**1/35. focusing on basic concepts, syntax, and common functions**

1. List Indexing

How can you access the third item in a list named `my\_list`?

- `if(my\_list[3]):`

- `if(my\_list[2]):`

- `if(3 in my\_list):`

- `if('third' in my\_list):`

2. Variable Assignment

Which of the following is the correct way to assign the value 10 to a variable named `my\_var`?

- `if(my\_var == 10):`

- `my\_var = 10`

- `if(my\_var = 10):`

- `my\_var == 10`

3. String Concatenation

How do you concatenate two strings `str1` and `str2` to form a new string `str3`?

- str3 = str1 + str2`

- `if(str1 . str2):`

- `str3 = str1, str2`

- `str3 = concat(str1, str2)`

4. Function Definition

Which of the following defines a function named `my\_func` that takes no arguments?

- `def my\_func():`

- `function my\_func():`

- `def my\_func(args):`

- `function my\_func(args):`

5. For Loop Syntax

How do you write a `for` loop that prints each character in the string `hello`?

- `for char in hello: print(char)`

- `for hello in char: print(char)`

- `for char in 'hello': print(char)`

- `for 'hello' in char: print(char)`

6. Dictionary Access

How do you access the value associated with the key `'age'` in a dictionary named `person`?

- `if(person['age']):`

- `if(age in person):`

- `if('age' = person):`

- `if(person = 'age'):`

7. Conditional Statement

Which of the following will check if the variable `a` is greater than the variable `b`?

- `if(a > b):`

- `if(a < b):`

- `if(a = b):`

- `if(a == b):`

8. List Comprehension

How can you create a list of squares of numbers from 0 to 9?

- `squares = [x\*\*2 for x in range(10)]`

- `squares = [x\*x for x in range(10)]`

- `squares = [for x in range(10): x\*\*2]`

- `squares = [x\*\*2 for in range(10)]`

9. Module Importing

How do you import the `math` module in a Python script?

- `import math`

- `include math`

- `using math`

- `#import math`

10. File Opening

Which of the following is the correct way to open a file `data.txt` for reading as a text file?

- `file = open('data.txt', 'r')`

- `open(file='data.txt', 'read')`

- `file.open('data.txt', 'r')`

- `file('data.txt', 'read')`

**2/35. focus on understanding code output**

1. Print Function with End Parameter

What will the following code print?

print("Hello", end=' ')

print("World!")

- `HelloWorld!`

- `Hello World!`

- `Hello`

- `World!`

2. String Formatting with f-string

What will the following code print?

name = "Alice"

age = 30

print(f"{name} is {age} years old.")

- `name is age years old.`

- `Alice is 30 years old.`

- `{name} is {age} years old.`

- `Alice is years old.`

3. String Multiplication

What will the following code output?

print("a" \* 3)

- `aaa`

- `a\*a\*a`

- `3a`

- `a3`

4. List Slicing

What will be printed by the following code?

my\_list = [10, 20, 30, 40, 50]

print(my\_list[1:4]) #ATTENTION 4 NON COMPRIS

- `[20, 30, 40]`

- `[10, 20, 30]`

- `[20, 30, 40, 50]`

- `[10, 20, 30, 40]`

5. Dictionary Default Value

What will be printed by the following code?

my\_dict = {'a': 1, 'b': 2, 'c': 3}

print(my\_dict.get('d', 'Not Found'))

- `None`

- `Not Found`

- `0`

- `d`

6. Tuple Unpacking

What will the following code print?

a, b, c = (1, 2, 3)

print(b)

- `1`

- `2`

- `3`

- `(2)`

7. Boolean Logic

What will the following code print?

print(not((True or False) and (False or True)))

- `True`

- `False`

- `True True`

- `False False`

8. List Comprehension with If Statement

What does this code print?

numbers = [1, 2, 3, 4, 5]

evens = [num for num in numbers if num % 2 == 1]

print(evens)

- `[1, 3, 5]`

- `[2, 4]`

- `[1, 2, 3, 4, 5]`

- `[]`

9. Using the Join Method

What will be printed by the following code?

my\_strings = ['Python', 'is', 'awesome']

print(' '.join(my\_strings))

- `Pythonisawesome`

- `Python is awesome`

- `['Python', 'is', 'awesome']`

- `Python-is-awesome`

10. Function Call

What will the following code print?

def greet(name):

return "Hello, " + name

print(greet("Bob"))

- `Hello, Bob`

- `Hello,`

- `Bob`

- `"Hello, " + name`

**3/35. understanding of Python code and its output**

1. Range Function

What will be printed by the following code?

print(len(range(5, 15, 3)))

- `5`

- `4`

- `3`

- `10`

2. Dictionary Key Access

What will be printed by the following code?

my\_dict = {'name': 'John', 'age': 25}

print(my\_dict['name'])

- `John`

- `25`

- `name`

- `KeyError`

3. Set Operations

What will be printed by the following code?

set1 = {1, 2, 3}

set2 = {3, 4, 5}

print(len(set1.union(set2)))

- `3`

- `5`

- `6`

- `4`

4. String Methods

What will be printed by the following code?

phrase = "Hello, World!"

print(phrase.replace("World", "Python"))

- `Hello, World!`

- `Hello, Python!`

- `Hello, !`

- `Hello, World!Python`

5. Exception Handling

What will be the output of the following code snippet?

try:

x = 1 / 0

except ZeroDivisionError:

print("Cannot divide by zero")

finally:

print("End of the program")

- `Cannot divide by zero`

- `End of the program`

- `Cannot divide by zero\nEnd of the program`

- `ZeroDivisionError`

**4/35. focus on the properties of data structures after certain operations are performed**

1. List Modification

Given the list `numbers = [1, 2, 3, 4, 5]`, which of the following statements are true after executing `numbers[1], numbers[3] = numbers[3], numbers[1]`?

