Python: Data Structures and Sequences

Want to Learn Python, Join our WhatsApp Channel:
https://whatsapp.com/channel/0029VaeGV0517En4iyZGWn2P

1. Tuple in Python

- In python, a tuple is an immutable sequence of elements. it is similar to a list, but the elements of a tuple cannot be modified once they are created.
- Tuple is a collection data type in python. It is useful for storing multiple related values as a single unit.
- Sequence types in python - list, tuple and range

Creating a Tuple in Python

A tuple is created by enclosing elements within parentheses () and separating them with commas. While parentheses are technically optional, it's generally considered best practice to use them for clarity and consistency.

- video: How to create a tuple in Python

Example

Some common ways to create tuples in Python include:

```
tup = (1,2,3)
print(tup) # Output: (1, 2, 3)
# check the type of variable
print(type(tup)) # Output: <class 'tuple'>

# another example to create tuple
tup1 = 4,5,6
print(tup1) # Output: (4, 5, 6)
```

```
# tuple with mixed datatypes
tup_mixed = (7, "String", 7.8)
print(tup_mixed)

# Tuples may be nested
nested_tup = tup1, (7,8)
print(nested_tup) #Output: ((4, 5,6), (7, 8))

# using the 'tuple()' function
tup3 = ([7,2,9])
print(tup3) #Output (7,2,9)

tup4 = tuple('string')
print(tup4) # Output: ('s','t','r','i','n','g')
```

A nested tuple is a tuple that contains one or more tuples as element.

Empty and single item tuple

- A special problem is the construction of tuples containing 0 or 1 item.
- Empty tuples are constructed by an empty pair of parentheses
- A tuple with one item is constructed by following a value with a comma (it is not sufficient to enclose a single value in parentheses.
- Example: How to create an Empty tuple and Single value tuple

Accessing the elements of Tuple

- Example #1: Learn how to print elements of a tuple in Python using while loop
- Example #2: How to: Access Tuple Items in Python
- Example #3: How to Calculate the Sum of a Tuple Using a For Loop

Unpacking tuples

Tuple unpacking allows you to assign the values of a tuple to multiple variables in a single step. Each element of the tuple is assigned to a corresponding variable.

- Example #1: Unpacking a Tuple in Python
- Example #2: How to Swap Variables in One Line of Code using Tuple Unpacking

Example:

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

print(a)  # Output: 1
```

```
print(b)  # Output: 2
print(c)  # Output: 3
```

- The number of variables on the left side must match the number of elements in the tuple, or you'll get a `ValueError`.

Example #2

```
# Tuples are immutable
# but they can contain mutable objects
```

Tuple methods

List

- A **list** in Python is one of the most commonly used data structures. It allows you to store a collection of items (which can be of different types) in a single variable. Lists are very flexible and easy to use, making them a great tool for beginners to understand.

Key Characteristics of Python Lists:

1. **Ordered**: The items in a list have a specific order, and this order will not change unless explicitly modified.
2. **Mutable**: You can change, add, or remove items after the list has been created.
3. **Heterogeneous**: A list can contain different data types, such as integers, strings, and even other lists.
4. **Indexed**: Each item in a list has an index, starting from `0` for the first item.

- video: How to use list in Python

Creating a List

You can create a list by placing items inside square brackets `[]`, separated by commas.

```
# A list of integers
numbers = [1, 2, 3, 4, 5]

# A list of strings
fruits = ["apple", "banana", "cherry"]

# A list of mixed data types
mixed_list = [1, "hello", 3.14, True]

# An empty list
empty_list = []
```

Accessing Elements in a List

You can access individual elements in a list using their index.

```
# Access the first element (index 0)
print(fruits[0])  # Output: apple

# Access the second element (index 1)
print(fruits[1])  # Output: banana

# Access the last element (index -1)
print(fruits[-1])  # Output: cherry
```

Modifying Elements in a List

Since lists are mutable, you can change an element in a list by assigning a new value to a specific index.

```
# Change the first element of the list
fruits[0] = "orange"
print(fruits)  # Output: ['orange', 'banana', 'cherry']
```

Adding Elements to a List

You can add elements to a list using methods like `append()` or `insert()`.

```
# Append an element to the end of the list
fruits.append("grape")
print(fruits)  # Output: ['orange', 'banana', 'cherry', 'grape']

# Insert an element at a specific position
fruits.insert(1, "mango")
print(fruits)  # Output: ['orange', 'mango', 'banana', 'cherry', 'grape']
```

Removing Elements from a List

Elements can be removed from a list using methods like `remove()`, `pop()`, or `del`.

```
# Remove a specific element by value
fruits.remove("banana")
print(fruits)  # Output: ['orange', 'mango', 'cherry', 'grape']

# Remove an element by index using pop
fruits.pop(2)
print(fruits)  # Output: ['orange', 'mango', 'grape']

# Remove an element by index using del
del fruits[0]
print(fruits)  # Output: ['mango', 'grape']
```

Slicing Lists

You can access a range of elements from a list using slicing.

