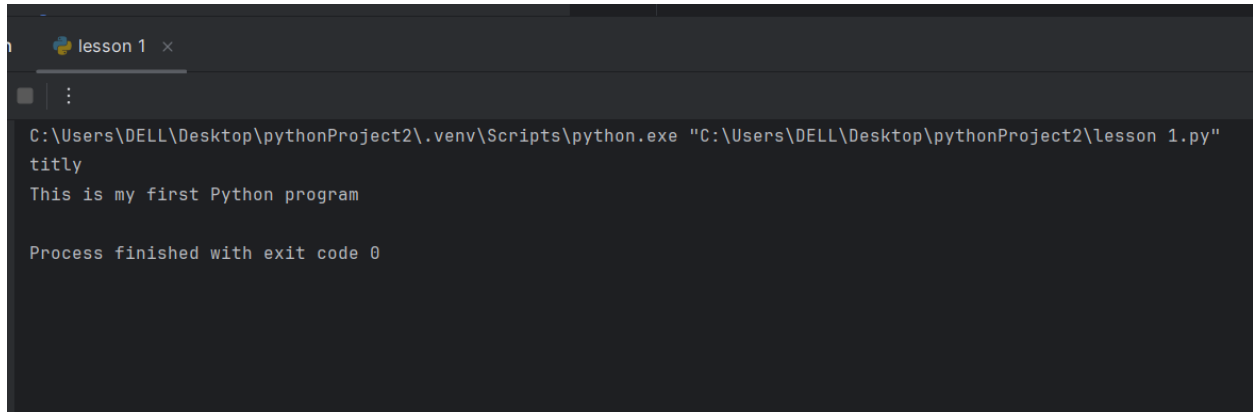


Lesson 1

Analysis: `print('titly')`: This line prints the string 'titly' to the console. The print function outputs whatever is inside the parentheses.

`print('This is my first Python program')`: This line prints the string 'This is my first Python program' to the console.

A terminal window titled 'lesson 1' showing the execution of a Python script. The command prompt shows the path to the Python executable and the script file. The output displays the string 'titly' followed by 'This is my first Python program' on a new line. The process ends with 'Process finished with exit code 0'.

```
C:\Users\DELL\Desktop\pythonProject2\.venv\Scripts\python.exe "C:\Users\DELL\Desktop\pythonProject2\lesson 1.py"
titly
This is my first Python program

Process finished with exit code 0
```

Lesson 2

Analysis: This code demonstrates basic variable assignment in Python. Variables `age`, `weight`, and `x` are assigned integer values. Python variables are dynamically typed, meaning you don't need to specify the type explicitly.

Printing Values: The `print()` function is used to display the values of these variables. Each `print()` call outputs a value to the console, one per line.

Formatting and Organization: For better readability and maintenance, you might want to add comments or organize the code into functions if it grows more complex.

A terminal window showing the execution of a Python script. The command prompt shows the path to the Python executable and the script file. The output displays the integer values 22, 75, and 5 on separate lines. The process ends with 'Process finished with exit code 0'.

```
C:\Users\DELL\Desktop\pythonProject2\.venv\Scripts\python.exe "C:\Users\DELL\Desktop\pythonProject2\lesson 2.py"
22
75
5

Process finished with exit code 0
```

Lesson 3

Analysis: String Concatenation: The script uses the `+` operator to concatenate strings and variables. This is a common way to combine strings with variable data in Python.

User Input: The `input()` function is used to get data from the user. It reads the input as a string, which is then stored in a variable.

Output Formatting: The script demonstrates basic string formatting by combining literals and user-provided data into a readable format.

This script is useful for learning how to handle user input and produce formatted output. If you want to improve it further, you could add error handling or additional formatting options to make it more robust.

```
C:\Users\DELL\Desktop\pythonProject2\.venv\Scripts\python.exe "C:\Users\DELL\Desktop\pythonProject2\lesson 3.py"
what is your name ?Titly
Username:Titly
where are you from ?Chittagong
Titly lives in Chittagong

Process finished with exit code 0
|
```

Lesson 4

Analysis: Type Conversion: It's crucial to convert the input to the correct type (int in this case) before performing arithmetic operations. Without this conversion, the arithmetic would involve string concatenation or result in an error.

