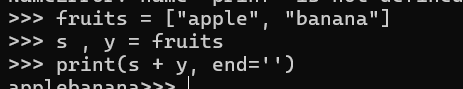
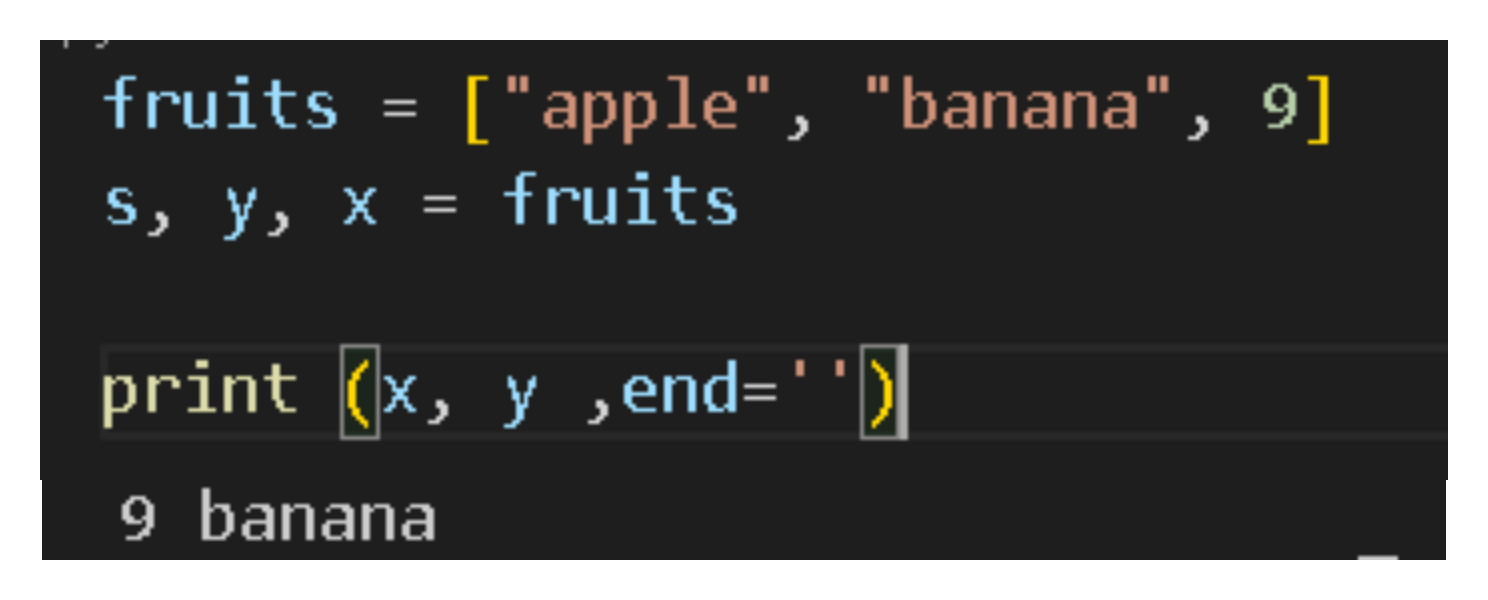
1. **Python - Output Variables :**

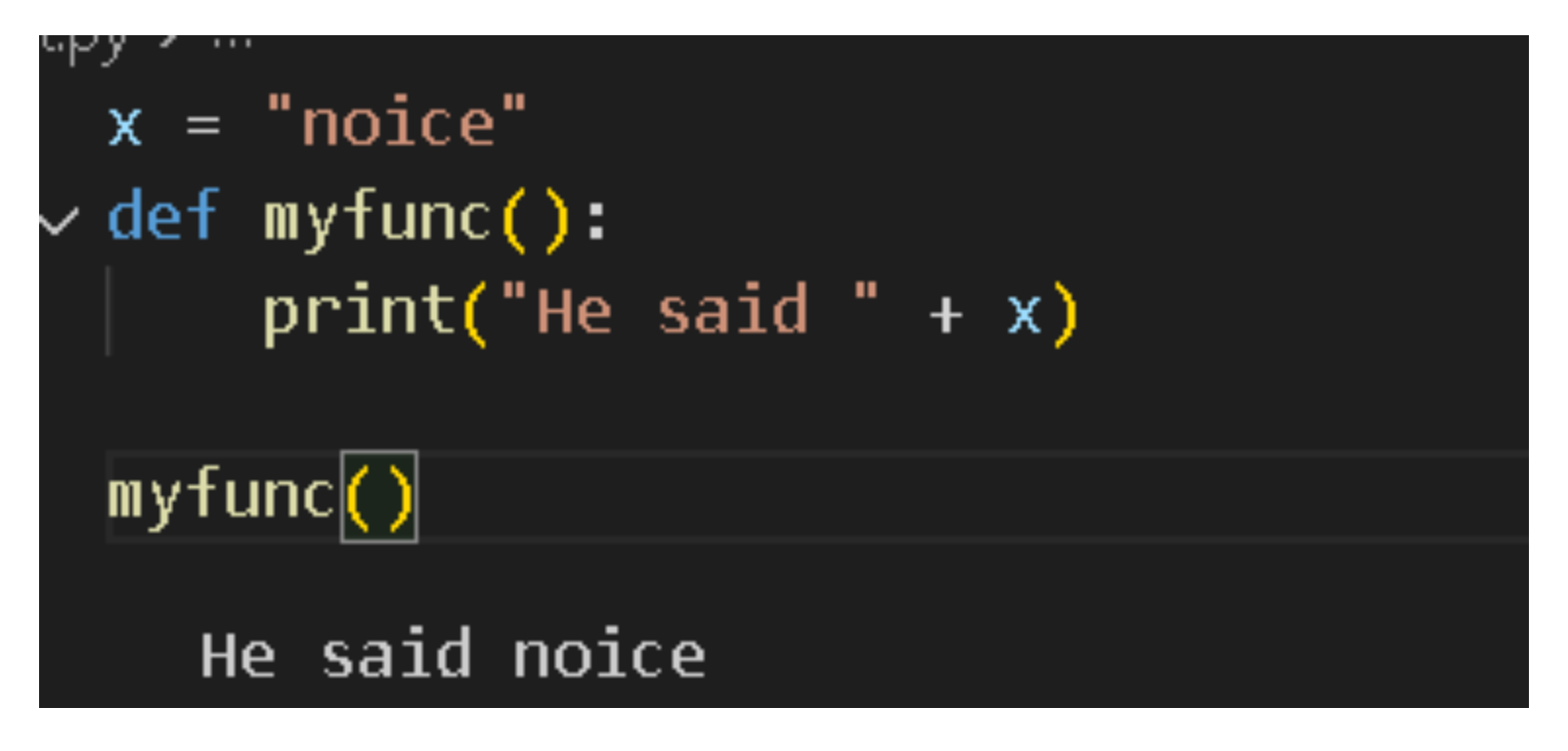
**=> Use “ + “ operator to output multiple variables ( *the same types* )**

