Python Code Examples

Andy

April 17, 2025

Page 1: String Manipulation Utilities

Python Code

```python
def is_palindrome(s):
    s = s.lower().replace(" ", "")
    return s == s[::-1]

def count_vowels(s):
    return sum(1 for char in s.lower() if char in "aeiou")

def reverse_words(sentence):
    return ' '.join(word[::-1] for word in sentence.split())

text = "Was it a car or a cat I saw"
print("Original:", text)
print("Is Palindrome?", is_palindrome(text))
print("Vowel Count:", count_vowels(text))
print("Reversed Words:", reverse_words(text))
```

Introduction

This code defines three string utility functions: one to check palindromes, one to count vowels, and one to reverse words individually.

Page 2: Recursive Factorial and Summation

Python Code

```python
def factorial(n):
    if n == 0:
        return 1
    return n * factorial(n - 1)

def recursive_sum(n):
    if n == 0:
        return 0
    return n + recursive_sum(n - 1)

for i in range(6):
    print(f"Factorial of {i} is {factorial(i)}")
    print(f"Sum of numbers 1 to {i} is {recursive_sum(i)}")
```

Introduction

This example defines recursive implementations of factorial and summation, and prints results for values from 0 to 5.

Page 3: Prime Checking and Listing

Python Code

```python
def is_prime(n):
    if n < 2:
        return False
    for i in range(2, int(n**0.5)+1):
        if n % i == 0:
            return False
    return True

def list_primes(limit):
    return [n for n in range(2, limit+1) if is_prime(n)]

print("Primes below 50:")
print(list_primes(50))
```

Introduction

This snippet checks if a number is prime and generates a list of primes below a given limit using list comprehension.

Page 4: Working with 2D Arrays

Python Code

```python
def create_grid(rows, cols, val=0):
    return [[val for _ in range(cols)] for _ in range(rows)]

def set_diagonal(grid, value):
    for i in range(min(len(grid), len(grid[0]))):
        grid[i][i] = value

def print_grid(grid):
    for row in grid:
        print(" ".join(map(str, row)))

grid = create_grid(5, 5, 1)
set_diagonal(grid, 9)
print("5x5 Grid with diagonal 9s:")
print_grid(grid)
```

Introduction

This program creates a 2D grid, sets the diagonal to a specific value, and prints the grid in a readable format.

Page 5: Simulating a Simple Counter Class

Python Code

```
class Counter:
    def __init__(self, start=0):
        self.value = start

    def increment(self, amount=1):
        self.value += amount

    def decrement(self, amount=1):
        self.value -= amount

    def reset(self):
        self.value = 0

    def display(self):
        print(f"Current value: {self.value}")

counter = Counter()
counter.display()
counter.increment(3)
counter.display()
counter.decrement()
counter.display()
counter.reset()
counter.display()
```

Introduction

This code defines a ‘Counter‘ class with methods to increment, decrement, reset, and display its value.

Page 6: List Filtering and Transformation

Python Code

```python
numbers = list(range(1, 21))

evens = [x for x in numbers if x % 2 == 0]
squares = [x**2 for x in evens]
greater_than_50 = [x for x in squares if x > 50]

print("Original:", numbers)
print("Even numbers:", evens)
print("Squares of evens:", squares)
print("Squares > 50:", greater_than_50)
```

Introduction

This code demonstrates list filtering and transformation using list comprehensions for a range of integers.

Page 7: Simple Grade Analysis

Python Code

```python
grades = [87, 92, 76, 81, 95, 67, 74, 88, 90, 100]

average = sum(grades) / len(grades)
above_90 = [g for g in grades if g > 90]
passed = [g for g in grades if g >= 75]

print("Grades:", grades)
print("Average:", round(average, 2))
print("Grades above 90:", above_90)
print("Passing grades (>=75):", passed)
```

Introduction

This code analyzes a list of grades, calculating average, listing top scores, and identifying passing scores.