Introduction:

This assignment focuses on hands-on learning in Python programming, inspired by Chapter 1 of "Think Python: How to think like a computer scientist." Divided into two parts, it emphasizes experiential learning through error introduction and practical problem-solving.

The Jupyter Notebook environment is utilized, ensuring interactive code execution and easy sharing. The Jupyter Notebook environment guarantees executable code, allowing output to be captured for insertion into the word file

Part 1:

In this section, deliberate errors are introduced in Python code to understand their impact. Questions range from printing names with missing quotation marks to exploring the distinctions between * and ** operators.

Each error provides valuable insights into Python behavior, fostering effective error interpretation.

a. Displaying name, code, output, and justification

```
# Attempting to print a name with missing closing quotation mark
print("Benewende)  # Missing closing quotation mark
```

Output:

```
  Cell In[19], line 2
    print("Benewende)  # Missing closing quotation mark
          ^
SyntaxError: unterminated string literal (detected at line 2)
```

Explanation:

Printing a name requires enclosing it in matching quotation marks. The first statement, if executed without the closing quotation mark, will yield a `SyntaxError: unterminated string literal` ([Jones, 2019]).

```
# Attempting to print a name with missing opening quotation mark
print(Pierre")  # Missing opening quotation mark
```

Output:

```
  Cell In[20], line 2
    print(Pierre")  # Missing opening quotation mark
                 ^
SyntaxError: unterminated string literal (detected at line 2)
```

Explanation:

Similarly, attempting to print a name without the opening quotation mark will result in a `SyntaxError: unterminated string literal` ([Jones, 2019]).

```
# Attempting to print a lastname with no quotation marks
print(BONKOUNGOU)     # No quotation marks
```

Output:

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[21], line 2
      1 # Attempting to print a lastname with no quotation mark
----> 2 print(BONKOUNGOU)

NameError: name 'BONKOUNGOU' is not defined
```

Explanation:

Attempting to print a lastname without any quotation marks will result in a `NameError` because Python interprets `BONKOUNGOU` as a variable, and it's not defined ([Downey, 2015]).

b. Difference in * and ** operators with an example

In Python, the * and ** operators are also used for multiplication and exponentiation, respectively.

- * is the multiplication operator;

- ** is the exponentiation operator. It raises the number on the left to the power of the number on the right.

For example:

```
# Using * and ** operators
a = 5
b = 3
print(f"a * b = {a * b}")  # Multiplication operator
print(f"a ** b = {a ** b}")  # Exponentiation operator
```

Output:

```
a*b = 15
a**b = 125
```

Explanation:

The `*` operator performs multiplication, while `**` performs exponentiation. Multiplication is the repeated addition of a number by itself, and exponentiation raises a number to the power of another number. Understanding the distinction is crucial for accurate arithmetic operations in Python ([Downey, 2015]).

c. Attempting to display an integer like 09

```
# Attempting to display an integer like 09
print(09)
```

Output:

```
  Cell In[23], line 2
    print(09)
          ^
SyntaxError: leading zeros in decimal integer literals are not permitted; use an 0o prefix for octal integers
```

Explanation:

It is not possible to display an integer with a leading zero like 09 in Python. Python interprets numbers starting with 0 as octal literals. However, 09 is not a valid octal number, leading to a `SyntaxError` ([Lutz, 2013]).

d. Using the type() function on a string and an integer

```
# Using the type() function on a string
print(type('67'))
# Using the type() function on an integer
print(type(67))
```

Output:

```
<class 'str'>
<class 'int'>
```

Explanation:

The `type` function in Python returns the data type of the object passed as an argument. Strings are enclosed in quotes (' '), while integers are not. Strings can contain characters, and they are immutable, whereas integers can only contain figures and are mutable. Understanding these differences is crucial for effective Python programming ([Downey, 2015]).

Part 2:

This part extends Python application to real-world scenarios. Python programs cover multiplying one's age, displaying geographical information, presenting examination schedules, and showcasing real-time temperature data.

a. Multiply the age by 2 and display

```
# Multiplying age by 2 and displaying the result
age = 33
```

```
result = age * 2
print(f"The age multiplied by 2 is: {result}")
```

Output:

```
The age multiplied by 2 is : 66
```

Explanation:

This code multiplies the age variable by 2 and prints the result, allowing for easy modification based on different age values ([Lutz, 2013]).

b. Display the name of the city, country, and continent

```
# Displaying information about the location
city = "Ouagadougou"
country = "Burkina Faso"
continent = "Africa"
print(f"City: {city}\nCountry: {country}\nContinent: {continent}")
```

Output:

```
City: Ouagadougou
Country: Burkina Faso
Continent: Africa
```

Explanation:

This code prints information about the city, country, and continent using formatted strings for better readability ([Downey, 2015]).

c. Display the examination schedule of our term

```
# Displaying examination schedule
exam_start_date = "11 January 2024"
exam_end_date = "14 January 2024"
```

```
print(f"Exam Schedule: {exam_start_date} to {exam_end_date}")
```

Output:

```
Exam Schedule: 11 January 2024 to 14 January 2024
```

Explanation:

The code displays the examination schedule with the start and end dates, providing a clear representation ([Rossum, 2003]).

d. Display the temperature ofcountry on the day the assignment

```
# Displaying the temperature of the country
temperature = 35  # Example temperature value
print(f"The temperature is {temperature} degrees Celsius.")
```

Output:

```
The temperature is 35 degrees Celsius.
```

Explanation:

This code prints the temperature of the country, and the value can be adjusted based on the actual temperature ([Lutz, 2013]).

Conclusion:

The combination of intentional experimentation and hands-on problem solving improves mastery of Python. The assignment promotes a hands-on, experiential approach to programming, facilitating a deeper understanding of Python syntax, semantics, and applications. You can find the code and outputs in the provided files: Programming_Assignment_Unit-1.py and Programming_Assignment_Unit-1.ipynb.

References:

- Downey, A. (2015). *Think Python: How to think like a computer scientist*. Green Tree Press. (Think Python)
- Jones, B. (2019). *Python Programming for Beginners*. Packt Publishing.
- Lutz, M. (2013). *Learning Python*. O'Reilly Media.
- Rossum, G. V. (2003). *Python Reference Manual*. Python Software Foundation.

Other resource:

- https://github.com/VirtuelsDev/UoPeople/blob/5fad7e099a7617494e26155fccbc9474e38eeaca/CS-1101_Programming-Fundamentals/Unit%201/Programming_Assignment_Unit-1.ipynb
- https://github.com/VirtuelsDev/UoPeople/blob/5fad7e099a7617494e26155fccbc9474e38eeaca/CS-1101_Programming-Fundamentals/Unit%201/Programming_Assignment_Unit-1.py