Floating-Point Arithmetic: The division $9/5$ results in a floating-point number (i.e., 1.8), so the result of the conversion will also be a floating-point number. This is why the output includes a decimal point.

Error Handling: The script lacks error handling. If the user enters a non-numeric value, the script will crash. Adding error handling can make the script more robust.

```
Run
C:\Users\DELL\Desktop\pythonProject2\.venv\Scripts\python.exe "C:\
temperature in celsius
```

Lesson 5

Analysis: String Indexing: Strings in Python are indexed starting from 0 for the first character. Negative indexing counts from the end of the string, with -1 representing the last character.

Slicing: The slicing notation [start:end] extracts a portion of the string from the start index up to, but not including, the end index. If the start index is omitted, it defaults to 0. If the end index is omitted, it defaults to the end of the string.

String Immutability: Strings in Python are immutable, meaning they cannot be changed after they are created. Operations like slicing create new strings rather than modifying the original string.

This script demonstrates basic string manipulation techniques, which are essential for many programming tasks involving text processing.

```
C:\Users\DELL\Desktop\pythonProject2\.venv\Scripts\python.exe "C:\Users\DELL\Desktop\pythonProject2\lesson 5.py"
Learn "Python" Programming
I am titly.This is my python course. And This is the best python course available online.
g
Lea
earn Python Programmin

Process finished with exit code 0
```

Lesson 6

Analysis: Discussion

String Concatenation:

Pros: It's simple and works in all versions of Python.

Cons: It can become unwieldy with more complex strings and expressions. Each part of the string must be manually concatenated, which can be error-prone and less readable.

F-Strings:

Pros:

Readability: F-strings improve readability by allowing you to embed variables and expressions directly within the string.

Performance: F-strings are generally faster than other string formatting methods like `str.format()` or `%` formatting.

Flexibility: You can include expressions, not just variables, and the syntax is clean and intuitive.

Cons: Requires Python 3.6 or newer. If you're working in an environment with an older Python version, f-strings won't be available.

Comparison Readability: F-strings tend to be more readable and concise, making the code easier to understand at a glance.

Performance: F-strings are typically faster because they are evaluated at runtime and require less processing compared to concatenation, especially when dealing with multiple variables.

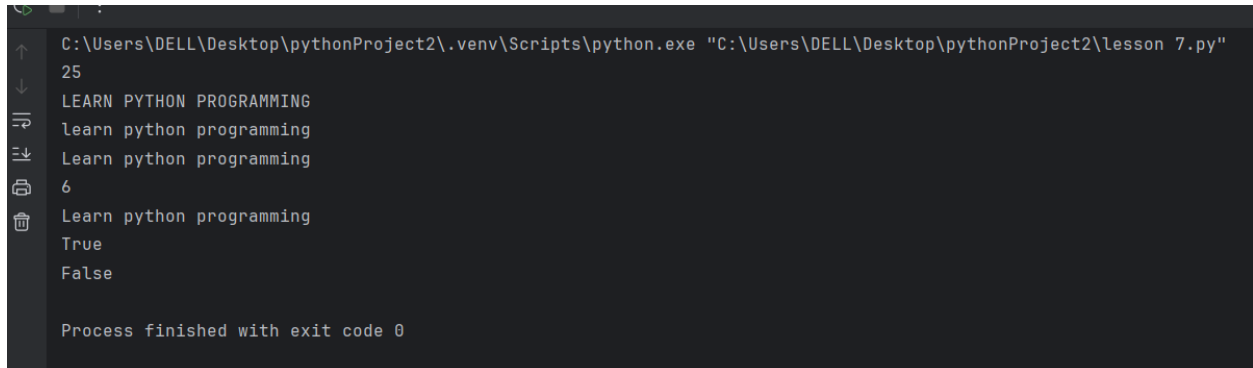
In summary, while both methods produce the same result in this case, f-strings offer a more modern and cleaner approach to string formatting, especially as complexity grows. For newer Python code, f-strings are generally recommended for their clarity and efficiency.

```
C:\Users\DELL\Desktop\pythonProject2\.venv\Scripts\python.exe "C:\Users\DELL\Desktop\pythonProject2\lesson 6.py"
titly(nf)teaches ai
titly(nf) teaches ai

Process finished with exit code 0
```

