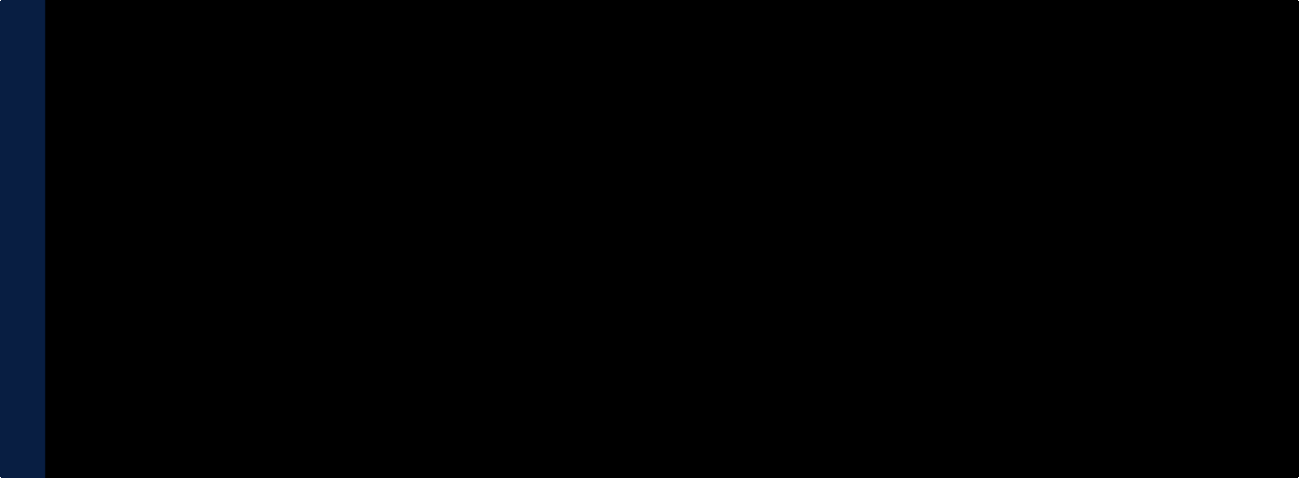
Learning Guide Unit 1

# PART 1

1. Trying to print my name with possible errors (Output)



1

2

3

4

5

6

7

8

9

10

11

12

13

14

>>> **print** ('T. Anguh Bleaise')

T. Anguh Bleaise

>>> **print** ('T. Anguh Bleaise)

File "<stdin>", line 1

**print** ('T. Anguh Bleaise)

^

SyntaxError: unterminated string literal (detected at line 1)

>>> **print** (T. Anguh Bleaise)

File "<stdin>", line 1

**print** (T. Anguh Bleaise)

^^^^^^^^^^^^^^^^

SyntaxError: invalid syntax. Perhaps you forgot a comma?

Explanations:

When I try >>> print ('T. Anguh Bleaise) , i get an error because printing something like your name will be in the form of a string. In Python, a print statement has to identify the start and the end of a statement. According to the error, Python is able to tell me that

start with

File "<stdin>", line 1

SyntaxError: unterminated string literal (detected at line 1)

incomplete.

but is unable to find where my string ends which is the missing ending quote. This error

means it is unable to print this statement as a string as it is

When I >>> print (T. Anguh Bleaise) , omitting both strings, I get

SyntaxError: invalid syntax. Perhaps you forgot a

comma? . This is not a valid syntax to print a string. It would have been good to print as as shown in the code above.

T. Anguh Bleaise



\*



\*\*

resulting to

>>> print ('T. Anguh Bleaise')

1. In Python, both the

and

operators are used for arithmetic operations, but they serve different purposes:

* 1. Operator:



\*

* + 1. The operator is used for multiplication.



