WIPRO SAMPLE QUESTIONS

**Sample questions python fundamentals:**

**1. List out few scripting languages:**

* Python
* JavaScript
* Perl
* Ruby
* Bash

**2. What is the another name of Python 3.2?**

* There is no official "another name" for Python 3.2.
* It is simply referred to as **Python 3.2** — a specific version of Python 3.x.

**3. Which command enables the interactive terminal mode?**

* python or python3 (typed in the command line or terminal)

**4. List out few mutable data types:**

* list
* dict (dictionary)
* set
* bytearray

**5. Which function returns the location of an object?**

* id()

**6. Which operator does integer division?**

* // (floor division operator)

**7. What is typecasting?**

* Typecasting is converting one data type to another.  
  Example: int("5") converts string "5" to integer 5.

**8. How to do multiline comment?**

* Using triple quotes:

python

CopyEdit

'''

This is a multiline comment

'''

**9. Which built-in function returns a sequence of integers?**

* range()

**10. How can we use else with for?**

* The else block runs after the for loop **only if it was not terminated by break**.

python

CopyEdit

for i in range(5):

print(i)

else:

print("Done without break")

**11. Can we write statements after pre-keyword?**

* If you meant **break keyword**, then **yes**, but those lines won't execute if break runs.  
  If you meant **pass**, then yes — it’s just a placeholder.
* **12. Which statement does nothing?**
* pass

**Sample question DSA**

1. Does list maintain insertion order?  
    Yes, lists in Python maintain the order in which items are inserted.
2. Can we access list items with negative index?  
    Yes, negative indexing allows access from the end of the list.  
   Example: list[-1] gives the last item.
3. Which function empties the list?  
   list.clear() empties the entire list.
4. Which class represents the group of characters?  
   The str (string) class represents a group (or sequence) of characters in Python.
5. What is slicing?  
   Slicing is extracting a portion of a sequence (like list or string) using start:stop[:step].  
   Example: my\_list[1:4] returns items from index 1 to 3.
6. Which function toggles the cases?  
   string.swapcase() toggles lowercase to uppercase and vice versa.
7. Does set take duplicate elements?  
    No, a set automatically removes duplicate elements.

**Sample Question Functions**

1. **List few built-in functions.**
   * print(): Used to output information to the console.
   * len(): Returns the length (number of items) of an object.
   * input(): Used to get user input from the console.
   * type(): Returns the type of an object.
   * int(), str(), float(): Used for type conversion.
   * sum(): Returns the sum of all items in an iterable.
   * min(), max(): Returns the smallest/largest item in an iterable.
   * range(): Generates a sequence of numbers.
   * dir(): Returns a list of attributes of an object or names in the current scope.
   * help(): Invokes the built-in help system.
2. **Which keyword marks the start of the function?**
   * Def
3. **Which keyword ends the execution of a function?**
   * Return
4. **Which allows you to logically organize your python files?**
   * **Modules** and **Packages** (Packages are collections of modules).
5. **How to import a module with alias name?**
   * Using the as keyword.
   * **Example:** import math as m (Now you can use m.sqrt() instead of math.sqrt())
6. **Which command lists the built in modules?**
   * You can get a list of loaded modules using sys.modules.keys() after import sys.
   * To explore *all* available standard library modules (not just loaded ones), there isn't a single "command" to list *all* of them directly in a concise way that fits this type of answer format. However, you can use help('modules') in an interactive Python session, which will list all installed modules. For built-in modules specifically, you might look at sys.builtin\_module\_names.
7. **Can we import multiple modules with single import statement?**
   * Yes, by separating them with commas.
   * **Example:** import os, sys, math

**Sample Questions Command Line Arguments**

1. Which module is required to access cmd line arguments?

To access command-line arguments in Python, the most commonly used and fundamental module is the **sys module**.

Specifically, the sys.argy list within the sys module holds the command-line arguments.

* sys.argv[0] is always the name of the script itself.
* sys.argv[1], sys.argv[2], and so on, are the arguments passed to the script.

For more complex command-line argument parsing (e.g., handling flags, optional arguments, type conversion, or generating help messages), the argparse module is generally preferred and recommended. However, for simple access, sys.argv is sufficient.