Lesson 7

Analysis: This script covers basic and essential string operations in Python, providing a good foundation for text manipulation tasks. Each method and operation has specific use cases, and understanding them helps in various text processing scenarios.



```
C:\Users\DELL\Desktop\pythonProject2\.venv\Scripts\python.exe "C:\Users\DELL\Desktop\pythonProject2\lesson 7.py"
25
LEARN PYTHON PROGRAMMING
learn python programming
Learn python programming
6
Learn python programming
True
False

Process finished with exit code 0
```

Lesson 8

Analysis: Basic Arithmetic Operations:

Subtraction (-): Computes the difference between two numbers.

Multiplication (*): Computes the product of two numbers.

Addition (+): Computes the sum of two numbers.

Division (/): Computes the quotient, resulting in a floating-point number.

Integer Division (//): Computes the quotient and returns the integer part only, discarding the remainder.

Modulus (%): Computes the remainder after division.

Operator Precedence:

Exponentiation (**) has the highest precedence, followed by multiplication (*), division (/), and addition (+). Parentheses can be used to group operations and control the order of evaluation.

Variable Assignment and Operations:

Variables can be used to store values and update them with arithmetic operations. This is useful for managing and manipulating data throughout a program.

This script covers fundamental arithmetic operations, variable manipulation, and demonstrates how to work with different types of numerical results in Python. Understanding these basics is essential for more advanced programming and problem-solving tasks.

```
C:\Users\DELL\Desktop\pythonProject2\.venv\Scripts\python.exe "C:\Users\DELL\Desktop\pythonProject2\.venv\Scripts\python.exe"
1
15
11
8.888888888888889
8
2
12
0
32

Process finished with exit code 0
```

Lesson 9

Analysis: `math.ceil()`: Useful when you need to round up to the nearest integer, often used in scenarios involving pagination or when exact rounding up is necessary.

`math.floor()`: Useful for rounding down to the nearest integer, which is often used in scenarios where you need to discard fractional parts and work with whole numbers.

`math.factorial()`: Calculates the factorial of a number, which is commonly used in permutations, combinations, and other mathematical computations.

`round()`: A built-in function for rounding floating-point numbers to the nearest integer. It can also accept a second argument to specify the number of decimal places.

`abs()`: Returns the absolute value of a number, which is useful in many contexts where only the magnitude of the number is relevant, regardless of its sign.

This script demonstrates key functions from both the `math` module and Python's built-in functions for numerical operations. Understanding these functions helps in performing various mathematical and data manipulation tasks, enhancing your ability to handle and process numerical data effectively in Python.

```
C:\Users\DELL\Desktop\pythonProject2\.venv\Scripts\python.exe "C:\Users\DELL\Desktop\pythonProject2\.venv\Scripts\python.exe"
6
5
120
6
5.7

Process finished with exit code 0
```

Lesson 10

Analysis: Boolean Values: The boolean variable `is_rainy` is used to control the flow of the program. It can be either `True` or `False`, which makes it ideal for conditional checks.

Conditional Statements:

`if`: Executes the block of code if the condition is `True`.

`else`: Executes the block of code if the condition is `False`.

Message Content:

The message 'carry umbrella' is appropriate for rainy weather.

The message 'carry not umbrella' seems awkward. A more natural phrasing would be 'don't carry an umbrella'.

Improved Code Example

For better readability and clarity, you might want to adjust the message for the `else` case:

python

Copy code

```
is_rainy = True
```

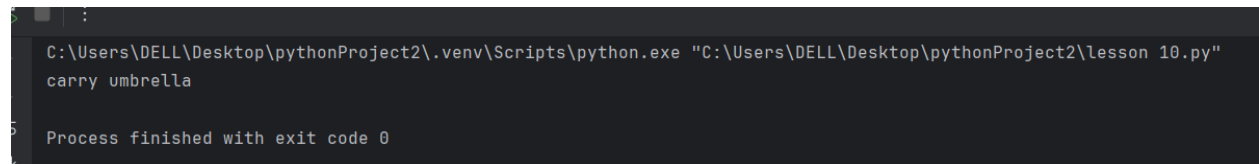
```
if is_rainy:
```

```
    print('Carry an umbrella')
```

```
else:
```

```
    print('No need for an umbrella')
```

