Question 1

a. With the aid of diagram and syntactic structure, differentiate the three selection statements.

- **Selection Statements**: Typically, selection statements include `if`, `else`, and `switch/case`. The PDF might explain these, but here's a summary:
    - **If-Else Statement**:
        - **Syntax**:

```
python
Copy code
if condition:
    # code block
elif another_condition:
    # another block
else:
    # else block
```

        - **Diagram**:
            - A flowchart with branching paths for each condition.
    - **Switch/Case Statement**:
        - **Syntax** (not native in Python, but conceptual):

```
python
Copy code
switch (expression):
    case value1:
        # code block
        break
    case value2:
        # code block
        break
    default:
        # code block
```

        - **Diagram**:
            - A central condition leading to multiple branches based on the case.

b. Discuss the essence of repetition structures and differentiate `do ... while` statement from `while ... endwhile` statement.

- **Repetition Structures**:
    - Repetition structures, also known as loops, allow a program to repeat a block of code.
    - **While loop**:
        - Syntax:

```
python
Copy code
while condition:
    # code block
```

    - **Do-While loop**:
        - Conceptual Syntax (not native in Python but similar logic):

```
python
Copy code
do:
    # code block
while condition
```

        - **Difference**: In `do...while`, the block is executed at least once before the condition is tested. In `while...endwhile`, the condition is tested first.

c. Itemize four advantages of procedures/functions and write a Python program to illustrate the concept of a user-defined void function.

- **Advantages**:
    1. **Code Reusability**: Functions allow the reuse of code blocks.
    2. **Modularity**: Programs can be broken down into smaller, manageable parts.
    3. **Simplified Testing**: Each function can be tested independently.
    4. **Abstraction**: Functions hide the implementation details from the user.
- **Python Example**:

```
python
Copy code
def greet():
    print("Hello, World!")
```

```
greet()  # This will print "Hello, World!"
```

Question 2

a. Discuss the similarities and differences between natural and formal languages.

- **Similarities**:
  - Both have syntax (structure) and semantics (meaning).
- **Differences**:
  - **Natural Languages**: Used by humans, ambiguous, and context-sensitive.
  - **Formal Languages**: Used in programming, unambiguous, and follows strict rules.

b. Explain the following basic Python string operations:

2. **Repetition**: Using * to repeat a string.
   - Example: `"Hello" * 3` results in `"HelloHelloHello"`
3. **Concatenation**: Using + to combine strings.
   - Example: `"Hello" + " World"` results in `"Hello World"`
4. **Indexing**: Accessing a specific character by position.
   - Example: `"Hello"[0]` results in `"H"`
5. **Slicing**: Extracting a portion of a string.
   - Example: `"Hello"[1:4]` results in `"ell"`

c. Write a program to calculate and output the overtime pay for employees. Overtime is paid N3,000 per hour for every hour worked over 40 hours per week.

- **Python Program**:

```
python
Copy code
def calculate_overtime(hours_worked):
    overtime_rate = 3000
    if hours_worked > 40:
        overtime_hours = hours_worked - 40
        overtime_pay = overtime_hours * overtime_rate
    else:
        overtime_pay = 0
```

```
    return overtime_pay

hours = int(input("Enter hours worked: "))
print("Overtime Pay: N", calculate_overtime(hours))
```

Question 3

a. Explain the following concepts: Encapsulation and Generalization with the aid of a sample Python program.

- **Encapsulation**: Bundling data and methods that operate on the data within a single unit, like a class.
- **Generalization**: Forming general concepts by abstracting common properties of instances.
- **Example**:

```
python
Copy code
class Animal:
    def __init__(self, name):
        self.name = name

    def speak(self):
        pass

class Dog(Animal):
    def speak(self):
        return "Woof!"

class Cat(Animal):
    def speak(self):
        return "Meow!"
```

b. Write a program to generate even numbers between 0 and 1000, the total number of even numbers generated, and the average of the even numbers.

- **Python Program**:

```
python
```

```
Copy code
even_numbers = [i for i in range(0, 1001) if i % 2 == 0]
total_count = len(even_numbers)
average = sum(even_numbers) / total_count

print("Even Numbers:", even_numbers)
print("Total Count:", total_count)
print("Average:", average)
```

Question 4

Fermat's Last Theorem Check

- **Python Program**:

```
python
Copy code
def check_fermat(a, b, c, n):
    if n > 2 and (a**n + b**n == c**n):
        return "Ojo kuu, Fermat was wrong!"
    else:
        return "Oga juu, that doesn't work."

# Example usage
print(check_fermat(3, 4, 5, 3))  # Example output
```

Question 5

ATM Machine Naira Breakdown

- **Python Program**:

```
python
Copy code
def atm_breakdown(amount):
    notes = [200, 100, 50, 20]
    breakdown = {}

    for note in notes:
```

```
        breakdown[note] = amount // note
        amount %= note

    return breakdown

amount = int(input("Enter amount (not less than 500): "))
if amount < 500:
    print("Invalid amount.")
else:
    print("Breakdown:", atm_breakdown(amount))
```