- `numbers` is now `[1, 4, 3, 2, 5]`

- `numbers` has the same length

- `numbers` contains a duplicate value

- The elements in `numbers` are still in ascending order

2. Tuple to List Conversion

Starting with `tup = (5, 6, 7)`, which of the following statements are true after executing `lst = list(tup)` then `lst.append(8)`?

- `lst` becomes `[5, 6, 7, 8]`

- `lst` is longer than `tup`

- `tup` remains unchanged

- `lst` contains only even numbers

3. Dictionary Keys and Values

If we have `d = {'a': 1, 'b': 2, 'c': 3}` and we execute `d['b'] = 4`, which statements are true?

- The value associated with key 'b' is now 4

- `d` has the same number of keys

- The keys of `d` are in alphabetical order

- The values of `d` are in ascending order

4. String Immutability

After executing `s = "abc"` and then trying to change it with `s[1] = 'd'`, what is true?

- `s` remains `"abc"`

- An error occurs because strings are immutable

- `s` becomes `"adc"`

- The length of `s` has changed

5. Set Symmetric Difference

Given `set1 = {1, 2, 3}` and `set2 = {2, 3, 4}`, after executing `set3 = set1 ^ set2`, what are the properties of `set3`?

- `set3` is `{1, 4}`

- `set3` has 2 elements

- `set3` is a subset of `set1`

- `set3` contains only elements that were in both `set1` and `set2`

**5/35. questions exploring error handling and code understanding:**

1. Immutable String Modification

What will happen if we run the following code snippet?

my\_string = "Hello, World!"

my\_string[7] = 'Python!'

print(my\_string)

- the output will be 'Hello, Python!'

- the code is incorrect, we get an error message

- the result will be: 'Hello, World!'

- the result will be: 'Hello, Python!'

2. List Index Out of Range

What happens when we run the code below?

numbers = [1, 2, 3]

numbers[5] = 10

print(numbers)

- the output will be: `[1, 2, 3, 10]`

- the code is incorrect, we get an error message

- the result will be: `[1, 2, 3, None, None, 10]`

- the result will be: `[1, 2, 3, 10, 10]`

3. Appending to a Tuple

Consider the following code snippet:

my\_tuple = (1, 2, 3)

my\_tuple.append(4)

print(my\_tuple)

- the output will be: `(1, 2, 3, 4)`

- the code is incorrect, we get an error message

- the result will be: `(1, 2, 3)`

- the result will be: `(1, 2, 3, (4,))`

4. Key Error in Dictionary Access

What happens when we execute this code?

my\_dict = {'a': 1, 'b': 2}

print(my\_dict['c'])

- the output will be: `None`

- the code is incorrect, we get a KeyError message

- the result will be: `1`

- the result will be: `{'a': 1, 'b': 2}`

5. Function Call with Incorrect Parameters

What is the result of running the following code?

def add(a, b):

return a + b

result = add(1)

print(result)

- the output will be: `1`

- the code is incorrect, we get a TypeError message

- the result will be: `None`

- the result will be: `a + b`

**6/35. creation and manipulation of random numbers in Python**

1. Random Float Generation

Which of the following will generate a random float number between 5.5 and 25.5?

- `random.uniform(5.5, 25.5)`

- `random.randrange(5.5, 25.5)`

- `random.randint(5.5, 25.5)`

- `random.random(5.5, 25.5)`

2. Random Choice from a List

If we have a list `colors = ['red', 'blue', 'green', 'yellow']`, how do we randomly pick one element from this list?

- `random.choice(colors)`

- `random.pick(colors)`

- `random.randint(0, len(colors)-1)`

- `random.select(colors)`

3. Random Elements without Replacement

How can you randomly select 3 unique elements from the list `[1, 2, 3, 4, 5, 6]` such that no element is repeated?

- `random.sample([1, 2, 3, 4, 5, 6], 3)`

- `random.choices([1, 2, 3, 4, 5, 6], k=3)`

- `random.randint(1, 6, 3)`

- `random.select([1, 2, 3, 4, 5, 6], 3)`

4. Shuffling a List

What is the correct way to shuffle the elements of the list `my\_list = [2, 11, 45, 23]`?

- `random.shuffle(my\_list)`

- `random.mix(my\_list)`

- `random.arrange(my\_list)`

- `my\_list.shuffle()`

5. Random Seed Setting

How do we ensure that the random numbers generated are reproducible in the subsequent runs?