\*

* + 1. When used with numbers, it multiplies them together.
    2. When used with sequences like lists or tuples, it performs repetition (concatenation of the sequence with itself a certain number of times).
  1. Operator:



\*\*

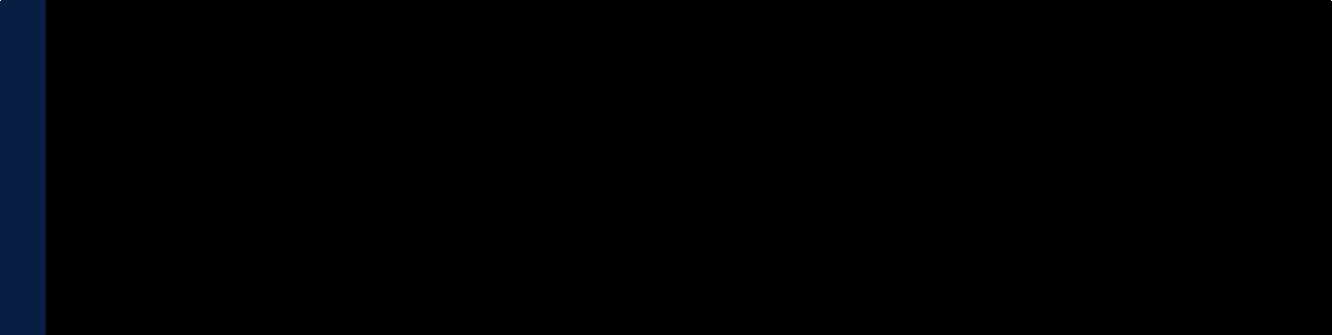
* + 1. The operator is used for exponentiation (raising a number to a power).



\*\*

* + 1. When used with two numbers, it raises the first number to the power of the second number.

Example output



1

2

3

4

5

6

7

8

9

*# \* operator for multiplication*

>>> 2 \* 3

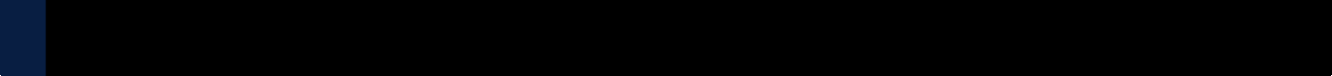
6

*# \* operator for repetition*

>>> [1,2,3] \* 3

[1, 2, 3, 1, 2, 3, 1, 2, 3]

*# \*\* operator for exponentiation*



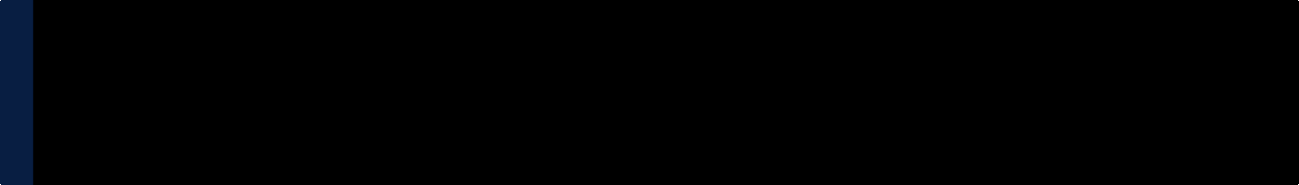
10 >>> 2 \*\* 3

11 8

1. In Python, it is not possible to display an integer leaded by zero (0). e.g., 09, 04, 02, and 01 etc. This is because in Python, leading zeros in integer literals indicate an octal (base 8) number. However, integers cannot start with a leading zero if the value exceeds (octal 7), as this would result in a syntax error as below.

