1. Complete the code to output the statement, “Diego’s favorite food is lasagna”. Remember that precise syntax must be used to receive credit.

1 point

```python
name = "Diego"
fav_food = "lasagna"
print(name + "’s favorite food is " + fav_food)

```

Run
Reset

```
Diego’s favorite food is lasagna
```

2. What's the value of this Python expression: `"big" > "small"`?

1 point

- True
- [✓ False]
- big
- small

3. What is the elif keyword used for?

1 point

- To mark the end of the if statement
- [✓ To handle more than two comparison cases]
- To replace the "or" clause in the if statement
- Nothing - it's a misspelling of the else-if keyword

4. Consider the following scenario about using if-elif-else statements:

1 point

Students in a class receive their grades as Pass/Fail. Scores of 60 or more (out of 100) mean that the grade is "Pass". For lower scores, the grade is "Fail". In addition, scores above 95 (not included) are graded as "Top Score".

Fill in the blanks in this function so that it returns the appropriate "Pass", "Fail", or "Top Score" grade.

```python
def exam_grade(score):
    if score > 95:
        grade = "Top Score"
    elif score >= 60:
        grade = "Pass"
    else:
        grade = "Fail"
    return grade


print(exam_grade(65)) # Should print Pass
print(exam_grade(55)) # Should print Fail
print(exam_grade(60)) # Should print Pass
print(exam_grade(95)) # Should print Pass
print(exam_grade(100)) # Should print Top Score
print(exam_grade(0)) # Should print Fail
```

Run
Reset

```
Pass
Fail
Pass
Pass
Top Score
Fail
```

5. In the following code, what would be the output?

1 point

```python
test_num = 12
if test_num > 15:
    print(test_num / 4)
else:
    print(test_num + 3)

```

- 12
- 3
- [✓ 15]
- 4

6. Fill in the blanks to complete the function. The “complementary_color” function receives a primary color name in all lower case, then prints its complementary color. Currently, the function only supports the primary colors of red, yellow, and blue. It returns "unknown" for all other colors or if the word has any uppercase characters.

1 point

```python
def complementary_color(color):
    if color == "blue":
        complement = "orange"
    elif color == "yellow":
        complement = "purple"
    elif color == "red":
        complement = "green"
    else:
        complement = "unknown"
    return complement

print(complementary_color("blue")) # Should print orange
print(complementary_color("yellow")) # Should print purple
print(complementary_color("red")) # Should print green
print(complementary_color("black")) # Should print unknown
print(complementary_color("Blue")) # Should print unknown
print(complementary_color("")) # Should print unknown
```

Run
Reset

```
orange
purple
```

```
green
unknown
unknown
unknown
```

7. Can you calculate the output of this code?

1 point

```python
def difference(x, y):
    z = x - y
    return z


print(difference(5, 3))
```

```
2
```

8. What's the value of this Python expression?

1 point

`x = 5*2`

`((10 != x) or (10 > x))`

- True
- [✓ False]
- 15
- 10

9. Fill in the blanks to complete the "safe_division" function. The function accepts two numeric variables through the function parameters and divides the "numerator" by the "denominator". The function's main purpose is to prevent a ZeroDivisionError by checking if the "denominator" is 0. If it is 0, the function should return 0 instead of attempting the division. Otherwise all other numbers will be part of the division equation. Complete the body of the function so that the function completes its purpose.

1 point

```python
def safe_division(numerator, denominator):
    # Complete the if block to catch any "denominator" variables
    # that are equal to 0.
    if denominator == 0:
        result = 0
    else:
        # Complete the division equation.
        result = numerator/denominator
    return result


print(safe_division(5, 5)) # Should print 1.0
print(safe_division(5, 4)) # Should print 1.25
print(safe_division(5, 0)) # Should print 0
print(safe_division(0, 5)) # Should print 0.0
```

Run

Reset

```
1.0
1.25
0
0.0
```

10. What are some of the benefits of good code style? Select all that apply.

1 point

- ☐ Makes sure the author will refactor it later
- ☐ Allows it to never have to be touched again
- ☑ Easier to maintain
- ☑ Makes the intent of the code obvious

Upgrade to submit