VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

ADVANCED PROGRAMMING (CO2039)

Assignment (Semester: 232)

Functional Programming in Python

Advisor: Trần Tuấn Anh, CSE-HCMUT
Students: Trần Đình Đăng Khoa - 2211649.

HO CHI MINH CITY, MAY 2024

Contents

1 Introduction

Functional programming (FP) is a programming paradigm that has gained significant traction in both academic and professional settings due to its powerful abstraction capabilities and emphasis on immutability and first-class functions. Unlike imperative programming, which is centered around changing state and iterating over instructions, functional programming treats computation as the evaluation of mathematical functions and avoids changing-state and mutable data.

Python, traditionally known as an imperative and object-oriented language, has incorporated several functional programming features that make it suitable for this paradigm. Python supports first-class functions, allowing functions to be treated as any other object, passed as arguments, and returned from other functions. Higher-order functions, which take other functions as parameters or return them as results, are also integral to Python. Furthermore, Python's lambda expressions facilitate the creation of anonymous functions, enabling succinct and expressive code.

The purpose of this report is to explore the application of functional programming principles in Python. We will delve into key functional programming concepts supported by Python, including first-class functions, higher-order functions, and lambda expressions. Additionally, we will demonstrate how to create a functional program in Python through a practical example that processes a list of numbers using map, filter, and reduce functions.

2 Python as a Functional Programming Language

Python, while primarily an imperative and object-oriented language, provides several features that facilitate functional programming. These include first-class functions, higher-order functions, anonymous functions (lambdas), and a rich standard library that supports functional constructs.

2.1 First-Class Functions

In Python, functions are first-class citizens. This means that functions can be assigned to variables, passed as arguments to other functions, and returned as values from other functions.

```python
def add(x, y):
    return x + y

f = add
print(f(2, 3))  # Output: 5
```

Listing 1: First-Class Functions

First-class functions are fundamental to functional programming as they allow for higher-order functions and functional composition. By assigning the function ‘add‘ to the variable ‘f‘, we demonstrate that functions in Python can be manipulated like any other object. This allows for flexible and dynamic function usage, promoting code reuse and modularity.

2.2 Higher-Order Functions

Higher-order functions are functions that take other functions as arguments or return functions as their result. They are essential for functional programming as they enable the creation of more abstract and reusable code.

```python
def apply_function(f, x, y):
    return f(x, y)

result = apply_function(add, 5, 10)
print(result)  # Output: 15
```

Listing 2: Higher-Order Functions

In this example, `apply_function` is a higher-order function because it takes another function `f` as an argument. This allows us to pass different functions to `apply_function`, making it highly versatile. Higher-order functions are powerful tools for building abstractions and composing behavior in a clean and readable way.

2.3 Anonymous Functions (Lambdas)

Python supports anonymous functions through the `lambda` keyword. These are useful for short, throwaway functions that are not reused elsewhere.

```python
square = lambda x: x * x
print(square(4))  # Output: 16
```

Listing 3: Anonymous Functions (Lambdas)

Lambda functions provide a concise way to define simple functions inline. They are particularly useful for functional programming constructs like `map`, `filter`, and `reduce`, where small functions are often passed as arguments. Despite their brevity, lambda functions can be used effectively to create clear and expressive code.

3 Creating a Functional Program in Python

We will create a functional program that processes a list of numbers. The program will filter out even numbers, square the remaining numbers, and then sum them up. This will demonstrate the use of `map`, `filter`, and `reduce` functions.

3.1 Implementation

```python
from functools import reduce

# List of numbers
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# Filter out even numbers
odd_numbers = list(filter(lambda x: x % 2 != 0, numbers))

# Square the remaining numbers
squared_numbers = list(map(lambda x: x * x, odd_numbers))

# Sum the squared numbers
sum_of_squares = reduce(lambda x, y: x + y, squared_numbers)

print(f"The sum of squares of odd numbers is: {sum_of_squares}")
```

Listing 4: Functional Program Demo

3.2 Explanation

This program showcases the core functional programming concepts in Python:

1. Filtering with `filter`:

   ```python
   odd_numbers = list(filter(lambda x: x % 2 != 0, numbers))
   ```

   - The `filter` function constructs an iterator from elements of the iterable (in this case, the list `numbers`) for which the function `lambda x: x % 2 != 0` returns `True`.
   - This lambda function checks if a number is odd by computing the remainder of the division by 2.
   - The `list` constructor is used to convert the resulting iterator back into a list, containing only the odd numbers.

2. Mapping with `map`:

```
squared_numbers = list(map(lambda x: x * x, odd_numbers))
```

- The `map` function applies the function `lambda x: x * x` to each item of the iterable (the list `odd_numbers`).
- This lambda function squares its input.
- Again, the `list` constructor is used to convert the map object into a list of squared numbers.

3. Reducing with `reduce`:

```
sum_of_squares = reduce(lambda x, y: x + y, squared_numbers)
```

- The `reduce` function from the `functools` module applies the function `lambda x, y: x + y` cumulatively to the items of the iterable (the list `squared_numbers`), from left to right, so as to reduce the iterable to a single value.
- This lambda function sums two numbers.
- The result is the sum of all squared numbers in the list.

These functional constructs — `filter`, `map`, and `reduce` — are powerful tools for data transformation and aggregation, allowing for clear and expressive code. The use of lambda functions in these constructs further enhances the readability and conciseness of the code, making it easy to understand the operations being performed on the data.

4 Conclusion

Functional programming is an essential paradigm that brings numerous benefits to software development, including improved modularity, easier reasoning about code, and a higher level of abstraction. Although Python is predominantly an imperative and object-oriented language, it effectively incorporates functional programming constructs, making it a versatile tool for developers.

By leveraging first-class functions, higher-order functions, and anonymous functions, Python enables the creation of concise and expressive functional programs. This report demonstrated how these features can be utilized to process a list of numbers in a functional style, showcasing the elegance and power of functional programming in Python.

The example provided illustrates the functional programming principles of immutability and function composition, highlighting Python's capability to support these concepts. As the software industry continues to evolve, the ability to blend multiple programming paradigms within a single language becomes increasingly valuable. Python's support for functional programming enhances its flexibility and applicability across various domains and problem sets.

Future work could explore more advanced functional programming techniques in Python, such as currying, memoization, and functional data structures. Additionally, integrating functional programming with Python's rich ecosystem of libraries can further enhance its utility in real-world applications.

5 References

- Python Functional Programming HOWTO
- Real Python - Functional Programming in Python
- Wikipedia - Functional Programming