# Exam 1

# Study Guide

# Solution

# True/False Questions and Answers

1. T/F: Python was created by Guido van Rossum in the late 1980s.

Answer: True.

2. T/F: Python is a low-level programming language focusing primarily on system operations and memory management.

Answer: False. Python is a high-level programming language.

3. T/F: Python's syntax allows programmers to write programs with fewer lines than some other programming languages.

Answer: True.

4. T/F: Python is only suitable for web development and cannot be used in data science.

Answer: False. Python is widely used in web development, data science, machine learning, and more.

5. T/F: Python is a proprietary software, and obtaining it requires a purchase.

Answer: False. Python is open-source.

6. T/F: In Python, variables must be declared before they are assigned a value.

Answer: False. Python does not require explicit variable declaration before assignment.

7. T/F: The assignment operator in Python is ==.

Answer: False. The assignment operator in Python is =.

8. T/F: Python variables are case-sensitive.

Answer: True.

9. T/F: Python allows using reserved words as variable names.

Answer: False. Reserved words cannot be used as variable names.

10. T/F: The ** operator in Python is used for multiplication.

Answer: False. The ** operator is used for exponentiation.

11. T/F: Python uses the = operator for both assignment and equality check.

Answer: False. The = operator is used for assignment, while == is used for equality check.

12. T/F: The input() function in Python always returns a value of the string type.

Answer: True.

13. T/F: In Python, the += operator is used to subtract a value from a variable.

Answer: False. The += operator is used to add a value to a variable.

14. T/F: Type casting can change a variable from an integer type to a string type.

Answer: True.

15. T/F: Lists in Python can only contain elements of the same data type.

Answer: False. Lists can contain elements of different data types.

16. T/F: The index of the first element in a Python list is 1.

Answer: False. The index of the first element is 0.

17. T/F: Lists in Python are immutable.

Answer: False. Lists are mutable, meaning their elements can be changed.

18. T/F: You can use negative indexing to access elements from the end of a list in Python.

Answer: True.

19. T/F: The len() function can be used to find the number of elements in a list.

Answer: True.

20. T/F: The elif statement can only be used after an else statement.

Answer: False. The elif statement is used after an if statement and before an else statement.

21. T/F: In Python, if statements cannot be nested within another if statement.

Answer: False. if statements can be nested within another if statement.

22. T/F: The else statement is executed only if the if statement's condition is True.

Answer: False. The else statement is executed when the if statement's condition is False.

23. T/F: Python requires parentheses around the condition in an if statement.

Answer: False. Parentheses are not required around the condition in an if statement in Python, though they can be used for clarity.

24. T/F: The == operator is used to assign a value to a variable in an if statement's condition.

Answer: False. The == operator is used for comparison, not assignment. The = operator is used for assignment.

25. T/F: A for loop can only iterate over lists in Python.

Answer: False. A for loop can iterate over any iterable object, including lists, tuples, dictionaries, sets, and strings.

26. T/F: The range() function in Python includes the end value specified in its arguments.

Answer: False. The range() function generates numbers up to, but not including, the end value.

27. T/F: You can use the break statement inside a for loop to exit the loop.

Answer: True.

28. T/F: Nested for loops are not allowed in Python.

Answer: False. Nested for loops are allowed and commonly used in Python.

29. T/F: List comprehension can replace most for loops that are used to create or modify lists.

Answer: True.

30. T/F: The condition for a while loop must be True for the loop to execute.

Answer: True.

31. T/F: It's impossible to create an infinite loop with a while loop in Python.

Answer: False.

32. T/F: In Python, a while loop can only execute a single statement as long as the condition remains true.

Answer: False. It can execute multiple statements in a block.

33. T/F: The break statement can be used to exit a while loop before the condition becomes False.

Answer: True.

34. T/F: The while loop cannot be used for iterating over a sequence of numbers generated by range().

Answer: False. While not its primary use, a while loop can iterate over a sequence generated by range() when properly coded.

35. T/F: A function in Python can only have one parameter or argument.

Answer: False.

36. T/F: A function defined without a return statement automatically returns None.

Answer: True.

37. T/F: In Python, arguments passed to functions are always passed by value.

Answer: False. Python uses "Call by Object Reference" or "Call by Assignment".

38. T/F: Variables defined inside a function are accessible from anywhere in the program.

Answer: False. They are only accessible within the function.

39. T/F: The global keyword allows a function to modify a variable defined outside of it.

Answer: True.

40. T/F: In Python, a class is a blueprint for creating objects.

Answer: True.

41. T/F: Objects in Python cannot have properties and behaviors.

Answer: False.

42. T/F: Inheritance allows a child class to inherit methods and properties from a parent class.

Answer: True.

