1. Which function overloads the == operator?
    a. __eq__()
    b. __isequal__()
    c. __same__()
    d. None of the above

    Explanation: https://www.pythontutorial.net/python-oop/python-__eq__/

2. Let A and B be objects of class Fun. Which functions are called serially when print(A+B) is executed?
    a. __str__(), __add__()
    b. __add__(), __str__()
    c. __sum__(), __str__()
    d. __str__(), __sum__()

    Explanation: __add__ will be called first, then __str__

3. What is hasattr(obj,name) used for?
    a. To access the attribute of the object
    b. To delete an attribute
    c. To check if an attribute exists or not
    d. To set an attribute

    Explanation: hasattr(obj,name) checks if an attribute exists or not and returns True or False

4. What is delattr(obj,name) used for?
    a. To access the attribute of the object
    b. To delete an attribute
    c. To check if an attribute exists or not
    d. To set an attribute

    Explanation: delattr(obj,name) deletes an attribute in a class.

5. Which of the following is not a type of inheritance?
    a. Double-level
    b. Multi-level
    c. Single-level
    d. Multiple-level

    Explanation: There is no inheritance called double-level in python

6. What type of inheritance is illustrated in the following Python code?
    class A():

```
    pass
class B(A):
    pass
class C(B):
    pass
```

a. Multiple inheritance
b. Hierarchical inheritance
c. Single-level inheritance
d. Multi-level inheritance

Explanation: In multilevel inheritance, features of the base class and the derived class are further inherited into the new derived class. This is similar to a relationship representing a child and a grandfather

7. Review the code below. What is the correct syntax for changing the price to 1.5?

```
fruit_info = {
        'fruit': 'apple',
        'count': 2,
        'price': 3.5
}
```

a. fruit_info['price'] = 1.5
b. my_list [3.5] = 1.5
c. 1.5 = fruit_info ['price]
d. my_list['price'] == 1.5

8. What will be the output of the following Python code?

```
class Test:
    def __init__(self):
        self.x = 0
class Derived_Test(Test):
    def __init__(self):
        Test.__init__(self)
        self.y = 1
def main():
    b = Derived_Test()
    print(b.x,b.y)
main()
```

a) Error because class B inherits A but variable x isn’t inherited
b) 0 0
c) 0 1
d) Error, the syntax of the invoking method is wrong

Explanation: Since the invoking method has been properly invoked, variable x from the main class has been properly inherited and it can also be accessed.

9. Is it necessary that all the abstract methods must be defined from an abstract class?
a) Depending on code
b) Yes, always
c) No, never
d) No, if function is not used, no definition is required

Explanation: That is the rule of programming language that each function declared, must have some definition. There can't be some abstract method that remains undefined. Even if it's there, it would result in a compile time error.

10.
```
from abc import ABC, abstractmethod
class AbstractClass(ABC):
        @abstractmethod
        def some_method():
            pass
ac = AbstractClass()

print(ac)
```

A. **TypeError**
B. ac
C. AbstractClass
D. None