**Python Assignment**

**1) What are the types of Applications?**

Applications are programs or software designed to perform specific tasks. The main types of applications include:

* **Web Applications**: Run on web browsers and accessed via the internet (e.g., Gmail, Facebook).
* **Mobile Applications**: Designed for smartphones or tablets (e.g., WhatsApp, Instagram).
* **Desktop Applications**: Installed on personal computers (e.g., MS Word, Photoshop).
* **Enterprise Applications**: Used in large organizations for business processes (e.g., ERP systems).
* **Embedded Applications**: Run on embedded systems like IoT devices (e.g., smart thermostats).

**2) What is programming?**

Programming is the process of writing, testing, and maintaining code to create software applications. It involves using programming languages like Python, Java, or C++ to instruct a computer to perform specific tasks. Programming is essential for creating websites, apps, games, and system tools.

**3) What is Python?**

Python is a high-level, interpreted programming language known for its simplicity and readability. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is widely used in web development, data analysis, artificial intelligence, machine learning, and more.

Example:

print("Hello, World!")

**7) How is memory managed in Python?**

Python uses a combination of reference counting and garbage collection for memory management:

* **Reference Counting**: Tracks the number of references to an object in memory. When the count drops to zero, the object is deleted.
* **Garbage Collection**: Automatically reclaims memory by removing objects that are no longer in use. It handles cyclic references that reference counting alone cannot resolve.

**8) What is the purpose of the continue statement in Python?**

The continue statement is used to skip the remaining part of the current loop iteration and proceed to the next iteration. It is commonly used in loops to avoid executing specific code under certain conditions.

Example:

for i in range(5):

if i == 3:

continue

print(i)

# Output: 0, 1, 2, 4

**17) What are negative indexes and why are they used?**

Negative indexes in Python allow you to access elements from the end of a sequence. For example, -1 refers to the last element, -2 refers to the second last element, and so on. Negative indexing is useful for slicing or accessing elements without knowing the exact length of a sequence.

Example:

list1 = [10, 20, 30, 40]

print(list1[-1]) # Output: 40

**25) What is a List? How will you reverse a list?**

A list in Python is a mutable, ordered collection of items that can hold elements of different data types. Lists are defined using square brackets [].

To reverse a list, you can use the following methods:

1. Using slicing:

list1 = [1, 2, 3, 4]

reversed\_list = list1[::-1]

print(reversed\_list)

1. Using reverse() method:

list1.reverse()

print(list1)

**26) How will you remove the last object from a list?**

You can remove the last object from a list using the pop() method. This method removes and returns the last element.

Example:

list1 = [1, 2, 3, 4]

list1.pop()

print(list1) # Output: [1, 2, 3]

**27) Suppose list1 is [2, 33, 222, 14, and 25], what is list1[-1]?**

The negative index -1 refers to the last element of the list. For list1 = [2, 33, 222, 14, 25], the value of list1[-1] is 25.

**28) Differentiate between append() and extend() methods.**

* **append()**: Adds a single element to the end of the list. Example:
* list1 = [1, 2, 3]
* list1.append(4)

print(list1) # Output: [1, 2, 3, 4]

* **extend()**: Adds elements of an iterable (e.g., list, tuple) to the end of the list. Example:
* list1 = [1, 2, 3]
* list1.extend([4, 5])

print(list1) # Output: [1, 2, 3, 4, 5]

**30) How will you compare two lists?**

To compare two lists, you can use the == operator, which checks if both lists have the same elements in the same order.

Example:

list1 = [1, 2, 3]

list2 = [1, 2, 3]

print(list1 == list2) # Output: True

**43) What is a tuple? Difference between list and tuple.**

A tuple is an immutable, ordered collection of items. It is defined using parentheses ().

**Differences:**

* Lists are mutable; tuples are immutable.
* Lists use square brackets []; tuples use parentheses ().
* Lists are slower; tuples are faster due to immutability.

Example:

list1 = [1, 2, 3]

tuple1 = (1, 2, 3)

**47) How will you create a dictionary using tuples in Python?**

You can create a dictionary using tuples by passing a list of key-value pairs to the dict() constructor.