- `random.seed(10)`

- `random.setstate(10)`

- `random.save(10)`

- `random.start(10)`

**7/35. file modes and operations in Python**

1. File Writing and Reading

If a file named `data.txt` contains the text "Hello World", what will be the content of the file after executing the following code?

with open('data.txt', 'w') as file:

file.write("Goodbye World")

with open('data.txt', 'r') as file:

print(file.read())

- `Hello World`

- `Goodbye World`

- `Hello WorldGoodbye World`

- The file will be empty

2. Appending to a File

What will be the content of `data.txt` after running the following code if the original content was "Python"?

with open('data.txt', 'a') as file:

file.write(" is fun!")

- `Python`

- ` is fun!`

- `Python is fun!`

- The file will be empty

3. Reading a Non-Existent File

What happens if we try to open a non-existent file in read mode?

with open('nonexistent.txt', 'r') as file:

print(file.read())

- The file is created.

- The content "None" is printed.

- An error is raised.

- The content "nonexistent.txt" is printed.

4. Exclusive Creation File Mode

What happens when we open a file with mode 'x' that already exists?

# Assuming 'existing.txt' is already present

with open('existing.txt', 'x') as file:

file.write("Trying to write.")

- The file's content is replaced with "Trying to write."

- A new file `existing.txt` is created and the old one is deleted.

- An error is raised.

- The content is appended to the file.

5. Binary File Writing

If you need to write bytes to a file, which mode should you use?

- 'wb' - write binary

- 'w' - write

- 'r+' - read and write

- 'ab' - append binary

**8/35. data manipulation with pandas**

1. Selecting Rows by Condition

How do you print rows from a DataFrame `df` where the column 'age' is greater than 30?

- `print(df[df['age'] > 30])`

- `print(df('age' > 30))`

- `print(df[['age' > 30]])`

- `print(df.where('age' > 30))`

2. Sorting a DataFrame

What is the correct way to sort the DataFrame `df` by the column 'salary' in descending order?

- `print(df.sort\_values('salary', ascending=False))`

- `print(df.order\_by('salary', ascending=False))`

- `print(df.sort('salary', descending=True))`

- `print(df.sort\_values('salary', descending=True))`

3. Dropping a Column

How can you drop the column named 'temp' from the DataFrame `df` without affecting the original DataFrame?

- `print(df.drop(columns=['temp']))`

- `print(df.remove('temp'))`

- `print(df.delete('temp'))`

- `print(df.drop('temp', axis=1))`

4. Grouping and Aggregating Data

How can you print the average 'score' for each 'team' from the DataFrame `df`?

- `print(df.groupby('team')['score'].mean())`

- `print(df.aggregate('team', 'score'))`

- `print(df.mean('score', by='team'))`

- `print(df.groupby('team').avg('score'))`

5. Renaming Columns

What is the correct way to rename the column 'old\_name' to 'new\_name' in DataFrame `df`?

- `print(df.rename(columns={'old\_name': 'new\_name'}))`

- `print(df.rename('old\_name', 'new\_name'))`

- `print(df.columns['old\_name'] = 'new\_name')`

- `print(df.set\_names('old\_name', 'new\_name'))`

**9/35. questions that could relate to using the Pygame module**

1. Pygame Window Title

How can you set the title of the window in Pygame?

- `pygame.display.set\_caption('Game Title')`

- `pygame.window.title('Game Title')`

- `pygame.set\_title('Game Title')`

- `pygame.display.title('Game Title')`

2. Pygame Event Handling

Which Pygame function is used to get a list of all the events that have happened?

- `pygame.event.get()`

- `pygame.get\_events()`

- `pygame.events()`

- `pygame.event.list()`

3. Pygame Frame Rate

How can you limit the frame rate to 60 frames per second in Pygame?

- `pygame.time.Clock().tick(60)`

- `pygame.fps.set(60)`

- `pygame.display.framerate(60)`

- `pygame.set\_fps(60)`

4. Loading Images in Pygame

What is the correct way to load an image called 'player.png' for use in a Pygame application?

- `player\_image = pygame.image.load('player.png')`

- `player\_image = pygame.load\_image('player.png')`

- `player\_image = pygame.assets.get('player.png')`

- `player\_image = pygame.get\_image('player.png')`

5. Pygame Quitting

How do you correctly terminate a Pygame application?