43. T/F: Encapsulation prevents the direct access of class properties from outside the class.

Answer: True.

44. T/F: Polymorphism in Python refers to the ability to redefine methods for derived classes.

Answer: True.

45. T/F: Every class in Python must have a constructor method defined.

Answer: False.

46. T/F: Instance variables are shared across all instances of a class.

Answer: False.

47. T/F: Method overriding is not supported in Python.

Answer: False.

48. T/F: The self parameter in class methods is optional.

Answer: False.

49. T/F: Static methods in Python cannot access instance variables.

Answer: True.

50. T/F: A Python module can only contain function definitions.

Answer: False. A module can contain function definitions, class definitions, variables, and executable statements.

51. T/F: You can import specific functions from a module without importing the entire module.

Answer: True.

52. T/F: The alias of a module can be used interchangeably with its original name after importing.

Answer: True. Once an alias is defined, it is used to refer to the module.

53. T/F: NumPy arrays can only hold elements of the same data type.

Answer: True. NumPy arrays are homogenous.

54. T/F: Python packages and modules refer to the same concept.

Answer: False. A package is a collection of modules organized in a directory hierarchy.

55. T/F: The open() function in Python can only create new files, not open existing ones.

Answer: False.

56. T/F: When opening a file in append mode ('a'), if the file does not exist, Python will create it.

Answer: True.

57. T/F: The readlines() method returns the entire content of the file as a single string.

Answer: False. It returns a list of lines.

58. T/F: Using the with statement to open files automatically closes the file when the block is exited.

Answer: True.

59. T/F: You cannot check if a file exists before attempting to delete it using the os module.

Answer: False.

60. T/F: Polymorphism in Python cannot be achieved through method overloading.

Answer: True.

61. T/F: Python supports operator overloading, allowing operators to be redefined for custom behaviors in classes.

Answer: True.

62. T/F: The __add__ method in Python can be overridden to change the behavior of the + operator for objects of a class.

Answer: True.

63. T/F: Encapsulation in Python restricts access to methods and variables from outside the class.

Answer: True.

64. T/F: In Python, private member variables can be accessed directly from outside the class without any restrictions.

Answer: False.

65. T/F: Protected members in Python are enforced by the language and cannot be accessed from outside the class.

Answer: False.

66. T/F: Method overloading in Python can be directly achieved by defining multiple methods with the same name but different parameters.

Answer: False.

67. T/F: The __init__ method cannot be considered a form of polymorphism when it involves default parameter values.

Answer: False.

68. T/F: Operator overloading is limited to arithmetic operators in Python.

Answer: False.

69. T/F: Encapsulation can be used to hide the implementation details of a class from the users of that class.

Answer: True.

# Multiple Choice Questions and Answers

**70.Who created Python?**

A) Linus Torvalds
B) James Gosling
C) Guido van Rossum
D) Dennis Ritchie

Answer: C) Guido van Rossum.

**71.Which of the following is a key feature of Python?**

A) Low-level operations
B) Object-oriented
C) Requires explicit memory management
D) Platform dependent

Answer: B) Object-oriented.

**72.Python can be used for:**

A) Web development only
B) Machine learning only
C) Data science only
D) All of the above

Answer: D) All of the above.

**73.Which of the following about Python is true?**

A) It has a small standard library.
B) It is primarily used for system administration tasks.
C) It is considered easy for beginners to learn.
D) It does not support cross-platform applications.

Answer: C) It is considered easy for beginners to learn.

**74.Python's design philosophy emphasizes:**

A) Code complexity
B) Readability and simplicity
C) Lengthy code for better understanding
D) Close to machine code for speed

Answer: B) Readability and simplicity.

**75.What is the correct way to assign the integer value 5 to the variable num?**

A) num == 5
B) 5 = num
C) num = 5
D) int num = 5

Answer: C) num = 5

**76.Which of the following is allowed as a Python variable name?**

A) 2things
B) thing2
C) for
D) None

Answer: B) thing2

**77.How can you confirm the value assigned to a variable?**

A) Using the assignment operator again
B) Printing the variable's value
C) Declaring the variable a second time
D) Checking the variable's type

Answer: B) Printing the variable's value

**78.What symbol is used in Python to start a comment?**

A) //
B) #
C) /*
D) $

Answer: B) #

**79.Which operator is used for floor division in Python?**

A) /
B) //
C) %
D) **

Answer: B) //

**80.What does the != operator in Python signify?**

A) Assignment
B) Equality
C) Less than
D) Not equal

Answer: D) Not equal

**81.Which of the following is a logical operator in Python?**

A) &
B) |
C) not
D) ==

Answer: C) not

**82.To convert a string to an integer, which function is used in Python?**

A) int()
B) str()
C) float()
D) bool()