****

**=> Use “ , “ operator to output the different types :**

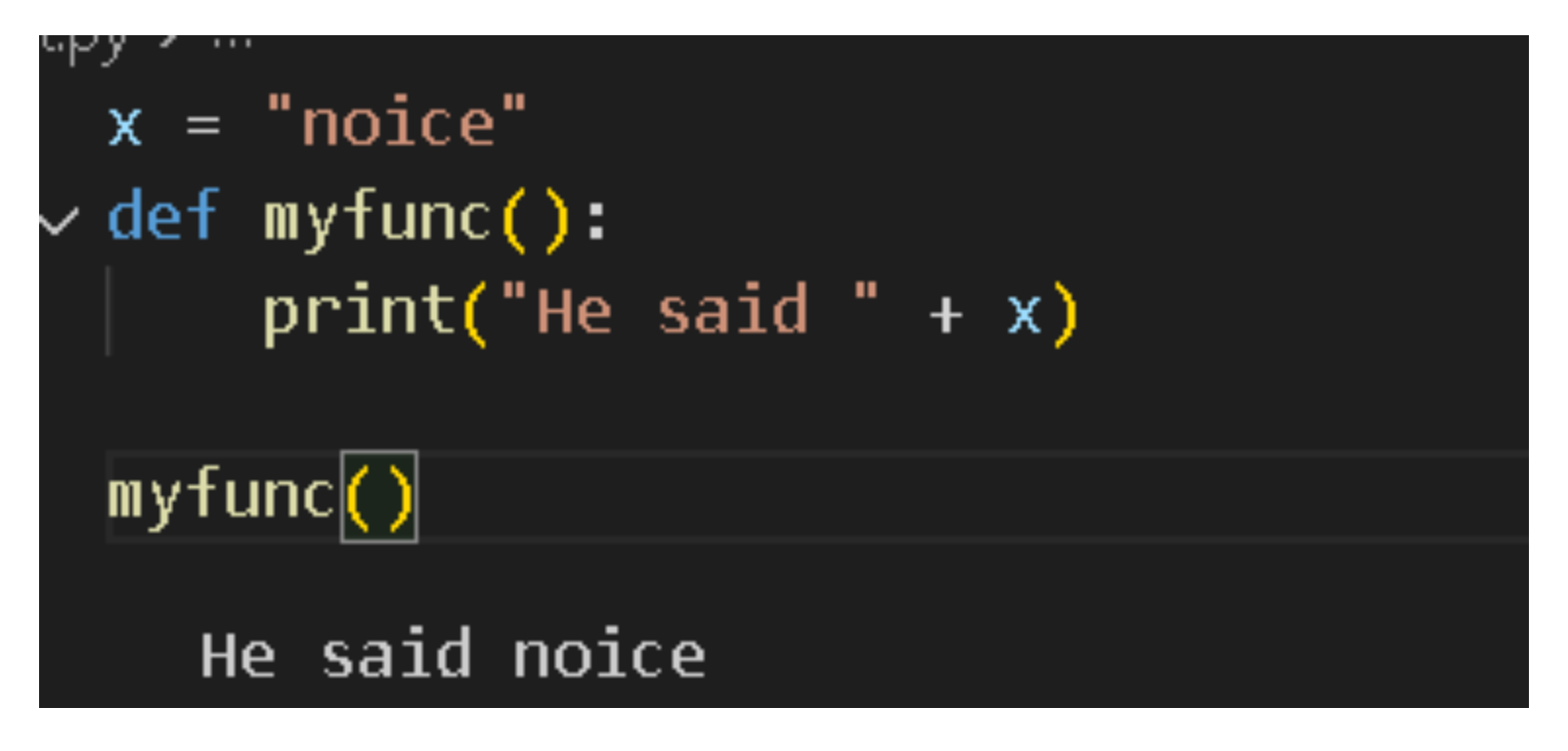
****

**=> Use “ + “ operator to output different types in print :**

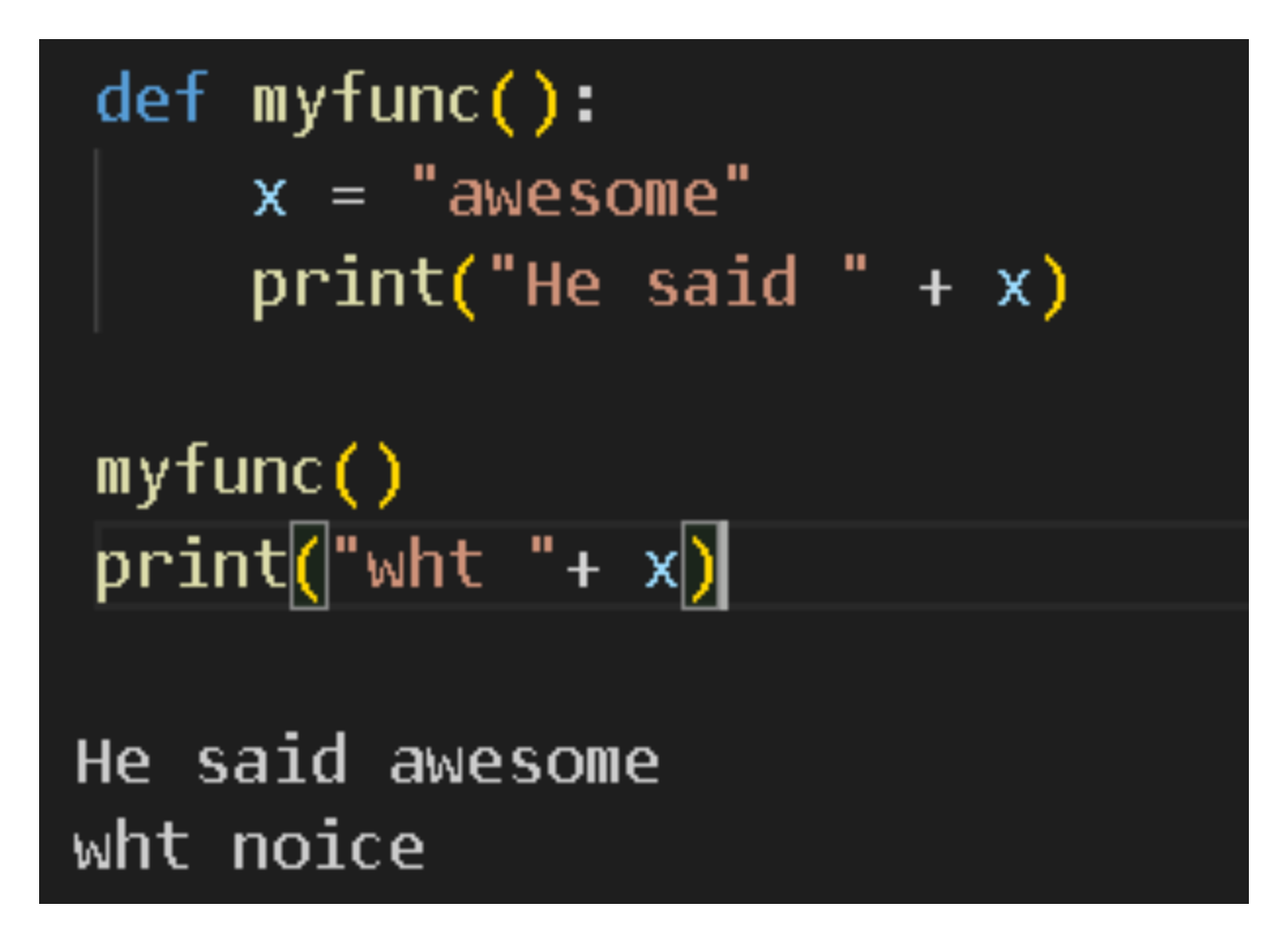
****

1. **Python - Global variables :** 
   1. **Global variables:**

* **Created outside of a function :**

****

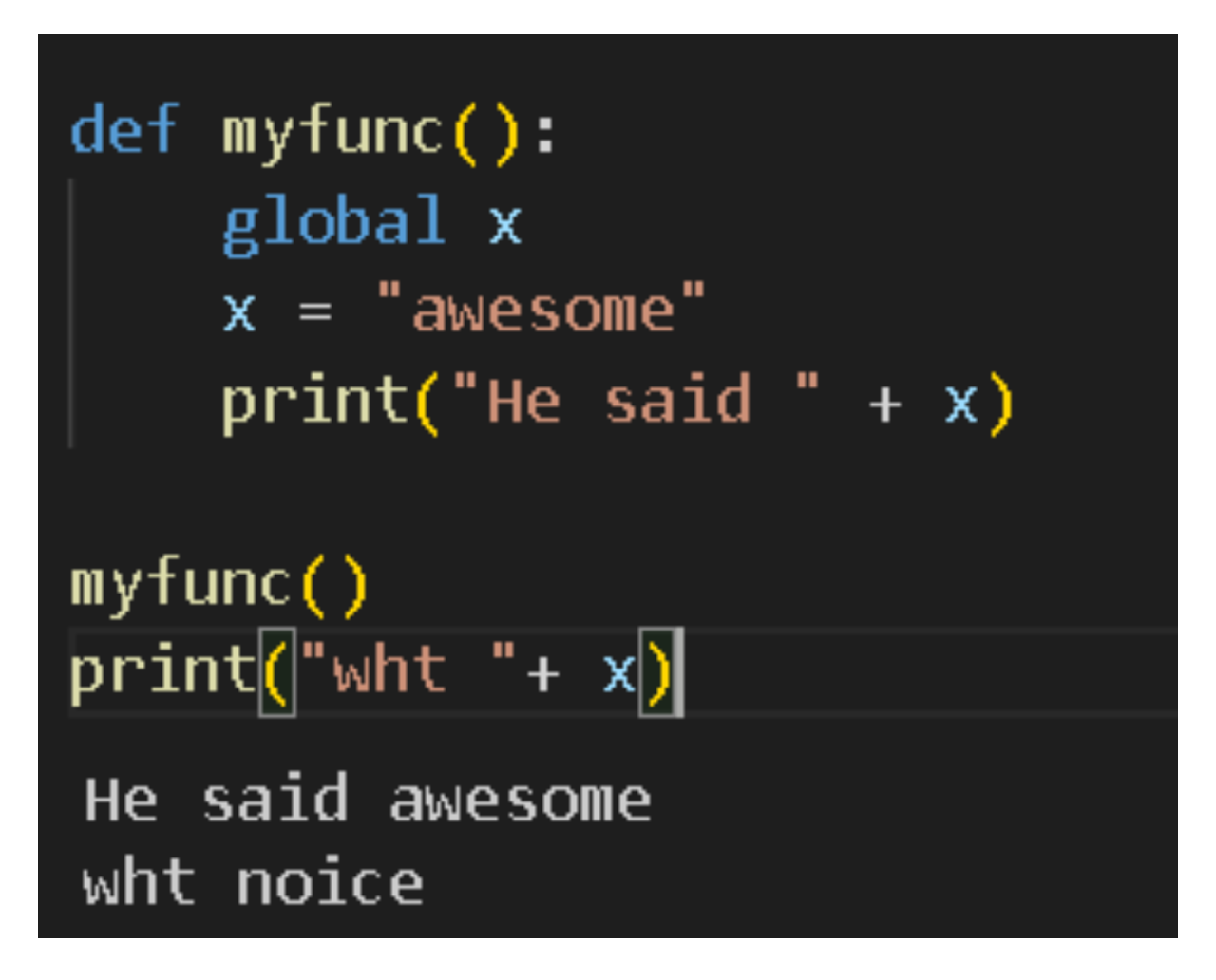
* **Or**

****

**=> If a variable is the same name inside a function -> this variable will be local, can only be used inside the function**

**=> The global variable with the same name will remain as it was, global and with the original value.**

* 1. **The global Keyword :**
* **If a variable is inside a function, using “Global” will make it global -> Mean : make that variable be used outside that function:**

****

* **We can use “global” for a variable outside a function:**

****

1. **Python Data Types :**

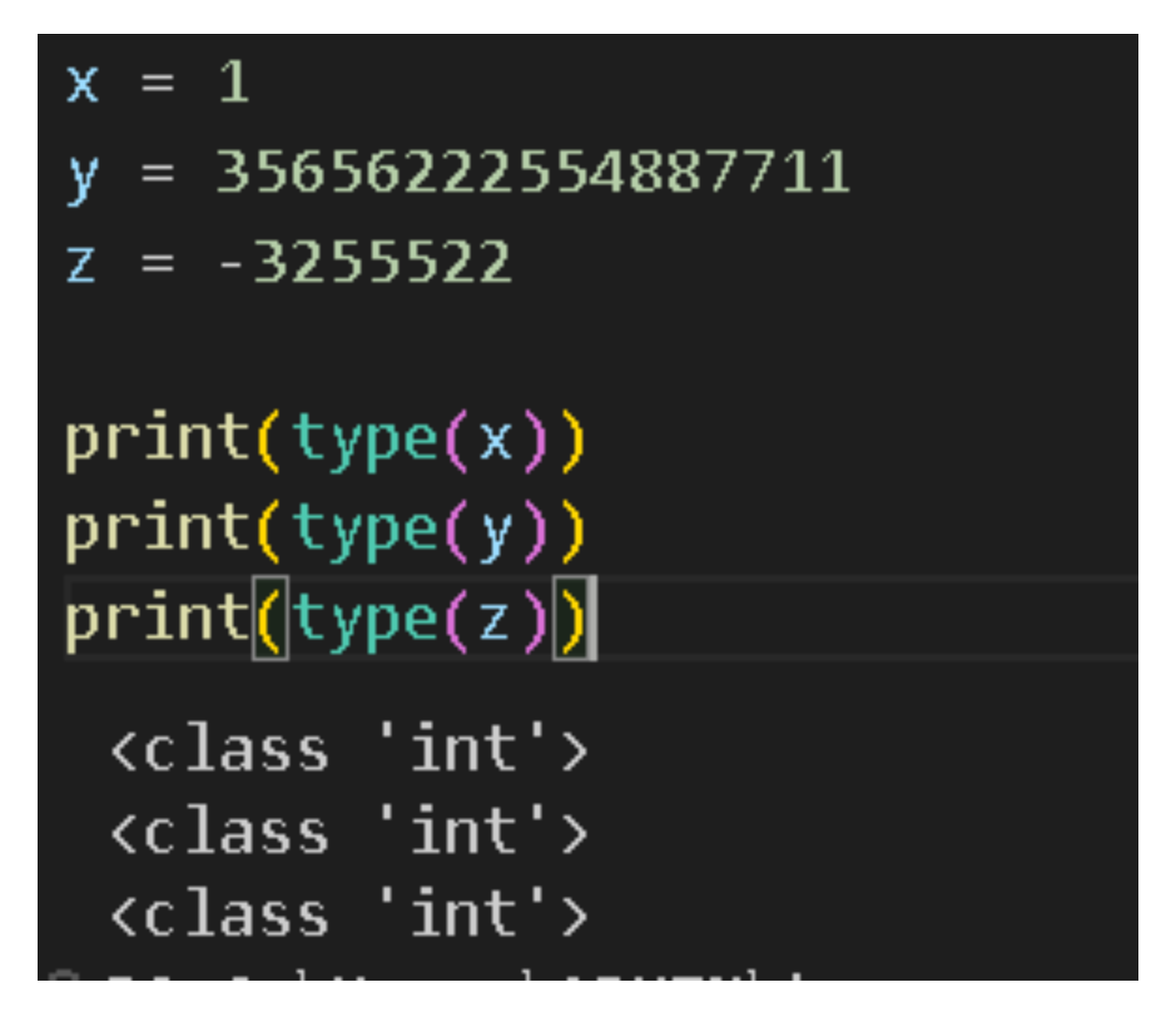
| **Text Type** | **str** |
| --- | --- |
| **Numeric Types** | **int, float, complex** |
| **Sequence Types** | **List, tuple, range** |
| **Mapping Type** | **dict** |
| **Set Types** | **set, frozenset** |
| **Boolean Type** | **bool** |
| **Binary Types** | **Bytes, bytearray, memory view** |
| **None Type** | **NoneType** |