```
# Get the first two elements
print(fruits[:2])  # Output: ['mango', 'grape']

# Get elements from the second to the end
print(fruits[1:])  # Output: ['grape']
```

Iterating Through a List

You can use a loop to iterate through all the elements in a list.

```
# Print each fruit in the list
for fruit in fruits:
    print(fruit)

# Output:
# mango
# grape
```

List Methods

Python lists come with many useful methods, such as:

- `append()`: Adds an element to the end of the list.
- `extend()`: Adds all elements of another list to the end.
- `insert()`: Inserts an element at a specified position.
- `remove()`: Removes the first occurrence of an element.
- `pop()`: Removes and returns an element at a specified position.
- `sort()`: Sorts the list in ascending order.
- `reverse()`: Reverses the order of elements in the list.

Example: Using Lists in a Simple Program

Here's a simple example to illustrate the use of lists in a practical scenario:

```
# Creating a shopping list
shopping_list = ["milk", "eggs", "bread"]

# Adding items to the list
shopping_list.append("butter")
shopping_list.append("apples")
```

```
# Removing an item
shopping_list.remove("eggs")

# Printing the final list
print("Final shopping list:", shopping_list)

# Output:
# Final shopping list: ['milk', 'bread', 'butter', 'apples']
```

- Video: 6 Ways to use List in For loop in Python
- Video: Read data from list using For loops in Python
- Python List Slicing
- Python Nested List
- How to modify a list by replacing multiple elements with a single element
- Adding and Removing Elements from a Python List
- Check if Data Structure is Empty Using 'not' Operator
- Remove duplicate elements from a list
- List Index Function: Find the Index of an Element in a List
- List pop() Method
- **Python set examples:**
- How to Find Duplicates in a List using Set and List Functions

Introduction to Python Dictionaries: Concepts, Usage, and Examples

In Python, a **dictionary** is a collection of key-value pairs. Each key in a dictionary is unique, and it is associated with a value. Dictionaries are used to store data values like a map or a real-life dictionary where each word (key) has a definition (value). They are mutable, meaning you can change, add, or remove items after the dictionary is created.

Creating a Dictionary

You create a dictionary using curly braces `{}` with keys and values separated by a colon `:`. Multiple key-value pairs are separated by commas.

```
# Example of a dictionary
student = {
    "name": "John",
    "age": 20,
    "courses": ["Math", "Science"]
}
```

Accessing Values

You can access the value associated with a specific key by using square brackets `[]` or the `get()` method.

```
# Accessing values
print(student["name"])  # Output: John
```

```
print(student.get("age"))  # Output: 20
```

Adding or Updating Elements

You can add a new key-value pair or update an existing one by assigning a value to the key.

```
# Adding a new key-value pair
student["grade"] = "A"

# Updating an existing value
student["age"] = 21
```

Removing Elements

You can remove elements using the `pop()` method or `del` keyword.

```
# Removing a specific key-value pair
student.pop("grade")

# Using del keyword
del student["age"]
```

Looping Through a Dictionary

You can loop through keys, values, or both using a for-loop.

```
# Looping through keys
for key in student:
    print(key)

# Looping through values
for value in student.values():
    print(value)

# Looping through key-value pairs
for key, value in student.items():
    print(f"{key}: {value}")
```

Example

Here's a full example demonstrating how to use a dictionary:

```
# Creating a dictionary
student = {
```

```
    "name": "John",
    "age": 20,
    "courses": ["Math", "Science"]
}

# Accessing and updating data
print(student["name"])  # Output: John
student["age"] = 21
print(student["age"])  # Output: 21

# Adding a new key-value pair
student["grade"] = "A"

# Removing a key-value pair
student.pop("courses")

# Looping through dictionary
for key, value in student.items():
    print(f"{key}: {value}")
```

Output

```
name: John
age: 21
grade: A
```

Key Points:

- **Dictionaries** store data as key-value pairs.
- Keys are unique, while values can be of any data type.
- Dictionaries are mutable and can be modified after creation.

This makes dictionaries a powerful tool for organizing and accessing data in Python!

- How to Print a Dictionary
- Python Dictionary with For Loop
- How to Merge Dictionaries with the | Operator
- dictionary copy() method

Sets

In Python, a **set** is an unordered collection of unique elements, meaning no duplicates are allowed. Sets are useful when you want to store multiple items but don't need to keep them in a particular order, and you want to ensure that each item only appears once.

Key Points about Sets:

- **Unordered:** The elements in a set do not have a specific order. For more details, see Appendix A
- **Unique:** Sets automatically remove any duplicate items.

- **Mutable:** You can add or remove elements from a set.
- **Immutable Elements:** The items in a set must be immutable (e.g., numbers, strings, or tuples).

Creating a Set

You can create a set using curly braces `{}` or the `set()` function.

