Variable_I/O_DataTypes

Assignment Solution

PW SKILLS

1. **Which of the following identifier names are invalid and why?**
   a. **Serial_no.**
   b. **1st_Room**
   c. **Hundred$**
   d. **Total_Marks**
   e. **total-Marks**
   f. **Total Marks**
   g. **True**
   h. **_Percentag**

Solution:

Invalid identifier names and reasons:

- **1st_Room:** Identifiers cannot start with a digit.
- **Hundred$:** Identifiers cannot contain special characters other than underscore (_).
- **total-Marks:** Identifiers cannot contain hyphens (-).
- **Total Marks:** identifier cannot contain space.
- **True:** Identifiers cannot contain spaces or the reserved keyword True.

Valid identifier names:

- **Serial_no:** Contains an underscore character, which is allowed.
- **Total_Marks:** Contains an underscore character, which is allowed.
- **_Percentag:** Starts with an underscore character, which is allowed.

2. **Write the corresponding Python assignment statements:**
   a. **Assign 10 to variable length and 20 to variable breadth.**
   b. **Assign the average of values of variables length and breadth to a variable sum.**
   c. **Assign a list containing strings 'Paper', 'Gel Pen', and 'Eraser' to a variable stationery.**
   d. **Assign the strings 'Mohandas', 'Karamchand', and 'Gandhi' to variables first, middle and last.**
   e. **Assign the concatenated value of string variables first, middle, and last to variable fullname.**

Solution:

```
a)
length = 10
breadth = 20

b) sum = (length + breadth) / 2

c) stationery = ['Paper', 'Gel Pen', 'Eraser']
d)
first = 'Mohandas'
middle = 'Karamchand'
last = 'Gandhi'

e) fullname = first + ' ' + middle + ' ' + last
```

3. What is the difference between static and dynamic variables in Python?

Solution:

Static variables are declared inside a function but outside any nested function. They are initialized when the function is defined and retain their value throughout the lifetime of the program. They are shared among all calls to the function.

Dynamic variables are declared inside a function but within a nested function. They are initialized when the nested function is called and are destroyed when the nested function exits. They are not shared among different calls to the nested function.

Key differences:

- **Scope:** Static variables have a function-level scope, while dynamic variables have a nested function-level scope.
- **Initialization:** Static variables are initialized when the function is defined, while dynamic variables are initialized when the nested function is called.
- **Lifetime:** Static variables persist throughout the lifetime of the program, while dynamic variables are destroyed when the nested function exits.
- **Sharing:** Static variables are shared among all calls to the function, while dynamic variables are not shared among different calls to the nested function.

Example:

```
Example:


def outer_function():
    static_variable = 10  # Static variable

    def inner_function():
        dynamic_variable = 20  # Dynamic variable

    inner_function()

outer_function()
```

In this example, static_variable is a static variable because it is declared outside the inner_function. It will be initialized to 10 when the outer_function is defined and will retain its value throughout the lifetime of the program.

dynamic_variable is a dynamic variable because it is declared inside the inner_function. It will be initialized to 20 when the inner_function is called and will be destroyed when the inner_function exits.

Note that dynamic variables can access static variables, but not vice versa.

4.
name = ["Mohan", "dash", "karam", "chandra","gandhi","Bapu"]
do the following operations in this list;

a. add an element "freedom_fighter" in this list at the 0th index.

b. find the output of the following ,and explain how?

```
name = [“freedomFighter”,"Bapuji","MOhan" "dash", "karam",
"chandra","gandhi"]
length1=len((name[-len(name)+1:-1:2]))
length2=len((name[-len(name)+1:-1]))
print(length1+length2)
```

c. add two more elements in the name ["NetaJi","Bose"] at the end of the list.

d. what will be the value of temp:

name = ["Bapuji", "dash", "karam", "chandra","gandi","Mohan"]
temp=name[-1]
name[-1]=name[0]
name[0]=temp
print(name)

Solution:

a) Add a element "freedom_fighter" in this list at 0th index.

```
name.insert(0, "freedom_fighter")
print(name)
Output:
['freedom_fighter', 'Mohan', 'dash', 'karam', 'chandra', 'gandhi',
'Bapu']
```

b) name.insert(0,"freedomFighter")

```
name=["freedomFighter","Bapuji","MOhan","dash","karam","chandra","g
andhi"]
length1=len((name[-len(name)+1:-1:2]))
length2=len((name[-len(name)+1:-1]))
print((name[-len(name)+1:-1:2]))
print((name[-len(name)+1:-1]))
print(length1+length2)
```

Let's break down the code step by step:

1. `name = ["Bapuji", "MOhan", "dash", "karam", "chandra", "gandhi"]`: This line initializes a list named `name` with six elements.