- `pygame.quit()`

- `pygame.exit()`

- `pygame.terminate()`

- `pygame.close()`

**10/35. focusing on Python lists and variables:**

1. List Element Replacement

After executing the following code, what will be the content of `list1`?

list1 = [1, 2, 3, 4]

list2 = list1

list2[2] = 10

- `[1, 2, 10, 4]`

- `[1, 2, 3, 4]`

- `[1, 2, 3, 10]`

- `[1, 10, 3, 4]`

2. Variable Assignment

Given the code below, what will `var\_a` and `var\_b` contain?

var\_a = [1, 2, 3]

var\_b = var\_a[:]

var\_b.append(4)

- `var\_a` contains `[1, 2, 3]` and `var\_b` contains `[1, 2, 3, 4]`

- `var\_a` contains `[1, 2, 3, 4]` and `var\_b` contains `[1, 2, 3, 4]`

- Both `var\_a` and `var\_b` contain `[1, 2, 3]`

- Both `var\_a` and `var\_b` contain `[1, 2, 3, 4]`

3. List and Element Deletion

What will be the output of the following code?

my\_list = ['a', 'b', 'c', 'd']

del my\_list[1]

my\_list.remove('c')

- `['a', 'd']`

- `['a', 'b', 'd']`

- `['a', 'c', 'd']`

- `['b', 'c', 'd']`

4. Copying Lists

If we execute the following code, what happens to `original` and `copy`?

original = [1, 2, 3]

copy = original.copy()

copy.append(4)

- `original` is `[1, 2, 3]` and `copy` is `[1, 2, 3, 4]`

- `original` is `[1, 2, 3, 4]` and `copy` is `[1, 2, 3, 4]`

- Both `original` and `copy` are `[1, 2, 3]`

- Both `original` and `copy` are `[1, 2, 3, 4]`

5. List Indexing and Slicing

What is the result of the following code snippet?

numbers = [10, 20, 30, 40, 50]

slice = numbers[1:-1]

slice[1] = 35

- `numbers` is `[10, 20, 35, 40, 50]`

- `numbers` is `[10, 20, 30, 40, 50]`

- `slice` is `[20, 35, 40]`

- `slice` is `[20, 30, 40]`

**11/35. concepts of interpretation and compilation**

1. Language Execution Models

Which of the following statements is true regarding interpreted languages?

- Interpreted languages are typically slower to execute than compiled languages.

- Interpreted languages require a compilation step before running.

- Interpreted languages convert the source code to machine code once and save it.

- Interpreted languages do not allow for dynamic typing.

2. Benefits of Interpretation

What is an advantage of interpreted languages?

- They can be more flexible and easier to debug.

- They have faster execution speed compared to compiled languages.

- They typically result in more optimized machine code.

- They are always statically typed.

3. Compilation and Execution

What does a compiler do?

- Translates the entire source code into machine code before execution.

- Translates each line of source code to machine code at runtime.

- Only checks the syntax of the source code without executing it.

- Executes the source code without translating it.

4. Just-In-Time Compilation

What is "Just-In-Time" (JIT) compilation?

- A process that compiles the entire program's source code before the program runs.

- A process that compiles portions of the code at runtime as needed.

- A compilation that happens after the program execution.

- A compilation method that only interpreted languages use.

5. Python Execution

How does Python execute code?

- Python compiles the source code to bytecode, which is then interpreted by the Python virtual machine.

- Python directly compiles the source code to machine code.

- Python interprets the source code line by line without any form of compilation.

- Python requires manual compilation before the interpreter can run the code.

**12/35. focus on Python expressions and their resulting types**

1. Python Expression Types

Given the following expressions, which one will result in a float?

- `10 / 2`

- `10 // 2`

- `10 % 2`

- `10 \* 2`

2. String Repetition

What is the result of the following Python expression?

- `'Python' \* 3`

- `'Python' + '3'`

- `'Python' / 3`

- `'Python' - 3`

3. String to Integer Conversion

Which Python function would correctly convert the string `'123'` to an integer?

- `int('123')`

- `str(123)`

- `float('123')`

- `bool('123')`

4. Boolean Values in Python

What is the type of the result of the expression `True and False`?

- Integer

- String

- Boolean

- Float

5. Python Division

Which division operator in Python always results in a float, even when the division is even?

- `/`

- `//`

- `%`

- `\*`

**13/35. focusing on list operations in Python:**

1. List Extension

What will `list1` contain after the following operations?

list1 = [1, 2, 3]

list2 = [4, 5]

list1.extend(list2)

- `[1, 2, 3, 4, 5]`

- `[4, 5, 1, 2, 3]`

- `[1, 2, 3]`

- `[4, 5]`

2. List Appending

What happens when you append a list to another list?

list1 = [1, 2]

list2 = [3, 4]

list1.append(list2)

- `list1` becomes `[1, 2, [3, 4]]`

- `list1` becomes `[1, 2, 3, 4]`

- `list2` becomes `[1, 2, 3, 4]`

- `list1` remains `[1, 2]`

3. List Slicing

If `list1 = [0, 1, 2, 3, 4, 5]`, what does `list1[1:5:2]` return?

- `[1, 3]`

- `[0, 2, 4]`

- `[1, 2, 3, 4]`

- `[2, 4]`

4. List Pop Method

What is the result of `pop()` method on `list1 = ['a', 'b', 'c', 'd']`?

- Removes and returns the last item in the list, which is `'d'`.

- Removes and returns the first item in the list, which is `'a'`.

- The list becomes empty.

- Removes the last item from the list but does not return it.

5. List Index Method

How do you find the index of the element `3` in the list `[1, 2, 3, 4, 3]`?

- `list.index(3)`

- `list.find(3)`

- `list.get(3)`

- `list.position(3)`

**14/35. handling errors and exceptions in Python**

1. Handling Division Errors

What type of error is caught when trying to divide by zero in Python?

