Q1. Rules for Python variables?

Ans. Variables are containers for storing data values. Python is not “statically typed”. We do not need to declare variables before using them or declare their type. A variable is created the moment we first assign a value to it. A Python variable is a name given to a memory location. Following are rules for python variable.

* A Pythonvariable name must start with a letter or the underscore character.
* A Python variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_ ).
* A Python variable name cannot start with a number.
* Variable in Python names are case-sensitive (python, Python, and PYTHON are three different variables).
* The reserved words or Keywords in Python cannot be used to name the variable in Python.

e.g \_name1 and name\_1is valid variable name but 1\_name and @name is invalid

Q2. Declaration and Initialization of Variables?

Python is dynamically typed, meaning we don't need to specify the variable type explicitly. We do not need to declare variables before using them or declare their type. A variable is created the moment we first assign a value to it.

* We can declare a variable with initial value i.e, initialization is the process of assigning a value to the variable.: variable\_name = initial\_value
* We can declare a variable without initializing it : variable\_name = None
* Python allows us to assign values to multiple variables in one line

x, y, z = "Orange", "Banana", "Cherry"

* We can assign the same value to multiple variables simultaneously with “=” operators. :

x = y = z = "Orange"

Q3. Redeclaring variables in Python?

Ans. We can redeclare Python variable once we have declared the variable and define variable in python already. but it's more accurately this redeclaring is reassigning . Python allows us to assign a new value to an existing variable, effectively changing its value or type.

variable\_name = initial\_value # Initial assignment

variable\_name = new\_value # Reassignment/Redeclaration

Python doesn't have strict "variables" unlike other languages.

Q4. Python Assign Values to Multiple Variables?

Ans. We can assign the same value to multiple variables simultaneously with “=” operators. :

x = y = z = "Orange"

Q5. Assigning different values to multiple variables?

Ans. Python allows us to assign different values to multiple variables with “,” operator.

x, y, z = "Orange", "Banana", "Cherry"

Q6. Can We Use the Same Name for Different Types?

Ans. No,If we use the same name, the variable starts referring to a new value and type. This is reassigning in another terms .The value which is reassigned becomes the present value for the variable(value gets overwritten).

e.g, x=8

x=”good”.

Value of x becomes “good” of str datatype.

Q7. What is variable interpolation in Python?

Ans. Variable interpolation in Python refers to the process of embedding variables within a string so that their values are included dynamically. This allows you to create strings where certain parts are replaced with the values of variables. In Python, there are several ways to achieve variable interpolation:

* Using f-strings :e.g.

name = 'World'

program = 'Python'

print(f'Hello {name}! This is {program}')

* Using the str.format() method :e.g

name = 'world'

print('Hello, {}'.format(name)):

* Using % formatting: e.g, print("%s %s" %('Hello','World',))
* Template Strings:

from string import Template

name = 'world'

program ='python'

new = Template('Hello $name! This is $program.')

print(new.substitute(name= name,program=program))

Q1. What are the built-in data types in Python?

Ans. Variables can store data of different types, and different types can do different things.

Python has the following data types built-in by default, in these categories:

Text Type: str

Numeric Types: int, float, complex

Sequence Types: list, tuple, range

Mapping Type:dict

Set Types: set, frozenset

Boolean Type:bool

Binary Types:bytes, bytearray, memoryview

None Type:NoneType

Q2. Explain the difference between lists and tuples in Python.?

|  |  |
| --- | --- |
| LIST | TUPLE |
| mutable | immutable |
| Dynamically | Fixed size |
| Enclosed in [ ] brackets | Enclosed in parenthesis { } |
| Takes more memory | Memory efficient |
| Useful for insertion and deletion operations. | Useful for read only operations like accessing elements. |

Q3. How can you convert a string to an integer in Python?

Ans. We can convert a string to an integer in Python using the int() function ,This is called typecasting.

e.g. s=”123”

x=int(s)

Note: If string contains non-numeric characters or is formatted incorrectly as an integer, attempting to convert it to an integer will raise a ValueError.

Q4. How do you check the data type of a variable in Python?

Ans. The type() function returns the type of the variable as a Python class.

e.g. x = 5

print(type(x)) # Output: <class 'int'>

Q5. Explain the concept of mutable and immutable data types in Python.

Ans. A mutable object is one whose internal state is changeable. On the contrary, once an immutable object in Python has been created, we cannot change it in any way. This property determines the ability of data types of being able to get updated after being initialized.