Answer: A) int()

**83.Which operator checks if two variables are the same object?**

A) ==
B) !=
C) is
D) not in

Answer: C) is

**84.What does the append() method do in Python lists?**

A) Deletes the last element of the list
B) Adds a new element to the beginning of the list
C) Adds a new element to the end of the list
D) Sorts the list in ascending order

Answer: C) Adds a new element to the end of the list

**85. Which of the following will create a sublist of the first three elements of a list named numbers?**

A) numbers[0:3]
B) numbers[1:3]
C) numbers[:3]
D) Both A and C

Answer: D) Both A and C

**86. How can you remove the first occurrence of an element in a list?**

A) list.delete(element)
B) list.remove(element)
C) list.pop(element)
D) list.erase(element)

Answer: B) list.remove(element)

**87. What is the result of using the list.insert(index, element) method?**

A) The element is added to the end of the list
B) The element replaces the element at the specified index
C) The element is inserted at the specified index, moving other elements to the right
D) The list is sorted after inserting the element

Answer: C) The element is inserted at the specified index, moving other elements to the right

**88. Which function would you use to combine the elements of another list to the end of the current list?**

A) list.append(list2)
B) list.combine(list2)
C) list.extend(list2)
D) list.add(list2)

Answer: C) list.extend(list2)

**89. Which statement is used to execute a block of code only if a specified condition is True?**

A) else
B) elif
C) if
D) while

Answer: C) if

**90. What will happen if the condition in an if statement evaluates to False?**

A) The program will raise an error.
B) The if block will be executed regardless.
C) The program immediately exits.
D) Any subsequent else or elif blocks may be considered for execution.

Answer: D) Any subsequent else or elif blocks may be considered for execution.

**91. How can you execute multiple conditions sequentially in Python?**

A) Using multiple if statements only
B) Using elif statements following an if statement
C) By using else statements alone
D) Python does not support multiple conditions

Answer: B) Using elif statements following an if statement

**92. Which of the following is the correct way to write an if statement in Python?**

A) if x > 10: print("Greater than 10.")
B) if (x > 10) print("Greater than 10.")
C) if x > 10 print:("Greater than 10.")
D) if x > 10 then print("Greater than 10.")

Answer: A) if x > 10: print("Greater than 10.")

**93. For an if-elif-else chain, when is the else block executed?**

A) After every if statement
B) When all if and elif conditions are False
C) Before any elif statements
D) It is never executed

Answer: B) When all if and elif conditions are False

**94.What does the continue statement do inside a for loop?**

A) Exits the loop immediately
B) Skips the rest of the code inside the loop for the current iteration
C) Pauses the loop execution temporarily
D) None of the above

Answer: B) Skips the rest of the code inside the loop for the current iteration

**95.Which of the following is correct about the range() function?**

A) range(5) generates numbers from 1 to 5
B) range(1, 5) includes the number 5
C) range(1, 5, 2) generates the sequence 1, 3, 5
D) range(5) generates the sequence 0, 1, 2, 3, 4

Answer: D) range(5) generates the sequence 0, 1, 2, 3, 4

**96.How can you iterate over a list in reverse order using a for loop?**

A) Use the reverse() function before the loop
B) Use the reversed() function in the loop declaration
C) Start the range with the last element and decrement
D) Python does not support iterating in reverse order

Answer: B) Use the reversed() function in the loop declaration

**97.Which of the following creates a list of squares of numbers from 0 to 4?**

A) [x**2 for x in range(5)]
B) [x*x for x in range(4)]
C) [x^2 for x in range(5)]
D) for x in range(5): x*x

Answer: A) [x**2 for x in range(5)]

**98.Which statement about while loops is correct?**

A) The loop condition must be updated within the loop to avoid infinite loops.
B) A while loop always executes at least once.
C) A while loop is the only loop that can execute a set of statements in Python.
D) while loops cannot use the else clause.

Answer: A) The loop condition must be updated within the loop to avoid infinite loops.

**99.What does the continue statement do in a while loop?**

A) Terminates the loop
B) Skips the rest of the code inside the loop for the current iteration and moves to the next iteration
C) Acts the same as a break statement
D) Resets the loop condition to True

Answer: B) Skips the rest of the code inside the loop for the current iteration and moves to the next iteration

**100. How can infinite loops be prevented in a while loop?**

A) By ensuring the loop condition becomes False at some point
B) By avoiding the use of variables within the loop
C) By not using break or continue statements
D) Infinite loops cannot be prevented

Answer: A) By ensuring the loop condition becomes False at some point

**101. Which scenario is best suited for a while loop?**

A) When the exact number of iterations is known
B) When iterating over each character in a string
C) When the number of iterations is not known in advance and depends on some condition
D) When you need to iterate over a list of known size

Answer: C) When the number of iterations is not known in advance and depends on some condition

**102. What will happen if the condition in a while loop never becomes False?**

A) The loop will terminate after a certain number of iterations to prevent freezing.
B) Python automatically adds a break statement after 1000 iterations.
C) The loop will continue to execute indefinitely, creating an infinite loop.
D) A runtime error will occur to stop the execution.