- `ValueError`

- `TypeError`

- `ZeroDivisionError`

- `KeyError`

2. Invalid Type Conversion

If you attempt to convert a string that does not contain a number to an integer, what error will occur?

- `ValueError`

- `TypeError`

- `ZeroDivisionError`

- `KeyError`

3. Accessing Non-existent Dictionary Key

What type of error is raised when trying to access a key that doesn’t exist in a dictionary?

- `ValueError`

- `TypeError`

- `ZeroDivisionError`

- `KeyError`

4. Appending to a Non-list Type

What type of error will occur if you try to use the `append` method on an integer in Python?

- `ValueError`

- `TypeError`

- `ZeroDivisionError`

- `AttributeError`

5. Using an Undefined Variable

What type of error does Python raise if you try to use a variable that has not been defined?

- `ValueError`

- `TypeError`

- `ZeroDivisionError`

- `NameError`

**15/35.**

1. Function Default Parameters

Which call is invalid if we have a function defined as `def compute(a, b=10, c=20)`?

- `compute(5)`

- `compute(a=5, c=30)`

- `compute(b=5, c=30)`

- `compute(5, b=15)`

2. Keyword Arguments

If a function is defined as `def printer(text, prefix='Error: ')`, which call will fail?

- `printer('An error occurred')`

- `printer(prefix='Warning: ', text='Low disk space')`

- `printer(text='User not found')`

- `printer('Error:', 'An error occurred')`

3. Mandatory and Optional Arguments

Given the function `def add(a, b, c=0, d=0)`, what is the minimum number of arguments you must pass?

- 1

- 2

- 3

- 4

4. Mixed Argument Types

For the function `def mix\_args(required, \*args, \*\*kwargs)`, which call is incorrect?

- `mix\_args('test', 1, 2, third=3)`

- `mix\_args()`

- `mix\_args('test', optional=1)`

- `mix\_args('test', 1, 2)`

5. Variable Scope Inside Functions

What will be printed after calling `def func(): x = 10; func(); print(x)`?

- 10

- 0

- `NameError` will be raised

- Nothing will be printed

**16/35.**

1. Logical Operations and Variable Assignment

Given the variables below, what will the value of the variable `result` be?

a = 4

b = 8

result = b > a and a \* 2 == b

- True

- False

- 8

- None

2. Modulo and Equality Check

Consider the following code snippet. What will be the value of `is\_equal`?

m = 10

n = 5

is\_equal = m % n == 0 and m != n

- True

- False

- 0

- 10

3. Combining Logical Operators

What will be the value of `outcome` after this code is executed?

p = 7

q = 14

outcome = q / p == 2 or p + q == 20

- True

- False

- 7

- 14

4. Comparison and Logical Operators

What will `final\_value` be in the following code?

i = 15

j = 5

final\_value = i / j == 3 and not j + i == 20

- True

- False

- 15

- 20

5. Nested Logical Conditions

After running the code below, what is the value of `can\_drive`?

age = 18

has\_license = True

can\_drive = age >= 18 and (has\_license or age > 21)

- True

- False

- 18

- None

**17/35. involving variable assignments and calculations**

1. Variable Assignment with Mixed Operators

What is the value of `var2` after the following code is executed?

var2 = (2 + 3 \* 2) \*\* 2 / 4

- `12.5`

- `20.25`

- `10`

- `6.25`

2. Modulus and Floor Division

What will be the value of `result` after this code snippet?

result = 15 % 4 // 2

- `1`

- `0`

- `3`

- `7.5`

3. Exponentiation and Subtraction

Consider the following statement. What will be the value of `var3`?

var3 = 5 \*\* 2 - 10

- `15`

- `25`

- `5`

- `-5`

4. Complex Arithmetic Expression

What is the final value of `expression` after executing this code?

expression = 8 / 2 \* (2 + 2)

- `16`

- `8`

- `32`

- `4`

5. Combined Operations with Floats

If we have the following code, what does `var4` hold?

var4 = 7 // 2 + 5.5

- `8.5`

- `3.5`

- `5.5`

- `9.0`

**18/35.**

1. Printing Stars

Given the following code snippet, how many stars (`\*`) will be printed?

i = 1

while (i <= 5):

if i % 2 != 0:

print("\*")

i += 1

- 2 stars

- 3 stars

- 5 stars

- 4 stars

2. Counting Loop Iterations

Consider the code below. How many times will "Python" be printed?

for i in range(1, 10, 2):

print("Python")

- 4 times

- 5 times

- 10 times

- 9 times

3. Nested Loops with Conditions

What will be the output of the following code snippet?

for i in range(3):

for j in range(2):

if j % 2 == 0:

print("Loop")

- "Loop" printed 3 times

- "Loop" printed 6 times

- "Loop" printed 2 times

- "Loop" printed once for each `i`

**19/35.**

1. Function Calculation with Default Parameters

Given the following function definition, what is the result of calling `compute(4)`?

def compute(x, y=3):

return x + 2 \* y

- `10`

- `14`

- `7`

- `5`

2. Nested Function Call

Consider the following two functions. What is the result of `outer(2)`?

def inner(z):

return z \* 3

def outer(x):

return inner(x + 1)

- `6`

- `9`

- `12`

- `3`

3. Function with Conditional Logic

What will `analyze(10)` return based on the function definition?

def analyze(value):

if value < 5:

return 'Low'

elif value < 10:

return 'Medium'

else:

return 'High'

- `'Low'`

- `'Medium'`

- `'High'`

- None of the above

**20/35. involving loops and conditional statements:**

1. Loop with Multiple Conditions

If we have the following loop, how many times will "Python" be printed?

for i in range(15):

if i % 3 == 0 or i % 4 == 0:

print("Python")

- Fill in the blank with the number of times "Python" will be printed.

