- It is used in compiling and linking the code of C, C++ and FORTRAN to be run on Windows Operating System.
- MinGW allows to compile and execute C++ program dynamically through Python program using g++.

4. Identify the module ,operator, definition name for the following: welcome.display()

Welcome → Module name

. → Dot operator

display() → Function call

5. What is sys.argv? What does it contain?

- **sys.argv** is the list of command-line arguments passed to the Python program.
- **argv** contains all the items that come along via the command-line input, it's basically an array holding the command-line arguments of the program.
- To use sys.argv, you will first have to import sys.
- **sys.argv[0]** is always the name of the program as it was invoked.
- **sys.argv[1]** is the first argument you pass to the program.
- **main(sys.argv[1])** :
  - Accepts the program file (Python program) and the input file (C++ file) as a list(array).
  - **argv[0]** contains the Python program which is need not to be passed because by default __main__ contains source code reference
  - **argv[1]** contains the name of the C++ file which is to be processed.

Section - D

Answer the following questions: **(5 Marks)**

1. Write any 5 features of Python.

- Python uses Automatic Garbage Collection.
- Python is a dynamically typed language.
- Python runs through an interpreter.
- Python code tends to be 5 to 10 times shorter than that written in C++.
- In Python, there is no need to declare types explicitly.
- In Python, a function may accept an argument of any type, and return multiple values without any kind of declaration beforehand.

2. Explain each word of the following command.

COMMAND: **Python <filename.py> -<i> <C++ filename without cpp extension>**

Where ,

Python	Keyword to execute the Python program from command-line
<filename.py >	Name of the Python program to executed
-< i >	Input mode
<C++ filename without cpp extension>	Name of C++ file to be compiled and executed

3. What is the purpose of sys,os,getopt module in Python. Explain

(i) Python's sys Module:

- This module provides access to some variables used by the interpreter and to functions that interact strongly with the interpreter.
- **sys.argv** is the list of command-line arguments passed to the Python program.
- **argv** contains all the items that come along via the command-line input, it's basically an array holding the command-line arguments of the program.
- To use **sys.argv**, you will first have to import sys.
- **sys.argv[0]** is always the name of the program as it was invoked.
- **sys.argv[1]** is the first argument you pass to the program.
- **main(sys.argv[1])** :
  - Accepts the program file (Python program) and the input file (C++ file) as a list(array).
  - **argv[0]** contains the Python program which is need not to be passed because by default__main__ contains source code reference
  - **argv[1]** contains the name of the C++ file which is to be processed.

(ii) Python's OS Module:

- The OS module in Python provides a way of using operating system dependent functionality.
- The functions that the OS module allows you to interface with the Windows operating system where Python is running on.
- **os.system():** Execute the C++ compiling command in the shell.
- For Example to compile C++ program g++ compiler should be invoked.
- **Command:** os.system ('g++' + <varaiable_name1> '-<mode>' + <variable_name2>

• **os.system**	• function system() defined in os module
• **g++**	• General compiler to compile C++ program under Windows Operating system.
• **variable_name1**	• Name of the C++ file without extension .cpp in string format
• **mode**	• To specify input or output mode. Here it is o prefixed with hyphen.

- **Example:**

  os.system('g++ ' + cpp_file + ' -o ' + exe_file) --

  g++ compiler compiles the file cpp_file and –o (output) send to exe_file

(iii) Python getopt Module:

- The getopt module of Python helps you to parse (split) command-line options and arguments.
- This module provides two functions to enable command-line argument parsing.
- **getopt.getopt method:**
  - This method parses command-line options and parameter list.
- **Syntax of getopt method:**

  <opts>,<args>=getopt.getopt(argv, options, [long_options])
  - Here is the detail of the parameters −
  - **argv** -- This is the argument list of values to be parsed (splited). In our program the complete command will be passed as a list.
  - **options** -- This is string of option letters that the Python program recognize as, for input or for output, with options (like 'i' or 'o') that followed by a colon (:).

