Solutions to Quiz-1
CSE/ISE 337 – Scripting Languages

1. Which of the following statements are TRUE?

a. An interpreter first translates a program to machine code and then executes it
b. An interpreter directly executes source code statement-by-statement.

c. An interpreter may translate source code to an intermediate representation before directly executing it.

d. A compiler always translates higher-level code to machine code before executing it

Feedback:
An interpreter always either executes a program statement-by-statement or translates the source code into an intermediate representation (IR) and executes the IR right away. On the other hand, a compiler translates the source code into machine code and then executes it.

2. Which of the following statements are FALSE? (Select all that apply)
a. Dynamic typing requires explicit type declarations
b. Static typing helps prevent type errors from occurring at runtime
c. Dynamic typing does not require explicit type declarations
d. Dynamically typed languages are less efficient than statically typed languages

Feedback:
Statically typed languages require explicit type declarations, but dynamically typed languages do not.

3. In which of the following scenarios would you use a scripting language?
a. Develop graphical interface to help users book flights
b. Develop an Email filter that will categorize incoming emails as 'Important', 'Social', or 'Promotions' based on the contents in the email. c. Develop the software required to automatically control the speed of a car d. Automatically test the correctness of a program that you have developed

Feedback: Scripting languages are best suited for gluing together existing components and not for developing software that requires a high degree of correctness, stability, and availability.

4. Which of the following statements are TRUE? (Select all that apply)
a. C is more strongly typed than Python

b. Python is more strongly typed than C
c. Python statements correspond to more machine instructions than C statements
d. Python and C are both are both interpreted languages

Feedback: Generally, the higher the language, the less strongly typed it is and the more machine instructions it takes.

5. What will be the result of executing the following code snippet in Python?

```
a = 'Python'
b = '3'
print(a+b)
```

Answer:
Python3

Feedback:
Strings can concatenated by the + operator

6. What will be the result of executing the following code snippet in Python3:

```
a = 'Python'
b = '3'
c = a + b
print c
```

Answer:
Syntax Error

Feedback:
In Python 3, print statements should be followed by parenthesis, e.g.,

print(s) 7. What is the result of executing the following code snippet in Python?

```
for n in range(1,5):
    for k in range(2,n):
        if n%k == 0:
            break
        else:
            print(n, 'is prime')
```

Answer:
3 is prime

Feedback:

The only time the *else* clause will be executed is when n = 3 and k = 2. 8. What value of the variable *num* will terminate the loop in the following code snippert?

```
sum = 0.0
count = 0
num = input("Enter your number: ")
while num:
  sum = sum + float(num)
  count = count + 1
  num = input("Enter your number: ")
print("average is", sum / count)
```

Answer:
The empty string ‘’.

Feedback:
The variable *num* is of type string. Hence, the while loop must be terminated by the empty string ‘’ (two single quotes).

9. This question asked for a valid implementations of Palindrome

Answer:

```
#implementation1
inp = input('Enter a sequence of characters: ')
n = len(inp)
for i in range(n):
 if(inp[i] != inp[(n-1)]):
  print('Not a palindrome')
  break
 n = n-1
else:
 print('A palindrome')

#implementation2
inp = input('Enter a sequence of characters: ')
n = len(inp)
for i in range(n):
 if(inp[i] != inp[-(i+1)]):
  print('Not a palindrome')
  break
else:
 print('A palindrome')
```

Feedback:
Notice that both the implementations use a *for-else* clause. In Python, when an *else* clause is used with the *for* loop, it means that the *else* clause will be executed when the *for* loop terminates normally (i.e., not induced by a *break* statement).

10. What is the output of the following code snippet:

```
def foo(s, d=','):
 r = ''
 for i in range(len(s)):
  if(s[i] != d):
   r = r + s[i]
 return r

print(foo('10,11,12'), end =' | ')
print(foo('10,11,12', ';'))
```

Answer:
101112 | 10,11,12

Feedback:
Notice in the first call to the function *foo()*, no argument is passed to parameter *d*. Hence, the default value of *d* = ',' is used. In the second call to *foo()*, the argument passed to parameter *d* is ';'. However, the argument provided to parameter *s* contains no ';' character. Hence, 10,11,12 is returned.