1.What is the output of the below code?

```
class Person:
    __belongs_to = 'Human Race'
    def __init__(self,name):
        self.__height=4.2
        self.__weight=55
        self.__name=name
    def can_speak(self,langauge):
        print(self.__name,'Can Speak',langauge)
```

Choose the correct option w.r.t the type and number of variables

a)The class Person has 3 instance variables, 2 local variables, 1 static variable.

b)The class Person has 3 instance variables, 1 local variables, 1 static variable.

c)The class Person has 2 instance variables, 3 local variables, 1 static variable.

d)The class Person has 4 instance variables, 2 local variables, no static variable.

2. What is the output of the below code?

```
class Demo:
    def __init__(self,num):
        self.__num=num
        self.value=0

    def display_total(self):
        total=self.__num+self.value
        return total

    def set_num(self,num):
        self.__num=num
```

```
obj = Demo(100)
obj.value = 200
obj.set_num(obj.value)
print(obj.display_total())
```

a)200

b)300

c)400

d)100

3.What is the output of the below code?

```
class ClaasA(metaclass=ABCMeta):
    def method1(self):
        return 45
    @abstractmethod
    def method2(self):
        pass
    @abstractmethod
    def method3(self):
        pass

class ClassB(ClassA):
    def method3(self):
        return 25
```

What should be done in so that object of ClassB gets created successfully

a)Implementation of method2() must be provided in classB

b)ClassB can be instantiated without any modification to the given code

c)Implementation of method1() must be provided in classB

d)Implementation of method3() must be removed from classB

4. Which function overloads the == operator?

a) __eq__()

b) __equ__()

c) __isequal__()

d) none of the mentioned

5. Let A and B be objects of class Foo. Which functions are called when print(A + B) is executed?

a) __add__(), __str__()

b) __str__(), __add__()

c) __sum__(), __str__()

d) __str__(), __sum__()

6. Which of the following is the most suitable definition for encapsulation?

a) Ability of a class to derive members of another class as a part of its own definition

b) Means of bundling instance variables and methods in order to restrict access to certain class members

c) Focuses on variables and passing of variables to functions

d) Allows for implementation of elegant software that is well designed and easily modified

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

```
class Demo:
    def __init__(self):
        self.a = 1
        self.__b = 1

    def display(self):
        return self.__b
obj = Demo()
print(obj.a)
```

a) The program has an error because there isn't any function to return self.a

b) The program has an error because b is private and display(self) is returning a private member

c) The program runs fine and 1 is printed

d) The program has an error as you can't name a class member using __b

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

```
class Demo:
    def __init__(self):
        self.a = 1
        self.__b = 1

    def display(self):
        return self.__b

obj = Demo()
print(obj.__b)
```

a) The program has an error because there isn't any function to return self.a

b) The program has an error because b is private and display(self) is returning a private member

c) The program has an error because b is private and hence can’t be printed

d) The program runs fine and 1 is printed

9.What gets printed?

```
class parent:
    def __init__(self, param):
        self.v1 = param


class child(parent):
    def __init__(self, param):
        self.v2 = param


obj = child(11)
print(obj.v1 + " " + obj.v2)
```

a)None None

b)None 11

c)11 None

d)Error

10.What gets printed?

```
class Account:
def __init__(self, id):
    self.id = id
    id = 666
acc = Account(123)
print(acc.id)
```

a)None

b)123

c)666

d)Error

11. John has written the python code given below:

```
class Account:
    def __init__(self,acc_number, balance_amt):
        self.__acc_number=acc_number
        self.__balance_amount=balance_amt
        self.account_type="Saving" #Line1

    def get_acc_number(self):
        return self.__acc_number

    def get_balance_amount(self):
        return self.__balance_amount

a=Account(123456,460000)

print(a.__acc_number,a.account_type,a.__balance_amount) #Line2
```

Note: Line numbers are for reference only.
John wants to display the account details but his code is resulting in an error due to improper variable access.
Which of the following option should he choose to remove the error and get his output?

a) Replace Line2 with:
print(a.get_acc_number(),a.account_type,a.get_balance_amount())
b) Replace Line2 with:
print(a.get_acc_number(),a.account_type,a.__balance_amount)
c) Remove methods get_acc_number() and get_balance_amount() from the class
d) Replace Line1 to: self.__account_type="Saving"

12. Consider the Python code given below:

```
class Base:
    def base_method(self):
        print("Inside baseMethod")
    def common_method(self):
        print("Inside commonMethod: Base")

class Derived(Base):
    def common_method(self):
        print("Inside commonMethod: Derived")
    def derived_method(self):
        print("Inside derivedMethod")
```

```
Line#    Code at module level
1.       base_obj=Base()
2.       derived_obj=Derived()
3.       _____.common_method()
4.       _____.base_method()
5.       _____.common_method()
```

Which reference variable/s should be used at line numbers 3,4,5 so as to get the below output?