```
# Creating a set using curly braces
my_set = {1, 2, 3, 4, 5}

# Creating a set using the set() function
my_set = set([1, 2, 3, 4, 5])
```

Example:

```
# Creating a set
fruits = {"apple", "banana", "cherry", "apple"}

# Displaying the set
print(fruits)  # Output: {'apple', 'banana', 'cherry'}
```

Notice how `apple` only appears once, even though we tried to add it twice.

Common Operations with Sets

1. **Adding Elements:** Use the `add()` method to add an item to a set.

   ```
   fruits = {"apple", "banana"}
   fruits.add("cherry")
   print(fruits)  # Output: {'apple', 'banana', 'cherry'}
   ```

2. **Removing Elements:** Use `remove()` or `discard()` to remove an item.

   ```
   fruits.remove("banana")
   print(fruits)  # Output: {'apple', 'cherry'}
   ```

3. **Set Operations:** Sets support mathematical operations like **union**, **intersection**, and **difference**.

   - **Union (`|`)**: Combines elements from both sets.
   - **Intersection (`&`)**: Finds common elements between sets.
   - **Difference (`-`)**: Finds elements in one set but not the other.

```
set1 = {1, 2, 3}
set2 = {3, 4, 5}

# Union
print(set1 | set2)  # Output: {1, 2, 3, 4, 5}

# Intersection
print(set1 & set2)  # Output: {3}

# Difference
print(set1 - set2)  # Output: {1, 2}
```

When to Use Sets?

- When you want to eliminate duplicates from a list.
- When you need to perform operations like union and intersection.
- When the order of elements doesn't matter.
- How to: Add or Remove Elements in a Set
- How to: Create Empty Set in Python
- Find the Union of Two Sets in Python

Why Integers, Strings, and Tuples Are Immutable in Python [1]

Integer (Immutable)

Integers are numbers without any fractional part. In Python, integers are immutable, meaning their value cannot be changed once they are created.

Example:

```
x = 10
print(x)  # Output: 10

x = 20  # This creates a new integer object and binds x to it
print(x)  # Output: 20
```

In this example, when `x` is reassigned from 10 to 20, a new integer object is created, and `x` is updated to reference the new object.

String (Immutable)

Strings are sequences of characters. In Python, strings are also immutable. Any operation that modifies a string will create a new string rather than altering the existing one.

Example:

```
s = "hello"
print(s)  # Output: hello

s = s + " world"  # This creates a new string object
print(s)  # Output: hello world
```

Here, concatenating `" world"` to `s` does not change the original string `"hello"`. Instead, a new string `"hello world"` is created and assigned to `s`.

Tuple (Immutable)

Tuples are ordered collections of elements. Like integers and strings, tuples are immutable. Once a tuple is created, you cannot change its contents.

Example:

```
t = (1, 2, 3)
print(t)  # Output: (1, 2, 3)

# Attempting to modify the tuple will raise an error
try:
    t[0] = 4
except TypeError as e:
    print(e)  # Output: 'tuple' object does not support item assignment

# You can create a new tuple
t = (4, 5, 6)
print(t)  # Output: (4, 5, 6)
```

In this example, trying to change the first element of `t` results in a `TypeError` because tuples are immutable. To change the contents, a new tuple must be created.

These examples illustrate that integers, strings, and tuples in Python are immutable, meaning their values cannot be changed after they are created.

Key Terms

True/False (Mark T for True and F for False)

Answer Key (True/False):

Multiple Choice (Select the best answer)

Tuple (MCQs)

1. **What is the output of the following code?** [Python Quiz #15]

```
a = ('34.5')
print(type(a))
```

- A) <class 'list'>
- B) <class 'tuple'>
- C) <class 'int'>
- D) <class 'str'>

Watch the video for the answer: https://youtube.com/shorts/uMtHVgPSymw

2. **What is the output of the following code?** [Python Quiz #46]

```
tup2= 3,4,5

nested_tup = tup2, (6,7)
print(nested_tup)
```

- A) ((3, 4, 5), (6, 7))
- B) [3, 4, 5, 6, 7]
- C) {(3, 4, 5), (6, 7)}
- D) (3, 4, 5, 6, 7)

Watch the video for the answer: Learn 5 Easy Ways to Create Tuples in Python

3. **What is the output of the following code?** [Python Quiz #47]

```
tup4 = tuple('python')
print(tup4)
```

```
- A) 'python'
- B) ['python']
- C) ('python')
- D) ('p', 'y', 't', 'h', 'o', 'n')
```

Watch the video for the answer: Learn 5 Easy Ways to Create Tuples in Python

List (MCQs)

What is the output of the following code? [Python Quiz #18]

