100 Python Questions with Answers

Functions (Questions 1-25)

1. Write a function that returns the sum of two numbers.

```python
def add_numbers(a, b):
    return a + b

# Usage: add_numbers(5, 3) returns 8
```

2. Create a function to check if a number is even or odd.

```python
def check_even_odd(num):
    if num % 2 == 0:
        return "Even"
    else:
        return "Odd"

# Usage: check_even_odd(4) returns "Even"
```

3. Write a function that takes a list and returns its length.

```python
def get_list_length(lst):
    return len(lst)

# Usage: get_list_length([1, 2, 3, 4]) returns 4
```

4. Create a function that returns the maximum of three numbers.

```python
def max_of_three(a, b, c):
    return max(a, b, c)

# Usage: max_of_three(10, 5, 8) returns 10
```

5. Write a function to calculate the factorial of a number.

python

```python
def factorial(n):
    if n == 0 or n == 1:
        return 1
    result = 1
    for i in range(2, n + 1):
        result *= i
    return result

# Usage: factorial(5) returns 120
```

6. Create a function that reverses a string.

python

```python
def reverse_string(s):
    return s[::-1]

# Usage: reverse_string("hello") returns "olleh"
```

7. Write a function to check if a string is a palindrome.

python

```python
def is_palindrome(s):
    s = s.lower().replace(" ", "")
    return s == s[::-1]

# Usage: is_palindrome("racecar") returns True
```

8. Create a function that counts vowels in a string.

python

```python
def count_vowels(s):
    vowels = "aeiouAEIOU"
    count = 0
    for char in s:
        if char in vowels:
            count += 1
    return count

# Usage: count_vowels("hello") returns 2
```

9. Write a function that returns the nth Fibonacci number.

python

```python
def fibonacci(n):
    if n <= 1:
        return n
    a, b = 0, 1
    for i in range(2, n + 1):
        a, b = b, a + b
    return b

# Usage: fibonacci(6) returns 8
```

10. Create a function that converts Celsius to Fahrenheit.

python

```python
def celsius_to_fahrenheit(celsius):
    return (celsius * 9/5) + 32

# Usage: celsius_to_fahrenheit(25) returns 77.0
```

11. Write a function that finds the GCD of two numbers.

python

```python
def gcd(a, b):
    while b:
        a, b = b, a % b
    return a

# Usage: gcd(48, 18) returns 6
```

12. Create a function that checks if a number is prime.

python

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

# Usage: is_prime(17) returns True
```

13. Write a function that returns the sum of digits of a number.

python

```python
def sum_of_digits(n):
    return sum(int(digit) for digit in str(abs(n)))

# Usage: sum_of_digits(123) returns 6
```

14. Create a function that generates a list of squares.

python

```python
def generate_squares(n):
    return [i**2 for i in range(1, n + 1)]

# Usage: generate_squares(5) returns [1, 4, 9, 16, 25]
```

15. Write a function with default parameters.

```python
def greet(name, greeting="Hello", punctuation="!"):
    return f"{greeting}, {name}{punctuation}"

# Usage: greet("Alice") returns "Hello, Alice!"
```

16. Create a function that accepts variable arguments.

```python
def sum_all(*args):
    return sum(args)

# Usage: sum_all(1, 2, 3, 4) returns 10
```

17. Write a function that accepts keyword arguments.

```python
def create_profile(**kwargs):
    return kwargs

# Usage: create_profile(name="John", age=25) returns {'name': 'John', 'age': 25}
```

18. Create a function that returns multiple values.

```python
def get_name_age():
    return "Alice", 30

# Usage: name, age = get_name_age()
```

19. Write a function that modifies a list in place.

python

```python
def double_list_values(lst):
    for i in range(len(lst)):
        lst[i] *= 2
    return lst

# Usage: double_list_values([1, 2, 3]) returns [2, 4, 6]
```

20. Create a nested function.

python

```python
def outer_function(x):
    def inner_function(y):
        return x + y
    return inner_function

# Usage: add_five = outer_function(5); add_five(3) returns 8
```

21. Write a function that returns a lambda function.

python

```python
def create_multiplier(n):
    return lambda x: x * n

# Usage: double = create_multiplier(2); double(5) returns 10
```

22. Create a function to find the second largest number in a list.

python

```python
def second_largest(lst):
    unique_nums = list(set(lst))
    unique_nums.sort()
    return unique_nums[-2] if len(unique_nums) >= 2 else None

# Usage: second_largest([1, 3, 2, 5, 4]) returns 4
```

23. Write a function that removes duplicates from a list.

python

```python
def remove_duplicates(lst):
    return list(set(lst))