Inside commonMethod: Derived
Inside baseMethod
Inside commonMethod: Base

Note: Line numbers are just for reference.
Choose TWO options that apply.

a) derived_obj,base_obj,base_obj
b) derived_obj,derived_obj,derived_obj
c) base_obj,base_obj,derived_obj
d) derived_obj,derived_obj,base_obj

13. Consider the below Python code.
Assume that necessary imports have been done.

```
class Shape(metaclass=ABCMeta):
    def __init__(self):
        print("I am in init")

    @abstractmethod
    def draw_shape(self):
        pass

    @abstractmethod
    def set_color(self):
        pass

class Circle(Shape):
    def draw_shape(self):
        print("Draw Circle")
```

Which of the following statement(s) is/are TRUE?
i) Class Circle cannot be instantiated as it does not implement set_color() method
ii) The above code will result in an error because class Shape has two abstract methods

a) Both i) and ii) are True
b) Neither i) nor ii) is True
c) Only i) is True
d) Only ii) is True

14. Consider the Python code given below.

```
class SomeException(Exception):
    pass

class Calculator:
    def div(self,a,b):
        c=a//b
        raise SomeException
try:
    cal=Calculator()
    cal.div(10,0)
except SomeException:
    print("Some Exception occurred")
except:
    print("Error")
finally:
    print("Finally out of divison method")
```

a) Some Exception occurred
   Error
   Finally out of divison method
b) Finally out of divison method
c) Error
   Finally out of divison method
d) Some Exception occurred
   Finally out of divison method

15. Tech Mindz company has a set of employees. Employees can either be permanent employees or temporary employees. Salary is calculated for all employees but calculation is different for both type of employees. In future there might be a few more categories of employees that may be added.

Company wants an optimal OOP solution for this scenario.

Suggestions from few employees are given below-

A) Jack: Create a class Employee with employeetype as attribute and one method called calculate_salary() which would have logic to calculate salary for different types of employee

B) Tom: Create an Employee class and two child classes namely PermanentEmployee and TemporaryEmployee having method calculate_salary()

C) Olive: Create an abstract Employee class having an abstract method calculate_salary()

Choose the best statement from given options.

a) Suggestions from Jack and Olive combined will result in an optimal solution

b) Suggestion given by Jack is sufficient to get an optimal solution

c) Suggestions from Tom and Olive combined will result in an optimal solution

d) Suggestion given by Tom is sufficient to get an optimal solution

16. What will be the output of the following Python code snippet?

```
class Sample:
    num=100
    def __init__(self,num):
        self.value=None

    def set_value(self,num):
        value=num

    def get_value(self):
        self.value=num
        return self.value

obj=Sample(200)
obj.set_value(10)
print(obj.get_value())
```

a) Code will not compile due to error in set_value() method
b) Code will not compile due to error in get_value() method
c) 100
d) 0

17. Identify the Python code to be written in lines Line 1, Line 2 and Line 3 so as to get the below output.

Emp id : 100
Emp id : 101
Emp Count 2

```
class Employee:
    __count=100
    def __init__(self,name):
        self.name=name
        #Line  1
        #Line  2

    @staticmethod
    def totalcount():
        #Line  3

emp1=Employee("Jack")
print("Emp id :",emp1.id)
emp2=Employee("Tom")
print("Emp id :",emp2.id)
Employee.totalcount()
```

Note: Line numbers are for reference only.

