Name:	Roll No:	Marks:

Programming and Data Structures with Python

Quiz 3, 29 Oct 2026, I Semester, 2026–2027

---

1. Consider the Python code on the right.

   Let `r1` and `r2` be two objects of class `Rectangle`. Suppose we want `r1 == r2` to return `True` if both have same area. Which of the following is the correct function to add to the `class` definition?

```
class Rectangle:
  def __init__(self, w, h):
    self.width = w
    self.height = h

  def area(self)
    return (self.width * self.height)
```

(a)
```
def __eq__(rect):
  return (self.area() ==
          rect.area())
```

(b)
```
def __eq__(self,rect):
  return (area() ==
          rect.area())
```

✔(c)
```
def __eq__(self,rect):
  return (self.area() ==
          rect.area())
```

(d)
```
def __eq__(rect):
  return (area() ==
          rect.area())
```

---

2. Consider the Python code on the right.

   Let `c1` and `c2` be two objects of class `Circle`. Suppose we want `c1 == c2` to return `True` if both have same area. Which of the following is the correct function to add to the `class` definition?

```
class Circle:
  def __init__(self, r):
    self.radius = r

  def area(self)
    import math
    return (math.pi * self.radius
                    * self.radius)
```

✔(a)
```
def __eq__(self,circ):
  return (self.area() ==
          circ.area())
```

(b)
```
def __eq__(self,circ):
  return (area() ==
          circ.area())
```

(c)
```
def __eq__(circ):
  return (self.area() ==
          circ.area())
```

(d)
```
def __eq__(circ):
  return (area() ==
          circ.area())
```

---

3. Consider the Python code on the right.

   Let `s1` and `s2` be two objects of class `Square`. Suppose we want `s1 == s2` to return `True` if both have same perimeter. Which of the following is the correct function to add to the `class` definition?

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

  def perimeter(self)
    return (4 * self.side)
```

(a)
```
def __eq__(sq):
  return (self.perimeter() ==
          sq.perimeter())
```

✔(b)
```
def __eq__(self,sq):
  return (self.perimeter() ==
          sq.perimeter())
```

(c)
```
def __eq__(self,sq):
  return (perimeter() ==
          sq.perimeter())
```

(d)
```
def __eq__(sq):
  return (perimeter() ==
          sq.perimeter())
```

4. Consider the Python code on the right.

   Let `t1` and `t2` be two objects of class `Triangle`. Suppose we want `t1 == t2` to return `True` if both have same perimeter. Which of the following is the correct function to add to the `class` definition?

```
class Triangle:
  def __init__(self, s1, s2, s3):
    self.side1 = s1
    self.side2 = s2
    self.side3 = s3

  def perimeter(self)
    return (self.side1 + self.side2
            + self.side3)
```

(a)
```
def __eq__(trg):
  return (self.perimeter() ==
          trg.perimeter())
```

(b)
```
def __eq__(trg):
  return (perimeter() ==
          trg.perimeter())
```

(c)
```
def __eq__(self,trg):
  return (perimeter() ==
          trg.perimeter())
```

✔(d)
```
def __eq__(self,trg):
  return (self.perimeter() ==
          trg.perimeter())
```

The next three questions refer to the following Python program.

```
def foobar(d,k):
  try:
    d[k].append(3)
  except KeyError:
    d[k] = [19]
  except AttributeError:
    d[k] = [28]
  except:
    d[k] = [36]
  else:
    d[k].append(45)

myd = {3:[17], 1:"world"}
foobar(myd,1)
foobar(myd,2)
foobar(myd,3)
```

Note:

- `KeyError` is triggered when a dictionary key is not valid.
- `AttributeError` is triggered when an invalid method (function) is applied to a value.

5. At the end of the program, what is the value of `myd[1][-1]`?

   Answer: 28.

   The value of `myd[1]` is updated from `"world"` to `[28]` because of an `AttributeError`.

6. At the end of the program, what is the value of `myd[2][-1]`?

   Answer: 19.

   The value of `myd[2]` is set to `[19]` because of an `KeyError`.

7. At the end of the program, what is the value of `myd[3][-1]`?

   Answer: 45.

   The value of `myd[3]` is updated to `[17,3]` in the main `try` block and then updated to `[17,3,45]` in the default `else` clause.

The next three questions refer to the following Python program.

```
def foobar(l,pos):
  try:
    l[pos].append(13)
  except AttributeError:
    l[pos] = [18]
  except IndexError:
    l.append([25])
  except:
    l.append([33])
  else:
    l[pos].append(41)

myl = ["world",[77]]
foobar(myl,0)
foobar(myl,1)
foobar(myl,2)
```

Note:

- `IndexError` is triggered when a list index is not valid.
- `AttributeError` is triggered when an invalid method (function) is applied to a value.

8. At the end of the program, what is the value of `myl[0][-1]`?

   Answer: 18.

   The value of `myl[0]` is updated from `"world"` to `[18]` because of an `AttributeError`.

9. At the end of the program, what is the value of `myl[1][-1]`?

   Answer: 41.

   The value of `myl[1]` is updated from `[77]` to `[77,13]` in the main `try` block and then to `[77,13,41]` in the `else` block.

10. At the end of the program, what is the value of `myl[2][-1]`?

    Answer: 25.

    The value of `myl[2]` is set `[25]` because of an `IndexError`.

The next three questions refer to the following Python program.

```
def foobar(d,k):
  try:
    d[k].append(3)
  except KeyError:
    d[k] = [103]
  except AttributeError:
```

```
    d[k] = [203]
  except:
    d[k] = [303]
  else:
    d[k].append(403)

myd = {1:[7], 2:"hello"}
foobar(myd,1)
foobar(myd,2)
foobar(myd,3)
```

Note:

- `KeyError` is triggered when a dictionary key is not valid.
- `AttributeError` is triggered when an invalid method (function) is applied to a value.

11. At the end of the program, what is the value of `myd[1][-1]`?

    Answer: 403.

    The value of `myd[1]` is updated from `[7]` to `[7,3]` in the main `try` block and then to `[7,3,403]` in the `else` block.

12. At the end of the program, what is the value of `myd[2][-1]`?

    Answer: 203.

    The value of `myd[2]` is updated from `"hello"` to `[203]` because of an `AttributeError`.

13. At the end of the program, what is the value of `myd[3][-1]`?

    Answer: 103.

    The value of `myd[1]` is set `[103]` because of an `KeyError`.

The next three questions refer to the following Python program.

```
def foobar(l,pos):
  try:
    l[pos].append(3)
  except AttributeError:
    l[pos] = [103]
  except IndexError:
    l.append([203])
  except:
    l.append([303])
  else:
    l[pos].append(403)

myl = [[7],"hello"]
foobar(myl,0)
foobar(myl,1)
foobar(myl,2)
```

Note:

- `IndexError` is triggered when a list index is not valid.
- `AttributeError` is triggered when an invalid method (function) is applied to a value.

14. At the end of the program, what is the value of `myl[0][-1]`?

    Answer: 403.

    The value of `myl[0]` is updated from `[7]` to `[7,3]` in the main `try` block and then to `[7,3,403]` in the `else` block.

15. At the end of the program, what is the value of `myl[1][-1]`?

    Answer: 103.

    The value of `myl[1]` is updated from `"hello"` to `[103]` because of an `AttributeError`.

16. At the end of the program, what is the value of `myl[2][-1]`?

    Answer: 203.

    The value of `myl[2]` is set `[203]` because of an `IndexError`.
