20CYS312 - Principles of Programming Languages
Exploring Programming Paradigms

Assignment-01

Presented by SUSHMANTH.V.M
CB.EN.U4CYS21077
TIFAC-CORE in Cyber Security
Amrita Vishwa Vidyapeetham, Coimbatore Campus

Feb 2024

AMRITA
VISHWA VIDYAPEETHAM

Outline

Scripting

- Interpretation, rapid development, and dynamic typing.
- High-level abstractions and platform independence.
- Python as a scripting language.

Scripting - Python

```
import csv

# Function to read data from a CSV file
def read_csv(file_path):
    data = []
    with open(file_path, 'r') as csvfile:
        reader = csv.DictReader(csvfile)
        for row in reader:
            data.append(row)
    return data

# Function to perform data transformation
def process_data(input_data):
    processed_data = []
    for entry in input_data:
```

Scripting - Python (Contd.) I

```
# Assume the CSV has 'name', 'age', and 'city' columns
name = entry['name']
age = int(entry['age'])
city = entry['city'].upper()

# Transform data (e.g., convert age to a category)
```

Scripting - Python (Contd.) I

```
            age_category = 'Young' if age < 30 else 'Old'

            # Create a new processed entry
            processed_entry = {
                'Name': name,
                'Age Category': age_category,
                'City': city
            }
            processed_data.append(processed_entry)

    return processed_data

# Function to write data to a new CSV file
def write_csv(output_data, output_file):
    fieldnames = ['Name', 'Age Category', 'City']
```

Scripting - Python (Contd.) I

```
    with open(output_file, 'w', newline='') as csvfile:
        writer = csv.DictWriter(csvfile, fieldnames=fieldnames)

        # Write the header
        writer.writeheader()

        # Write the processed data
        for entry in output_data:
            writer.writerow(entry)

# Main script
if __name__ == "__main__":
    # Input and output file paths
    input_file_path = 'input_data.csv'
    output_file_path = 'output_data.csv'
```

Scripting - Python (Contd.) I

```
# Read data from CSV
raw_data = read_csv(input_file_path)

# Process data
processed_data = process_data(raw_data)

# Write processed data to a new CSV file
write_csv(processed_data, output_file_path)

print("Script executed successfully. Check 'output_data.csv'
```

Meta programming

- Code generation, reflection, and code manipulation.
- Domain-Specific Languages (DSLs) and template metaprogramming.
- Ruby as a language for meta-programming.

Meta programming - Ruby

```ruby
# Define a simple class
class MyClass
  def existing_method
    puts "This is an existing method."
  end
end

# Instantiate an object of the class
obj = MyClass.new
```

Meta programming - Ruby (Contd.) I

```ruby
# Call the existing method
obj.existing_method

# Meta-programming: Define a new method dynamically
MyClass.class_eval do
  define_method :new_method do
    puts "This is a dynamically added method."
  end
end

# Call the dynamically added method
obj.new_method
```

Comparison and Discussions

- **Similarities:** Dynamic typing, high-level abstractions, ease of learning.
- **Differences:** Paradigm focus, use cases, code structure, flexibility vs. readability, community and usage.
- Choose Python for readability and ease of use (scripting).
- Choose Ruby for dynamic code manipulation (meta-programming).
- Both languages offer versatility for different paradigms.

References

- https://stackoverflow.com/questions/514644/what-exactly-is-metaprogramming
- https://www.ruby-lang.org/en/documentation/
- https://www.geeksforgeeks.org/introduction-to-scripting-languages/
- chatgpt for codes