```
a = [1, 2, 3]
b = [4, 5, 6]
c = a + b
```

```
print(c)
```

- A) [1, 2, 3]
- B) [4, 5, 6]
- C) [1, 2, 3, 4, 5, 6]
- D) Error

Watch the video for the answer: https://youtube.com/shorts/rEDmm9ry7wE?si=ce2iYVXHbCEjLm6W

What is the output of the following code? [Python Quiz #19]

```
a = [1, 2, 3, 4]
b = a
b[0] = 0
print(a)
```

- A) [0, 2, 3, 4]
- B) [1, 2, 3, 4]
- C) [0, 1, 2, 3, 4]
- D) [1, 0, 3, 4]

Watch the video for the answer: https://youtu.be/ZWB4dfUYz1k?si=aYNoLd_81_-9oLfR

What is the output of the following code? [Python Quiz #20]

```
my_list = [1, 2, 3, 4, 5]
print(my_list[1::2])
```

- **A)** [1, 3, 5]
- **B)** [2, 4]
- **C)** [2, 4, 5]
- **D)** [1, 3]

Watch the video for the answer: https://youtube.com/shorts/UH5znVEFehI

What is the output of the following code? [Python Quiz #21]

```
my_list = [8, 9, 11, 12]
print(my_list[1:-1])
```

- A) [8, 9, 11, 12]
- B) [9, 11, 12]
- C) [9, 11]
- D) [8, 9, 11]

Watch the video for the answer: https://youtube.com/shorts/PBBnTGfFm4o?si=Z1TlMu24412nVKG_

What is the output of the following code? [Python Quiz #26]

```
my_list = [
    "apple",
    "banana",
    "cherry"
]

print(len(my_list))
```

Options:

- A) 6
- B) 3
- C) [apple, banana, cherry]
- D) Error

Watch the video for the answer: https://youtube.com/shorts/t7SVFVPrPlk?si=h9nAfaOPofOLnwu4

What is the output of the following code? [Python Quiz #32] Code Explanation:

```
a = [1, 2, 3]  # Creates a list named 'a' with elements 1, 2, and 3
b = a.copy()   # Creates a copy of list 'a' and assigns it to 'b'
a[0] = 4       # Modifies the first element of list 'a' to 4
print(b)       # Prints the contents of list 'b'
```

- A) [4, 2, 3]
- B) [1, 2, 3]
- C) [2, 4, 3]
- D) Error

Watch the video for the answer: https://youtube.com/shorts/Jub8TgDntRQ?si=wt8lPaFgr4BMEk7E

What is the output of the following code? Python Quiz #34

```
a = [1, 2, 3]
b = a[:]
b[0] = 4
print(a)
```

- A) [1, 2, 3]
- B) [4, 2, 3]
- C) [4, 4, 3]
- D) [4, 2, 3]

Watch the video for the answer: https://youtube.com/shorts/rslioE6VWOQ

What is the output of the following code? [Python Quiz #35] Code:

```
my_list = [1, 2, 3, 4, 5]
for i in range(len(my_list)):
    my_list[i] *= 2
print(my_list)
```

Options:

- A) [1, 2, 3, 4, 5]
- B) [2, 4, 6, 8, 10]
- C) [1, 4, 9, 16, 25]
- D) None

Watch the video for the answer: https://youtube.com/shorts/QffZTQasQSs?si=6ZW4auXECcvTGQgn

Dictionary (MCQs)

What is the output of the following code? [Python Quiz #25]

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

for key in my_dict:
    print(my_dict[key])
```

```
- A) abc
- B) 123
- C) {1, 2, 3}
- D) {a, b, c}
```

Watch the video for the answer: https://youtube.com/shorts/wofaOXA0SVA?si=EY4-_ndR8_qbB6zF

What is the output of the following code? [Python Quiz #65]

```
my_dict = {"name": "Alice", "age": 30}
print(my_dict["city"])
```

What will happen when you run this code?

```
- A) It will print `None`.
- B) It will print an empty string `""`.
- C) It will raise a `KeyError`.
- D) It will print `city`.
```

Correct Answer:

C. It will raise a `KeyError`.

Set (MCQs)

What is the output of the following code?

```
my_set = {1, 2, 3}
result = my_set.add(4)

print(result)
```

```
- A) Error
- B) None
- C) {1, 2, 3, 4}
- D) {4}
```

What is the data type of the following value? {1, 2, 3} a) list b) tuple c) set d) dict

Answer: c) set

What is the data type of the following value? {1, 2, 3} a) list b) tuple c) set d) dict

Answer: c) set

What is the data type of the following value? (1, 2, 3) a) list b) tuple c) set d) dict

Answer: b) tuple

What is the data type of the following value? {"name": "Alice", "age": 25} a) list b) tuple c) set d) dict

Answer: d) dict

What is the data type of the following value? [1, 2, 3] a) list b) tuple c) set d) dict

Answer: a) list

What is the correct way to create a set in Python?