This script demonstrates a basic use of conditional statements to handle different scenarios based on a boolean variable. Understanding how to use `if-else` statements is fundamental for making decisions in Python programming.



```
C:\Users\DELL\Desktop\pythonProject2\.venv\Scripts\python.exe "C:\Users\DELL\Desktop\pythonProject2\lesson 10.py"
carry umbrella

Process finished with exit code 0
```

Lesson 11

Analysis: This script demonstrates how to use logical operators (`and`, `or`) in conditional statements to control the flow of the program based on multiple conditions. Understanding these operators helps in making more complex decisions in Python programming.

```
C:\Users\DELL\Desktop\pythonProject2\.venv\Scripts\python.exe "C:\Users\DELL\Desktop\pythonProject2\lesson 11.py"
we are interested

Process finished with exit code 0
```

Lesson 12

Analysis: This script demonstrates how to use conditional statements to evaluate different scenarios based on the value of price. Understanding and correctly implementing these conditions helps in making clear and accurate decisions in your programs.

```
C:\Users\DELL\Desktop\pythonProject2\.venv\Scripts\python.exe "C:\Users\DELL\Desktop\pythonProject2\lesson 12.py"
the price is not 10
it is expenssive
it is cheap

Process finished with exit code 0
```

Lesson 13

Analysis: The loop prints the numbers from 0 to 4 and then prints the final message after the loop completes. This is a straightforward demonstration of how a while loop operates, incrementing a counter until a specified condition is no longer met.

```
C:\Users\DELL\Desktop\pythonProject2\.venv\Scripts\python.exe "C:\Users\DELL\Desktop\pythonProject2\lesson 13.py"
0
1
2
3
4
the while loop ended

Process finished with exit code 0
```

Lesson 14

Analysis: break Statement: It exits the loop immediately when the correct guess is made.

else Clause: This is unique to while loops and for loops in Python. It runs only if the loop completes normally, meaning it wasn't exited via break.

This pattern effectively handles both successful and unsuccessful attempts while allowing for clear feedback to the user.

```
C:\Users\DELL\Desktop\pythonProject2\.venv\Scripts\python.exe "C:\Users\DELL\Desktop\pythonProject2\lesson 14.py"
Guess the price: 6
Guess the price: 1
Guess the price: 4
Guess the price: 2
Guess the price: 10
you've won!

Process finished with exit code 0
```

Lesson 15

Analysis: First Loop: Processes and prints characters of a string.

Second Loop: Processes and prints elements of a tuple.

Third Loop: Demonstrates an edge case with the range function where the step is larger than the range.

Fourth Loop: Calculates and prints the sum of values in a list.

Each loop showcases different ways to iterate over collections or ranges in Python and highlights some common patterns and behaviors.

```
C:\Users\DELL\Desktop\pythonProject2\.venv\Scripts\python.exe "C:\Users\DELL\Desktop\pythonProject2\LESSON 15.py"
p
y
t
h
o
n
eng
rice
oil
5
270

Process finished with exit code 0
```

Lesson 16

Analysis: The outer loop iterates over x values 0, 1, 2, and 3.

For each x, the inner loop iterates over y values 0, 1, 2, and 3.

As a result, every combination of (x, y) is printed in a grid-like pattern, where x changes more slowly than y.

The output displays a Cartesian product of the ranges for x and y. This structure is often useful for generating grids, matrices, or combinations of values.


```
C:\Users\DELL\Desktop\pythonProject2\.venv\Scripts\python.exe "C:\Users\DELL\Desktop\pythonProject2\lesson 16.py"
(0,0
(0,1
(0,2
(0,3
(1,0
(1,1
(1,2
(1,3
(2,0
(2,1
(2,2
(2,3
(3,0
(3,1
(3,2
(3,3

Process finished with exit code 0
```

Lesson 17

Analysis: The list modification replaced an element with a nested list and demonstrated slicing to access part of the updated list.

The loop efficiently finds the maximum value in a list of integers by iterating through the elements and updating a variable whenever a higher value is found.