**Sample Questions File**

1. **How to open a file in create mode?**

* In Python, you can open a file in "write" mode ('w') or "exclusive creation" mode ('x').
  + **'w' (write mode):** If the file exists, its content is truncated (emptied). If the file does not exist, a new file is created.
  + **'x' (exclusive creation mode):** This mode creates a new file, but if the file already exists, it raises a FileExistsError. This is safer if you want to ensure you are creating a *new* file and not overwriting an existing one.

1. **Which function reads one line at a time?**

* In Python, the readline() method of a file object reads a single line from the file.

1. **Which module is required to delete files or folders?**

* In Python, the os module (specifically, os.remove() for files and os.rmdir() for empty directories, or shutil.rmtree() for non-empty directories) is used for deleting files and folders.
* os.remove(path): Deletes a file.
* os.rmdir(path): Deletes an empty directory.
* shutil.rmtree(path): Deletes a directory and all its contents (files and subdirectories). You would need to import shutil for this.

**Sample Questions Exception Handling**

1. **What will be the output for print(5-'A')?**
   * **Output:** TypeError: unsupported operand type(s) for -: 'int' and 'str'
   * **Explanation:** Python does not allow direct subtraction between an integer (5) and a string ('A'). This operation attempts to combine incompatible data types, leading to a TypeError.
2. **Which block is always executed irrespective of the exception?**
   * The finally block.
   * **Explanation:** Code within a finally block will always execute, whether an exception occurred in the try block or not, and whether it was handled by an except block or not. It's commonly used for cleanup operations like closing files or network connections.
3. **Which block is executed when there is no exception?**
   * The else block (in a try-except-else-finally structure).
   * **Explanation:** The else block associated with a try...except statement is executed only if the code in the try block completes without raising any exceptions.
4. **Which is the base class for all the exception classes?**
   * BaseException
   * **Explanation:** All built-in exceptions in Python are derived from the BaseException class. A more common base class for *most user-defined* exceptions and the standard errors you typically catch is Exception, which is a subclass of BaseException.
5. **Do we need any module to implement exception handling?**
   * No.
   * **Explanation:** Exception handling (try, except, else, finally) is a built-in feature of the Python language. You do not need to import any specific module to use it.

**Sample Questions List/Lambda Function**

1. **Is List comprehension faster than for loop?**
   * **Generally, yes.**
   * **Explanation:** List comprehensions are often faster than equivalent for loops, especially for simple transformations and filtering. This is because:
     + They are implemented in C under the hood, making them more optimized.
     + They involve fewer bytecode instructions compared to explicit for loops, which translates to faster execution.
     + They avoid the overhead of calling append() repeatedly.
   * For very complex logic or when side effects are desired, a for loop might be clearer or more appropriate, but for simple list construction, comprehensions typically offer a performance boost.
2. **Is List iterator or iterables?**
   * A Python list is an **iterable**.
   * **Explanation:**
     + An **iterable** is an object that can be looped over (e.g., with a for loop). When you call iter() on an iterable, it returns an iterator. Examples include lists, tuples, strings, dictionaries, and sets.
     + An **iterator** is an object that represents a stream of data. It has a \_\_next\_\_() method that returns the next item in the sequence. When there are no more items, it raises a StopIteration exception.
   * So, a list is an iterable because you can iterate over it. When you do so, Python implicitly creates an iterator from the list to manage the iteration process.
3. **Is Lambda keyword must in lambda functions?**
   * **Yes.**
   * **Explanation:** The lambda keyword is the defining keyword for creating anonymous (unnamed) functions in Python. It's an integral part of their syntax: lambda arguments: expression
   * You cannot create a lambda function without using the lambda keyword.