A) Line1: Employee.__count=100
Line2: self.id=Employee.__count+1
Line3: print("Emp Count",Employee.__count-100)

B) Line1: self.id=Employee.__count
Line2: Employee.__count+=1
Line3: print("Emp Count",Employee.__count-100)

C) Line1: self.id=100
Line2: Employee.__count+=1
Line3: print("Emp Count",Employee.__count)

D) Line1: self.id=Employee.__count+1
Line2: Employee.__count+=1
Line3: print("Emp Count",Employee.__count)

a) A

b) B

c) C

d) D

18. Consider the Python code given below.

```
class Base:
    def __init__(self):
        self.__value=200
    def get_value(self):
        return self.__value+1

class Child(Base):
    def get_num(self):
        num=100
        return num

class GrandChild(Child):
    def __init__(self):
        self.num=200

child=Child()
grandchild=GrandChild()
print(grandchild.get_value())
```

What changes should be done in the above code so as to get the output as 201?

a) Make the instance variable of Base class public
b) Add a constructor with statement super().__init__() in Child class
c) Add statement super().__init__() in the constructor of GrandChild class
d) Add a constructor in the Child class and initialize num to 201

19. Consider the Python code given below.

```
class ClassA:
    def __init__(self):
        self.__var_one=100

    def method_one(self):
        return self.__var_one

class ClassB(ClassA):

    def __init__(self,var_two):
        #Line 1 _____________________
        self.__var_two=var_two

    def method_two(self):
        final_val=self.__var_two + self.method_one() #Line2
        return final_val

bobj=ClassB(50)
print(bobj.method_two())
```

What changes should be done in the above code so as to get the output as 150?
Note: Line numbers are for reference only

a) At Line 1 add: super().__init__()
b) At Line 1 add: super().__init__(var_two)
c) No need to make any change in the code, it will produce 150 as output
d) No need to add anything at Line1.
 Change Line2 as: final_val=self.__var_two + super().method_one(

20. What will be the output of the Python code given below?

```
class ExceptionOne(Exception):
    pass

class Test:
    counter=1
    @staticmethod
    def perform_test(temp_list,n):
        try:
            if(temp_list[n]>=5):
                Test.counter+=1
                temp_list[n]-=2
                raise ExceptionOne()
                temp_list[n]=5
            else:
                raise Exception()
        except Exception:
            Test.counter-=5
        except ExceptionOne:
            Test.counter+=5
        print("Data:",temp_list[n])

try:
    t=Test()
    t.perform_test([2,4,7,5,1],3)
finally:
    print("Counter:",Test.counter)
```

a) Data: 3
 Counter: -3
b) Data: 3
 Counter: 7
c) Counter: -3
d) Data: 3

21. Consider the Python code given below.

```
class Alpha:
    def one(self):
        return 10
    def two(self):
        return self.one()

class Beta(Alpha):
    def one(self):
        return 10
    def three(self):
        return 10

class Gamma(Beta):
    def one(self):
        return 10
    def four(self):
        return self.one()
    def three(self):
        return super().two()
    def five(self):
        return 10


gobj=Gamma()
num1=gobj.two()+gobj.three()+gobj.four()
num2=gobj.three()+gobj.four()+gobj.one()
num3=gobj.one()+gobj.two()+gobj.three()
```

Which of the following statement(s) is/are TRUE?

i) Value of num1, num2, num3 will be the same
ii) Error in class Gamma as it cannot override method one() which has already been overriden by parent class Beta
iii) Error in class Gamma as method three() is giving call to method two() using super() which is not written in parent class Beta
iv) Value of num1 and num2 will be the same but num3 will be different

a) Only iv)
b) Both ii) and iii)
c) Only i)
d) Only ii)

22. Consider the Python code given below.

```
class ClassAA:
    def method1(self):
        return 10
    def method2(self):
        return 20

class ClassBB:
    def method3(self):
        return 60
    def method4(self):
        return 70

class ClassCC(ClassAA):
    def method1(self):
        return 30
    def method4(self):
        return 40


class ClassDD(ClassBB):
    def method1(self):
        return 50
    def method2(self):
        return 60

obj1=ClassCC()
obj2=ClassDD()
```

Match the following print statements with corresponding output

Print Statements	**Outputs**
1. print(obj1.method2()+obj2.method4())	A. 100
2. print(obj1.method1()+obj2.method1())	B. 80
3. print(obj1.method4()+obj2.method3())	C. 90

a) 1-C, 2-B, 3-A
b) 1-A, 2-B, 3-C
c) 1-A, 2-C, 3-A
d) 1-C, 2-A, 3-A