* 1. **Getting the Data Type : Use ( print(type(x)) )**
  2. **Setting the Data Type :**

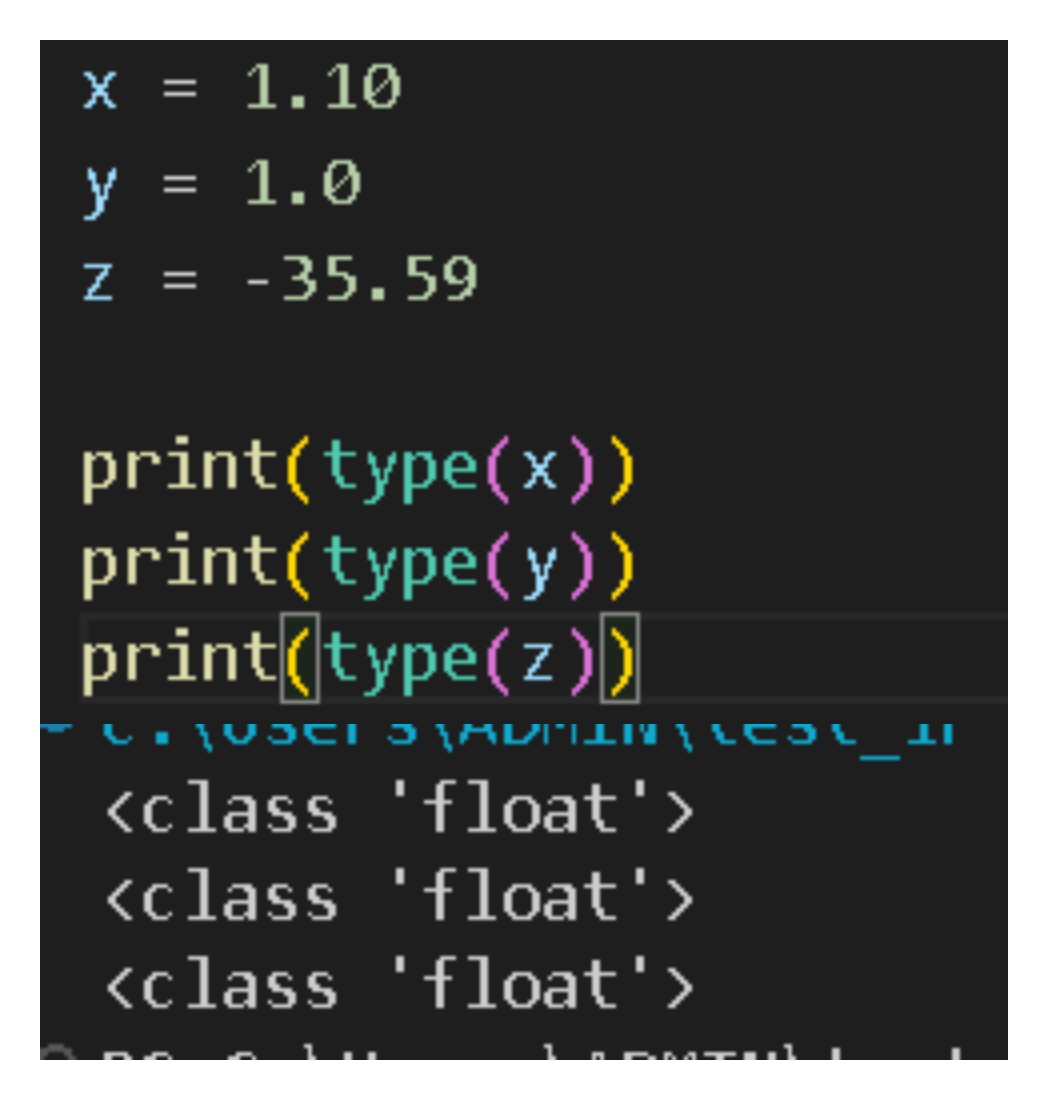
| **Example** | **Data Type** |
| --- | --- |
| **x = “nice”** | **str** |
| **x = 20** | **int** |
| **x = 20.5** | **float** |
| **x = 1j** | **complex** |
| **x = [“apple”, “banana”]** | **list** |
| **x = (“apple”, “banana”)** | **tuple** |
| **x = range(6)** | **range** |
| **x = {“name” : John, “age” : 36 }** | **dict** |
| **x={“apple”, “banana”}** | **set** |
| **x=frozenset({“apple”, “banana”})** | **frozenset** |
| **x=True** | **bool** |
| **x=b”Hello”** | **bytes** |
| **x=bytearray(5)** | **bytearray** |
| **x=memoryview(byte(5))** | **memoryview** |
| **x=None** | **NoneType** |

1. **Python Numbers**
   1. **Python Numbers :** 
      1. **Int :**

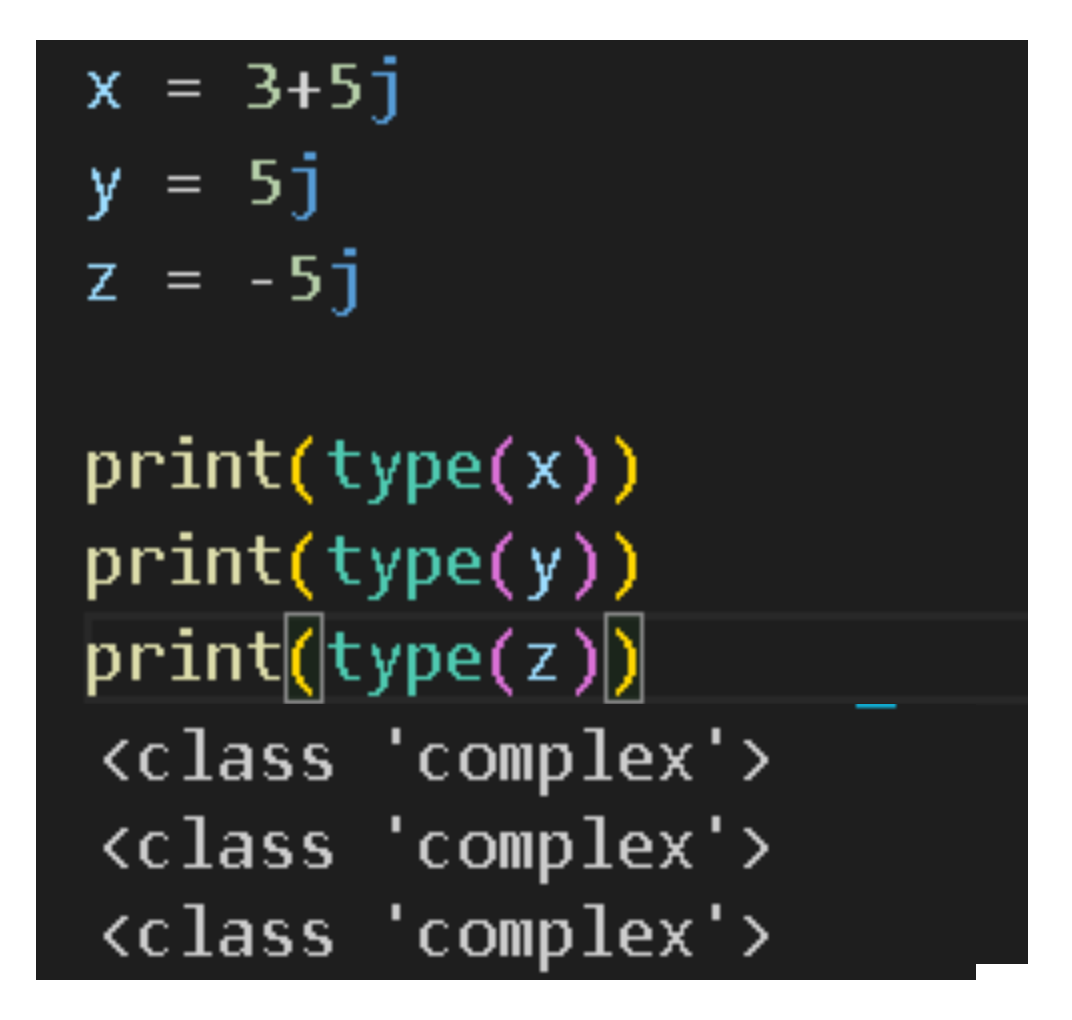
* **A whole number, positive or negative, without decimals , of unlimited length**

****

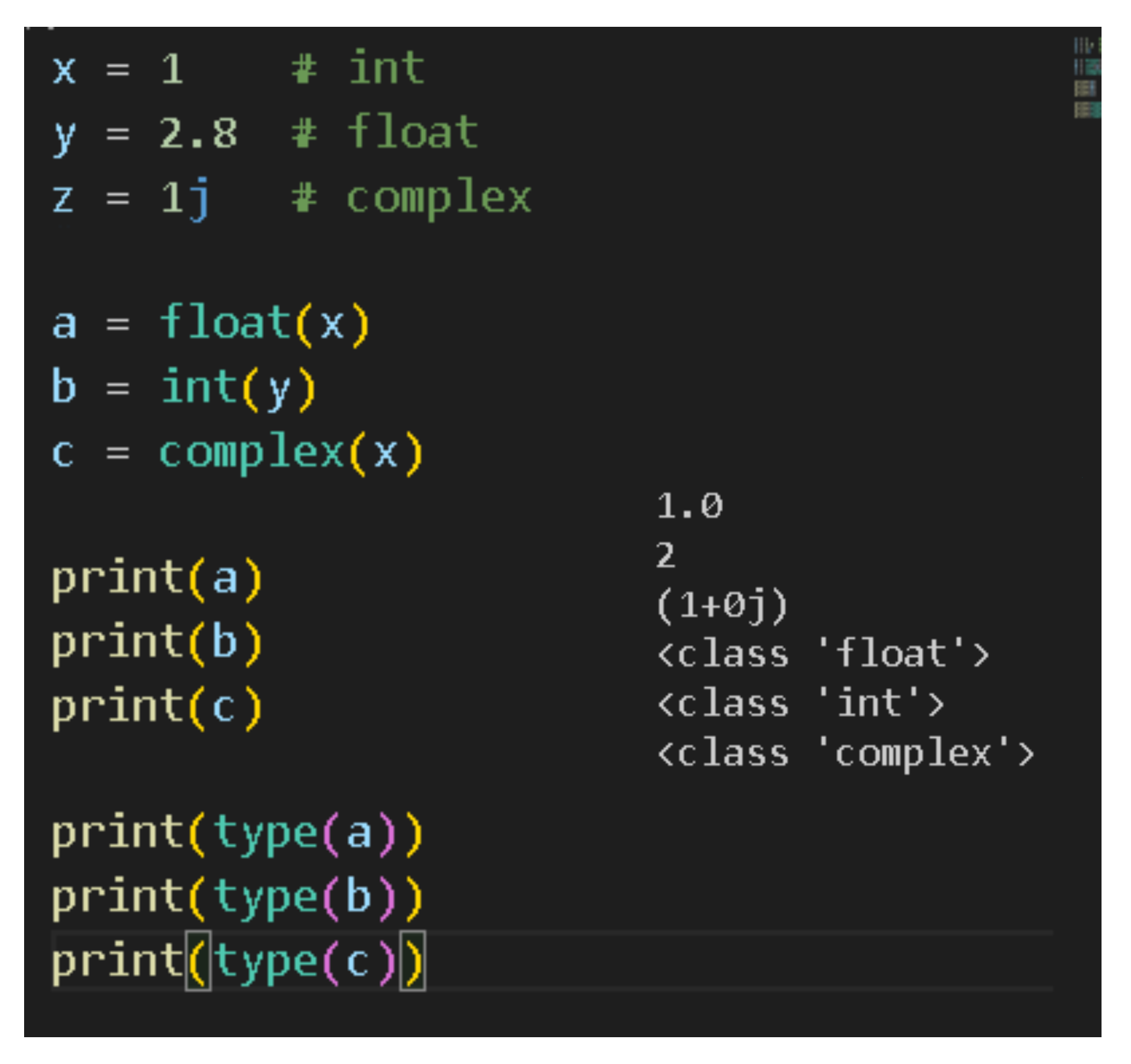
* + 1. **Float : (Float point number)**
* **A number, positive or negative, containing one or more decimals.**