a) set = [1, 2, 3] b) set = (1, 2 ,3) c) set = {1, 2 ,3} d) set = <1, 2 ,3>

Answer: c) set = {1, 2 ,3}

#58 What is the correct way to create a loop in Python?

a) for i in range(10): b) while i < 10: c) repeat i = 0 to 9: d) either a or b

Answer: d) either a or b

#53 What is the correct way to create a dictionary in Python?

a) dict = [key: value] b) dict = (key: value) c) dict = {key: value} d) dict = <key: value>

Answer: c) dict = {key: value}

#52 What is the correct way to create a list in Python?

a) list = [1, 2, 3] b) list = (1, 2, 3) c) list = {1, 2, 3} d) list = <1, 2, 3>

Watch this video for the answer:

#47 In Python, What is the output of the following code?

a = [1, 2, 3, 4] b = [5, 7, 4, 9,2] print (a[-1] in b)

Follow me https://www.facebook.com/yasirbhutta786

True False Error None

#35 What is the output of the following code?

def fun(arr): arr = arr[::-1]

arr = [1,2,3,4,5] fun(arr) print(arr)

Watch the Video Tutorial for the Answer: https://youtube.com/shorts/Ix-h6WR-vQM?feature=share

#python #pythonpoll #MCQsTest #yasirbhutta

[1,2,3,4,5] [5,4,3,2,1] [] Error

Answer: A.

#24 What is the syntax to create a union of two sets in Python?

Watch the Video Tutorial for the Answer: https://youtu.be/YDyCNYCUK9A

#python #pythonpoll #MCQsTest #yasirbhutta

a. set1 + set2 b. set1.union(set2) c. set1 & set2 d. set1.merge(set2)

Answer: b. set1.union(set2)

#26 What is the result of the following code?

set1 = {1, 2, 3} set2 = {3, 4, 5} set3 = set1.union(set2) print(set3)

Watch the Video Tutorial for the Answer: https://youtu.be/YDyCNYCUK9A

#python #pythonpoll #MCQsTest #yasirbhutta

a. {1, 2, 3} b. {3, 4, 5} c. {1, 2, 3, 4, 5} d. {3} Answer: c. {1, 2, 3, 4, 5}

Is it possible to use the union operator | to combine sets with different data types? a. Yes b. No Answer: b. No

#25 Can the union of two sets contain duplicates?

Watch the Video Tutorial for the Answer: https://youtu.be/YDyCNYCUK9A

#python #pythonpoll #MCQsTest #yasirbhutta

a. Yes b. No Answer: b. No

#27 What happens when you try to add an element to a set that already exists in the set?

Watch the Video Tutorial for the Answer: https://youtu.be/YDyCNYCUK9A

#python #pythonpoll #MCQsTest #yasirbhutta

a. A TypeError occurs b. The element is added as a duplicate c. Nothing happens, the element is not added

Answer: c. Nothing happens, the element is not added.

What is the output of the following code?

list1 = [10,20,10,'10',20] set1 = set(list1) print(set1)

5 3 2 4 Answer: B) 3

What is the output of the following code?

set1 = {1,2,3} set2 = {1,2,3} print(*(set1+set2))

Error {2,4,6} [2,4,6] [1,2,3]

Answer: A) error + operator is not supported for set

What does the "in" operator do in Python? a) check if a value is present in a list b) check if a value is equal to another value c) check if a variable is defined Answer: a) check if a value is present in a list

Can the "in" operator be used with dictionaries in Python? a) yes b) no Answer: a) yes

How would you check if the value "dog" is in the list ['cat', 'dog', 'elephant'] using the "in" operator? a) 'dog' in ['cat', 'dog', 'elephant'] b) ['dog'] in ['cat', 'dog', 'elephant'] Answer: a) 'dog' in ['cat', 'dog', 'elephant']

Can the "in" operator be used to check if an element is present in a set in Python? a) yes b) no Answer: a) yes

How would you check if the value "dog" is in the set {'cat', 'dog', 'elephant'} using the "in" operator? a) 'dog' in {'cat', 'dog', 'elephant'} b) {'dog'} in {'cat', 'dog', 'elephant'} Answer: a) 'dog' in {'cat', 'dog', 'elephant'}

#14 Is the "in" operator faster for checking if an element is present in a set compared to a list in Python?

https://lucasmagnum.medium.com/pythontip-list-vs-set-performance-experiments-dfbe4f72d47f

a) yes b) no Answer: a) yes

#13 What does the "in" operator do in Python?

a) check if a value is present in a list b) check if a value is equal to another value c) check if a variable is defined Answer: a) check if a value is present in a list

Can the "in" operator be used with dictionaries in Python?

a) yes b) no Answer: a) yes

How would you check if the value "dog" is in the list ['cat', 'dog', 'elephant'] using the "in" operator?

a) 'dog' in ['cat', 'dog', 'elephant'] b) ['dog'] in ['cat', 'dog', 'elephant'] Answer: a) 'dog' in ['cat', 'dog', 'elephant']

What is the operator used to check if an element is present in a set in python?

in contains has exit

Answer: A) in

What is the output of the following code?

```
t=(1,2,4,3) t[1:3] a) (1, 2) b) (1, 2, 4) c) (2, 4) d) (2, 4, 3)
```

#8 What is the output of the following code?

z=set('stri$ng') print('r' in z)

Python YouTube Playlist: https://www.youtube.com/playlist?list=PLKYRx0Ibk7Vi-CC7ik98qT0VKK0F7ikja

a) Error b) True c) False d) No output

Which of the following is a Python tuple?