**Sample Questions AI/ML**

1. **What is Artificial Intelligence?**
   * **Artificial Intelligence (AI)** refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The term may also be applied to any machine that exhibits traits associated with a human mind, such as learning and problem-solving. AI aims to enable machines to perform cognitive functions such as perceiving, reasoning, learning, interacting, and problem-solving, traditionally associated with human intelligence.
2. **What is Machine Learning?**
   * **Machine Learning (ML)** is a subset of Artificial Intelligence (AI) that enables systems to learn from data, identify patterns, and make decisions with minimal human intervention. Instead of being explicitly programmed for every task, ML algorithms learn from vast amounts of data, improving their performance over time as they are exposed to more data. It's about teaching computers to learn from experience.
3. **Difference between Rule-based AI and ML-based AI?**
   * **Rule-based AI:**
     + Relies on a predefined set of "if-then" rules created by human experts.
     + Knowledge is explicitly programmed into the system.
     + Works well for well-defined problems with clear, static rules.
     + Lacks flexibility and adaptability to new, unforeseen situations or data.
     + Examples: Expert systems, early chatbots, simple decision trees.
   * **ML-based AI:**
     + Learns patterns and rules directly from data.
     + Knowledge is implicitly derived by the algorithm from examples.
     + Excels at complex problems with vast or dynamic data, identifying patterns humans might miss.
     + Highly adaptable and can generalize to new, unseen data.
     + Examples: Image recognition, natural language processing, recommendation systems, spam filters.
4. **Difference between Supervised Machine Learning and Unsupervised Machine Learning?**
   * **Supervised Machine Learning:**
     + **Data:** Uses labeled data, meaning the input data has corresponding output labels (the "correct answers").
     + **Goal:** To learn a mapping from inputs to outputs, so it can predict outputs for new, unseen inputs.
     + **Process:** The model is "supervised" during training, continuously adjusting its parameters based on the errors it makes compared to the correct labels.
     + **Common Tasks:** Classification (e.g., spam detection, image classification) and Regression (e.g., predicting house prices, stock prices).
     + **Examples:** Linear Regression, Logistic Regression, Support Vector Machines (SVM), Decision Trees, Random Forests, Neural Networks.
   * **Unsupervised Machine Learning:**
     + **Data:** Uses unlabeled data; there are no predefined output labels.
     + **Goal:** To find hidden patterns, structures, or relationships within the data.
     + **Process:** The model explores the data on its own to discover inherent groupings or features.
     + **Common Tasks:** Clustering (grouping similar data points), Dimensionality Reduction (reducing the number of features), Association Rule Mining (finding relationships between variables).
     + **Examples:** K-Means Clustering, Hierarchical Clustering, Principal Component Analysis (PCA), Apriori algorithm.
5. **Difference between Strong AI and Weak AI?**
   * **Weak AI (Narrow AI):**
     + **Definition:** AI systems designed and trained for a particular task or a narrow set of tasks. They can perform specific tasks very well, often exceeding human capabilities, but they do not possess genuine intelligence, consciousness, or broad cognitive abilities.
     + **Capabilities:** Simulate human intelligence in limited contexts.
     + **Current State:** Most of the AI we see and use today (e.g., virtual assistants like Siri/Alexa, recommendation engines, facial recognition, autonomous cars) falls under Weak AI.
     + **Consciousness:** Does not possess true consciousness or self-awareness.
   * **Strong AI (General AI or AGI - Artificial General Intelligence):**
     + **Definition:** Hypothetical AI systems that possess human-like cognitive abilities, including reasoning, problem-solving, learning, understanding language, and adapting to various environments, just like a human. It would be able to perform any intellectual task that a human being can.
     + **Capabilities:** Mimic human intelligence in its full breadth and depth.
     + **Current State:** Strong AI does not yet exist and is a subject of ongoing research and debate.
     + **Consciousness:** Would theoretically possess consciousness, sentience, and self-awareness.
6. **What are applications of AI?**
   * **Healthcare:** Disease diagnosis, drug discovery, personalized medicine, robotic surgery, medical imaging analysis.
   * **Finance:** Fraud detection, algorithmic trading, credit scoring, personalized financial advice.
   * **Retail & E-commerce:** Recommendation engines, customer service chatbots, demand forecasting, inventory management, personalized shopping experiences.
   * **Education:** Personalized learning, intelligent tutoring systems, automated grading.
   * **Transportation:** Self-driving cars, traffic management, logistics optimization.
   * **Manufacturing:** Predictive maintenance, quality control, robotic automation, supply chain optimization.
   * **Customer Service:** Chatbots, virtual assistants, sentiment analysis.
   * **Agriculture:** Precision farming, crop monitoring, disease detection, automated harvesting.
   * **Security:** Facial recognition, cybersecurity threat detection, surveillance.
   * **Entertainment:** Content recommendation (Netflix, Spotify), game AI, personalized advertising.
   * **Natural Language Processing (NLP):** Language translation, spam filtering, sentiment analysis, speech recognition.
   * **Computer Vision:** Image recognition, object detection, facial recognition, augmented reality.