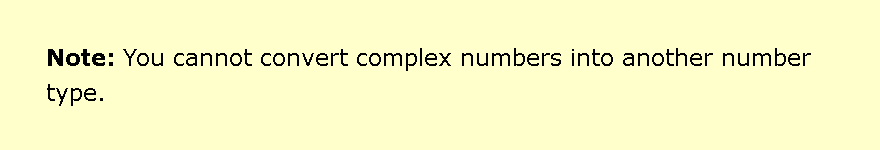
****

* + 1. **Complex :**
* **Numbers are written with a “j” as the imaginary part.**

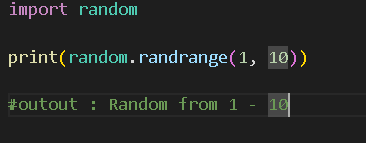
****

* + 1. **Type Conversion :**
* **Can convert from 1 type to another with int(), float(), complex() methods:**

****

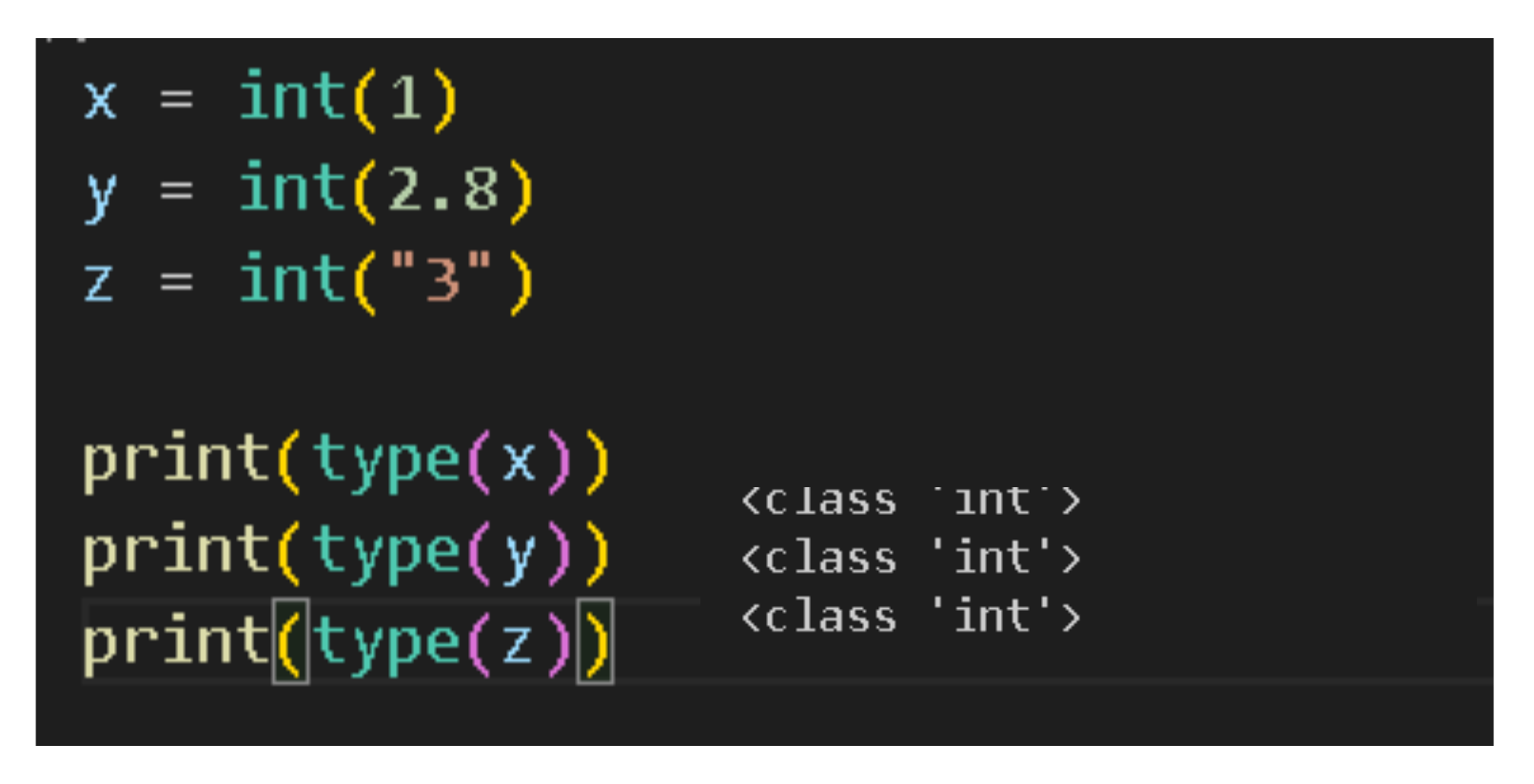
****

* + 1. **Random Number :**
* **In Python, we have to call random() to use :**

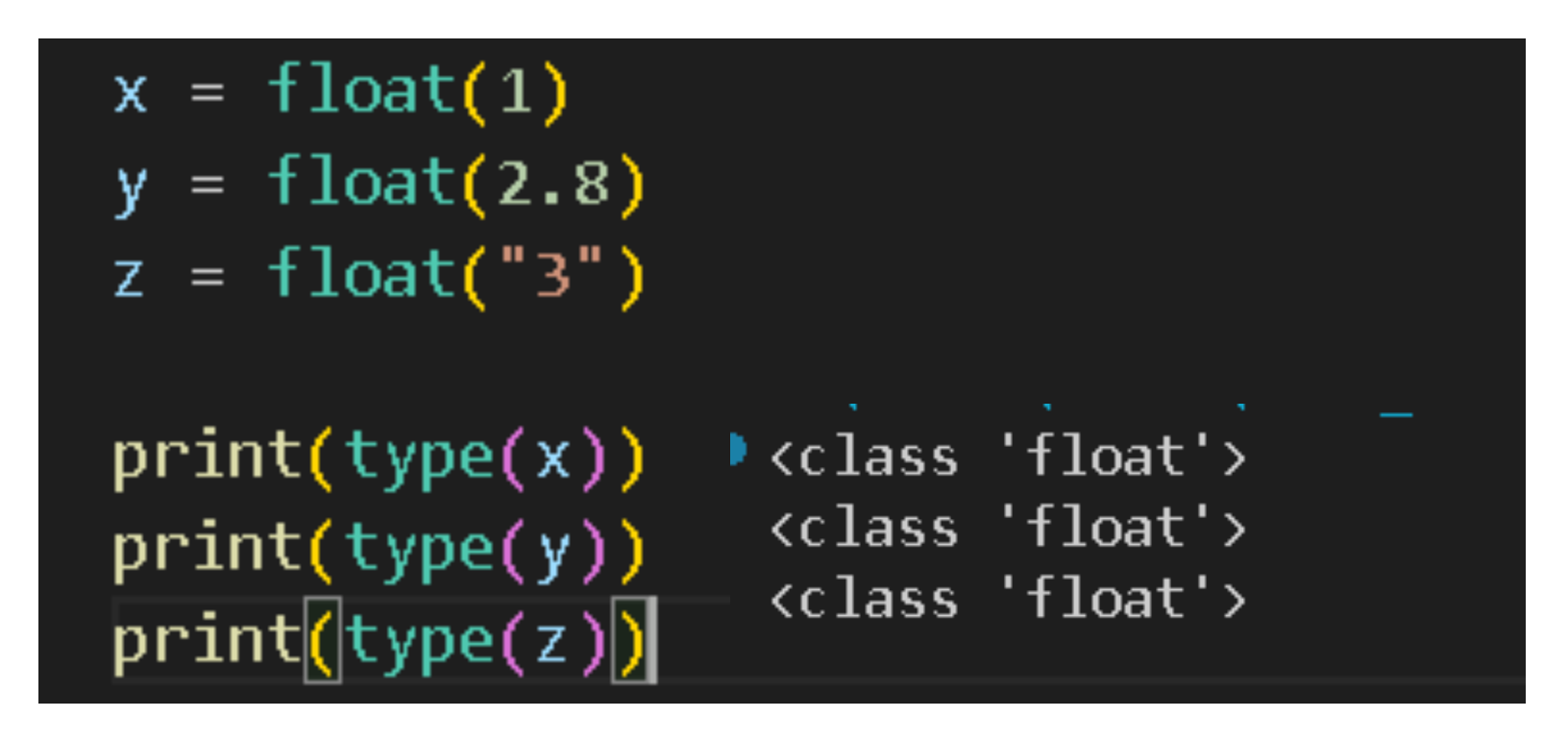
****

1. **Python Casting :** 
   1. **Specify a Variable Type :**

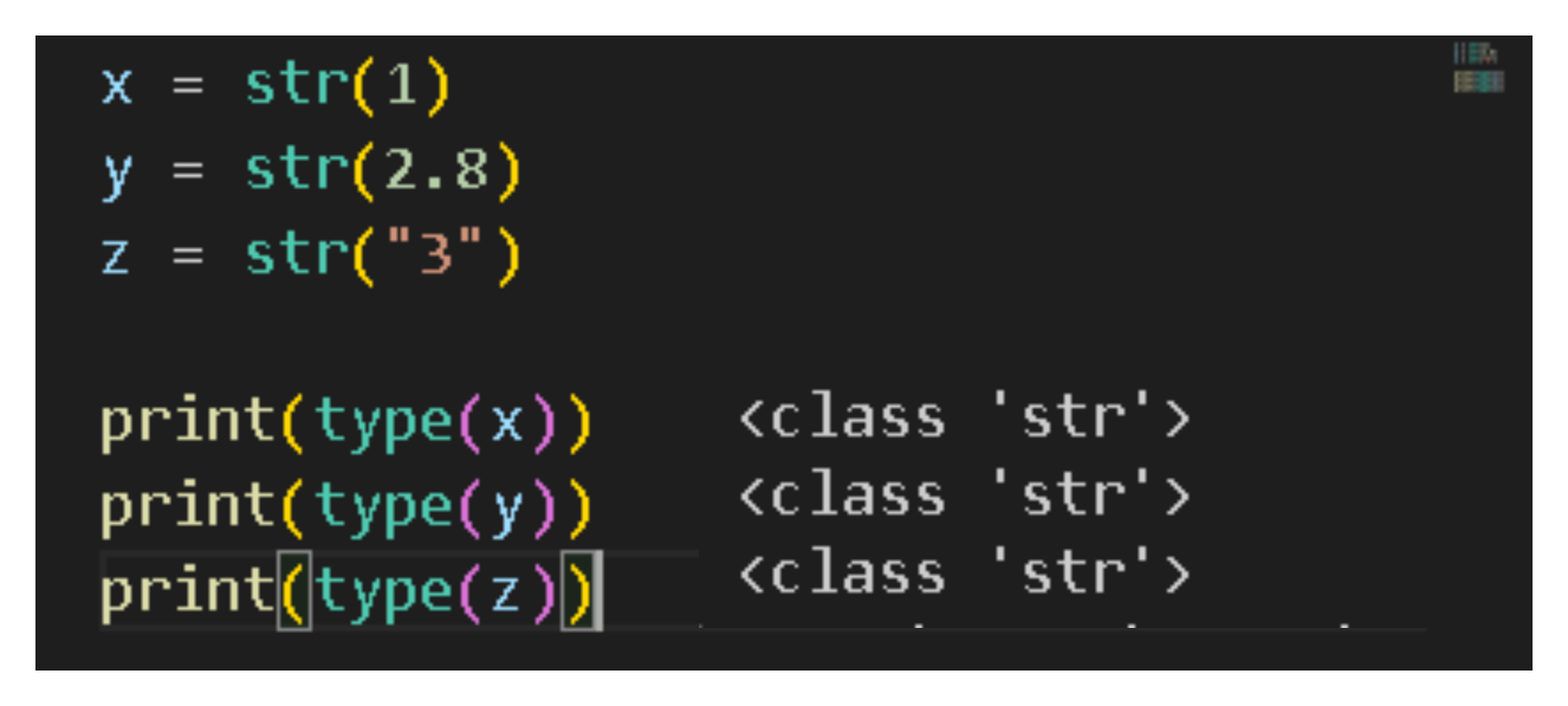
* **int(): constructs an integer number from an integer literal, a float literal, or a string literal.**

****

* **float(): constructs a float number from an integer literal, a float literal or a string literal.**

