

Preguntas PCAP (Tanda 2)

Which of the following is the output of the below Python code?

```
list1 = [1, 3]
```

```
list2 = list1
```

```
list1[0] = 4
```

```
print(list2)
```

☐ [1, 3]

☒ [4, 3]

☐ [1, 4]

☐ [1, 3, 4]

Which is the output of the following Python code fragment?

```
x = True
```

```
y=False
```

```
z= False
```

```
if not x or y:
```

```
    print (1)
```

```
elif not x or not y and z:
```

```
    print (2)
```

```
elif not x or y or not y and x:
```

```
    print (3)
```

```
else:
```

```
    print (4)
```

☐ 4

☐ 2

☒ 3

☐ None

What will be the output of the following code?

```
class Test:  
    def __init__(self, s):  
        self.s = s
```

```
    def print(self):  
        print(s)
```

```
a = Test("Python Class")  
a.print()
```

- ☐ The program gives an error because there is no constructor for class Test.
- ☐ Signature for the print method is incorrect, so an error is thrown.
- ☐ The correct output is .
- ☐ The above code will execute correctly on changing print(s) to print(self.s).

What will be the output of the following code?

```
1 class Test:
1     def __init__(self, s):
1         self.s = s
1
1     def print(self):
1         print(self.s)
1
1 msg = Test()
1 msg.print()
```

- ☐ The program has an error because class Test does not have a constructor.
- ☐ The above code produces an error because the definition of print(s) does not include .
- ☐ It executes successfully but prints nothing.
- ☒ The program has an error because of the constructor call is made without an argument.

What will be the output of the following code?

```
1 class Test:
1     def __init__(self, s = "Welcome"):
1         self.s = s
1
1     def print(self):
1         print(self.s)
1
1 msg = Test()
1 msg.print()
```

- ☐ The program has an error because the constructor is not present in class Test.
- ☐ The above code produces an error because the definition of print(s) does not contain .
- ☐ It executes successfully but prints nothing.
- ☐ The program has an error because of the constructor call is made without an argument.
- ☒ The program executes successfully and prints Welcome.

What will be the output of the following code snippet?

```
1 | class Test:
1 |     def __init__(self):
1 |         self.x = 1
1 |         self.__y = 1
1 |
1 |     def getY(self):
1 |         return self.__y
1 |
1 | val = Test()
1 | print(val.x)
```

☐ The program has an error because x is private and cannot be accessed outside of the class.

☐ The program has an error because you cannot name a variable using .

☒ The program runs fine and prints 1.

☐ The program runs fine and prints nothing.

What will be the output of the following code snippet?

```
1 class Test:
1     def __init__(self):
1         self.x = 1
1         self.__y = 1
1
1     def getY(self):
1         return self.__y
1
1 val = Test()
1 print(val.__y)
```

☒ The program has an error because y is private and should not access it from outside the class.

☐ The program has an error because you cannot name a variable using __y.

☐ The program runs fine and prints 1.

☐ The program runs fine and prints nothing

What will be the output of the following code snippet?

```
1 class Test:
1     def __init__(self):
1         self.x = 1
1         self.__y = 1
1
1     def getY(self):
1         return self.__y
1
1 val = Test()
1 val.x = 45
1 print(val.x)
```

- ☐ The program has an error because x is private and should not access it from outside the class.
- ☐ The program has an error because you cannot name a variable using __y.
- ☐ The program runs fine and prints 1.
- ☒ The program runs fine and prints 45.

Which of the following is a private data field in the given code snippet?

```
1 class Test:
1     def __init__(self):
1         __a = 1
1         self.__b = 1
1         self.__c__ = 1
1         __d__ = 1
```

☐ __a

☒ __b

☐ __c__

☐ __d__

What will be the output of the following code snippet?

```
1 class Test:
1     def __init__(self):
1         self.x = 1
1         self.__y = 1
1
1     def getY(self):
1         return self.__y
1
1 val= Test()
1 val.__y = 45
1 print(val.getY())
```

- ☒ The program has an error because y is private and should not access it from outside the class.
- ☐ The program has an error because you cannot name a variable using __y.
- ☐ The code runs fine and prints 1.
- ☐ The code executes successfully and prints 45.