Answer: C) The loop will continue to execute indefinitely, creating an infinite loop.

**103. Which of the following is true about lambda functions in Python?**

A) They can have multiple expressions.
B) They are defined using the def keyword.
C) They can take any number of arguments but have only one expression.
D) They cannot take arguments.

Answer: C) They can take any number of arguments but have only one expression.

**104. What does the *args parameter in a function definition allow?**

A) It specifies that the function should take a fixed number of arguments.
B) It allows the function to accept an arbitrary number of arguments.
C) It restricts the function to only keyword arguments.
D) It unpacks the arguments from a list.

Answer: B) It allows the function to accept an arbitrary number of arguments.

**105. How is a variable defined inside a function treated?**

A) As a global variable
B) As a local variable
C) As a static variable
D) As an instance variable

Answer: B) As a local variable

**106. What is the scope of a variable declared outside of any function in Python?**

A) Local scope
B) Global scope
C) Nonlocal scope
D) Block scope

Answer: B) Global scope

**107. What is the purpose of the __init__ method in a class?**

A) To perform some action when the class is called
B) To initialize a newly created object
C) To delete an instance of a class
D) To perform calculations

Answer: B) To initialize a newly created object

**108. Which keyword is used for creating a base class in Python?**

A) parent
B) super
C) class
D) base

Answer: C) class

**109. How do you create a subclass that inherits from a superclass?**

A) class Subclass(super):
B) class Subclass(Superclass):
C) subclass Subclass extends Superclass:
D) class Subclass inherits Superclass:

Answer: B) class Subclass(Superclass):

**110. Which of the following is true about encapsulation?**

A) It exposes all the details of a class to the outside world.
B) It restricts access to methods and variables to prevent data from direct modification.
C) It allows for unrestricted access to any class's variables.
D) It is not supported by Python.

Answer: B) It restricts access to methods and variables to prevent data from direct modification.

**111. What does polymorphism allow in object-oriented programming?**

A) Changing the way loops work in Python
B) Allowing one interface for multiple forms of data
C) Encrypting data automatically
D) Creating multiple instances of the same class

Answer: B) Allowing one interface for multiple forms of data

**112. Which method is automatically called when an object is created?**

A) __create__()
B) __init__()
C) __new__()
D) __start__()

Answer: B) __init__()

**113. How can you define a private variable in a Python class?**

A) By prefixing the variable name with an underscore _
B) By prefixing the variable name with two underscores __
C) By using the private keyword
D) By declaring it outside the class

Answer: B) By prefixing the variable name with two underscores __

**114. What is method overriding?**

A) Changing the return type of a method
B) Adding new methods to a class that do not exist in the superclass
C) Providing a new implementation of a method inherited from a parent class
D) Removing a method from a parent class

Answer: C) Providing a new implementation of a method inherited from a parent class

**115. Which of the following is not a pillar of object-oriented programming?**

A) Inheritance
B) Encapsulation
C) Polymorphism
D) Recursion

Answer: D) Recursion

**116. How do you define a static method in a class?**

A) Using the @staticmethod decorator
B) Prefixing the method name with static
C) Declaring it with the static keyword
D) By declaring it in the global scope outside of any class

Answer: A) Using the @staticmethod decorator

**117. What is a Python module?**

A) A Python interpreter
B) A built-in Python function
C) A file containing Python definitions and statements
D) A Python package

Answer: C) A file containing Python definitions and statements

**118. How can you import a module named math with an alias m?**

A) import math as m
B) import m from math
C) alias math as m
D) module math as m

Answer: A) import math as m

**119. Which statement is true about the from keyword used in Python imports?**

A) It is used to import the entire module only.
B) It allows importing only specified parts from a module.
C) It renames the module upon importing.
D) It prevents the import of a module.

Answer: B) It allows importing only specified parts from a module.