2. Conditional Loop Execution

Given the following code snippet, what will be the final value of `a`?

a = 0

for i in range(10):

if i < 5 and not i % 2==0:

a += 1

- Fill in the blank with the final value of `a`.

3. Nested Loop with Condition

In the nested loop below, how many times will "Looping" be printed?

for i in range(4):

for j in range(3):

if j < 2:

print("Looping")

- Fill in the blank with the number of times "Looping" will be printed.

21/35.

1. Loop for Printing Characters

If we want to print the character 'A' five times, what should we write in the gap?

for i in range(\_\_):

print("A")

- Fill in the blank with the correct range to print 'A' five times.

2. Conditional Loop for Selective Printing

For the following loop, how many times will 'B' be printed?

for i in range(1, 10):

if i % 3 == 0:

print("B")

- Fill in the blank with the number of times 'B' will be printed.

3. Nested Loops for Pattern Printing

If we want to print the pattern 'C' three times in one line and repeat this for four lines, what should we write in the gaps?

for i in range(\_\_):

for j in range(\_\_):

print("C", end='')

print()

- Fill in the blanks with the correct range to create the described pattern.

**22/35.** involving dictionaries and iterations

1. Dictionary Iteration

Given the dictionary `data = {'first': (1, 2), 'second': (3, 4)}`, what will be the output of the following code?

for key in data.keys():

print(data[key][0], end="-")

- Fill in the blank with the expected output.

2. Accessing Dictionary Elements

If `info = {'x': [10, 20, 30], 'y': [40, 50, 60]}`, what will be the result of the following code?

result = ""

for k in info:

result += str(info[k][1])

print(result)

- Fill in the blank with the output of the print statement.

3. Selective Printing from a Dictionary

Consider `pairs = {'one': [1, 'a'], 'two': [2, 'b'], 'three': [3, 'c']}`. If we want to print only the numerical parts of the values, what should the following code look like?

for \_\_ in pairs:

print(\_\_)

- Fill in the blanks to print `1 2 3` with correct iteration and access.

**23/35.** involving exception handling

1. Exception Handling with ValueError

What will the following code output if a non-integer value is entered by the user?

try:

number = int(input("Enter an integer: "))

print(number)

except ValueError:

print("This is not an integer.")

else:

print("Thank you for entering an integer.")

- Fill in the blank with the expected output when a non-integer is entered.

2. Multiple Exceptions

What will be printed by the following code snippet if the input is `0`?

try:

value = int(input("Enter a number: "))

result = 10 / value

except ValueError:

print("Please enter a valid number.")

except ZeroDivisionError:

print("Cannot divide by zero.")

else:

print("Everything went well.")

finally:

print("The try-except block is finished.")

- Fill in the blank with the sequence of messages that will be printed.

3. File Handling Exception

What will the following code output if the file `nonexistent.txt` does not exist?

try:

with open('nonexistent.txt', 'r') as file:

print(file.read())

except FileNotFoundError:

print("The file could not be found.")

- Fill in the blank with the expected output.

**24/35.**

1. Counting Loop Iterations

How many times will the word "Looping" be printed with the following code?

count = 10

while count > 0:

print("Looping")

count -= 2

- Fill in the blank with the number of times "Looping" will be printed.

2. Conditional Loop with Break

If we want to stop the loop after printing "Python" three times, what should we add to the following code?

i = 0

while True:

print("Python")

i += 1

if \_\_:

break

- Fill in the blank with the correct condition to break the loop after printing "Python" three times.

3. Loop with Continue

How many numbers will be printed by the following code?

for i in range(1, 10):

if i % 3 == 0:

continue

print(i)

- Fill in the blank with the total count of numbers that will be printed.

**25/35.**

1. Loop Break with a Condition

What will be printed if we want to stop the loop after printing the first 3 letters of the word "coding"?

for letter in "coding":

if letter == "i":

break

print(letter, end="")

- Fill in the blank with what will be printed.

2. Conditional Loop Execution

If we want to print all letters except for vowels in the word "example", what condition should we add to the following code?

for char in "example":

if \_\_\_\_\_\_: # Condition to skip vowels

continue

print(char, end="")

- Fill in the blank with the correct condition.

3. Printing with a Loop

What will the following code print?

word = "parallel"

for i in word:

if i == "a":

print("A", end="")

continue

print(i, end="")

- Fill in the blank with what will be printed by the code.

**26/35.**

1.