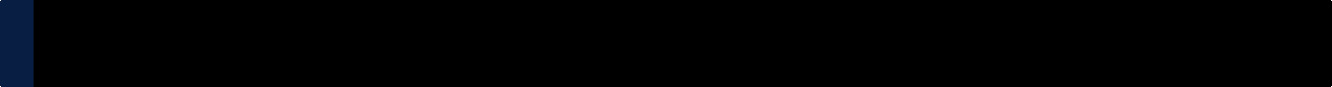


0o7

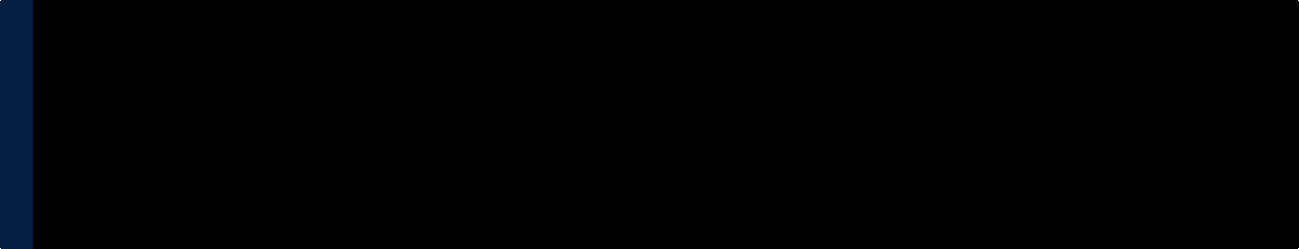


1. >>> **print** (01)
2. File "<stdin>", line 1
3. **print** (01)
4. ^
5. SyntaxError: leading zeros **in** decimal integer literals are **not** permitted; use an 0o prefix **for** octal integers

For me, If I need to represent a number with a leading zero for display purposes in Python 3, I can convert it to a string and then display it:



1. >>> **print** ('09')
2. 09
3. Differences between commands type('67') and type(67)



1

2

3

4

5

6

7

*# value type for string*

>>> type('67')

<**class** 'str'>

*# value type for integers*

>>> type(67)

<**class** 'int'>

# Explanation:

>>> type('67') . This command checks the type of the string '67' . The output will be <class 'str'> , indicating that string data type.

>>> type(67) . This command checks the type of the integer 67 . The output will be <class 'int'> , indicating that data type.



67

'67'



67

is a

is an integer

'67'

The difference in output occurs because

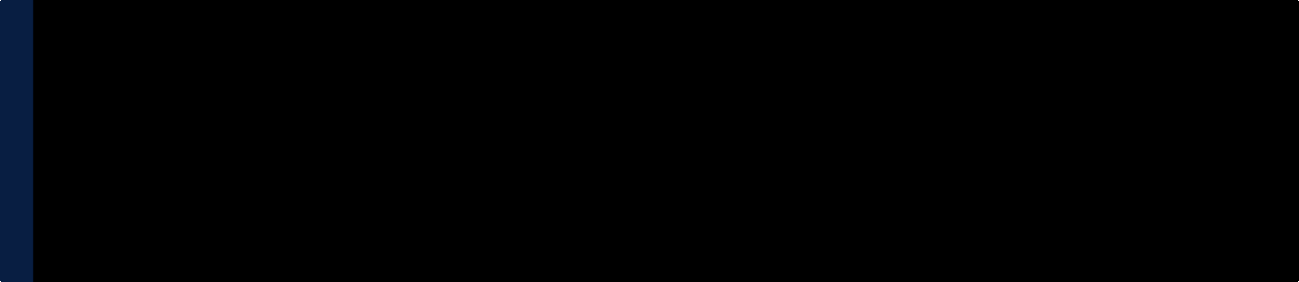
is enclosed in single quotes, making it a string literal, while

is a numeric literal without

quotes, making it an integer. Python treats these as different data types, So it displays different output when checking their types.

# PART 2

1. Displaying my age multiplied by 2



1

2

3

4

5

6

7

8

*# My age (32), multiplied by 2*

>>> **print** (32 \* 2)

64

*# Declaring a variable*

>>> age = 32

>>> **print** (age \* 2) 64

# Explanation:

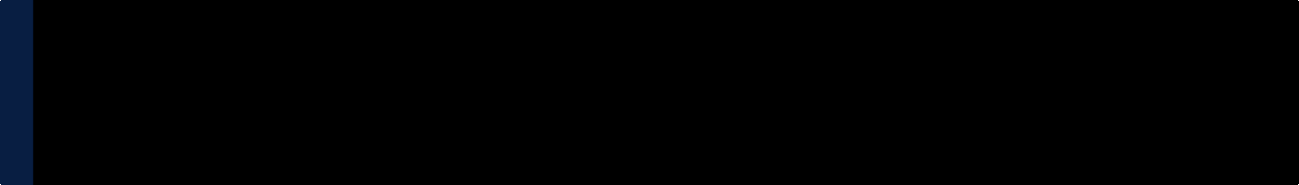
Input required here are both integers.