Python YouTube Playlist: https://www.youtube.com/playlist?list=PLKYRx0Ibk7Vi-CC7ik98qT0VKK0F7ikja

a) {1, 2, 3} b) {} c) [1, 2, 3] d) (1, 2, 3)

#9 Which of the following Python statements will result in the output: 6

A = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

Python YouTube Playlist: https://www.youtube.com/playlist?list=PLKYRx0Ibk7Vi-CC7ik98qT0VKK0F7ikja

a) A[2][1] b) A[1][2] c) A[3][2] d) A[2][3]

Answer: b Explanation: The output that is required is 6, that is, row 2, item 3. This position is represented by the statement: A[1][2].

11# Which of the following statements is used to create an empty set in Python?

related video:https://youtube.com/shorts/nmI7BGXPA4I

a) ( ) b) [ ] c) { } d) set() Explanation: { } creates a dictionary not a set. Only set() creates an empty set.

#15 What method is used to add an element to the end of a list in Python?

Watch the Video Tutorial for the Answer:https://youtu.be/x98wvk-4MHw

#python #pythonpoll #MCQsTest

A) append() B) insert() C) extend() D) append_list() Answer: A) append()

#16 How do you add an element to a specific index in a list in Python?

Watch the Video Tutorial for the Answer: https://youtu.be/x98wvk-4MHw

#python #pythonpoll #MCQsTest

A) add_index() B) insert() C) extend() D) append() Answer: B) insert()

#17 What function is used to remove the last element from a list in Python?

Watch the Video Tutorial for the Answer:https://youtu.be/x98wvk-4MHw

#python #pythonpoll #MCQsTest #yasirbhutta

A) del_last() B) remove_last() C) pop() D) delete_end() Answer: C) pop()

#18 How can you remove the first occurrence of an element from a list in Python?

Watch the Video Tutorial for the Answer: https://youtu.be/x98wvk-4MHw

#python #pythonpoll #MCQsTest #yasirbhutta

A) remove() B) delete_first() C) pop_first() D) del_first() Answer: A) remove()

What is the output of the following code?

list1 = [12,98,23,70,66] list1[1:4] = [20] print(list1)

[12,20,66] [12,20,23,70,66] [12,98,23,20,66] Error

Which of the following is not a core data type in Python? a) List b) Tuple c) Dictionary d) Class

How would you create a tuple with only one element?

a) my_tuple = (1,) b) my_tuple = (1) c) my_tuple = [1] d) my_tuple = {1}

What is the result of the following code?

x = [1, 2, 3] y = x y[1] = 4 print(x)

a) [1, 2, 3] b) [1, 4, 3] c) [4, 2, 3] d) [1, 2, 4]

What is the result of the following code?

x = [1, 2, 3] y = x y = [4, 5, 6]

print(x) a) [1, 2, 3] b) [4, 5, 6] c) [1, 4, 5, 6] d) [4, 5, 6, 1, 2, 3]

How would you add an element to the end of a list in Python? a) list.append(element) b) list += element c) list.push(element) d) list = list + [element]

#5 What is the result of the following code?

x = {1: 'a', 2: 'b', 3: 'c'} y = x.copy() y[1] = 'd' print(x)

related video: https://youtube.com/shorts/PXp9uzvKFdU?feature=share

a) {1: 'a', 2: 'b', 3: 'c'} b) {1: 'd', 2: 'b', 3: 'c'} c) {1: 'd', 2: 'b', 3: 'c', 4: 'd'} d) {1: 'a', 2: 'b', 3: 'c', 1: 'd'}

How do you create an empty tuple in Python?

related video: https://youtu.be/nGIWcYXj580

a) tuple() b) {} c) () d) [] Answer: c) ()

How do you create a tuple with multiple elements in Python? related video: https://youtu.be/QpRiHuQycXg

a) tuple(1, 2, 3) b) (1, 2, 3) c) {1, 2, 3} d) [1, 2, 3] Answer: b) (1, 2, 3)

What is the correct way to create a tuple with a single element in Python? related video: https://youtu.be/nGIWcYXj580

a) tuple(1) b) (1,) c) {1} d) [1] Answer: b) (1,)

How do you add an element to an existing tuple in Python? a) tuple.append(element) b) tuple + (element,) c) tuple.extend(element) d) Tuples are immutable, so it is not possible to add an element to an existing tuple. Answer: d) Tuples are immutable, so it is not possible to add an element to an existing tuple.