If we have a recursive function that calls itself when `n` is not a multiple of 4, and we pass the number 16 to it, how many times will the function be called before it stops?

def recursive\_count(n):

if n % 4 != 0:

return recursive\_count(n-1)

else:

return 0

print(recursive\_count(16))

- Select the correct number of times the function will be called.

2.

A recursive function calls itself while the input `n` is greater than 0 and decreases `n` by 2 each time. How many times will the function be called with the initial value of `n` being 5?

def count\_calls(n):

if n > 0:

return count\_calls(n-2)

else:

return 1

print(count\_calls(5))

- Select the correct number of times the function will be called.

3.

Given a recursive function that calls itself only if `n` is even, what will be the total number of calls including the first call if the function is called with `n` as 8?

def even\_recurse(n):

if n % 2 == 0:

return even\_recurse(n-1)

else:

return 1

print(even\_recurse(8))

- Select the correct number of times the function will be called.

**27/35.**

.

**28/35.**

. You are requesting to generate multiple programming-related questions similar to the ones you've uploaded. Here are five questions that reflect a similar style and complexity:

1. What is the output of the following code snippet?

numbers = [1, 2, 3, 4, 5]

output = [n \* n for n in numbers if n % 2 == 0]

print(output)

- [ ] `[1, 4, 9, 16, 25]`

- [ ] `[4, 16]`

- [ ] `[1, 9, 25]`

- [ ] `SyntaxError`

2. Which line of code correctly appends an element 'apple' to the list 'fruits'?

fruits = ['banana', 'cherry']

- [ ] `fruits.append['apple']`

- [ ] `fruits.add('apple')`

- [ ] `fruits.append('apple')`

- [ ] `append(fruits, 'apple')`

3. Which of the following is a valid dictionary creation in Python?

- [ ] `d = {('a', 1), ('b', 2)}`

- [ ] `d = {'a': 1, 'b': 2}`

- [ ] `d = dict('a' = 1, 'b' = 2)`

- [ ] `d = dict[['a', 1], ['b', 2]]`

4. How does the 'continue' statement work inside a loop?

- [ ] It stops the loop and exits.

- [ ] It skips the current iteration and moves to the next iteration.

- [ ] It repeats the current iteration indefinitely.

- [ ] It has no effect on the loop.

5. What will be the result of the following function call?

def multiply\_by\_two(x):

return x \* 2

result = multiply\_by\_two(5.5)

print(result)

- [ ] `5.5`

- [ ] `11.0`

- [ ] `TypeError`

- [ ] `10`

**29/35.**

1.

num1 = input("Enter first number: ")

num2 = input("Enter second number: ")

result = int(num1) \* int(num2)

print(result)

What will be the output if the user enters '3' for the first number and '4' for the second number?

- A) 7

- B) 12

- C) 34

- D) Error

2.

string = input("Enter a word: ")

print(string[::-1])

What will be the output if the user enters 'hello'?

- A) olleh

- B) hello

- C) oellh

- D) Error

3.

value = input("Enter a boolean value (True/False): ")

if value == 'True':

print(not bool(value))

else:

print(bool(value))

What will be the output if the user enters 'True'?

- A) True

- B) False

- C) 'True'

- D) 'False'

4.

hours = input("Enter number of hours: ")

rate = input("Enter rate per hour: ")

total\_pay = float(hours) \* float(rate)

print("Total pay:", total\_pay)

What will be the output if the user enters '8' for hours and '15.5' for rate?

- A) Total pay: 124

- B) Total pay: 123.5

- C) Total pay: 124.0

- D) Error

5.

number = input("Enter a number to double: ")

print(f"The double of {number} is {2 \* int(number)}")

What will be the output if the user enters '5'?

- A) The double of 5 is 10

- B) The double of 5 is 55

- C) 10

- D) Error

Answers:

1. B) 12

2. A) olleh

3. B) False

4. A) Total pay: 124

5. A) The double of 5 is 10

**30/35.**

1. Given the following snippet, what will be the output?

numbers = [5, 2, 5, 2, 2]

for x\_count in numbers:

output = ''

for count in range(x\_count):

output += 'x'

print(output)

- A series of 'x' corresponding to each number

- A single line of 'x'

- An error message

- None of the above

2. What will the following code print?

def print\_max(a, b):

if a > b:

return a

else:

return b

print(print\_max(3, 5))

- 3

- 5

- Nothing, there's an error

- 8

3. What is the final value of `y` after this code is executed?

y = 10

for x in range(5):

y -= x

print(y)

- 5

- 10

- 15

- None of the above

4. What does the following function return when we call `calc(8)`?

def calc(x):

return x \* x

# What's the result of this call?

calc(8)

- 64

- 16

- "8"

- An error

5. What will be printed when the following code is executed?

def is\_even(num):

return num % 2 == 0

print(is\_even(4), is\_even(9))

- True True

- True False

- False True

- False False

**31/35.**

1. What will the following code output?

a = "5"

b = 5

if a == b:

print("Match")

else:

print(int(a) + b)