# Usage: remove_duplicates([1, 2, 2, 3, 3, 4]) returns [1, 2, 3, 4]
```

24. Create a function that counts character frequency.

python

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

# Usage: char_frequency("hello") returns {'h': 1, 'e': 1, 'l': 2, 'o': 1}
```

25. Write a function that flattens a nested list.

python

```python
def flatten_list(nested_list):
    result = []
    for item in nested_list:
        if isinstance(item, list):
            result.extend(flatten_list(item))
        else:
            result.append(item)
    return result

# Usage: flatten_list([1, [2, 3], [4, [5, 6]]]) returns [1, 2, 3, 4, 5, 6]
```

For Loops (Questions 26-40)

26. Write a for loop to print numbers 1 to 10.

python

```python
for i in range(1, 11):
    print(i)
```

27. Create a for loop to sum numbers in a list.

```python
numbers = [1, 2, 3, 4, 5]
total = 0
for num in numbers:
    total += num
print(total)  # Output: 15
```

28. Write a for loop to print even numbers from 2 to 20.

```python
for i in range(2, 21, 2):
    print(i)
```

29. Create a for loop to iterate through a string.

```python
word = "Python"
for char in word:
    print(char)
```

30. Write a for loop with enumerate to get index and value.

```python
fruits = ["apple", "banana", "orange"]
for index, fruit in enumerate(fruits):
    print(f"{index}: {fruit}")
```

31. Create a nested for loop to print a multiplication table.

```python
for i in range(1, 6):
    for j in range(1, 6):
        print(f"{i} x {j} = {i*j}")
```

32. Write a for loop to create a list of squares.

python

```python
squares = []
for i in range(1, 6):
    squares.append(i**2)
print(squares)  # [1, 4, 9, 16, 25]
```

33. Create a for loop to count occurrences of each character.

python

```python
text = "hello world"
char_count = {}
for char in text:
    if char in char_count:
        char_count[char] += 1
    else:
        char_count[char] = 1
print(char_count)
```

34. Write a for loop to reverse a list.

python

```python
original = [1, 2, 3, 4, 5]
reversed_list = []
for i in range(len(original) - 1, -1, -1):
    reversed_list.append(original[i])
print(reversed_list)  # [5, 4, 3, 2, 1]
```

35. Create a for loop to find the maximum value in a list.

python

```python
numbers = [3, 7, 2, 9, 1]
max_val = numbers[0]
for num in numbers[1:]:
    if num > max_val:
        max_val = num
print(max_val)  # 9
```

36. Write a for loop with break statement.

python

```
for i in range(1, 11):
    if i == 6:
        break
    print(i)  # Prints 1 to 5
```

37. Create a for loop with continue statement.

python

```
for i in range(1, 11):
    if i % 2 == 0:
        continue
    print(i)  # Prints odd numbers: 1, 3, 5, 7, 9
```

38. Write a for loop to iterate through a dictionary.

python

```
student_grades = {"Alice": 85, "Bob": 92, "Charlie": 78}
for name, grade in student_grades.items():
    print(f"{name}: {grade}")
```

39. Create a for loop to generate Fibonacci sequence.

python

```
n = 10
fib_sequence = []
a, b = 0, 1
for i in range(n):
    fib_sequence.append(a)
    a, b = b, a + b
print(fib_sequence)
```

40. Write a for loop to check if all elements in a list are positive.

python

```python
numbers = [1, 2, 3, 4, 5]
all_positive = True
for num in numbers:
    if num <= 0:
        all_positive = False
        break
print(all_positive)  # True
```

While Loops (Questions 41-55)

41. Write a while loop to print numbers 1 to 5.

python

```python
i = 1
while i <= 5:
    print(i)
    i += 1
```

42. Create a while loop to find the sum of digits.

python

```python
num = 12345
sum_digits = 0
while num > 0:
    sum_digits += num % 10
    num //= 10
print(sum_digits)  # 15
```

43. Write a while loop to reverse a number.

python

```python
num = 12345
reversed_num = 0
while num > 0:
    reversed_num = reversed_num * 10 + num % 10
    num //= 10
print(reversed_num)  # 54321
```

44. Create a while loop to find factorial.

```python
n = 5
factorial = 1
i = 1
while i <= n:
    factorial *= i
    i += 1
print(factorial)  # 120
```

45. Write a while loop with user input validation.

```python
# Simulated user input validation
user_input = "invalid"
valid_inputs = ["yes", "no"]
while user_input not in valid_inputs:
    print("Please enter 'yes' or 'no'")
    # user_input = input()  # In real scenario
    user_input = "yes"  # Simulated valid input
```