Overall, these operations illustrate basic list manipulation and iteration techniques in Python.

```
C:\Users\DELL\Desktop\pythonProject2\.venv\Scripts\python.exe "C:\Users\DELL\Desktop\pythonProject2\lesson 17.py"
[['bread', 'egg']
['salt', ['bread'], 'egg', 'rice']]
98

Process finished with exit code 0
```

Lesson 18

Analysis: Matrix Initialization and Modification:

Demonstrates how to access, modify, and print elements of a 2D list.

Nested Loop for Iteration:

Shows how to use nested loops to iterate through a 2D list, allowing access to every individual element.

This code effectively demonstrates how to work with matrices (2D lists) in Python, including basic operations like access, modification, and iteration.

```
C:\Users\DELL\Desktop\pythonProject2\.venv\Scripts\python.exe "C:\Users\DELL\Desktop\p
[[1, 3], [4, 5], [6, 7]]
3
[[1, 45], [4, 5], [6, 7]]
1
45
4
5
6
7

Process finished with exit code 0
```

Lesson 19

Analysis: List Modifications: Demonstrated adding, inserting, removing, sorting, and reversing operations on a list.

Membership and Counting: Checked for the presence of elements and counted occurrences.

Copying and Clearing: Showed how to copy a list and then clear its contents.

This code provides a good overview of various list operations in Python, illustrating how to manipulate and interact with lists effectively.

```
C:\Users\DELL\Desktop\pythonProject2\.venv\Scripts\python.exe "C:\Users\DELL\Desktop\pythonProject2\lesson 19.py"
[20, 14, 10, 7, 5, 1, 1]
True
False
2
2
[]

Process finished with exit code 0
```

Lesson 20

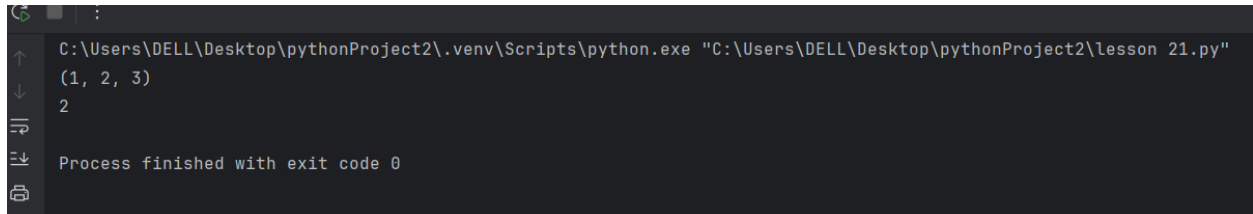
Analysis: this code takes a list of numbers and removes duplicates by adding only the first occurrence of each number to a new list, which is then printed.

```
C:\Users\DELL\Desktop\pythonProject2\.venv\Scripts\python.exe "C:\Users\DELL\Desktop\pythonProject2\LESSON 20.py"
[1, 2, 4, 5, 7, 8, 10]

Process finished with exit code 0
```

Lesson 21

Analysis in this code, `number_tuple` is a tuple containing the values 1, 2, and 3. The `print(number_tuple)` statement prints the entire tuple, while `print(number_tuple[1])` prints the second element of the tuple, which is 2 (since indexing starts at 0).

A terminal window showing the execution of a Python script. The command is `C:\Users\DELL\Desktop\pythonProject2\.venv\Scripts\python.exe "C:\Users\DELL\Desktop\pythonProject2\lesson 21.py"`. The output is `(1, 2, 3)` followed by `2` on the next line. The process finished with exit code 0.

```
C:\Users\DELL\Desktop\pythonProject2\.venv\Scripts\python.exe "C:\Users\DELL\Desktop\pythonProject2\lesson 21.py"
(1, 2, 3)
2
Process finished with exit code 0
```


Lesson 22

Analysis: The tuple `number_tuple` contains three integers.

You can sum the elements of a tuple in different ways: by indexing, using unpacked variables, or with other methods.

The output for summing the values in the tuple will be 6 in both cases.

Make sure to use `print` in lowercase to avoid syntax errors.

