Eval Function / ZIP Function

Eval Function

Think of `eval()` as a python “interpreter within your code” - it take a string that contains Python code and actually runs it. It’s like telling Python ‘hey, treat this text as actual code and execute it.”

Eval function evaluate python expression which are written as strings.

```
# Basic arithmetic
result = eval("2 + 3")  # result = 5

# Using variables
x = 10
result = eval("x * 2")  # result = 20

# Expression with multiple operations
result = eval("2 * 3 + 4")  # result = 10
```

In Data Science

1. Reading Data

    ```
    # Converting string representations of lists/dictionaries to
    # actual Python objects

    data_str = "{'name': 'John', 'age': 30}"
    data = eval(data_str)  # Converts to actual dictionary
    ```

2. Dynamic Calculation

    ```
    # When you have mathematical formulas as strings
    formula = "x**2 + 2*x + 1"
    x = 5
    result = eval(formula)  # Calculates with x = 5
    ```

Note

eval() can be dangerous if used with untrusted input since it can execute any Python code. In data science, it's better to use safer alternatives like:

- `pandas.eval()` for dataframe operations
- `ast.literal_eval()` for parsing strings into Python data structures
- `numpy.vectorize()` for mathematical operations

Zip Function

`zip()` combines multiple iterables (lists, tuples, etc.) into pairs/tuple

```
# Empty zip
list(zip())
```

```
numbers = [1,2,3]
letters = ['a', 'b', 'c']
zipped = zip(numbers, letters) # Create pairs: (1,a), (2,b), (3,c)
```

```
1 lst1=["Krish","Sam","John"]
2 lst2=["a",'b','c']
3 lst3=[1,2,3]
```

```
1 output=zip(lst1,lst2,lst3)
2 for i,j,k in output:
3     print(i,j,k)
```

```
Krish a 1
Sam b 2
John c 3
```

```
# REAL WORLD EXAMPLE - Making Student Records

# Creating student records from separate lists
names = ['John', 'Emma', 'Mike']
grades = [85, 92, 78]
subjects = ['Math', 'Math', 'Math']

# Zip them together to create student records
student_records = list(zip(names, grades, subjects))
```

```
# Result:
# [('John', 85, 'Math'),
#  ('Emma', 92, 'Math'),
#  ('Mike', 78, 'Math')]

# You can easily loop through the records
for name, grade, subject in student_records:
    print(f"{name} got {grade} in {subject}")
```

```
# Combining customer data
customer_ids = [101, 102, 103]
names = ['Alice', 'Bob', 'Charlie']
purchases = [150, 200, 175]

for cust_id, name, amount in zip(customer_ids, names, purchases):
    print(f"Customer {cust_id}: {name} spent ${amount}")
```

Key Points to Remember:

- `zip()` creates an iterator of tuples
- The length of the zipped result is the length of the shortest input
- To see the results, you often need to convert to list: `list(zip(...))`
- You can unzip using zip(*zipped_object)