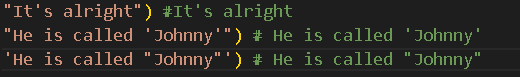
****

* **str(): constructs a string from a wide variety of data types, including strings, integer literals, float literals.**

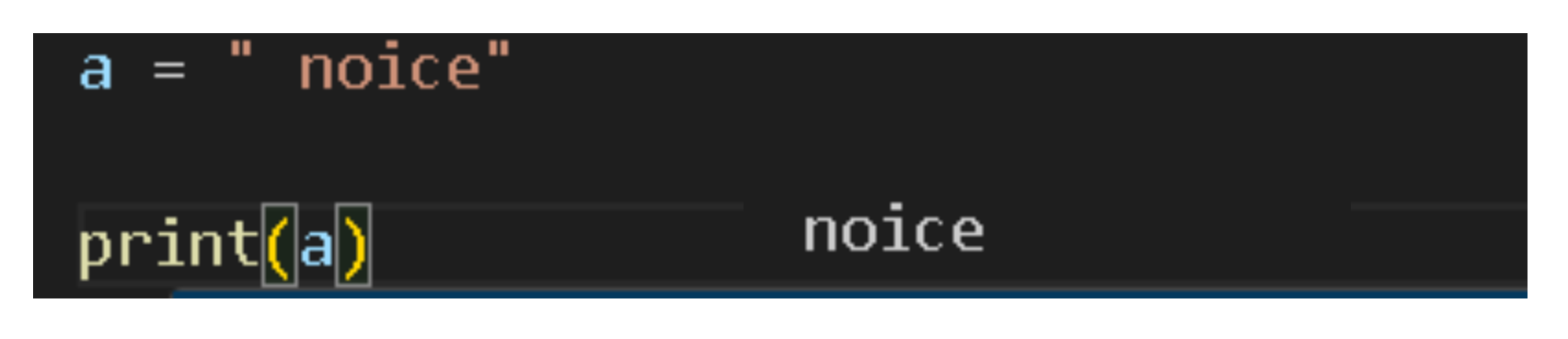
****

1. **Python Strings :** 
   1. **Strings :**

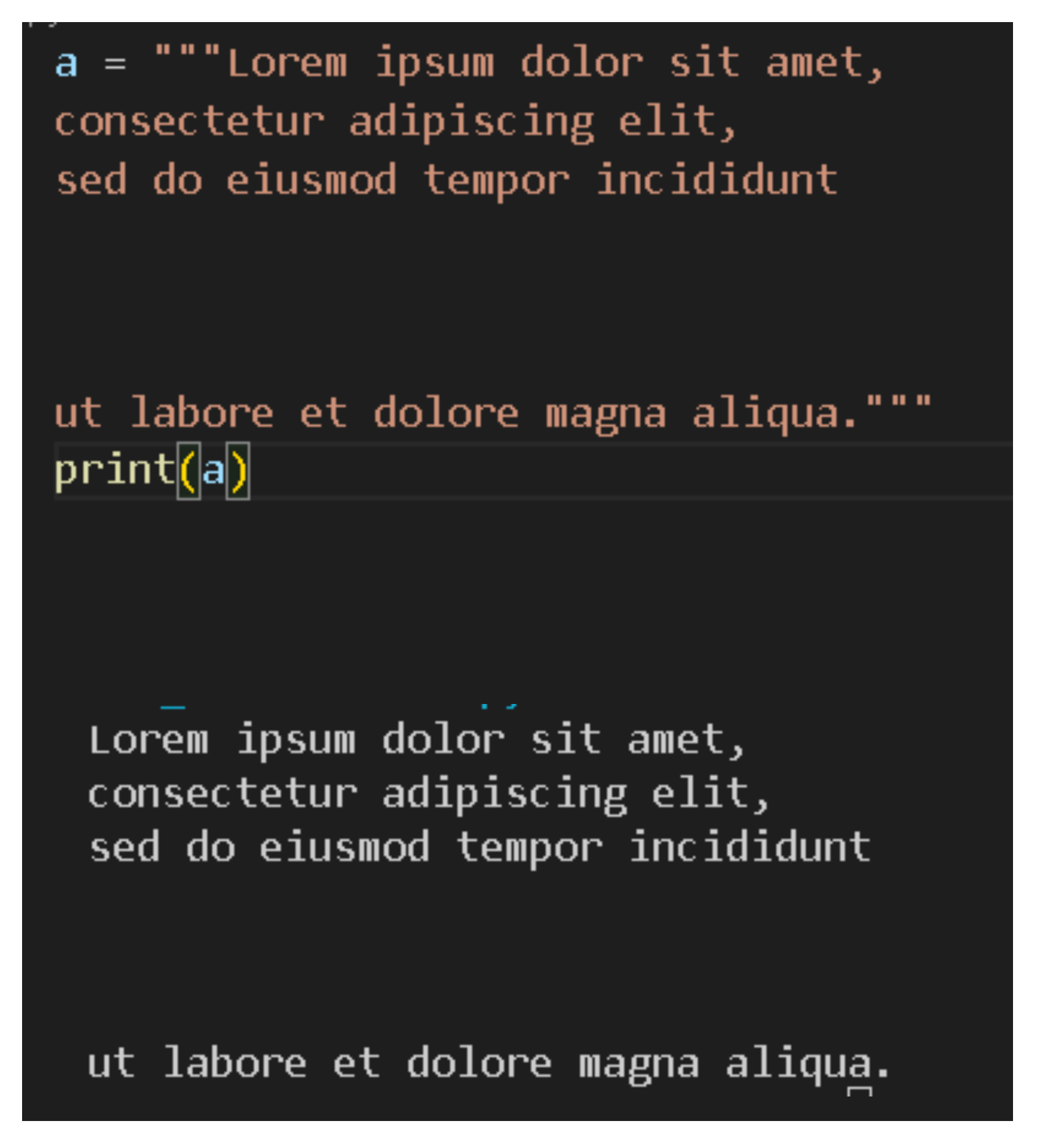
* **‘Sth’ is the same as “Sth”**
  1. **Quotes Inside Quotes :**
* **Can use quotes inside a string, as long as they dont match the quotes surrounding the string:**

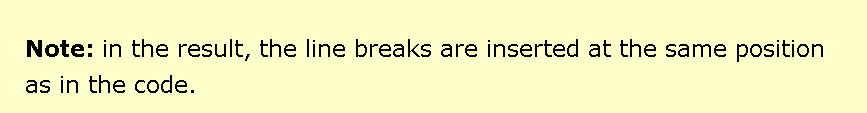
****

* 1. **Assign String to a Variable :**

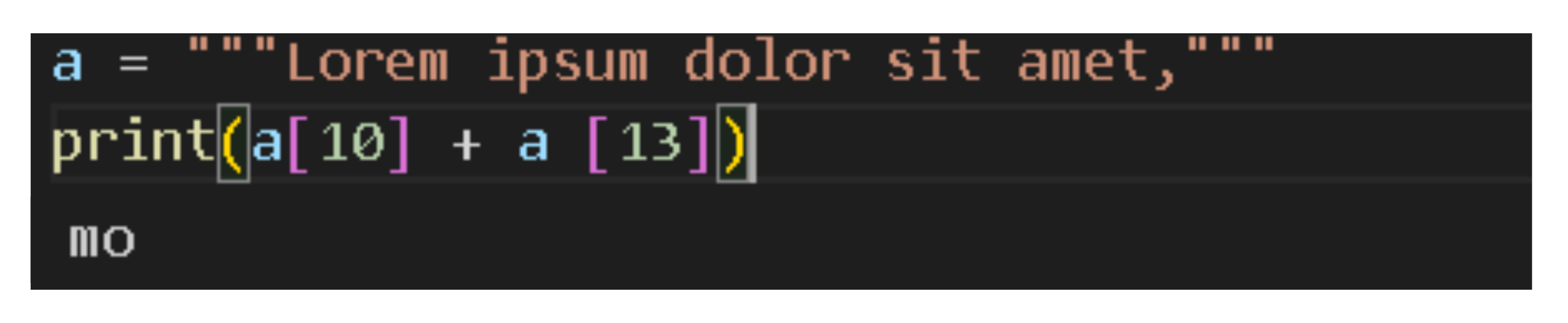
****

* 1. **Multiline Strings :**
* **CAn use a multiline string to a variable by using three quotes :**

