1. Which of the following statements are true about web apps? **(Select two)** **0.75 / 1 point**

- [ ] Web apps help link offline apps.
- [x] All web apps are APIs.

Correct
Correct! All web apps are APIs.

- [x] Web apps support CRUD actions.

Correct
Correct! Web apps support CRUD actions.

2. In which testing phase do you verify that the application functions within the larger framework? **1 / 1 point**

- ( ) User testing
- ( ) Performance testing
- ( ) Unit testing
- (•) Integration testing

Correct
Correct! Integration tests verify that all involved programs continue functioning as expected after integration. Integration testing also verifies that the application functions within a larger framework.

3. Which of the following statements is TRUE about PyLint? **1 / 1 point**

- ( ) PyLint is a Python-based package creation tool.
- (•) PyLint is a Python static code analysis tool.
- ( ) PyLint is a Python code compiler.
- ( ) PyLint is a Python IDE.

Correct
Correct! PyLint is a Python static code analysis tool. You can use the PyLint library to check the compliance of your Python code with PEP8 guidelines.

4. Which of the following is the right way to name constants based on the PEP8 guidelines? 1 / 1 point

- ○ date_of_birth
- ◉ DATE_OF_BIRTH
- ○ dateOfBirth
- ○ Date_Of_Birth

Correct
Correct! The right way to name constants based on the PEP8 guidelines is DATE_OF_BIRTH.

5. When gathering requirements for an app to manage events, the customer mentions that the project's objective is to improve customer retention. 0 / 1 point

Which of the following requirements describes the above scenario?

- ○ Business requirements
- ○ Constraints
- ○ Technical requirements

6. Why is it a good practice to build a unit testing class? 1 / 1 point

- ○ To check if functions are returning the correct values
- ○ To import the unittest library
- ◉ To call unit tests from a single class object
- ○ To add one or more assertion methods

Correct
Correct! It is a good practice to build a unit testing class as it allows you to call the unit tests from a single class object

7. For this question, review the partial module code and the unit test code: 1 / 1 point

math.py

```
def add(a, b):
    return a + b

def subract(a, b):
    return a - b
```

Test_subtract.py

```
import unittest
from math import add, subtract

class TestMain(unittest.TestCase):
    def test_add(self):
        self.assertEqual(add(6, 4), 10)

    def test_subract(self):
        self.assertEqual(subract(6, 4), 3)
```

When the unit test is executed, the test fails. Which of the following code statement will you modify for the unit test to pass?

- ○ from math import add
- ◉ self. assertEqual (subract (6, 4), 3)
- ○ def test_subract(self)
- ○ self. assertEqual (add (6, 4), 10)

Correct

Correct! The values in the assetEqual () function are incorrect. The assertEqual () function checks if the provided values are the same.

8. Consider the following code in the file *my_code.py* a directory named *mypackage* with *__init__.py*. **1 / 1 point**

```
def print_hello():
 print("Hello")
```

Which of the following **two** statements are the right ways to access the function in the method?

- [ ] *from mypackage import print_hello*

  print_hello ()
- [x] *from mypackage.my_code import print_hello*

  print_hello ()

Correct
Correct! The function must be called from the file, and the file must be called from the package.

- [ ] *import my_code*

  print_hello ()
- [x] *from mypackge import my_code*

  my_code.print_hello ()

Correct
Correct! The function needs to be called from the file and the file from the package.

9. Which of the following is the general structure for testing a package? **1 / 1 point**

- ( ) {package_name} (parameters)
- ( ) import {package_name}
- (•) {package_name}. {module_name}. {function_name} (parameters)
- ( ) run {package_name}. {module_name}. {function_name} (parameters)

Correct
Correct! The general structure for testing your package is {package_name}. {module_name}. {function_name} (parameters)

10. Which of the following is the correct folder structure for the package "MyPackage"? **1 / 1 point**

- ( ) MyPackage -> module_1.py, module_2.py
- (•) MyPackage -> module_1.py, module_2.py, __init__.py
- ( ) MyPackage -> module_1.py, module_2.py, init.py
- ( ) MyPackage -> module_1-> module_1.py MyPackage -> module_2-> module_2.py

Correct
Correct! The module files are stored in the package folder along with a __init__.py file, which imports all modules.