Python Lambda Functions

Basic Lambda Functions

```
# Simple lambda function
square = lambda x: x ** 2
print("Square of 5:", square(5))

# Lambda with multiple arguments
add = lambda x, y: x + y
print("3 + 4 =", add(3, 4))

# Lambda with conditional
max_of_two = lambda a, b: a if a > b else b
print("Max of 10 and 20:", max_of_two(10, 20))

# Regular function vs lambda
def multiply(x, y):
    return x * y

multiply_lambda = lambda x, y: x * y

print("\nRegular:", multiply(3, 4))
print("Lambda:", multiply_lambda(3, 4))
```

Lambda with Sorting

```
# Sort list of tuples by second element
students = [('Alice', 85), ('Bob', 92), ('Charlie', 78), ('David', 88)]

# Sort by score (descending)
by_score = sorted(students, key=lambda x: x[1], reverse=True)
print("By score:", by_score)

# Sort by name
by_name = sorted(students, key=lambda x: x[0])
print("\nBy name:", by_name)

# Sort dictionaries
people = [
    {'name': 'Alice', 'age': 25},
    {'name': 'Bob', 'age': 30},
    {'name': 'Charlie', 'age': 22}
]
by_age = sorted(people, key=lambda x: x['age'])
print("\nBy age:", by_age)
```

Lambda with map(), filter(), reduce()

```
# map: apply function to each element
numbers = [1, 2, 3, 4, 5]
doubled = list(map(lambda x: x * 2, numbers))
print("Doubled:", doubled)

squares = list(map(lambda x: x**2, numbers))
print("Squared:", squares)

# filter: keep elements that satisfy condition
evens = list(filter(lambda x: x % 2 == 0, numbers))
print("\nEven numbers:", evens)
```

```
greater_than_3 = list(filter(lambda x: x > 3, numbers))
print("Greater than 3:", greater_than_3)

# reduce: combine elements
from functools import reduce
product = reduce(lambda x, y: x * y, numbers)
print("\nProduct:", product)

sum_all = reduce(lambda x, y: x + y, numbers)
print("Sum:", sum_all)
```

Lambda in Data Processing

```
# Process list of dictionaries
data = [
    {'name': 'Item1', 'price': 10.0, 'quantity': 2},
    {'name': 'Item2', 'price': 15.5, 'quantity': 1},
    {'name': 'Item3', 'price': 7.25, 'quantity': 3}
]

# Calculate total for each item
totals = list(map(lambda x: x['price'] * x['quantity'], data))
print("Totals:", totals)

# Filter expensive items
expensive = list(filter(lambda x: x['price'] > 10, data))
print("\nExpensive items:", expensive)

# String manipulation
words = ['hello', 'world', 'python', 'lambda']
uppercase = list(map(lambda s: s.upper(), words))
print("\nUppercase:", uppercase)

long_words = list(filter(lambda s: len(s) > 5, words))
print("Long words:", long_words)
```

Advanced Lambda Patterns

```
# Lambda with multiple operations
data = [1, -2, 3, -4, 5]
abs_doubled = list(map(lambda x: abs(x) * 2, data))
print("Absolute value doubled:", abs_doubled)

# Nested lambda
apply_twice = lambda f, x: f(f(x))
result = apply_twice(lambda x: x + 3, 5)
print("\nApply +3 twice to 5:", result)

# Lambda returning lambda
def make_multiplier(n):
    return lambda x: x * n

times_3 = make_multiplier(3)
times_5 = make_multiplier(5)
print("\n7 * 3 =", times_3(7))
print("7 * 5 =", times_5(7))

# Complex sorting
items = [(1, 'b'), (2, 'a'), (1, 'a'), (2, 'b')]
# Sort by second element, then first
sorted_items = sorted(items, key=lambda x: (x[1], x[0]))
print("\nSorted:", sorted_items)
```