**120. What does the NumPy function np.array() do?**

A) Creates a new Python list
B) Converts a Python list into a NumPy array
C) Sorts an existing NumPy array
D) Deletes elements from a NumPy array

Answer: B) Converts a Python list into a NumPy array

**121. Which of the following is a feature of NumPy arrays?**

A) Ability to hold elements of multiple data types
B) Slow execution speed for mathematical operations
C) Support for large multi-dimensional arrays and matrices
D) Limited to one-dimensional arrays

Answer: C) Support for large multi-dimensional arrays and matrices

**122. Which mode should you use with the open() function to read an existing file?**

A) 'w'
B) 'r'
C) 'a'
D) 'x'

Answer: B) 'r'

**123. What does the write() method do?**

A) Reads the content of a file
B) Deletes the content of a file
C) Appends content at the end of the file
D) Writes content to a file

Answer: D) Writes content to a file

**124. How do you delete a file in Python?**

A) file.delete('filename.txt')
B) os.remove('filename.txt')
C) delete.file('filename.txt')
D) os.delete('filename.txt')

Answer: B) os.remove('filename.txt')

**125. Which method is used to read the entire content of a file as a single string?**

A) read()
B) readline()
C) readlines()
D) readall()

Answer: A) read()

**126. What is the result of calling os.rmdir('myfolder')?**

A) Deletes a file named 'myfolder'
B) Reads the content of 'myfolder'
C) Deletes the directory 'myfolder' if it is empty
D) Creates a new directory named 'myfolder'

Answer: C) Deletes the directory 'myfolder' if it is empty

**127. Which of the following is not a form of polymorphism in Python?**

A) Method Overriding
B) Method Overloading
C) Operator Overloading
D) Function Overriding

Answer: D) Function Overriding

**128. How can polymorphism be achieved in Python?**

A) By using default parameter values in methods
B) Through inheritance and method overriding
C) By creating multiple classes with the same methods
D) Both A and B

Answer: D) Both A and B

**129. What is encapsulation?**

A) A technique to inherit methods from parent classes
B) A process of wrapping variables and methods into a single entity
C) A method to overload operators in Python
D) A way to create abstract classes in Python

Answer: B) A process of wrapping variables and methods into a single entity

**130. Which of the following is true about private variables in Python?**

A) They start with a single underscore _
B) They can be accessed directly from outside the class
C) They start with two underscores __
D) They are enforced by the Python runtime

Answer: C) They start with two underscores __

**131. What does operator overloading allow in Python?**

A) Changing the value of existing variables
B) Creating new types of operators
C) Redefining existing operators to work with user-defined objects
D) Removing default operators from Python

Answer: C) Redefining existing operators to work with user-defined objects

**132. Which is a correct way to implement operator overloading for +?**

A) Defining a __plus__ method in your class
B) Defining a __add__ method in your class
C) Overriding the plus function in the global scope
D) Python does not support operator overloading

Answer: B) Defining a __add__ method in your class

**133. Which access modifier indicates a protected member in Python?**

A) public
B) private
C) _ (single underscore)
D) __ (double underscore)

Answer: C) _ (single underscore)

**134. Which statement is true about encapsulation in Python?**

A) It enforces strict data hiding as in Java.
B) It relies on the programmer's discipline rather than language enforcement.
C) Private members can be accessed using direct attribute names.
D) It cannot be achieved in Python due to its dynamic nature.

Answer: B) It relies on the programmer's discipline rather than language enforcement.

**135. How can you achieve method overloading in Python?**

A) By overloading the constructor method to accept different types of arguments
B) Using default parameter values and *args, **kwargs for dynamic argument passing
C) Creating multiple methods with the same name within a class
D) Python does not allow method overloading

Answer: B) Using default parameter values and *args, **kwargs for dynamic argument passing

**136. What is the purpose of the __eq__ method in Python?**

A) To compare the memory addresses of two objects
B) To determine if two strings are equal in length
C) To check the equality of two objects based on their content
D) To assign a new value to an object

Answer: C) To check the equality of two objects based on their content

```
x = 10
y = "10"
z = 10.0
print(x == y, x == int(y), x == z, x == int(z))
```

**137. What is the output of this code?**

A) False False True True
B) False True True True
C) True True False True
D) False True False True

Answer: B

```
my_list = [1, "2", [3, 4, "five"], "six", 7]
result = [str(item)[0] for item in my_list if type(item) in [list, str]]
print("".join(result))
```

**138. What does this code print?**

A) 2fives
B) 25s
C) 2fs
D) 235six7
E) 2[s

Answer: E

```
def check_prime(number):
    for divisor in range(2, int(number ** 0.5) + 1):
        if number % divisor == 0:
            return False
    return True if number > 1 else False

print(check_prime(4), check_prime(11))
```

**139. What is the correct output?**

A) True True
B) False True
C) True False
D) False False

Answer: B

```
class Animal:
    def speak(self):
        return "Some sound"

class Dog(Animal):
    def speak(self):
        return "Woof"

animals = [Animal(), Dog()]
sounds = [animal.speak() for animal in animals]
print(sounds)
```