46. Create a while loop to generate Fibonacci sequence.

```python
n = 8  # Number of terms
a, b = 0, 1
count = 0
while count < n:
    print(a)
    a, b = b, a + b
    count += 1
```

47. Write a while loop to find GCD.

```python
a, b = 48, 18
while b:
    a, b = b, a % b
print(a)  # 6
```

48. Create a while loop to check if number is prime.

```
python

num = 17
i = 2
is_prime = True
while i * i <= num:
    if num % i == 0:
        is_prime = False
        break
    i += 1
print(is_prime)  # True
```

49. Write a while loop to count digits in a number.

```
python

num = 12345
count = 0
while num > 0:
    count += 1
    num //= 10
print(count)  # 5
```

50. Create a while loop with break statement.

```
python

i = 1
while True:
    if i > 5:
        break
    print(i)
    i += 1
```

51. Write a while loop with continue statement.

python

```python
i = 0
while i < 10:
    i += 1
    if i % 2 == 0:
        continue
    print(i)  # Prints odd numbers
```

52. Create a while loop to find power of a number.

python

```python
base = 2
exponent = 5
result = 1
while exponent > 0:
    result *= base
    exponent -= 1
print(result)  # 32
```

53. Write a while loop to remove specific element from list.

python

```python
numbers = [1, 2, 3, 2, 4, 2, 5]
target = 2
i = 0
while i < len(numbers):
    if numbers[i] == target:
        numbers.pop(i)
    else:
        i += 1
print(numbers)  # [1, 3, 4, 5]
```

54. Create a while loop for binary search.

python

```
def binary_search(arr, target):
    left, right = 0, len(arr) - 1
    while left <= right:
        mid = (left + right) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] < target:
            left = mid + 1
        else:
            right = mid - 1
    return -1

# Usage: binary_search([1, 2, 3, 4, 5], 3) returns 2
```

55. Write a while loop to calculate square root.

python

```
num = 16
guess = num / 2
while abs(guess * guess - num) > 0.001:
    guess = (guess + num / guess) / 2
print(f"Square root of {num} is approximately {guess}")
```

If Statements (Questions 56-65)

56. Write an if statement to check if a number is positive.

python

```
num = 5
if num > 0:
    print("Number is positive")
```

57. Create an if statement to check voting eligibility.

python

```
age = 20
if age >= 18:
    print("Eligible to vote")
```

58. Write an if statement to check if a string is empty.

```python
text = ""
if not text:
    print("String is empty")
```

59. Create an if statement to check if a number is divisible by 3.

```python
num = 9
if num % 3 == 0:
    print("Number is divisible by 3")
```

60. Write an if statement with multiple conditions using 'and'.

```python
score = 85
attendance = 90
if score >= 80 and attendance >= 85:
    print("Student passed with good performance")
```

61. Create an if statement with multiple conditions using 'or'.

```python
weather = "sunny"
if weather == "sunny" or weather == "partly cloudy":
    print("Good day for outdoor activities")
```

62. Write an if statement using 'in' operator.

```python
fruit = "apple"
fruits_list = ["apple", "banana", "orange"]
if fruit in fruits_list:
    print("Fruit is available")
```

63. Create an if statement to check list length.

```python
my_list = [1, 2, 3, 4, 5]
if len(my_list) > 3:
    print("List has more than 3 elements")
```

64. Write an if statement with string comparison.

```python
password = "secret123"
if password == "secret123":
    print("Access granted")
```

65. Create an if statement using not operator.

```python
is_weekend = False
if not is_weekend:
    print("It's a weekday")
```

If-Else Statements (Questions 66-75)

66. Write an if-else to check if number is even or odd.

```python
num = 7
if num % 2 == 0:
    print("Even")
else:
    print("Odd")
```

67. Create an if-else for age category.

```python
age = 25
if age < 18:
    print("Minor")
else:
    print("Adult")
```

68. Write an if-else to find maximum of two numbers.

```python
a, b = 10, 15
if a > b:
    max_num = a
else:
    max_num = b
print(f"Maximum is {max_num}")
```

69. Create an if-else for grade evaluation.

```python
score = 75
if score >= 70:
    grade = "Pass"
else:
    grade = "Fail"
print(f"Grade: {grade}")
```

70. Write an if-else for login validation.

```python
username = "admin"
password = "123456"
if username == "admin" and password == "123456":
    print("Login successful")
else:
    print("Invalid credentials")
```

71. Create an if-else for temperature check.

```python
temperature = 25
if temperature > 30:
    print("It's hot outside")
else:
    print("Temperature is comfortable")
```