What will be the output of the following code snippet?

```
1 def main():
1     myCounter = Counter()
1     num = 0
1
1     for i in range(0, 100):
1         increment(myCounter, num)
1
1     print("myCounter.counter =", myCounter.counter, ", number of times =", num)
1
1 def increment(c, num):
1     c.counter += 1
1     num += 1
1
1 class Counter:
1     def __init__(self):
1         self.counter = 0
1
1 main()
```

☐ counter is 101, number of times is 0

☒ counter is 100, number of times is 0

☐ counter is 100, number of times is 100

☐ counter is 101, number of times is 101

What code can we put at the third line of the definition of class B to invoke its superclass's constructor?

```
1 class A:
1     def __init__(self, i = 1):
1         self.i = i
1
1
1 class B(A):
1     def __init__(self, j = 2):
1         _____
1         self.j = j
1
1
1 def main():
1     b = B()
1     print(b.i, b.j)
1
1
1 main()
```

☐ **super().__init__(self)**

☐ **super().__init__()**

☐ **A.__init__()**

☐ **A.__init__(self)**

☐ **B and D**

What will be the output of the following code snippet?

```
1 class A:
1     def __init__(self, x = 1):
1         self.x = x
1
1 class B(A):
1     def __init__(self, y = 2):
1         super().__init__()
1         self.y = y
1
1 def main():
1     b = B()
1     print(b.x, b.y)
1
1 main()
```

☐ 00

☐ 01

☐ 12

☐ 02

What will be the output of the following code snippet?

```
1 class A:
1     def __init__(self):
1         self.__x = 1
1         self.y = 10
1
1     def print(self):
1         print(self.__x, self.y)
1
1 class B(A):
1     def __init__(self):
1         super().__init__()
1         self.__x = 2
1         self.y = 20
1
1 c = B()
1 c.print()
```

☐ 110

☐ 120

☐ 210

☐ 220

What will be the output of the following code snippet?

```
1 class A:
1     def __init__(self, x = 0):
1         self.x = x
1
1     def func1(self):
1         self.x += 1
1
1 class B(A):
1     def __init__(self, y = 0):
1         A.__init__(self, 3)
1         self.y = y
1
1     def func1(self):
1         self.y += 1
1
1 def main():
1     b = B()
1     b.func1()
1     print(b.x, b.y)
1
1 main()
```

☐ 20

☐ 31

☐ 40

☐ 30

What will be the output of the following code snippet? What will be the output of the following code snippet?

```
1 class A:
1     def __new__(self):
1         self.__init__(self)
1         print("A's __new__() invoked")
1
1     def __init__(self):
1         print("A's __init__() invoked")
1
1 class B(A):
1     def __new__(self):
1         print("B's __new__() invoked")
1
1     def __init__(self):
1         print("B's __init__() invoked")
1
1 def main():
1     b = B()
1     a = A()
1
1     main()
```

☐ B's __new__() invoked A's __init__() invoked

☐ B's __new__() invoked A's __new__() invoked

☐ B's __new__() invoked A's __init__() invoked A's __new__() invoked

☐ A's __init__() invoked A's __new__() invoked

What will be the output of the following code snippet?

```
1 class A:
1     def __init__(self, num):
1         self.x = num
1
1 class B(A):
1     def __init__(self, num):
1         self.y = num
1
1 obj = B(11)
1 print ("%d %d" % (obj.x, obj.y))
```

☐ None 11

☐ 11 None

☐ 11 11

☐ AttributeError: 'B' object has no attribute 'x'

What will be the output of the following code snippet?

```
1 class A:
1     def __init__(self):
1         self.x = 1
1
1     def func(self):
1         self.x = 10
1
1 class B(A):
1     def func(self):
1         self.x += 1
1         return self.x
1
1 def main():
1     b = B()
1     print(b.func())
1
1 main()
```

☐ 1

☐ 2

☐ 10

☐ x is not accessible from the object of class B.

What will be the output of the following code snippet?

```
1 class A:
1     def __str__(self):
1         return "A"
1
1 class B(A):
1     def __init__(self):
1         super().__init__()
1
1 class C(B):
1     def __init__(self):
1         super().__init__()
1
1 def main():
1     b = B()
1     a = A()
1     c = C()
1     print(a, b, c)
1
1 main()
```

☐ BBB

☐ ABC

☐ CBA

☐ AAA

What will be the output of the following code snippet?

```
1 class A:
1     def __init__(self, x = 2, y = 3):
1         self.x = x
1         self.y = y
1
1     def __str__(self):
1         return "A"
1
1     def __eq__(self, num ):
1         return self.x * self.y == num.x * num.y
1
1 def main():
1     a = A(1, 2)
1     b = A(2, 1)
1     print(a == b)
1
1 main()
```

☐ True

☐ False

☐ 2

☐ 1