The multiplication operator (\*) in Python is used to obtain the result.

A variable can be declared to hold/store my current age so it can be used else where. Results belongs to of type

<class 'int'>

1. Displaying the name of the city, country and continent I am living in.

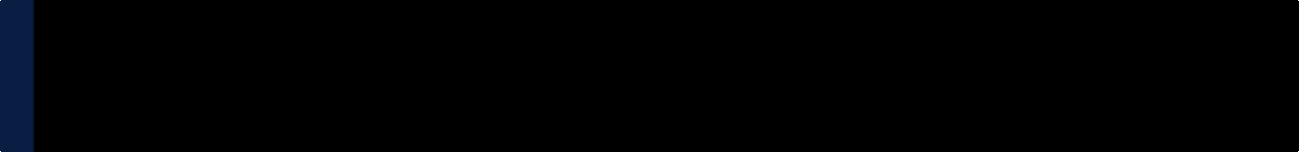


1. >>> City = 'Douala'
2. >>> Country = 'Cameroon'
3. >>> Continent = 'Africa'
4. >>> **print** ('My city is ' +City+ ', country is ' +Country+ ', and continent is ' +Continent)
5. My city **is** Douala, country **is** Cameroon, **and** continent **is** Africa

# Explanation:

I declared 3 variables to store my current city, country and continent (Storing it as a string). After, I used sting concatenation in Python to display as a sentence.

1. Displaying examination schedule (i.e., the starting and the ending day) of this term



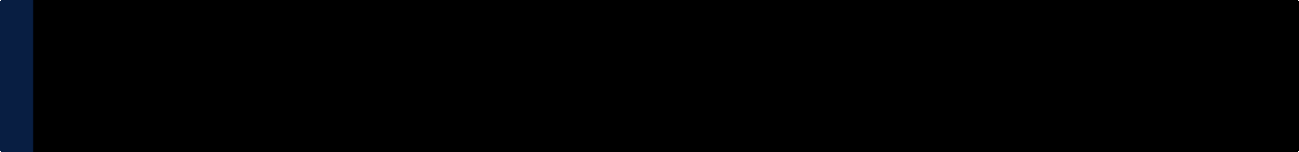
1. >>> start\_date = '6 June 2024, 12:04 AM (GMT-5)'
2. >>> end\_date = '9 June 2024, 11:55 PM (GMT-5)'
3. >>> **print** ('The Examination schedule for this term starts on ' +start\_date+ 'and ends on ' +end\_date)
4. The Examination schedule **for** this term starts on 6 June 2024, 12:04 AM (GMT-5)**and** ends on 9 June 2024, 11:55 P

# Explanation:

Start date and end date for exam schedule are store in a string variable.

Print statement is used along side string concatenation in Python to display the result.

1. Display the temperature of your country on the day the assignment is attempted by you



1 >>> current\_date = '15/04/2024'

1. >>> temp\_of\_day = '30 degrees celsius'
2. >>> **print** ('Current temperature of ' +current\_date+ ' is '+temp\_of\_day)
3. Current temperature of 15/04/2024 **is** 30 degrees celsius

# Explanation:

Current date and end temperature of the day are store in a string variable.

Print statement is used along side string concatenation in Python to display the result.

# Lessons learned from Part 2 :

1. In Python, variables are used to store data values. Unlike some other programming languages, Python does not require explicit declaration of variables before they are used. When you assign a value to a variable, Python automatically creates the variable and assigns the value to it. They can be reassigned and also helps in code debugging.
2. String concatenation can be used in combining multiple strings into a single string. We can use the + operator which is the most common way to concatenate a string.