72. Write an if-else for list emptiness check.

python

```python
my_list = []
if my_list:
    print("List has elements")
else:
    print("List is empty")
```

73. Create an if-else for string length validation.

python

```python
user_input = "hello"
if len(user_input) >= 5:
    print("Input is valid")
else:
    print("Input too short")
```

74. Write an if-else for number sign determination.

python

```python
num = -5
if num >= 0:
    print("Number is non-negative")
else:
    print("Number is negative")
```

75. Create an if-else for membership check.

python

```python
item = "apple"
shopping_list = ["milk", "bread", "eggs"]
if item in shopping_list:
    print("Item is in the list")
else:
    print("Item not found in the list")
```

If-Elif-Else Statements (Questions 76-85)

76. Write an if-elif-else for grade calculation.

python

```python
score = 85
if score >= 90:
    grade = "A"
elif score >= 80:
    grade = "B"
elif score >= 70:
    grade = "C"
elif score >= 60:
    grade = "D"
else:
    grade = "F"
print(f"Grade: {grade}")
```

77. Create an if-elif-else for age categories.

python

```python
age = 25
if age < 13:
    category = "Child"
elif age < 20:
    category = "Teenager"
elif age < 60:
    category = "Adult"
else:
    category = "Senior"
print(f"Category: {category}")
```

78. Write an if-elif-else for BMI calculation.

```python
bmi = 22.5
if bmi < 18.5:
    status = "Underweight"
elif bmi < 25:
    status = "Normal weight"
elif bmi < 30:
    status = "Overweight"
else:
    status = "Obese"
print(f"BMI Status: {status}")
```

79. Create an if-elif-else for season determination.

```python
month = 7
if month in [12, 1, 2]:
    season = "Winter"
elif month in [3, 4, 5]:
    season = "Spring"
elif month in [6, 7, 8]:
    season = "Summer"
else:
    season = "Fall"
print(f"Season: {season}")
```

80. Write an if-elif-else for traffic light.

```python
light = "yellow"
if light == "red":
    action = "Stop"
elif light == "yellow":
    action = "Caution"
elif light == "green":
    action = "Go"
else:
    action = "Invalid light"
print(action)
```

81. Create an if-elif-else for discount calculation.

python

```python
amount = 150
if amount >= 200:
    discount = 0.20
elif amount >= 100:
    discount = 0.15
elif amount >= 50:
    discount = 0.10
else:
    discount = 0
final_amount = amount * (1 - discount)
print(f"Final amount: ${final_amount}")
```

82. Write an if-elif-else for number comparison.

python

```python
a, b = 10, 15
if a > b:
    result = "a is greater"
elif a < b:
    result = "b is greater"
else:
    result = "a and b are equal"
print(result)
```

83. Create an if-elif-else for day of week.

python

```python
day = 3
if day == 1:
    day_name = "Monday"
elif day == 2:
    day_name = "Tuesday"
elif day == 3:
    day_name = "Wednesday"
elif day == 4:
    day_name = "Thursday"
elif day == 5:
    day_name = "Friday"
elif day == 6:
    day_name = "Saturday"
elif day == 7:
    day_name = "Sunday"
else:
    day_name = "Invalid day"
print(day_name)
```

84. Write an if-elif-else for calculator operations.

python

```python
num1, num2 = 10, 5
operation = "+"
if operation == "+":
    result = num1 + num2
elif operation == "-":
    result = num1 - num2
elif operation == "*":
    result = num1 * num2
elif operation == "/":
    result = num1 / num2 if num2 != 0 else "Cannot divide by zero"
else:
    result = "Invalid operation"
print(f"Result: {result}")
```

85. Create an if-elif-else for password strength.

python

```python
password = "MyPass123!"
length = len(password)
if length < 6:
    strength = "Weak"
elif length < 10:
    strength = "Medium"
elif length < 15:
    strength = "Strong"
else:
    strength = "Very Strong"
print(f"Password strength: {strength}")
```

Classes and Objects (Questions 86-100)

86. Create a simple class with attributes.

python

```python
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

# Usage
person1 = Person("Alice", 25)
print(person1.name)  # Alice
print(person1.age)   # 25
```

87. Write a class with methods.

```python
class Calculator:
    def add(self, a, b):
        return a + b

    def subtract(self, a, b):
        return a - b

# Usage
calc = Calculator()
print(calc.add(5, 3))      # 8
print(calc.subtract(5, 3)) # 2
```

88. Create a class with class variables.

```python
class Student:
    school_name = "ABC High School"

    def __init__(self, name, grade):
        self.name = name
        self.grade = grade

# Usage
student1 = Student("John", "A")
print(Student.school_name)  # ABC High School
print(student1.name)        # John
```