    Here colon is used to denote the mode.
  - **long_options** -- This parameter is passed with a list of strings. Argument of Long options should be followed by an equal sign ('=').
  - In our program the C++ file name will be passed as string and 'i' also will be passed along with to indicate it as the input file.
- **getopt()** method returns value consisting of two elements.
- Each of these values are stored separately in two different list (arrays) **opts and args** .
- **Opts** contains list of splitted strings like mode, path and args contains any string if at all not splitted because of wrong path or mode.
- **args** will be an empty array if there is no error in splitting strings by getopt().

- **Example:**
- **opts, args = getopt.getopt (argv, "i:",['ifile='])**
  - where opts contains -- ('-i', 'c:\\pyprg\\p4')]
  - -i: -- **option** nothing but **mode** should be followed by **:**
  - 'c:\\pyprg\\p4' -- **value** nothing but the **absolute path of C++ file.**
- In our examples since the entire command line commands are parsed and no leftover argument, the second argument args will be empty [].
- If args is displayed using print() command it displays the output as [].
- **Example:**
- >>>print(args)
- []

4. Write the syntax for getopt() and explain its arguments and return values.

Python getopt Module:

- The **getopt** module of Python helps you to parse (split) command-line options and arguments.
- This module provides two functions to enable command-line argument parsing.
- **getopt.getopt method:**
  - This method parses command-line options and parameter list.
- **Syntax of getopt method:**

<opts>,<args>=getopt.getopt(argv, options, [long_options])

  - Here is the detail of the parameters −
  - **argv** -- This is the argument list of values to be parsed (splited). In our program the complete command will be passed as a list.
  - **options** -- This is string of option letters that the Python program recognize as, for input or for output, with options (like 'i' or 'o') that followed by a colon (:). Here colon is used to denote the mode.
  - **long_options** -- This parameter is passed with a list of strings. Argument of Long options should be followed by an equal sign ('=').
  - In our program the C++ file name will be passed as string and 'i' also will be passed along with to indicate it as the input file.

- **getopt()** method returns value consisting of two elements.
- Each of these values are stored separately in two different list (arrays) **opts and args** .
- **Opts** contains list of splitted strings like mode, path and args contains any string if at all not splitted because of wrong path or mode.
- **args** will be an empty array if there is no error in splitting strings by getopt().
- **Example:**
- **opts, args = getopt.getopt (argv, "i:",['ifile='])**
  - where opts contains -- ('-i', 'c:\\pyprg\\p4')]
  - -i: -- **option** nothing but **mode** should be followed by **:**
  - 'c:\\pyprg\\p4' -- **value** nothing but the **absolute path of C++ file.**
- In our examples since the entire command line commands are parsed and no leftover argument, the second argument args will be empty [].
- If args is displayed using print() command it displays the output as [].
- **Example:**
- >>>print(args)
- []

5. Write a Python program to execute the following c++ coding.

C++ CODE:

```
#include <iostream>
using namespace std;
int main()
{ cout<<“WELCOME”;
return(0);
}
```

The above C++ program is saved in a file welcome.cpp

PYTHON PROGRAM:

```
import sys, os, getopt
def main(argv):
  cpp_file = ''
  exe_file = ''
  opts, args = getopt.getopt(argv, "i:",['ifile='])
  for o, a in opts:
    if o in ("-i", "--ifile"):
```

```
    cpp_file = a + '.cpp'
    exe_file = a + '.exe'
    run(cpp_file, exe_file)
def run(cpp_file, exe_file):
  print("Compiling " + cpp_file)
  os.system('g++ ' + cpp_file + ' -o ' + exe_file)
  print("Running " + exe_file)
 print("----------------------")
 print
 os.system(exe_file)
 print
if __name__ =='__main__':      #program starts executing from here
 main(sys.argv[1:])
```

STEPS TO IMPORT CPP CODE INTO PYTHON CODE:

- **Select File→New** in Notepad and type the above Python program.
- Save the File as **welcome.py**.
- Click the Run Terminal and open the command window
- Go to the folder of Python using cd command.
- Type the command**: Python c:\pyprg\welcome.py -i c:\pyprg\welcome_cpp**

OUTPUT:

```
----------------------------------------
            WELCOME
----------------------------------------
```

PREPARED BY

J. BASKARAN M.Sc., B.Ed. (C.S)
jbaskaran89@gmail.com

J. ILAKKIA M.Sc., M.Phil., B.Ed. (C.S)
jilakkia@gmail.com