Python Modules

Python modules are files containing Python code that can define functions, classes, and variables. Using modules helps in organizing and reusing code.

Why Use Modules?

- **Organization**: Modules help organize related functions, classes, and variables.
- **Reusability**: Code can be reused across different projects by importing modules.
- **Namespace**: Modules create a separate namespace, avoiding conflicts between identifiers.

Importing Modules

You can import modules using the `import` keyword.

```
import math
import datetime
import random
import re
import json
import collections
import itertools
```

math Module

The `math` module provides mathematical functions.

Common Functions

```
import math

# Constants
print(math.pi)  # Output: 3.141592653589793
print(math.e)  # Output: 2.718281828459045

# Trigonometric functions
print(math.sin(math.pi / 2))  # Output: 1.0
print(math.cos(0))  # Output: 1.0

# Logarithmic functions
print(math.log(1))  # Output: 0.0
print(math.log10(100))  # Output: 2.0

# Power and square root
```

```
print(math.pow(2, 3))  # Output: 8.0
print(math.sqrt(16))  # Output: 4.0

# Rounding functions
print(math.floor(3.7))  # Output: 3
print(math.ceil(3.7))  # Output: 4
```

datetime Module

The `datetime` module supplies classes for manipulating dates and times.

Common Classes and Methods

```
import datetime

# Getting current date and time
now = datetime.datetime.now()
print(now)

# Creating specific dates
new_year = datetime.datetime(2024, 1, 1)
print(new_year)

# Date arithmetic
one_day = datetime.timedelta(days=1)
tomorrow = now + one_day
print(tomorrow)

# Formatting dates
formatted_date = now.strftime("%Y-%m-%d %H:%M:%S")
print(formatted_date)
```

random Module

The `random` module implements pseudo-random number generators for various distributions.

Common Functions

```
import random

# Generating random numbers
print(random.random())  # Output: Random float between 0.0 and 1.0
print(random.randint(1, 10))  # Output: Random integer between 1 and 10

# Random choice
```

```python
choices = ['apple', 'banana', 'cherry']
print(random.choice(choices))  # Output: Random choice from the list

# Shuffling a list
random.shuffle(choices)
print(choices)  # Output: Shuffled list

# Sampling
print(random.sample(choices, 2))  # Output: 2 random elements from the list
```

re Module

The `re` module provides regular expression matching operations.

Common Functions

```python
import re

# Searching for patterns
pattern = r"\bword\b"
text = "A word in a sentence."
match = re.search(pattern, text)
print(match.group())  # Output: word

# Finding all matches
matches = re.findall(r"\d+", "There are 123 numbers and 456 in this text.")
print(matches)  # Output: ['123', '456']

# Replacing patterns
new_text = re.sub(r"\d+", "#", "There are 123 numbers and 456 in this text.")
print(new_text)  # Output: There are # numbers and # in this text.
```

json Module

The `json` module provides functions for parsing JSON strings and converting Python objects to JSON.

Common Functions

```python
import json

# Converting Python objects to JSON
data = {"name": "Alice", "age": 25}
json_str = json.dumps(data)
print(json_str)  # Output: '{"name": "Alice", "age": 25}'
```

```python
# Parsing JSON strings to Python objects
parsed_data = json.loads(json_str)
print(parsed_data)  # Output: {'name': 'Alice', 'age': 25}

# Reading from and writing to JSON files
with open("data.json", "w") as file:
    json.dump(data, file)

with open("data.json", "r") as file:
    loaded_data = json.load(file)
print(loaded_data)  # Output: {'name': 'Alice', 'age': 25}
```

collections Module

The `collections` module provides specialized container datatypes.

Common Classes

```python
import collections

# Named tuple
Point = collections.namedtuple('Point', ['x', 'y'])
p = Point(1, 2)
print(p.x, p.y)  # Output: 1 2

# Counter
counter = collections.Counter("aabbcc")
print(counter)  # Output: Counter({'a': 2, 'b': 2, 'c': 2})

# defaultdict
default_dict = collections.defaultdict(int)
default_dict['a'] += 1
print(default_dict)  # Output: defaultdict(<class 'int'>, {'a': 1})

# deque
d = collections.deque([1, 2, 3])
d.appendleft(0)
d.append(4)
print(d)  # Output: deque([0, 1, 2, 3, 4])
```

itertools Module

The `itertools` module provides functions for creating iterators for efficient looping.

Common Functions

```
import itertools

# Infinite iterators
counter = itertools.count(start=1, step=2)
print(next(counter))  # Output: 1
print(next(counter))  # Output: 3

# Combinatoric iterators
permutations = list(itertools.permutations([1, 2, 3]))
print(permutations)  # Output: [(1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2), (3, 2

combinations = list(itertools.combinations([1, 2, 3], 2))
print(combinations)  # Output: [(1, 2), (1, 3), (2, 3)]

# Grouping
groups = itertools.groupby("aaabbbcc")
for key, group in groups:
    print(key, list(group))  # Output: ('a', ['a', 'a', 'a']), ('b', ['b', 'b', 'b']), ('c'
```

Conclusion

Python's standard library provides a rich set of modules and functions that facilitate various tasks. Understanding and utilizing these modules can significantly enhance your programming skills and efficiency.

Stay Updated

Be sure to this repository to stay updated with new examples and enhancements!

License

This project is protected under the MIT License.

Contact

Panagiotis Moschos - pan.moschos86@gmail.com

Note: This is a Python script and requires a Python interpreter to run.

Happy Coding

Made with by Panagiotis Moschos (https://github.com/pmoschos)