89. Write a class with private attributes.

python

```python
class BankAccount:
    def __init__(self, balance):
        self.__balance = balance  # Private attribute

    def get_balance(self):
        return self.__balance

    def deposit(self, amount):
        self.__balance += amount

# Usage
account = BankAccount(1000)
print(account.get_balance())  # 1000
account.deposit(500)
print(account.get_balance())  # 1500
```

90. Create a class with static methods.

python

```python
class MathUtils:
    @staticmethod
    def is_even(num):
        return num % 2 == 0

    @staticmethod
    def square(num):
        return num ** 2

# Usage
print(MathUtils.is_even(4))  # True
print(MathUtils.square(5))   # 25
```

91. Write a class with class methods.

```python
class Person:
    population = 0

    def __init__(self, name):
        self.name = name
        Person.population += 1

    @classmethod
    def get_population(cls):
        return cls.population

# Usage
p1 = Person("Alice")
p2 = Person("Bob")
print(Person.get_population())  # 2
```

92. Create a class with inheritance.

```python
class Animal:
    def __init__(self, name):
        self.name = name

    def speak(self):
        pass

class Dog(Animal):
    def speak(self):
        return f"{self.name} says Woof!"

class Cat(Animal):
    def speak(self):
        return f"{self.name} says Meow!"

# Usage
dog = Dog("Buddy")
cat = Cat("Whiskers")
print(dog.speak())  # Buddy says Woof!
print(cat.speak())  # Whiskers says Meow!
```

93. Write a class with property decorators.

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

    @property
    def radius(self):
        return self._radius

    @radius.setter
    def radius(self, value):
        if value <= 0:
            raise ValueError("Radius must be positive")
        self._radius = value

    @property
    def area(self):
        return 3.14159 * self._radius ** 2

# Usage
circle = Circle(5)
print(circle.area)      # 78.53975
circle.radius = 3
print(circle.area)      # 28.27431
```

94. Create a class with multiple inheritance.

```python
class Flyable:
    def fly(self):
        return "I can fly!"

class Swimmable:
    def swim(self):
        return "I can swim!"

class Duck(Flyable, Swimmable):
    def __init__(self, name):
        self.name = name

    def quack(self):
        return f"{self.name} says Quack!"

# Usage
duck = Duck("Donald")
print(duck.fly())    # I can fly!
print(duck.swim())   # I can swim!
print(duck.quack())  # Donald says Quack!
```

95. Write a class with operator overloading.

```python
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __add__(self, other):
        return Point(self.x + other.x, self.y + other.y)

    def __str__(self):
        return f"Point({self.x}, {self.y})"

# Usage
p1 = Point(1, 2)
p2 = Point(3, 4)
p3 = p1 + p2
print(p3)  # Point(4, 6)
```

96. Create a class with context manager methods.

python

```python
class FileManager:
    def __init__(self, filename, mode):
        self.filename = filename
        self.mode = mode

    def __enter__(self):
        self.file = open(self.filename, self.mode)
        return self.file

    def __exit__(self, exc_type, exc_value, traceback):
        self.file.close()

# Usage
# with FileManager('test.txt', 'w') as f:
#     f.write('Hello, World!')
```

97. Write a class with iterator methods.

python

```python
class NumberSequence:
    def __init__(self, start, end):
        self.start = start
        self.end = end

    def __iter__(self):
        return self

    def __next__(self):
        if self.start >= self.end:
            raise StopIteration
        current = self.start
        self.start += 1
        return current

# Usage
seq = NumberSequence(1, 5)
for num in seq:
    print(num)  # Prints 1, 2, 3, 4
```

98. Create a class representing a car with methods.

python

```python
class Car:
    def __init__(self, make, model, year):
        self.make = make
        self.model = model
        self.year = year
        self.odometer = 0

    def describe_car(self):
        return f"{self.year} {self.make} {self.model}"

    def read_odometer(self):
        return f"This car has {self.odometer} miles on it."

    def update_odometer(self, mileage):
        if mileage >= self.odometer:
            self.odometer = mileage
        else:
            print("You can't roll back an odometer!")

# Usage
my_car = Car("Toyota", "Camry", 2020)
print(my_car.describe_car())     # 2020 Toyota Camry
my_car.update_odometer(23500)
print(my_car.read_odometer())    # This car has 23500 miles on it.
```

99. Write a class for a simple bank account system.

python

```python
class BankAccount:
    def __init__(self, account_holder, initial_balance=0):
        self.account_holder = account_holder
        self.balance = initial_balance
        self.transaction_history = []

    def deposit(self, amount):
        if amount > 0:
            self.balance
```