Quiz 02

Due Feb 5 at 10pm **Points** 10 **Questions** 5 **Time Limit** None

Instructions

Answer the following questions in your own words. Do NOT simply cut and paste the information from the slides. You will receive a score of 0 if you copy the prose from the slides.

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	1,290 minutes	10 out of 10

Score for this quiz: **10** out of 10
Submitted Feb 5 at 5:54pm
This attempt took 1,290 minutes.

Question 1

2 / 2 pts

How can Python programmers avoid adding instance variables caused by typing mistakes?

Your Answer:

We can use __slots__ to avoid adding instance variables caused by typing mistakes.

Use the __slots__ attribute to explicitly specify the attributes for the class.

Question 2

2 / 2 pts

I want to define a class 'Quiz' that is initialized with a single parameter x. Python didn't complain when I defined the class but I get an error **"TypeError: object() takes no parameters"** when I try to create an instance of class Quiz as show below.

```
class Quiz:
    def init(self, x):
        self.x = x
```

```
>>> q = Quiz(10)

TypeError: object() takes no parameters


>>> question = Quiz(3)

TypeError: object() takes no parameters
```

Fix the code so I can create and initialize instances of class Quiz properly.

Your Answer:

```
class Quiz:
    def __init__(self, x):
        self.x = x
```

Question 3

2 / 2 pts

Explain how encapsulation can help to improve your code. Why should you use encapsulation? What can go wrong if you don't?

Your Answer:

- Encapsulation helps packing an information and the methods that work on that information in a same class.
- Encapsulation helps in hiding the data instances so that user can not use it indirectly in any other method. It also helps in code re-usability and adding additional features in future.
- It may happen that you made your project but in future if you want to add some extra features or other developer handles your code it becomes very difficult to understand the code if it's not encapsulated.

Question 4

2 / 2 pts

Describe a situation where you might raise an exception. Write the code to raise a ValueError exception to warn users about an invalid value, x, which is negative, but should be >= 0

Your Answer:

```
x:int = int(input("Enter the Number : "))
if x<0:
    raise ValueError("X is an Invalid and a Negative integer")
```

if x < 0:

raise ValueError('x has value' + str(x) + 'but must be >= 0')

Question 5

2 / 2 pts

What is the output of the following code:

```
def raise_exception():
    raise ValueError
    print("leaving raise_exception()")

def inner():
    raise_exception()
    print("leaving inner()")

def outer():
    inner()
    print("leaving outer()")

def way_out():
    try:
        outer()
    except ValueError:
```

```
        print("way_out(): caught a ValueError")
    print("leaving way_out()")

way_out()
```

Your Answer:

way_out(): caught a ValueError
leaving way_out()

way_out(): caught a ValueError leaving way_out()

way_out() calls outer().
outer() calls inner().
inner() calls raise_exception().
raise_exception() raises a ValueError exception that is not caught in raise_exception() so Python checks in inner(). Inner() does not catch the exception so Python pops the stack and returns to outer(). Outer() does not catch the exception so control returns to way_out(). way_out() catches the exception, prints the "caught a ValueError" message then executes the next line which prints the "leaving way_out()" message.

Quiz Score: **10** out of 10