Python mutable data types:

* Lists
* Dictionaries
* Sets
* User-Defined Classes (It depends on the user to define the characteristics of the classes)

Python immutable data types:

* Numbers (Integer, Float, Complex, Decimal, Rational & Booleans)
* Tuples
* Strings
* Frozen Sets

Q6. How can you create a dictionary in Python?

Ans. In dictionary keys must be unique and immutable (strings, numbers, or tuples typically) and values can be of any data type and can be duplicated. Keys are case-sensitive.We can access, modify, add, or remove elements from the dictionary using key-based operations.

* Creating an empty dictionary

my\_dict = {} or dict()

* Creating a dictionary with some initial key-value pairs

my\_dict = {"name": "Alice", "age": 30, "city": "New York"}

* Adding key-value pairs to an existing dictionary

my\_dict["country"] = "USA"

my\_dict["occupation"] = "Engineer" #country and occupation are keys and USA and Engineer are their values respectively.

Q7. What is the purpose of the id() function in Python?

Ans. The id() function in Python is used to get the identity (memory address) of an object. Each object in Python has a unique identity, which is essentially the memory address where the object is stored in the computer's memory.

E.g, x = 5

print(id(x))

Uses:

* Object identity comparison: We can use id() to compare whether two variables reference the same object in memory.
* Debugging: It can be helpful for debugging to know the memory address of an object.
* Understanding object behavior: Understanding how objects are stored in memory can help you understand certain behaviors in Python, especially when dealing with mutability and immutability.

Q8. How do you remove an element from a list in Python?

Ans. we can remove an element from list in various ways:

* Using the remove() method: This method removes the first occurrence of a specified value from the list.

E.g x=[2,5,7,6,5,8,7]

x.remove(7). Remove() takes the element.

* Using the pop() method: This method removes and returns the element at a specified index. If no index is specified, it removes and returns the last element.

e.g,

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

removed\_element = lst.pop(2)

* Using the del statement: This statement removes an element at a specified index or a slice of elements.

e.g, lst = [1, 2, 3, 4, 5]

del lst[2]

* Using list comprehension: You can create a new list that excludes the elements you want to remove.

e.g, my\_list = [1, 2, 3, 4, 5]

my\_list = [x for x in my\_list if x != 3]

Q9. What is the purpose of the len() function in Python?

Ans.

The len() function in Python is used to get the length (the number of elements) of various objects, such as strings, lists, tuples, dictionaries, and sets. It returns the number of items in an object. The len() function is commonly used in situations where we need to determine the size or length of a data structure,

* when iterating over elements
* checking for empty objects,
* performing operations that depend on the size of the object.

E.g, lst = [1, 2, 3, 4, 5]

print(len(lst))

Q10. How do you check if a variable is of a certain data type in Python?

Ans.

* If you have a single parameter, the Python typeof function will return the type of that parameter.e.g, x=”hello”

print(type(x)) #Gives output as class <str>

* To check if a variable is of a certain data type in Python using the isinstance() function. This function takes two arguments: the variable you want to check and the data type you want to check against. It returns True if the variable is of the specified data type, and False otherwise.

e.g, x= "hello"

print(isinstance(x, str)) #returns True

Q11. What is a set in Python? How is it different from a list?

Ans.

* In Python, a set is an unordered collection of unique elements.
* Sets are mutable, meaning you can add or remove elements from them.
* But they don't support indexing or slicing like lists do.
* Sets do not allow duplicate elements; if you try to add an element that already exists in the set, it won't be added again.

|  |  |
| --- | --- |
| LIST | SETS |
| Ordered | unordered |
| Duplicates element allowed | Unique only allowed |
| Supports indexing and slicing | Doesnt support |
| Implementation Linkedlist and ArrayList | Implantation in Hashsets,treesets |

Q12. How can you convert a list to a tuple in Python?

Ans. Conversion can be done using tuple() constructor. he tuple() constructor takes an iterable (like a list) as an argument and returns a new tuple containing the same elements as the iterable.

e.g, x\_l= [1, 2, 3, 4, 5]

x\_t= tuple(x)

print(x\_t) # Output: (1, 2, 3, 4, 5)

Q13. How can you convert a float to an integer in Python?

Ans.We can convert a float to an integer in Python using the int() function. This function truncates the decimal part of the float and returns the integer part. int() always rounds towards zero. So, , and

* if the float is positive, it rounds down

my\_float = 3.9

my\_integer = int(my\_float)

print(my\_integer) # Output: 3

* if the float is negative, it rounds up.

my\_float = -3.9

my\_integer = int(my\_float)

print(my\_integer) # Output: -3