Understanding Python through a complete example

1. Function Definition and Parameters

Theory: In Python, functions are blocks of reusable code. They can take parameters as input, perform operations, and return a result.

Exercise: Define a function `calculate_average` that takes four parameters (x1, x2, x3, x4) and returns their average.

```
def calculate_average(x1, x2, x3, x4):
    average = (x1 + x2 + x3 + x4) / 4
    return average
```

2. Conditional Statements

Theory: Conditional statements (if, elif, else) allow you to execute different blocks of code based on certain conditions.

Exercise: Write a function `convert_to_text_grade` that takes a numeric grade as a parameter and returns the corresponding text grade using conditional statements.

```
def convert_to_text_grade(numeric_grade):
    if numeric_grade >= 7.5:
        return "Very good"
    elif numeric_grade >= 5:
        return "Good"
    else:
        return "Fail"
```

3. Variable Assignment and Basic Operations

Theory: Variables are used to store and manage data. Basic operations (+, -, *, /) are essential for performing calculations.

Exercise: Assign values to variables `t1` and `t2` representing grades.

```
t1 = 6.0
t2 = 4.0
```

4. Function Invocation

Theory: Functions are invoked (called) to execute the code within them and obtain the result.

Exercise: Use the `calculate_average` function to find the average of `t1` and `t2`.

```
average = calculate_average(t1, t2)
print("Average:", average)
```

5. Rounding Numbers

Theory: The `round()` function is used to round numerical values to a specified number of decimal places.

Exercise: Round the calculated average to two decimal places.

```
average = round(average, 2)
print("Rounded Average:", average)
```

6. String Concatenation

Theory: Strings can be concatenated using the `+` operator.

Exercise: Concatenate strings to print the final grade in the specified format.

```
text_average = convert_to_text_grade(average)
print("Final grade: " + str(average) + " (" + text_average + ")")
```

7. Application Example

Here's a basic example that calculates the average of two grades, converts it to a text grade, and prints the result:

```
# Function to calculate average
def getAverage(x1, x2):
    x = (x1 + x2) / 2
    return x

# Function to convert numeric grade to text
def getTextGrade(ngrade):
    text = ""
    if ngrade >= 7.5:
        text = "Very good"
    elif 5 <= ngrade < 7.5:
        text = "Good"
    else:
        text = "Fail"
    return text

# Assign values to variables
```

```
t1 = 6.0
t2 = 4.0

# Calculate average
average = getAverage(t1, t2)

# Round the average to 2 decimals
average = round(average, 2)

# Convert numeric average to text grade
text_average = getTextGrade(average)

# Print the final grade
print("Final grade: " + str(average) + " (" + text_average + ")")
```

By working through these exercises, you'll reinforce the fundamental concepts needed to understand and write the Python code needed to create a get_average.py application. Each exercise corresponds to a key aspect of the code, helping you build a solid foundation in Python programming.