A terminal window showing the execution of a Python script. The command is `C:\Users\DELL\Desktop\pythonProject2\.venv\Scripts\python.exe "C:\Users\DELL\Desktop\pythonProject2\lesson 22.py"`. The output is `(1, 2, 3)` followed by `6` on the next line, and `6` on the line after that. The process finished with exit code 0.

```
C:\Users\DELL\Desktop\pythonProject2\.venv\Scripts\python.exe "C:\Users\DELL\Desktop\pythonProject2\lesson 22.py"
(1, 2, 3)
6
6
Process finished with exit code 0
```

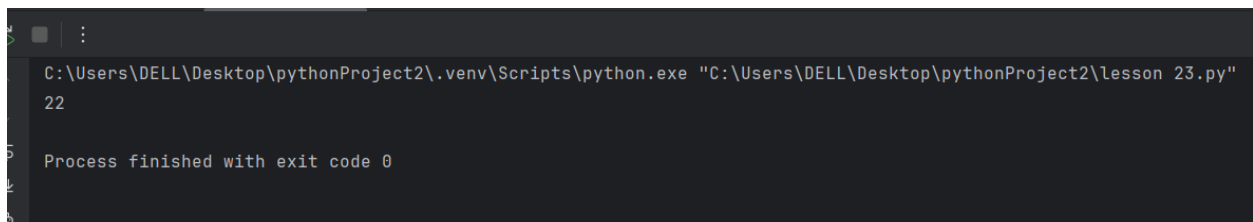
Lesson 23

Analysis : Dictionary Definition: The User dictionary holds a name and an age.

Accessing Values: Use `User['Age']` to get the value associated with the key 'Age'.

Printing: Use `print()` with the lowercase function name to display the value.

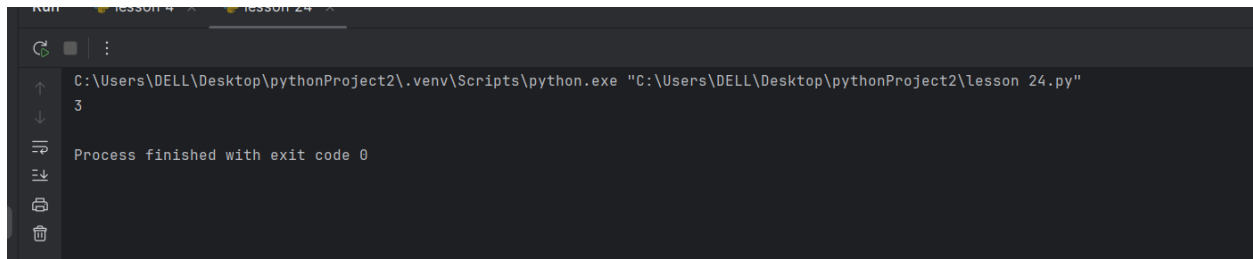
The c code will print 22, which is the value associated with the 'Age' key in the User dictionary.

A terminal window showing the execution of a Python script. The command is `C:\Users\DELL\Desktop\pythonProject2\.venv\Scripts\python.exe "C:\Users\DELL\Desktop\pythonProject2\lesson 23.py"`. The output is `22`. The process finished with exit code 0.

```
C:\Users\DELL\Desktop\pythonProject2\.venv\Scripts\python.exe "C:\Users\DELL\Desktop\pythonProject2\lesson 23.py"
22
Process finished with exit code 0
```

Lesson 24

Analysis: This will define a function named `add` that takes two parameters and returns their sum. The print statement will then output the result of calling `add(1, 2)`, which is 3.



The screenshot shows a terminal window with a dark background. At the top, there are tabs for 'Run', 'Lesson 4', and 'Lesson 24'. The terminal displays the command `C:\Users\DELL\Desktop\pythonProject2\.venv\Scripts\python.exe "C:\Users\DELL\Desktop\pythonProject2\lesson 24.py"` and the output `3`. Below the output, it says 'Process finished with exit code 0'. On the left side of the terminal, there is a vertical toolbar with icons for running, stepping through, and other debugging actions.

```
C:\Users\DELL\Desktop\pythonProject2\.venv\Scripts\python.exe "C:\Users\DELL\Desktop\pythonProject2\lesson 24.py"
3
Process finished with exit code 0
```