**140. What is the output of this code?**

A) ["Some sound", "Some sound"]
B) ["Some sound", "Woof"]
C) ["Woof", "Woof"]
D) ["Animal", "Dog"]

Answer: B

```
class Counter:
    __secretCount = 0

    def count(self):
        self.__secretCount += 1
        print(self.__secretCount)

counter = Counter()
counter.count()
counter.count()
print(counter.__secretCount)
```

**141. What happens when this code is run?**

A) Prints 1, 2, and an AttributeError
B) Prints 1, 2, and 0
C) Prints 1, 2, and 2
D) Only prints 1 and 2

Answer: A

```
i, j = 5, 10
while i < j:
    print(i, end="")
    i += 2
    j -= 1
```

**142. What is the correct output?**

A) 57
B) 579
C) 575
D) 57910

Answer: A

```
input_dict = {"a": 1, "b": 2, "c": 3, "d": 4}
output_dict = {key:val for key, val in input_dict.items() if val % 2 == 0}
print(output_dict)
```

**143. What does this code print?**

A) {"a": 1, "c": 3}
B) {"b": 2, "d": 4}
C) {"a": 2, "c": 4}
D) {1: "a", 3: "c"}

Answer: B

```
def multiply(x, y):
    return x * y

def multiply(sequence, times):
    return sequence * times

print(multiply(2, 3), multiply('A', 3))
```

**144. What is the output?**

A) 6 'AAA'
B) TypeError
C) 6 6
D) 'AAA' 'AAA'

Answer: A

```
class SecretNumber:
    def __init__(self):
        self.__number = 42

    def guess(self, number):
        return "Correct!" if number == self.__number else "Try again!"

obj = SecretNumber()
print(obj.guess(42), obj.guess(obj._SecretNumber__number))
```

**145. What is the output?**

A) "Correct!" "Try again!"
B) "Try again!" "Correct!"
C) "Correct!" "Correct!"
D) AttributeError

Answer: C

```
class A:
    def say(self):
        return "A",

class B(A):
    def say(self):
        return super().say() + ("B",)

class C(A):
    def say(self):
        return super().say() + ("C",)

class D(B, C):
    pass

obj = D()
print(obj.say())
```

**146. What is the output?**

A) ('A', 'B', 'C')
B) ('A', 'C', 'B')
C) ('A', 'B')
D) ('A', 'C')

Answer: B

```
my_list = [1, 2, 3, 4, 5]
result = my_list[1:4:2] + my_list[:1:-2]
print(result)
```

**147. What is the output of this code?**

A) [2, 4, 5]
B) [2, 4, 4, 2]
C) [2, 4]
D) [2, 4, 5, 3]

Answer: D

```
def format_string(name, age):
    return f"My name is {name} and I am {age} years old."

print(format_string(age=30, name="John"))
```

**148. What does this code print?**

A) My name is 30 and I am John years old.
B) SyntaxError
C) My name is John and I am 30 years old.
D) TypeError

Answer: C

```
result = ['Even' if i % 2 == 0 else 'Odd' for i in range(1, 5)]
print(result)
```

**149. What is the correct output?**

A) ['Odd', 'Even', 'Odd', 'Even']
B) ['Even', 'Odd', 'Even', 'Odd']
C) ['Odd', 'Odd', 'Even', 'Even']
D) SyntaxError

Answer: A

```
a = [1, 2, 3]
b = [1, 2, 3]
print(a == b, a is b)
```

**150. What is the output?**

A) True False
B) True True
C) False True
D) False False

Answer: A

```
result = []
for i in range(1, 6):
    if i % 2 == 0:
        for j in range(i, 0, -1):
            if j % 3 == 0:
                result.append((i, j))
                break
    else:
        result.append((i, -1))

print(result)
```

**151. What is the output of the code above?**

A) [(1, -1), (2, 0), (3, -1), (4, 3), (5, -1)]
B) [(1, -1), (2, 3), (3, -1), (4, 3), (5, -1)]
C) [(1, -1), (2, -1), (3, -1), (4, 3), (5, -1)]
D) [(1, -1), (2, 3), (3, -1), (4, 0), (5, -1)]
E) [(1, -1), (3, -1), (4, 3), (5, -1)]

Answer: E

```
x, y = 1, 5
while x < y:
    print(f"x={x} y={y}", end='; ')
    x += 1
    while y > x:
        y -= 1
        if y % 2 == 0:
            print(f"y={y}", end='; ')
            break
```