2. `name.insert(0, "freedomFighter")`: This line inserts the string "freedomFighter" at the beginning of the list `name`.

3. `length1 = len((name[-len(name)+1:-1:2]))`: This line calculates the length of a sublist of `name`. It starts from the second element (index -len(name)+1) to the second-to-last element (index -1), with a step size of 2. It stores the length of this sublist in the variable `length1`.

4. `length2 = len((name[-len(name)+1:-1]))`: This line calculates the length of another sublist of `name`. It starts from the second element (index -len(name)+1) to the second-to-last element (index -1), without specifying a step size. It stores the length of this sublist in the variable `length2`.

5. `print((name[-len(name)+1:-1:2]))`: This line prints the sublist of `name` starting from the second element to the second-to-last element, with a step size of 2.

6. `print((name[-len(name)+1:-1]))`: This line prints the sublist of `name` starting from the second element to the second-to-last element, without specifying a step size.

7. `print(length1 + length2)`: This line prints the sum of `length1` and `length2`.
Given the output:
- The sublist `['Bapuji', 'karam']` is printed, containing elements at even indices starting from the second position.
- The sublist `['Bapuji', 'MOhandash', 'karam', 'chandra']` is printed, containing elements from the second position to the second-to-last position without any step.
- The sum of the lengths of these two sublists is `6`, as both sublists contain a total of 4 elements (`length1` = 2, `length2` = 4), which sums up to 6.

c).Add two more elements in the name ["NetaJi","Bose"] in end of list.

```
name.extend(["NetaJi", "Bose"])
print(name)
```

d)What will be the value of temp?

```
name = ["Bapuji", "dash", "karam", "chandra", "gandhi", "Mohan"]
temp = name[-1]
name[-1] = name[0]
name[0] = temp
print(name)
```

Output:

['Mohan', 'dash', 'karam', 'chandra', 'gandhi', 'Bapuji']

This code swaps the first and last elements of the list. First, it stores the last element in the temporary variable temp. Then, it assigns the first element to the last element. Finally, it assigns temp to the first element.

5. Find the output of the following.

```
animal = ['Human','cat','mat','cat','rat','Human', 'Lion']
print(animal.count('Human'))
print(animal.index('rat'))
print(len(animal))
```

Solution:

```
animal = ['Human','cat','mat','cat','rat','Human', 'Lion']
print(animal.count('Human'))
print(animal.index('rat'))
print(len(animal))
```

✓ 0.0s

```
2
4
7
```

6. tuple1=(10,20,"Apple",3.4,'a',["master","ji"],("sita","geeta",22),[{"roll_no":1},{"name":"Navneet"}])

a. print(len(tuple1))

b. print(tuple1[-1][-1]["name"])

c. c)fetch the value of roll_no from this tuple.

d. d)print(tuple1[-3][1])

e. fetch the element "22" from this tuple.

Solution:

a)&b)

```
tuple1=(10,20,"Apple",3.4,'a',["master","ji"],("sita","geeta",22),[{"roll_no":1},{"name":"Navneet"}])

print(len(tuple1))
print(tuple1[-1][-1]["name"])
```

✓ 0.0s

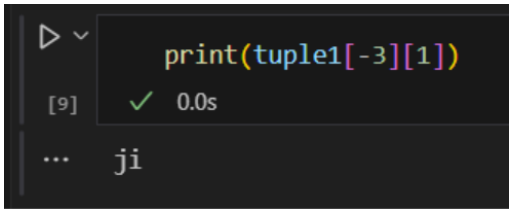
```
8
Navneet
```

```
print(tuple1[-1][0]["roll_no"])
```