How do you remove an element from a tuple in Python? a) tuple.remove(element) b) tuple.pop(element) c) Tuples are immutable, so it is not possible to remove an element from a tuple. d) del tuple[element] Answer: c) Tuples are immutable, so it is not possible to remove an element from a tuple.

What is the output of this code in PYTHON? list1=[1,2,3,4,5] print(list1[:4].pop())

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

What is the output of this code in PYTHON? list1=[1,2,34,5] print(list1[:4].pop())

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

Answer key (Mutiple Choice):

Fill in the Blanks

Answer Key (Fill in the Blanks):

Exercises

Review Questions

References and Bibliography

[1] B. E. Prep, “Difference Between Mutable and Immutable in Python | Mutable vs Immutable Objects,” BYJU’S Exam Prep, Sep. 24, 2023. https://byjusexamprep.com/gate-cse/difference-between-mutable-and-immutable (accessed Jul. 27, 2024).

Here's an expanded list of 50 exercises designed to help beginners understand and practice using `for` loops in Python. These exercises cover a variety of basic, intermediate, and advanced concepts to provide a comprehensive learning experience.

Basic Exercises

6. **Print elements of a list**: Use a `for` loop to print each element in the list `[10, 20, 30, 40, 50]`.
7. **Print elements of a matrix**: Use nested `for` loops to print each element of a 2x2 matrix, e.g., `[[1, 2], [3, 4]]`.
8. **Find the largest number in a list**: Write a `for` loop to find the largest number in the list `[34, 78, 23, 89, 12]`.
9. **Remove duplicates from a list**: Write a `for` loop to remove duplicates from the list `[1, 2, 2, 3, 4, 4, 5]`.
10. **Calculate the average of a list**: Use a `for` loop to calculate the average of numbers in the list `[10, 20, 30, 40, 50]`.

Advanced Exercises

23. **Transpose a matrix**: Write a `for` loop to transpose a 2x3 matrix, e.g., `[[1, 2, 3], [4, 5, 6]]` should become `[[1, 4], [2, 5], [3, 6]]`.
24. **Find the common elements in two lists**: Write a `for` loop to find the common elements in the lists `[1, 2, 3, 4]` and `[3, 4, 5, 6]`.
25. **Flatten a list of lists**: Write a `for` loop to flatten the list `[[1, 2], [3, 4], [5, 6]]` into `[1, 2, 3, 4, 5, 6]`.
26. **Print all the indexes of a list**: Write a `for` loop to print all the indexes of the list `[10, 20, 30, 40, 50]`.
27. **Generate a list of the first 10 square numbers**: Write a `for` loop to create a list of the first 10 square numbers.
28. **Print numbers in a list that are greater than 10**: Use a `for` loop to print numbers from the list `[5, 12, 17, 9, 3, 21]` that are greater than 10.
29. **Find the sum of even numbers in a list**: Write a `for` loop to calculate the sum of all even numbers in the list `[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]`.
30. **Calculate the product of all numbers in a list**: Write a `for` loop to calculate the product of all numbers in the list `[1, 2, 3, 4]`.
31. **Find the minimum value in a list**: Write a `for` loop to find the smallest number in the list `[34, 78, 23, 89, 12]`.
32. **Convert a list of Celsius temperatures to Fahrenheit**: Use a `for` loop to convert the list `[0, 10, 20, 30, 40]` to Fahrenheit using the formula `(Celsius * 9/5) + 32`.

33. **Reverse the elements of a list**: Write a `for` loop to reverse the elements of the list `[1, 2, 3, 4, 5]`.
34. **Calculate the sum of squares of a list of numbers**: Write a `for` loop to calculate the sum of the squares of numbers in the list `[1, 2, 3, 4, 5]`.
35. **Find the intersection of two lists**: Use a `for` loop to find the common elements in the lists `[1, 2, 3, 4, 5]` and `[4, 5, 6, 7, 8]`.
36. **Count the frequency of each element in a list**: Write a `for` loop to count the occurrences of each element in the list `[1, 2, 2, 3, 3, 3, 4, 4, 4, 4]`.
37. **Generate a dictionary with numbers and their squares**: Write a `for` loop to create a dictionary where keys are numbers from 1 to 5 and values are their squares.
38. **Print the transpose of a 3x3 matrix**: Write a `for` loop to transpose the matrix `[[1, 2, 3], [4, 5, 6], [7, 8, 9]]`.

