Here are the correct answers to the questions:

1. C) %
2. B) 0
3. A) 36
4. D) 0
5. B) 4
6. C) the finally block will be executed no matter if the try block raises an error or not.
7. A) It is used to raise an exception.
8. C) in defining a generator

Explanations:

1. % is the modulo operator, used to find the remainder after division.
2. // is the floor division operator, which rounds down the result to the nearest whole number.
3. << is the left shift operator, which shifts the bits of a number to the left by a specified number of positions.
4. & is the bitwise AND operator, which performs a bitwise AND operation on the binary representations of two numbers.
5. | is the bitwise OR operator, which performs a bitwise OR operation on the binary representations of two numbers.
6. The finally block is always executed, regardless of whether an exception occurs or not. It's often used for essential cleanup tasks, such as closing files or releasing resources.
7. The raise keyword is used to manually raise an exception, signaling an error or unexpected condition.
8. The yield keyword is used to create generators, which are functions that can pause and resume their execution, producing a sequence of values one at a time.

Here are the correct answers to questions 9 and 10:

9. A) _abc and C) abc2

Explanation:

- Variable names in Python must start with a letter (A-Z or a-z) or an underscore (_).
- They can only contain letters, numbers, and underscores.
- They cannot start with a number.

10. A) yield and B) raise

Explanation:

- Keywords are reserved words in Python that have specific meanings and cannot be used as variable names.
- "look-in" is not a keyword in Python.

List of valid variable names from the options:

- _abc (starts with an underscore)
- abc2 (starts with a letter and contains only letters and numbers)

List of keywords from the options:

- yield (used to create generators)
- raise (used to raise exceptions)

Question 11:

Python

```python
def factorial(num):
    if num == 0:
        return 1
    else:
        return num * factorial(num - 1)

number = int(input("Enter a non-negative integer: "))
if number < 0:
    print("Factorial is not defined for negative numbers.")
else:
    result = factorial(number)
    print(f"The factorial of {number} is {result}.")
```

Question 12:

Python

```python
def is_prime(num):
    if num <= 1:
        return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True

number = int(input("Enter a positive integer: "))
if is_prime(number):
    print(f"{number} is a prime number.")
else:
    print(f"{number} is a composite number.")
```

Question 13:

Python

```python
def is_palindrome(text):
    return text == text[::-1]
```

```
string = input("Enter a string: ")
if is_palindrome(string):
    print(f"{string} is a palindrome.")
else:
    print(f"{string} is not a palindrome.")
```

Question 14:

Python

```
import math

def third_side(a, b):
    return math.sqrt(a**2 + b**2)

side1 = float(input("Enter the first side: "))
side2 = float(input("Enter the second side: "))
hypotenuse = third_side(side1, side2)
print("The third side (hypotenuse) is:", hypotenuse)
```

Question 15:

Python

```
def char_frequency(text):
    char_counts = {}
    for char in text:
        char_counts[char] = char_counts.get(char, 0) + 1
    return char_counts

string = input("Enter a string: ")
frequencies = char_frequency(string)
print("Character frequencies:")
for char, count in frequencies.items():
    print(f"{char}: {count}")
```