- Match

- 10

- TypeError

- 55

2. Consider the code snippet below, what will it print?

list1 = [1, 2, 3]

list2 = [4, 5, 6]

if sum(list1) == sum(list2):

print("Sums are equal")

else:

print("Sums are different", sum(list1), sum(list2))

- Sums are equal

- Sums are different 6 15

- Sums are different 15 6

- An error occurs

3. What does the following function output when called with `check\_number(0)`?

def check\_number(n):

if n:

return "Not Zero"

return "Zero"

print(check\_number(0))

- Not Zero

- Zero

- True

- False

4. Analyze the code below. What will be the final output?

x = 10

y = "20"

z = "Thirty"

if str(x) == y:

print("x equals y")

elif x == int(y):

print("x equals int(y)")

else:

print("x plus y equals", x + int(y))

- x equals y

- x equals int(y)

- x plus y equals 30

- TypeError

5. What is printed when this code is executed?

def print\_value(a, b):

if a == b:

print("a equals b")

elif a < b:

print("a is less than b")

elif a > b:

print("a is greater than b")

else:

print("No match")

print\_value(3, 3.0)

- a equals b

- a is less than b

- a is greater than b

- No match

**32/35.**

1. Given the following code, what will be the output?

def calculate\_difference(x, y):

return x - y

result = calculate\_difference(10, calculate\_difference(5, 3))

print(result)

- 8

- 2

- 5

- -2

2. What will the following code snippet output?

def multiply\_by\_self(n):

return n \* n

number = 7

output = multiply\_by\_self(multiply\_by\_self(number))

print(output)

- 49

- 2401

- 14

- 343

3. If the following Python code is executed, what will it print?

count = 0

for i in range(1, 5):

if i % 2 == 0:

count += 1

print(count)

- 1

- 2

- 4

- 3

4. What will be the output of the following function when we call `increment\_by\_two(3)`?

def increment\_by\_two(n):

if n < 10:

return increment\_by\_two(n + 2)

else:

return n

print(increment\_by\_two(3))

- 10

- 11

- 3

- Infinite loop

5. Analyze the code snippet below. What will be the output after execution?

text = "programming"

for letter in text:

if letter in "aeiou":

text = text.replace(letter, "")

print(text)

- programming

- prgrmmng

- programmin

- prgrmng

**33/35.**

1. Given the following code snippet, what will be the output?

nums = [5, 3, 8, 6]

result = nums[0] \* nums[-1]

print(result)

- (A) 15

- (B) 30

- (C) 48

- (D) SyntaxError

2. If the code below is executed, what will be printed to the console?

x = "global"

def check\_scope():

global x

x = "local"

return x

print(check\_scope(), x)

- (A) global local

- (B) local global

- (C) local local

- (D) SyntaxError

3. What is the result of the following code execution?

def divider(numerator, denominator):

try:

return numerator / denominator

except ZeroDivisionError:

return "Cannot divide by zero."

print(divider(10, 0))

- (A) Infinity

- (B) Cannot divide by zero.

- (C) 0

- (D) An error occurs that is not caught by the except block.

4. Consider the Python code snippet below. What will it output when run?

count = 10

while count > 0:

count -= 3

print(count, end=' ')

- (A) 7 4 1 -2

- (B) 10 7 4 1

- (C) 7 4 1

- (D) The code will result in an infinite loop.

5. What will the following Python code snippet print out?

sequence = ['a', 'b', 'c', 'd']

for item in sequence:

if item == 'c':

break

print(item, end=' ')

- (A) a b c d

- (B) a b c

- (C) a b

- (D) The code will raise a ValueError.

**34/35.**

1. Debugging a Function Call

def compute\_sum(x, y):

return x + z

a = 5

b = 7

print(compute\_sum(a, b))

Which error is likely to occur when the above code is executed?

- NameError

- SyntaxError

- TypeError

- No error will occur

2. Understanding Loops and Conditions

count = 0

for num in range(10):

if num % 4 == 0:

count += 1

print(count)

What will be the output of the above code snippet?

- 2

- 3

- 4

- An error occurs

3. Working with Lists

my\_list = [10, 20, 30, 40, 50]

for i in range(len(my\_list)):

if my\_list[i] == 30:

my\_list[i] = my\_list[i] \* 2

print(my\_list)

What is the final state of `my\_list` after the code is executed?

- [10, 20, 60, 40, 50]

- [20, 40, 60, 80, 100]

- [10, 20, 30, 40, 50]

- A ValueError occurs

4. String Operations

initial\_string = "coding"

modified\_string = ""

for char in initial\_string:

modified\_string = char.upper() + modified\_string

print(modified\_string)

What will be the value of `modified\_string` after the code execution?

- "CODING"

- "GNIDOC"

- "Gnidoc"

- An AttributeError occurs

5. Dictionary Key-Value Pairing

details = {'name': 'Alice', 'age': 28, 'job': 'Engineer'}

person = "name is {name}, age is {age}, and job is {job}".format(\*\*details)

print(person)

What will be the output of the `person` variable?

- "name is Alice, age is 28, and job is Engineer"

- "name is {name}, age is {age}, and job is {job}"

- A KeyError occurs

- A TypeError occurs

**35/35.**