What is the output of the following code? Python Quiz #23

```
a = [1, 2, 3, 4, 5]
for i in range(0, len(a), 2):
    print(a[i])
```

- A) 135
- B) 24
- C) 12345
- D) Syntax error

Watch the video for the answer: https://youtube.com/shorts/yrTnxNkrXH4?si=EUAzZiONKORP4nIU

> What is the output of the following code?

```
fruits = ["Apple", "Banana", "Cherry"]
for i, fruit in enumerate(fruits):
    if i == 1:
        print(fruit)
```

1. ☐ Apple
2. ☐ Banana
3. ☐ Cherry
4. ☐ Error

related video: https://youtu.be/-FErgsl9njQ

Which of the following is the correct way to iterate over elements in a list using a for loop? a. for item in my_list: b. for i = 0; i < len(my_list): c. for element = my_list: d. for (i, item) in enumerate(my_list):

How can you iterate over both the index and elements of a list using a for loop? a. for i in my_list: b. for (i, element) in enumerate(my_list): c. for element in range(my_list): d. for index, element in my_list:

What is the output of the following code?

```
numbers = [1, 2, 3, 4, 5]
for x in numbers:
    print(x * 2)
```

a. 2 4 6 8 10 b. 1 2 3 4 5 c. 1 4 9 16 25 d. 2 4 8 16 32

How many times will the following loop execute?

```
for i in [1, 2, 3, 3]:
  print("Hello")
```

```
- A) 3
- B) 4
- C) 1
- D) Infinitely
```

Appendices

Appendix A: Unordered Nature of Sets in Python

In Python, sets are **unordered collections**. This means that when you create a set, the elements are stored in no particular order. Unlike lists or tuples, sets do not maintain the order of elements based on how you inserted them.

What Does "Unordered" Mean?

1. **No Indexing:** You cannot access elements in a set using an index like you can with lists or tuples. For example, in a list, you can access elements using indices (`my_list[0]`), but this is not possible in sets because there is no guaranteed order.
2. **No Sequence:** When you add elements to a set, Python may internally arrange them in an unpredictable way for efficiency. Thus, the order of elements in a set may seem random when you print or inspect the set.

Example of Unordered Nature

```
my_set = {1, 2, 3, 4, 5}
```

```
# Display the set
print(my_set)  # Output could be {1, 2, 3, 4, 5} or {3, 1, 4, 2, 5}
```

If you run the above code multiple times, you might see the elements displayed in a different order each time, but the set still contains the same unique elements.

Why Does This Happen?

Sets are implemented using a **hash table** data structure, which optimizes for operations like checking membership (`in`), adding, and removing elements. The elements are stored in memory based on their hash values (computed using the `hash()` function). This hashing process determines where the elements are placed, and their physical position in memory may not correspond to the order in which they were added.

Therefore, when you print or inspect a set, Python shows the elements in a seemingly arbitrary order, influenced by how the hash table organizes them.

Example: Adding Elements and Observing the Unordered Nature

```
my_set = {10, 20, 30, 40, 50}

# Adding new elements
my_set.add(60)
my_set.add(70)

# Display the set
print(my_set)  # Output could be {40, 10, 50, 20, 70, 30, 60}
```

Even though `60` and `70` were added at the end, they might appear at different positions when the set is printed. This demonstrates that sets do not preserve the order of insertion.

Implications of Being Unordered

1. **No Guarantee of Order:** Every time you perform an operation on a set or print it, you should not expect the elements to appear in the same order as you inserted them. You cannot rely on the order of elements for any kind of sequential processing.
2. **Cannot Be Sorted Within a Set:** Sets do not support sorting directly because their main property is that they are unordered. However, if you need a sorted version of a set, you can convert it to a list and then sort it.

   ```
   my_set = {30, 10, 20}
   sorted_list = sorted(my_set)
   print(sorted_list)  # Output: [10, 20, 30]
   ```

3. **Iterating Over a Set:** Even though sets are unordered, you can still iterate over them. However, the order in which you retrieve elements during iteration is not guaranteed to be the same every time.

```
for element in my_set:
    print(element)  # The order of printing is unpredictable
```

Practical Uses of Unordered Sets

1. **Fast Membership Testing:** The unordered nature allows sets to perform very fast membership tests (`in` and `not in`). Checking if an element is in a set is much faster than in lists because of the underlying hash table.

Example:

```
my_set = {1, 2, 3, 4, 5}

# Checking if an element exists
if 3 in my_set:
    print("3 is in the set")
else:
    print("3 is not in the set")
```

2. **Duplicate Elimination:** Sets are used to quickly eliminate duplicates from collections, regardless of the order of elements. For example:

```
my_list = [1, 2, 2, 3, 4, 4, 5]
unique_elements = set(my_list)
print(unique_elements)  # Output could be {1, 2, 3, 4, 5}
```

Summary

The unordered nature of sets means:

- You cannot rely on the order of elements.
- Indexing and slicing operations are not supported.
- Elements are stored in positions based on their hash values, not their insertion order.

While this lack of order might seem limiting, it is also what makes sets powerful for specific tasks like ensuring uniqueness and performing fast lookups.