[32] ✓ 0.0s

```
1
```

d)



e).

```
tuple1[6][-1]
```

7. What are the three main types of number data types in Python?

Solution:

- int (integer)
- float (floating-point number)
- complex (complex number)

8. Give examples of each of the numeric data types mentioned.

Solution:

- int: 10, -25, 0
- float: 3.14, -9.8, 1.23e5
- complex: 1+2j, 3-4j, 0+1j

9. How is the boolean data type related to the integer data type in Python?

Solution:

The boolean data type is a subtype of the integer data type. Boolean True is equivalent to the integer value 1, and boolean False is equivalent to the integer value 0.

10. What is the special significance of the boolean values True and False?

Solution:

Boolean values are used in conditional statements and other logical operations. True represents a true condition or a non-zero value, while False represents a false condition or a zero value.

11. Explain the concept of sequence data types in Python.

Solution:

Sequence data types are ordered collections of items. They include strings, lists, and tuples. Each item in a sequence has an index, and sequences can be accessed and modified using these indexes.

12. What is the primary difference between a list and a tuple in Python?

Solution:

Lists are mutable, meaning their elements can be added, removed, or modified. Tuples, on the other hand, are immutable, meaning their elements cannot be changed once they are created.

13. What is a set in Python, and what distinguishes it from a list?

Solution:

A set is an unordered collection of unique items. It is similar to a list, but it does not allow duplicate elements. Sets are useful for removing duplicates from a list or finding the intersection or union of two sets.

14. Explain the purpose of using the None data type in Python.

Solution:

The None data type represents the absence of a value. It is used as a placeholder or to indicate that a variable has not yet been assigned a value.

15. How is a dictionary different from other data types in Python?

Solution:

Dictionaries are mapping data types that store data in key-value pairs. They are unordered, meaning the order of the key-value pairs is not guaranteed. Dictionaries allow for fast access to data based on the keys.

16. Describe the syntax for creating a dictionary in Python.

Solution:

A dictionary is created using curly braces {}. Key-value pairs are separated by colons (:), and multiple key-value pairs are separated by commas (,).

```
my_dict = {"name": "John", "age": 30, "city": "New York"}
```

17. What is the significance of keys and values in a dictionary?

Solution:

Keys are unique identifiers used to access the corresponding values in a dictionary. Values are the data associated with the keys.

18. Explain the purpose of "pop","popitem","clear()" in a dictionary with suitable example.

Solution:

- **pop(key):** Removes the key-value pair with the specified key from the dictionary and returns the value.
- **popitem():** Removes and returns a random key-value pair from the dictionary.
- **clear():** Removes all key-value pairs from the dictionary, leaving it empty.

my_dict = {"name": "John", "age": 30, "city": "New York"}

```
#Remove the "age" key-value pair
removed_value = my_dict.pop("age")
print(removed_value)  # Output: 30
#Remove a random key-value pair

removed_key, removed_value = my_dict.popitem()
print(removed_key)  # Output: "name"
print(removed_value)  # Output: "John"

#Clear the dictionary

my_dict.clear()
print(my_dict)  # Output: {}
```

19. Difference between discord and remove.

Solution:

Both discard() and remove() methods are used to remove elements from a set. However, there is a subtle difference between the two:

- **discard():** Removes the specified element from the set if it exists. If the element is not present, it does nothing.
- **remove():** Removes the specified element from the set. If the element is not present, it raises a KeyError exception.

20. tWhat do you mean by FrozenSet? Explain it with suitable example.

Solution:

A frozenset is an immutable set. Once created, its elements cannot be added, removed, or modified. Frozen sets are useful when you want to create a set that will not change over time.

```
my_frozenset = frozenset([1, 2, 3, 4, 5])
#Attempt to add an element to the frozenset

my_frozenset.add(6)  # Raises an AttributeError: 'frozenset' object has no attribute 'add'
```

21. **Differentiate between mutable and immutable data types in Python and give examples of mutable and immutable data types.**

Solution:

- **Mutable data types:** Can be modified after creation. Examples: list, dictionary, set
- **Immutable data types:** Cannot be modified after creation. Examples: int, float, string, tuple, frozenset