**152. What is the sequence printed by the code?**

A) x=1 y=5; y=4; x=2 y=4; y=3; x=3 y=3;
B) x=1 y=5; y=4; x=2 y=4; x=3 y=3;
C) x=1 y=5; y=4; x=2 y=4;
D) x=1 y=5; y=4; x=2 y=3; x=3 y=2;

Answer: C

```
data = [5, 3, 9, 1, 10]
processed = []
for d in data:
    if d % 3 == 0:
        processed.append(d * 2)
    elif d % 2 == 0:
        continue
    else:
        processed.append(d - 1)

if processed[-1] > 5:
    processed.append(processed[-1] + 5)
else:
    processed.append(processed[-1] - 5)

print(processed)
```

**153. What is the final state of the processed list?**

A) [4, 6, 0, 9]
B) [4, 6, 0, 15]
C) [4, 6, 18, 0, -5]
D) [4, 6, 18, 0, 10]

Answer: C

```
x = [0]
y = x
y.append(1)
z = x + [2]
print(x, z)
```

**154. What is printed by this code?**

A) [0, 1] [0, 1, 2]
B) [0] [0, 2]
C) [0, 1, 2] [0, 1, 2]
D) [0, 2] [0, 2]

Answer: A

```
s = "python"
t = s.upper().lower()
print(s == t, s is t)
```

**155. What is the output of the code?**

A) True True
B) True False
C) False True
D) False False

Answer: B

```
def append_to(element, to=[]):
    to.append(element)
    return to

list1 = append_to(12)
list2 = append_to(42)

print(list1)
```

**156. What does this code print?**

A) [12]
B) [42]
C) [12, 42]
D) SyntaxError

Answer: C

```
a = 256
b = 256
c = 257
d = 257

print(a is b, c is d)
```

**157. What is the output of the code?**

A) True True
B) True False
C) False True
D) False False

Answer: B

```
x = 'global'
def test():
    x = 'local'
    return x

print(test(), x)
```

**158. What is the output of this code?**

A) local local
B) global global
C) local global
D) global local

Answer: C

```
y = 5
def set_x(z):
    global y
    y = z
    return y

set_x(10)
print(y)
```

**159. What does this code print?**

A) 5
B) 10
C) Error
D) None
Answer: B

```
a = 20
def change_and_print():
    a = 30
    print(a)

change_and_print()
print(a)
```

**160. What is the output of the code?**

A) 20 20
B) 30 20
C) 30 30
D) Error
Answer: B

```
a = 20
def function():
    print(a)
    a = 30
    return a
print(function(), a)
```

**161. What is the output of the code?**

A) 30 20
B) 20 20
C) 20 30
D) Error: UnboundLocalError

Answer: D

# Short Answer Questions and Answers

**What is Python and why is it popular?**

Answer: Python is a high-level, interpreted programming language known for its readability, simplicity, and versatility. It is popular due to its wide range of applications from web development to data science, its large standard library, and the supportive community around it.

**Describe two areas where Python is extensively used and explain why it is suited for these areas.**

Answer: Python is extensively used in data science and web development. In data science, Python's simplicity and the powerful libraries like NumPy and pandas make data manipulation and analysis easy. In web development, frameworks like Django and Flask allow for the quick development of secure and maintainable websites. Python's versatility and the rich ecosystem of libraries make it suited for these areas.

**Explain the difference between == and is operators in Python.**

Answer: The == operator checks if the values of two operands are equal, while the is operator checks whether both operands refer to the same object in memory.

**Describe how to use the input() function to receive numerical input from a user and calculate the sum of two numbers.**

Answer: To receive numerical input, use the input() function to capture user input as a string and then convert the strings to integers or floats using the int() or float() function. Sum the numbers by adding the converted values. For example: num1 = int(input("Enter first number: ")), num2 = int(input("Enter second number: ")), and then sum = num1 + num2.

**Explain how you can access the last element of a list named items.**

Answer: You can access the last element of the list items by using negative indexing. For example, items[-1] will give you the last element of the list.

**Describe the difference between the sort() method and the sorted() function in Python.**

Answer: The sort() method sorts the list in place, modifying the original list. The sorted() function returns a new list that is sorted, leaving the original list unchanged.

**Explain the use of the break statement in a for loop.**

Answer: The break statement is used to exit a for loop before completing all iterations. It immediately stops the loop's execution and exits the loop block.

**Describe how you would use a nested for loop to print a 3x3 square of asterisks (*).**

Answer: A nested for loop involves using one loop inside another. To print a 3x3 square of asterisks, you can use an outer for loop to iterate through each row and an inner for loop to print asterisks for each column in that row. Each iteration of the inner loop prints an asterisk, and after the inner loop completes, the outer loop moves to the next line.