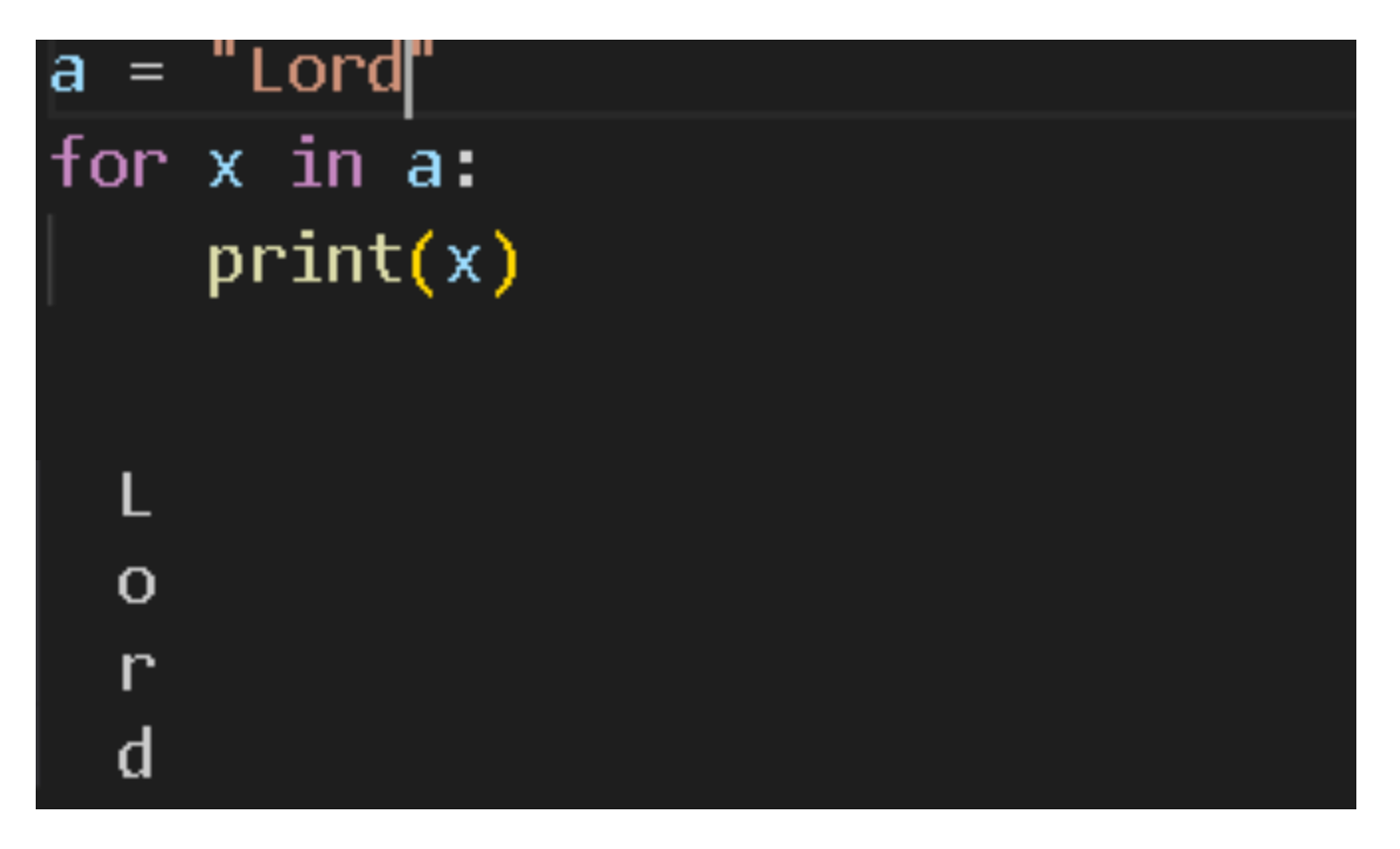
****

****

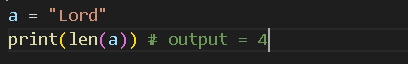
* 1. **Strings are arrays :**
* **Strings in Python are arrays of bytes representing unicode characters.**
* **A single character is simply a string with a length of 1**
* **-> Use Square brackets “[ ]” to access elements of the string.**

****

* 1. **Looping through a String :**
* **Cause strings are arrays -> can loop through the characters in a string : for**

****

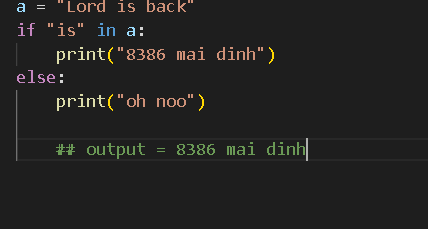
* 1. **String Length :**
* **To get the length of a string : len()**

****

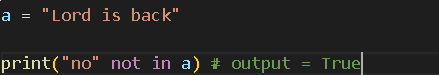
* 1. **Check String:**
* **To check if a certain phrase or character is present in a string : in**

****

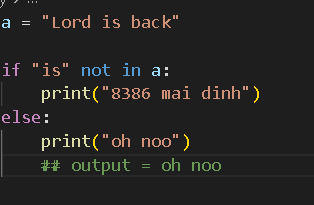
* **Or can use if :**

****

* 1. **Check if NOT :**
* **Check if a certain phrase or character is Not present in a string : not in**

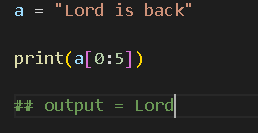
****

* **OR can use if :**

****

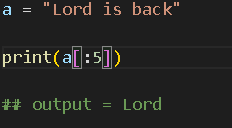
1. **Slicing Strings :** 
   1. **Slicing :**

* **Return a range of characters by using slice syntax.**
* **Specify the start and the end , separated by a colon (“ : “)**

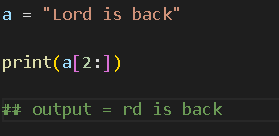
****

**\*\*\*\* : print(a[x:y]) => x <= a < y**

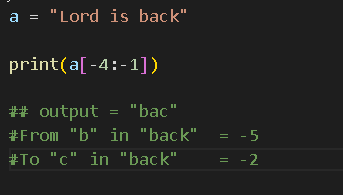
* 1. **Slice From the Start:**
* **Leaving out the start index-> the range will start at the first chac :**

****

* 1. **Slice To the End :**
* **Leaving out the end -> the range will go to the end :**

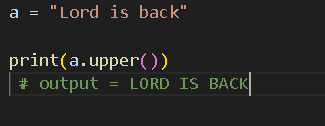
****

* 1. **Negative Indexing :**
* **Use negative indexes to start the slice from the end of the string:**

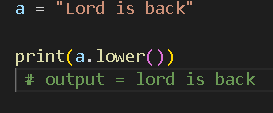
****

1. **Python - Modify Strings :** 
   1. **Upper Case :**

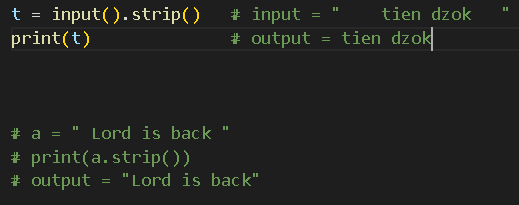
* **The upper() method returns the string in upper case:**

****

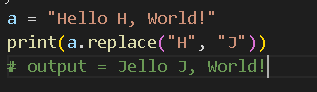
* 1. **Lower Case :**
* **The lower() method returns the string in lower case:**

****

* 1. **Remove Whitespace :**
* **Whitespace is the space before and/or after the actual text, and very often wants to remove this space.**
* **The strip() method removes any whitespace from the beginning or the end:**

****

* 1. **Replace String : string.replace(old, new, count) ( số lần xuất hiện )**
* **The replace() method replaces a string with another string:**

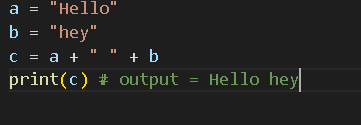
** => Replace all “H” to “J”**

* 1. **Split String :**
* **The split() method splits the string into substrings if it finds instances of the separator: (chia chuỗi thành các chuỗi con nếu tìm đc các trường hợp của dâu split():**

****

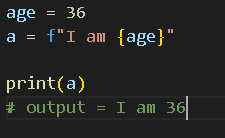
1. **Python - String Concatenation :** 
   1. **String Concatenation :**

* **To concatenate, or combine, two strings -> use the “+” operator.**

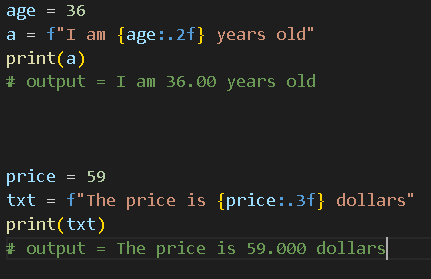
****

1. **Python - Format - Strings :** 
   1. **F-Strings :**

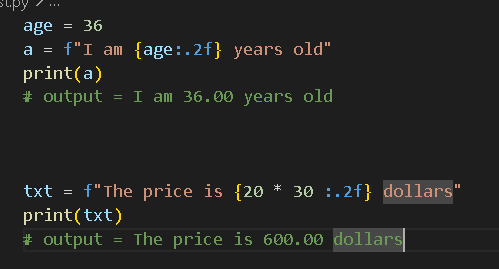
* **Put an “f” in front of the string literal and add curly brackets {} as placeholders for variables and other operations.**

****

* 1. **Placeholders and Modifiers :**
* **A placeholder can contain variables, operations, functions and modifiers to format the value.(the same as f-string)**
* **A placeholder can include a modifier to format the value**
* **A modifier is included by adding a colon “:” followed by a legal formatting type, like : .2f**

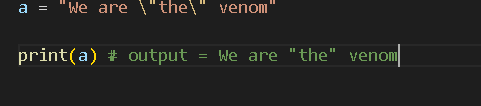
****

* **A placeholder can contain python code, like math:**

****

1. **Python - Escape Characters :** 
   1. **Escape Character : Insert**

* **An escape character is a backslash “\”followed by a character that we want to insert.**

****

**=> To put the in “ ” : Each \” is an escape character.**

* **Escape Characters :**

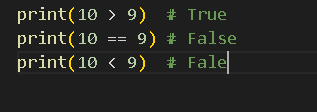
| **\’ or \”** | **Single Quote**  **(insert the right “ or ‘ into the string)** |
| --- | --- |
| **\\** | **Backslash**  **(insert the right \ into the string)** |
| **\n** | **New Line** |
| **\r**  **(ex: a\rb = b)**  **Replace all b to a**  **(ex:"Hello\rWor!" =**  **Wor!o)** | **Carriage Return**  **(Remove the cursor back to the start)** |
| **\t** | **Tab(+1 tab)** |
| **\b**  **(ex: "Hello \bWorld!" = HelloWorld!)**  **(ex:”hello\bw” =**  **hellw+** | **Backspace**  **(Remove 1 space behind the command)**  **(can remove 1 element if 1 element in front of it)** |
| **\f**  **(like printers)** | **Form Feed(indicate a page break in some contexts)** |
| **\ooo**  **(o is a number)**  **(ex: “\110\145” = He)** | [**Octal**](https://www.w3schools.com/python/trypython.asp?filename=demo_string_octal) **value(Octaldecimal number)** |
| **\xhh**  **(x is unchanged)**  **(ex: "\x48\x65" = He** | [**Hex value**](https://www.w3schools.com/python/trypython.asp?filename=demo_string_hex) **(Hexadecimal number)** |

