Built-in Functions

The Python interpreter has a number of functions and types built into it that are always available.

Built-in Functions

[need to add details]

See also:

- Built-in Functions - Python 3.12.1 documentation

Built-in Sequence Functions

- #1 Python zip() Function
- #2 Python Zip Function: Handling Lists with Different Numbers of Elements
- Python Iterators and Iterables: How to Loop Over Lists and Iterators
- How to Iterate Over Tuples with the Enumerate Function
- Finding the maximum value in a list using a one-liner
- Find the sum of all even numbers between 1 and 100 using a one-liner
- Counting the number of occurrences of an element in a list

Key Terms

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

Answer Key (True/False):

Multiple Choice (Select the best answer)

Answer key (Mutiple Choice):

What is the output of the following Python code?

numbers = [10, 32, 98, 38, 47, 34] print(max(numbers))

```
- **A.** 10
- **B.** 32
- **C.** 98
- **D.** 38
```

Watch this video for the answer: https://youtube.com/shorts/x7zh_WqO1e4

What is the output of the following code? [Python Quiz #11]

```
def my_func(fruit):
    if fruit:
```

~~~
        return  True
    return  False

fruits = ["apple","", "orange"
,"mango" ]

print(list(filter(my_func,fruits)))
```python

**A)** ['apple', 'orange', 'mango']
**B)** ['apple', '', 'orange', 'mango']
**C)** ['', '', '']
**D)** ['True', 'False', 'True', 'True']

**Watch this video for the answer:** [https://youtube.com/shorts/gkGpJfxsDew]
(https://youtube.com/shorts/gkGpJfxsDew)

#43 What is the output of the zip() function when one iterable is shorter than the
others?

Watch the Video Tutorial for the Answer: https://youtube.com/shorts/TOxTxP9x4ME?
feature=share

#python #pythonpoll #MCQsTest #yasirbhutta

a) The function raises an exception
b) The function returns only the tuples that have corresponding elements in all
iterables
c) The function pads the shorter iterable with None values
d) The function pads the shorter iterable with the last value of the iterable

Answer: b) The function returns only the tuples that have corresponding elements
in all iterables

#42 What does the zip() function in Python do?

Watch the Video Tutorial for the Answer: https://youtube.com/shorts/7ix3cDWAsUc?
feature=share

#python #pythonpoll #MCQsTest #yasirbhutta

### MCQs: Iterators and `zip()` in Python

#### **Question 1**

What is the output of the following code?

```python
list_a = [1, 2, 3]
list_b = ['x', 'y', 'z']
zipped = zip(list_a, list_b)

print(list(zipped))
print(list(zipped))
~~~

A)

```
[(1, 'x'), (2, 'y'), (3, 'z')]
[(1, 'x'), (2, 'y'), (3, 'z')]
```

B)

```
[(1, 'x'), (2, 'y'), (3, 'z')]
[]
```

C)

```
[]
[]
```

D)

```
Error: zip object has already been used.
```

Correct Answer: B

for more details, see Appendix A: Exploring the One-Time Use Behavior of `zip()` and Iterators in Python

Question 2

Why does the second `print(list(zipped))` in the following code output an empty list?

```
list_a = [1, 2, 3]
list_b = ['x', 'y', 'z']
zipped = zip(list_a, list_b)

print(list(zipped))
print(list(zipped))
```

A) The `zip` object is immutable.
B) The `zip` object is an iterator, and it can be consumed only once.
C) The `zip` object is reset after every `list()` call.
D) The `zip` object contains only one item by default.

Correct Answer: B

for more details, see Appendix A: Exploring the One-Time Use Behavior of `zip()` and Iterators in Python

Question 4

What does the `zip()` function return in Python?

A) A tuple of paired elements.
B) A dictionary mapping keys from the first list to values from the second list.
C) An iterator that pairs elements from the input iterables.
D) A list of tuples pairing elements from the input iterables.

Correct Answer: C

a) Compresses a file into a ZIP archive b) Combines two or more iterables into a single iterable of tuples, where each tuple contains one element from each iterable c) Creates a dictionary from two iterables, with keys from one iterable and values from another iterable d) Calculates the checksum of a file

Answer: b) Combines two or more iterables into a single iterable of tuples, where each tuple contains one element from each iterable

Q: What is the syntax for the zip() function in Python? a) zip(iterable1, iterable2) b) zip(iterable1, iterable2, ...) c) zip(*iterables) d) zip(iterables)

Answer: c) zip(*iterables)

Q: How can you use the zip() function to unzip a list of tuples? a) list(zip(*tuples_list)) b) tuple(zip(*tuples_list)) c) set(zip(*tuples_list)) d) dict(zip(*tuples_list))

Answer: a) list(zip(*tuples_list))

Fill in the Blanks

Answer Key (Fill in the Blanks):

Exercises

Review Questions

References and Bibliography

[1]“Built-in Functions — Python 3.12.4 documentation,” docs.python.org. https://docs.python.org/3/library/functions.html

Appendices

Appendix A: Exploring the One-Time Use Behavior of `zip()` and Iterators in Python

The behavior you're observing is due to the fact that `zip()` returns an **iterator**, not a list. Iterators in Python are designed to be **consumed** as they are iterated over. Once an iterator is exhausted, it cannot be reused.

Step-by-step Explanation:

1. **First `list(zipped_lists)`**:
   - The `zip` object (`zipped_lists`) is converted to a list.
   - During this process, the iterator is consumed, meaning all pairs are retrieved and stored in the resulting list: `[(1, 'a'), (2, 'b'), (3, 'c')]`.
2. **Second `list(zipped_lists)`**:
   - At this point, the `zip` object (`zipped_lists`) has already been fully consumed in the first `list()` call.
   - Since there are no more elements left in the iterator, the second call returns an empty list: `[]`.

How to Fix It:

If you need to reuse the results of `zip`, you can:

1. **Convert to a list immediately**:

   ```
   zipped_lists = list(zip(list_a, list_b))
   print(zipped_lists)
   print(zipped_lists)
   ```

   Output:

   ```
   [(1, 'a'), (2, 'b'), (3, 'c')]
   [(1, 'a'), (2, 'b'), (3, 'c')]
   ```

2. **Create a new `zip` object each time**:

   ```
   list_a = [1, 2, 3]
   list_b = ['a', 'b', 'c']
   print(list(zip(list_a, list_b)))
   print(list(zip(list_a, list_b)))
   ```

   Output:

   ```
   [(1, 'a'), (2, 'b'), (3, 'c')]
   [(1, 'a'), (2, 'b'), (3, 'c')]
   ```

This ensures you always have a fresh iterator to work with.