**Explain the purpose of the else clause in a while loop.**

Answer: The else clause in a while loop is executed when the loop condition becomes False. It is a way to run a block of code once after the loop has finished iterating, provided the loop was not terminated by a break statement.

**Describe how to implement a while loop that asks the user for input and breaks the loop if the user enters 'quit'.**

Answer: Initialize the loop with a condition that remains True, then within the loop, prompt the user for input using the input() function. Check if the input is 'quit', and if so, use the break statement to exit the loop. Otherwise, continue with the loop's intended functionality.

**Explain the difference between parameters and arguments in the context of Python functions.**

Answer: Parameters are the names listed in the function's definition, whereas arguments are the real values passed to the function when it is called.

**Describe how you would use a lambda function to add two numbers in Python.**

Answer: A lambda function to add two numbers can be defined as add = lambda x, y: x + y, and it can be called with two arguments like add(2, 3) to get the result 5.

**Explain the concept of a class in Python.**

Answer: A class in Python is a blueprint for creating objects. It defines a set of attributes and methods that the instantiated objects will possess. Classes encapsulate data for the objects created from them.

**What is inheritance in Python and how is it beneficial?**

Answer: Inheritance allows a class (subclass) to inherit attributes and methods from another class (superclass), promoting code reuse and the creation of hierarchical class structures. It simplifies maintenance and enhances readability.

**Describe encapsulation with an example in Python.**

Answer: Encapsulation involves hiding the internal representation of an object from its outside view. For example, by defining variables as private (using __ prefix), a class can hide its internal state and require that all interaction be performed through an object's methods, protecting the object from inconsistent state changes.

**How does polymorphism work in Python? Give an example.**

Answer: Polymorphism allows methods to do different things based on the object it is acting upon. For example, the same method name might behave differently on different classes (e.g., draw() method for shapes where each shape class (Circle, Square) has a different implementation of draw()).

**What is the significance of the self parameter in class methods?**

Answer: The self parameter refers to the instance calling the method, allowing access to the attributes and methods of the class. It's used to access variables that belongs to the class.

**Explain the difference between a module and a package in Python.**

Answer: A module is a single file (or files) that is imported under one import and used. A package is a collection of modules in directories that give a package hierarchy.

**Describe how to use NumPy to convert a list of temperatures from Celsius to Fahrenheit.**

Answer: First, create a NumPy array from the list of Celsius temperatures. Then, use the formula F = C * 9/5 + 32 on the array to convert each Celsius temperature to Fahrenheit, leveraging NumPy's ability to perform element-wise operations on arrays.

**Explain how to safely open and read a file in Python.**

Answer: Use the with statement to ensure that the file is properly closed after its suite finishes, even if an exception is raised. For reading, open the file in 'r' mode with open('filename.txt', 'r') as part of the with statement and then call .read() or similar methods to read its content.

**Describe how to write content to a new file and ensure that the file does not exist beforehand.**

Answer: Use the open() function with the 'x' mode, which is exclusively for creating a new file. If the file already exists, the operation will fail. Example: with open('newfile.txt', 'x') as f: f.write('Hello, world!'). This approach avoids overwriting an existing file.

**Explain polymorphism in Python and provide an example scenario where it might be used.**

Answer: Polymorphism allows objects to adopt multiple forms. It's used in method overriding where a subclass redefines a method of its superclass, enabling different behaviors while sharing the same method name. For example, a draw() method can be defined in a Shape class and overridden in Circle and Rectangle subclasses to draw the specific shapes.

**Describe how encapsulation is implemented in Python and its benefits.**

Answer: Encapsulation is implemented using public, protected (_ prefix), and private (__ prefix) access modifiers. It hides the internal state of an object from outside interference and misuse, leading to more modular and secure code. For example, a BankAccount class can encapsulate the balance as a private attribute to prevent unauthorized access and modification.

**What is operator overloading and how can it be applied in Python?**

Answer: Operator overloading allows custom implementation of the behavior of Python's built-in operators for objects of user-defined classes. For instance, __add__ can be overridden to allow adding two objects of a Vector class, resulting in a new Vector object that represents the vector sum of the original vectors.

**How does method overloading differ from operator overloading in Python?**

Answer: Method overloading, having multiple methods with the same name but different parameters, is not directly supported in Python. Instead, dynamic argument handling techniques (*args, **kwargs) are used. Operator overloading, however, allows redefining how built-in operators work with user-defined objects, like changing the behavior of + or * for custom classes.

**Discuss the role of magic methods in Python with examples.**

Answer: Magic methods, identified by double underscores (e.g., __init__, __str__), enable operator overloading and define special behaviors for user-defined classes. For example, __add__ can be used to define addition for class instances, and __str__ can be implemented to return a human-readable string representation of an object.