1. **Python - String Methods :**

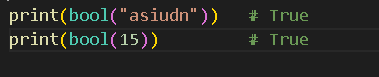
| **capitalize()** | **Converts the first character to uppercase** |
| --- | --- |
| [**casefold()**](https://www.w3schools.com/python/trypython.asp?filename=demo_ref_string_casefold) | **Converts string into lower case(bigger, stronger, more comprehensive than lower())** |
| **count()** | **Returns the number of times a specified value occurs in a string** |
| **encode()**  **(ex:”My name is Ståle" =b'My name is St\xc3\xe5le'** | **Returns an encoded version of the string.** |
| **endswith()**  **(endswith(“...”)** | **Returns true if the string ends with the specified value** |
| **expandtabs(n)** | **Sets the tab size of the string**  **(Replace (\t) with n - 1 space)** |
| **find()**  **string.find(“...”, start, end)**  *(safely check for the existence of a substring)* | **Searches the string for a specified value and returns the position of where it was found first**  **Return -1 if no found** |
| [**format()**](https://www.w3schools.com/python/trypython.asp?filename=demo_ref_string_format) | **Formats specified values in a string** |
| **format\_map()** | **Formats specified values in a string** |
| **index()**  **string.index(..., start, end)**    *(need to confirm that substring exist and want to handle the case where it error)* | **Searches the string for a specified value and returns the position of where it was found**  **Have to use ValueError if not found** |
| **isalnum()** | **Return true if all characters are alphanumeric** |
| **isascii()** | **Return true if all characters are ascii characters** |
| **isalpha()**  **(***like a, b , y,..***)** | **Return true if all characters are in the alphabet** |
| **isdecimal()** | **Return true if all characters are decimals** |
| **isdigit()** | **Return true if all characters are digits(numbers)** |
| **isidentifier()**  *(def.isidentifier() -> False*  *invalid-variable.isidentifier()->false* | **Return true if the string is an identifier** |
| **islower()** | **Return true if all characters are lower case** |
| **isnumeric()** | **Return true if all characters are numeric(include fraction, Roman numeral,..)** |
| **isprintable()** | **Return true if all characters are printable**  **(NO contains a tab(\t) or a newline(\n),...)** |
| **isspace()**  *(only space)* | **Return true if all characters are whitespaces** |
| **istitle()**  *(a complete string)* | **Return true if the string follows the rules of a title** |
| **isupper()** | **Return true if all characters are upper case** |
| **join()**  **‘..’.join(string)**   * *The elements must be substrings* * *The string is empty -> return empty strings* | **Join the elements of an iterable to the end of the string.** |
| [**ljust()**](https://www.w3schools.com/python/trypython.asp?filename=demo_ref_string_ljust)  string.ljust(width, ‘...’)   * *width > len(str)* * *‘..’ only 1 character* | **Return a left justified version of the string** |
| **lower()** | **Concerts a string into lower case** |
| **lstrip()** | **Return a left trim version of string**  **(***remove all the characters from the left)* |
| **partition()** | **Return a string where the string is parted into three parts** |
| **rfind()**  **(***to find if substring exists without exceptions)* | **Search the string for a specified value and return the last position of where it was found** |
| **rindex()**  **(***to ensure if the substring exists* | **Search the string for a specified value and return the last position of where it was found** |
| **rjust()**  **(***opposite direction to ljust)* | **Return a right justified version of the string** |
| **rpartition()**  **(***start scanning from the right and end at the left***)** | **Return a tuple where the string is parted into three parts** |
| **rsplit()**  **(***similar to split but start from the right***)** | **Split the string at the specified separator and return a list** |
| **rstrip()**  **(***opposite to the lstrip()***)** | **Return a right trim version of the string** |
| **splitlines()**  **(***divide the string to some substrings by* **(*\n*))** | **Split the string at the line breaks and return a list** |
| **startswith()** | **Return true if the string starts with the specified value** |
| **strip()**  **(***Remove substrings from right and left of string***)** | **Return a trimmed version of the string** |
| **swapcase()**  **(***Scanning, swapping each character***)** | **Swaps cases, lower case become upper case and vice versa** |
| **title()**  **(***Convert the first character of 1 substring***)** | **Convert the first character of each word to upper case** |
| **maketrans()** | **Return a translation table to be used in translations**  **(***like replace() but more complex and combinate with str.translate()* **)** |
| **translate()**  **string.translate(table)** | **Return a translated string**  **(table where using maketran() to map characters to their replacements.)** |
| **upper()** | **Convert s string into upper case** |
| **zfill()**  **string.zfill(width)** | **Fill the string with a specified number of 0 values at the beginning** |

1. **Python Booleans :** 
   1. **Boolean Values :**

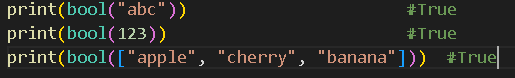
* **When compare 2 values, the expression is evaluated and Python returns the Boolean:**

****

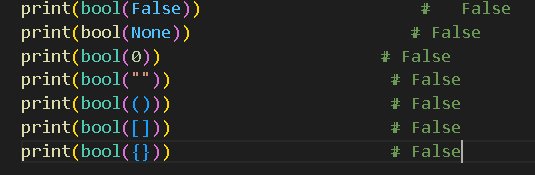
* 1. **Evaluate Values and Variables :**
* **The bool() function allows us to evaluate any value and give True or False in return.**

****

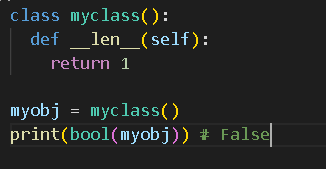
* 1. **Most Values are True :**
* **Almost any value is evaluated to True if it has some sort of content.**
* **Any string is True, except empty strings.**
* **Any number is True, except 0.**
* **Any list, tuple, set, and dictionary are True, except empty ones.**

****

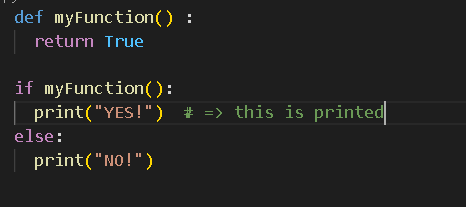
* 1. **Some Values are False :**
* **Some values are False : the empty values, such as : ( ), [ ] , { }, the number 0, the value NONE :**

****

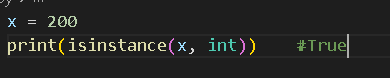
* **An obj that is made from a class a \_\_len\_\_ function that return 0 or False :**

****

* 1. **Functions can Return a Boolean:**

****

* **Python has many built-in functions that return a boolean value :**

** => Determine if an obj is of a certain data type.**

1. **Python Operators :** 
   1. **Python Operators :**

* **Operators are used to perform operations on variables and values.**
  1. **Python Arithmetic Operators :**

| **+** | **Addition** | **x + y** |
| --- | --- | --- |
| **-** | **Subtraction** | **x - y** |
| **\*** | **Multiplication** | **x \* y** |
| **/** | **Division** | **x / y** |
| **%** | **Modulus** | **x % y**  **(***the remainder after division)* |
| **\*\*** | **Exponentiation** | **x \*\* y**  **(x^y)** |
| **//** | **Floor division**  **(***làm tròn xuống***)** | **x // y**  *(rounds the result down to the nearest whole number)* |

* 1. **Python Assignment Operators :**

| **Explain** | **Binary(Nhị phân)** | **Math** |
| --- | --- | --- |
| **&=** | **x&=3**  **(x AND 3)**  **(***Bit nào có 2 bit 1 thì lấy)* | **x = x % 3** |
| **|=** | **x|=3**  **(x OR 3)**  **(***bit nào có 1 thì lấy***)** | **x = x | 3** |
| **^=** | **x^=3**  **(x XOR 3)**  **(***đối chiếu, bit nào có 1 bit 1 thì lấy, có 2 bit 1 thì 0)* | **x = x ^ 3** |
| **>>=** | **x>>=3**  **(***Remove 3 elements in the right***)** | **x = x >> 3**  **(***x = x // 2 ^ 3***)** |
| **<<=** | **x<<=3**  **(***Remove 3 elements in the left and add 3 element “0” in the right***)** | **x = x << 3**  **(***x \* 2^3***)** |
| **:=** | **print(x := 3)** | **x = 3**  **print(x)** |

* 1. **Python Comparison Operators :**

| **==** | **Equal** | **x == y** |
| --- | --- | --- |
| **!=** | **Not equal** | **x != y** |
| **>** | **Greater than** | **x > y** |
| **<** | **Less than** | **x < y** |
| **>=** | **Greater than or equal to** | **x >= y** |
| **<=** | **Less than or equal to** | **x <= y** |

* 1. **Python Logical Operators :**

| **and** | **Return true if both statements are true** | **x < 5 and x < 2** |
| --- | --- | --- |
| **or** | **Return true if one statements is true** | **x < 5 or x < 2** |
| **not** | **Reverse the result, return false if the result is true** | **not(x < 5 and x < 2)** |

* 1. **Python Identity Operators :**

| **is** | **Returns true if both variable are the same obj** | **x is y** |
| --- | --- | --- |
| **is not** | **Returns true if both variable are not the same obj** | **x is not y** |

* 1. **Python Membership Operators :**

| **in** | **Returns true if a sequence with specified value is present in the obj** | **x in y** |
| --- | --- | --- |
| **not in** | **Returns true if a sequence with specified value is not present in the obj** | **x not in y** |

* 1. **Python Bitwise Operators :**

| **&** | **AND** | **Set each bit to 1 if both bits are 1** | **x & y** |
| --- | --- | --- | --- |
| **|** | **OR** | **Set each bit to 1 if one of 2 bits is 1** | **x | y** |
| **^** | **XOR** | **Set each bit to 1 if only one of 2 bits is 1** | **x ^ y** |
| **~** | **NOT** | **Invert all the bits**  **(đảo ngược)** | **~x** |
| **<<** | **Zero fill left shift** | **Shift left by pushing zeros (0) in from the right and let the leftmost bits fall off** | **x << n** |
| **>>** | **Signed right shift** | **Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off** | **x >> n** |

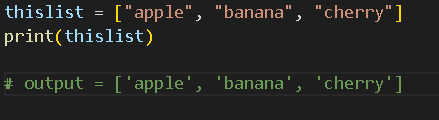
* 1. **Operator Precedence :**
* **Describes the order in which operations are performed.**

****

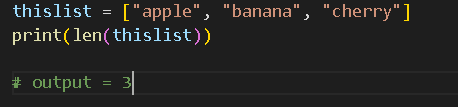
| **()** | **Parentheses** |
| --- | --- |
| **\*\*** | **Exponentiation** |
| **+x -x ~x** | **Unary plus, unary minus, bitwise NOT** |
| **+ / // %** | **Multiplication, division, floor division, modulus** |
| **+ -** | **Add, Sub** |
| **<< >>** | **Bitwise left, right shifts** |
| **&** | **Bitwise AND** |
| **^** | **Bitwise XOR** |
| **|** | **Bitwise OR** |
| **== != > >= < <= is is not in not in** | **Comparisons, identity, membership operators** |
| **not** | **Logical NOT** |
| **and** | **AND** |
| **or** | **OR** |

1. **Python Lists :** 
   1. **List :**

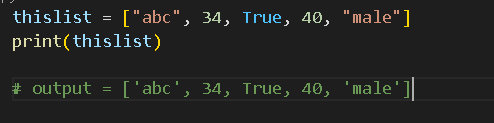
* **Used to share multiple items in a single variable**
* **List are one of 4 built-in data types in Python used to store collections of data** 
  + **The other: Tuple, Set, Dictionary .**

****

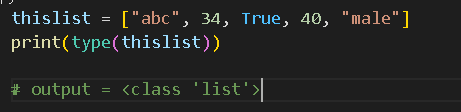
* 1. **List Length :**
* **To determine how many items a list has : len()**

****

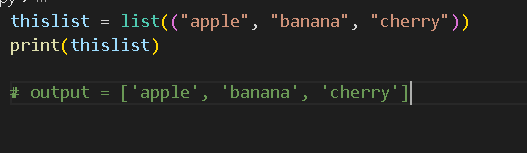
* 1. **List Items - Data Types :**
* **Can contain different data types :**