Example:

tuples = [("a", 1), ("b", 2), ("c", 3)]

dict1 = dict(tuples)

print(dict1) # Output: {'a': 1, 'b': 2, 'c': 3}

**51) How do you traverse through a dictionary object in Python?**

You can traverse a dictionary using a for loop.

Example:

dict1 = {'a': 1, 'b': 2, 'c': 3}

for key, value in dict1.items():

print(key, value)

**52) How do you check the presence of a key in a dictionary?**

You can check for a key in a dictionary using the in operator.

Example:

dict1 = {'a': 1, 'b': 2}

print('a' in dict1) # Output: True

**65) How many basic types of functions are available in Python?**

Python has two basic types of functions:

* **Built-in Functions**: Predefined functions like len(), print(), etc.
* **User-defined Functions**: Functions created by the programmer using the def keyword.
* **Anonymous functions (lambda function):**  a function that is not stored in a program file, but is associated with a variable whose data type is function\_handle .

**66) How can you pick a random item from a list or tuple?**

Use the random.choice() function from the random module.

Example:

import random

list1 = [1, 2, 3, 4]

print(random.choice(list1))

**67) How can you pick a random item from a range?**

Use the random.randrange() function to pick a random number from a range.

Example:

import random

print(random.randrange(1, 10))

**68) How can you get a random number in Python?**

Use the random.random() function to generate a random float between 0 and 1.

Example:

import random

print(random.random())

**69) How will you set the starting value in generating random numbers?**

Use the random.seed() function to set the starting value (seed) for generating random numbers.

Example:

import random

random.seed(10)

print(random.random())

**70) How will you randomize the items of a list in place?**

Use the random.shuffle() function to randomize the items in a list.

Example:

import random

list1 = [1, 2, 3, 4]

random.shuffle(list1)

print(list1)

**71) What is the File function in Python? What are keywords to create and write a file?**

The file function refers to Python’s ability to work with file objects for reading, writing, and appending. The keywords used include:

* **open()**: Opens a file.
* **Modes**: ‘w’ (write), ‘r’ (read), ‘a’ (append).

Example:

file = open("example.txt", "w")

file.write("Hello, File!")

file.close()

**83) Explain Exception handling? What is an Error in Python?**

Exception handling in Python is the process of managing runtime errors using try, except, else, and finally blocks. Errors are issues in a program that may cause it to stop execution. Exceptions are a type of error that can be handled programmatically.

Example:

try:

x = 10 / 0

except ZeroDivisionError:

print("Cannot divide by zero")

**84) How many except statements can a try-except block have? Name some built-in exception classes:**

A try-except block can have multiple except clauses to handle different exceptions. Some built-in exception classes include:

* ZeroDivisionError
* ValueError
* TypeError
* KeyError

**85) When will the else part of try-except-else be executed?**

The else block is executed only if no exception occurs in the try block.

Example:

try:

print(10 / 2)

except ZeroDivisionError:

print("Error")

else:

print("No Error")

**86) Can one block of except statements handle multiple exceptions?**

Yes, you can handle multiple exceptions in a single except block by specifying them as a tuple.

Example:

try:

x = int("a")

except (ValueError, TypeError):

print("Handled multiple exceptions")

**87) When is the finally block executed?**

The finally block is executed regardless of whether an exception occurs or not. It is typically used for cleanup actions.

Example:

try:

print(10 / 0)

except ZeroDivisionError:

print("Error")

finally:

print("Cleanup")

**88) What happens when 1 == '1' is executed?**

When 1 == '1' is executed, it returns False because the integer 1 and the string '1' are different types and not equal.

**89) How do you handle exceptions with try/except/finally in Python? Explain with coding snippets.**

the "try" block contains code that might raise an exception, the "except" block handles the exception if it occurs, and the "finally" block always executes regardless of whether an exception is raised or not.

# Python code to illustrate

def divide(x, y):

try:

# Floor Division : Gives only Fractional, Part as Answer

result = x // y

except ZeroDivisionError:

print("Sorry ! You are dividing by zero ")

else:

print("Yeah ! Your answer is :", result)

finally:

# this block is always executed

# regardless of exception generation.

print('This is always executed')

# Look at parameters and note the working of Program

divide(3, 2)

divide(3, 0)