****

* **Type() :**

****

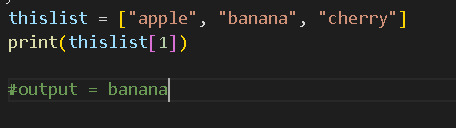
* 1. **The list() Constructor :**
* **Using list() constructor to make a list :**

****

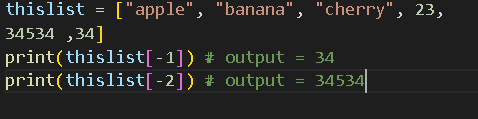
* 1. **Python Collections (Arrays) :**
* **List : a collection which is ordered and changeable -> Allow duplicate members.(trùng lặp)**
* **Tuple : a collection which is ordered and unchangeable -> Allow duplicate members.**
* **Set : a collection which is unordered, unchangeable, and unindexed -> No duplicate members.(***can remove and/or add items***)**
* **Dictionary : a collection which is order and changeable -> No duplicate members.**

1. **Python - Access List Items :** 
   1. **Access Items :**

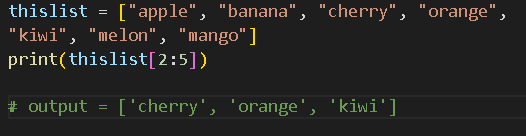
* **List items are indexed and we can access them by referring to the index number :**

****

* + 1. **Negative indexing :**
* **Mean: Start from the end.**
  + **-1 refers to the last item**
  + **-2 refers to the second last item**

****

* + 1. **Range of Indexes :**
* **We can specify a range of indexes by specifying where to start and where to end the range.’**

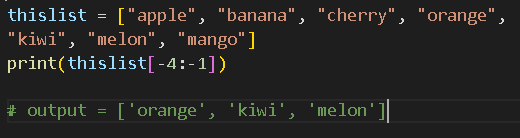
****

**\*\*\*\* : 2 <= thislist < 5**

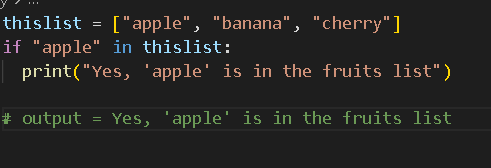
* **We can leave out the start or the end.**

****

* + 1. **Range of Negative indexes :**
* **Specify negative indexes if we want to start the search from the end of the list :**

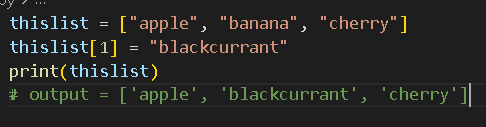
****

* 1. **Check if Item Exists :**
* **To determine if a specified item is present in the list use the in keyword :**

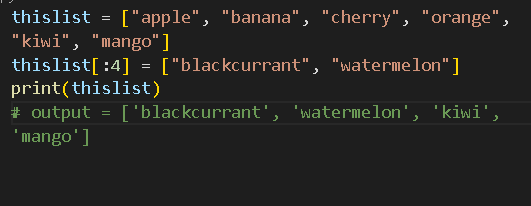
****

1. **Python - Change List Items :** 
   1. **Change item value :**

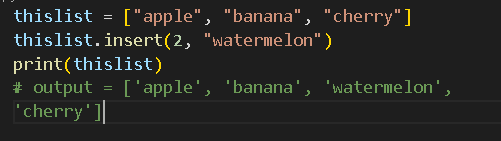
* **To change the value of a specific item :**

****

* 1. **Change a Range of item values :**
* **Define a list with the new values and refer to the range of index numbers where we want to insert the new values :**

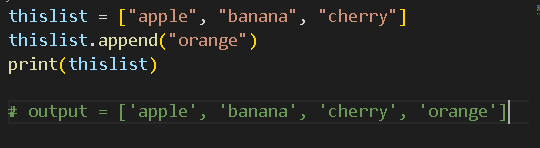
****

* **Replace a substring in the list to another substring which can contain many values. -> The length of the list will change.**
  1. **Insert Items :**
* **To insert a new list items, we can use insert() method :**

****

1. **Python - Add List Items :** 
   1. **Append Items :**

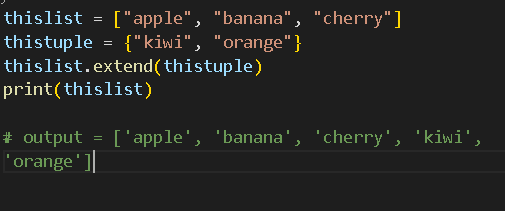
* **append() to add an item to the end of the list :**

****

* 1. **Extend List :**
* **extend() to append elements from another to the current list :**

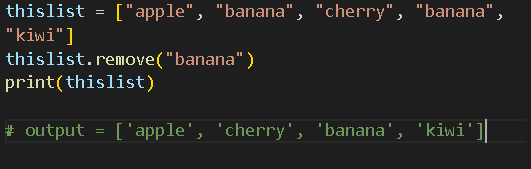
****

* 1. **Add any iterable :**
* **extend() does not have to append lists, we can add any iterable obj .**

****

1. **Python - Remove List Items :** 
   1. **Remove specified item :**

* **The remove() removes the specified item :**

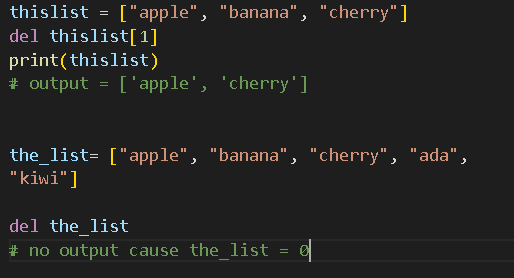
****

**=> Can only remove 1 start scanning from the left**

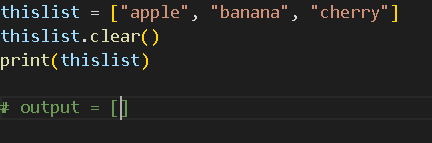
* 1. **Remove specified index :**
* **The pop() removes the specified index :**

****

* **The del keyword can also delete the list :**

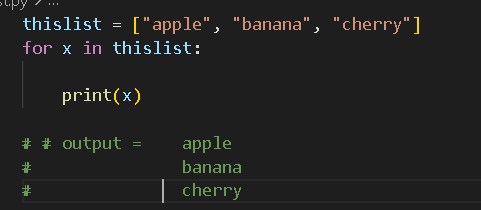
****

* 1. **Clear the list :**
* **clear() empties the list :**

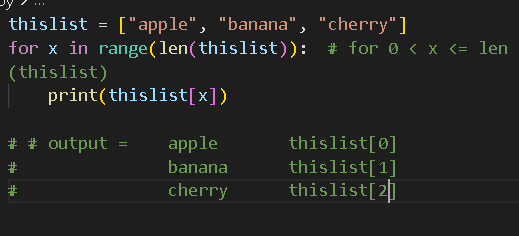
****

1. **Python Loop Lists :** 
   1. **Loop Through a List :**

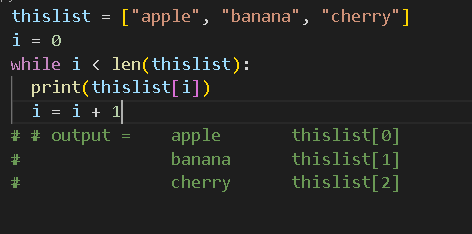
* **Using a for loop :**

****

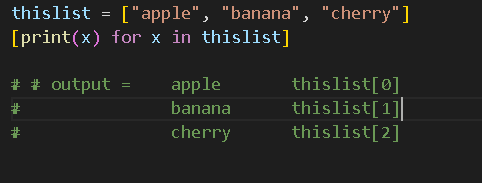
* 1. **Loop Through the Index Numbers :**
* **Loop through the list items by referring to their index number:**

****

* 1. **Using a while loop :**
* **Use len() function to determine the length of the list, then start at 0 and loop the way through the list items by referring to their index:**

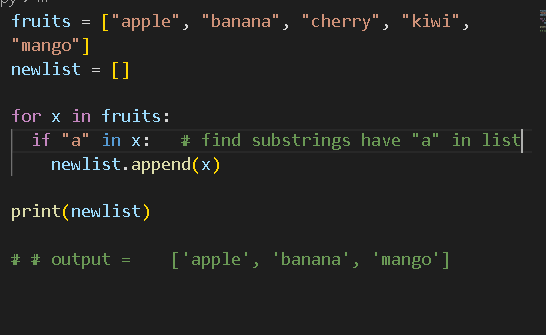
****

* 1. **Looping using list comprehension :**
* **Offer the shortest syntax for looping through lists :**

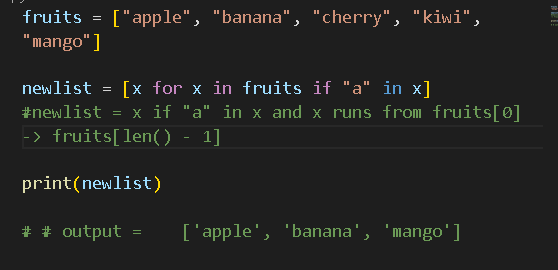
****

1. **Python - List Comprehension :** 
   1. **List Comprehension :**

* **Offer a shorter syntax when we want to create a new list based on the values of an existing list.**

****

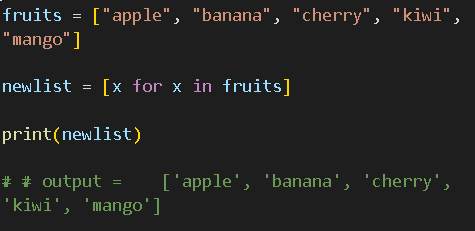
* **We can do that with one line code :**

****

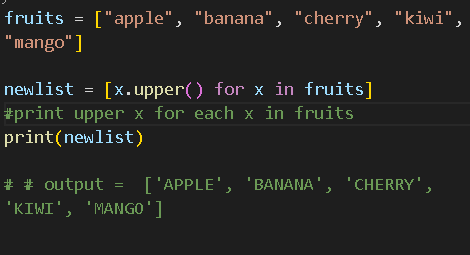
* 1. **The Syntax :**

| **newlist = [*expression* for *item* in *iterable* if *condition* == True ]** |
| --- |

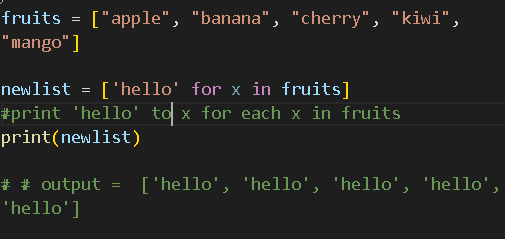
* **An example without if :**

****

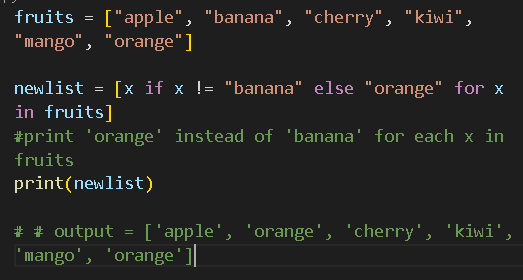
* **Expression :**

****

* **We can set the outcome to whatever we like :**

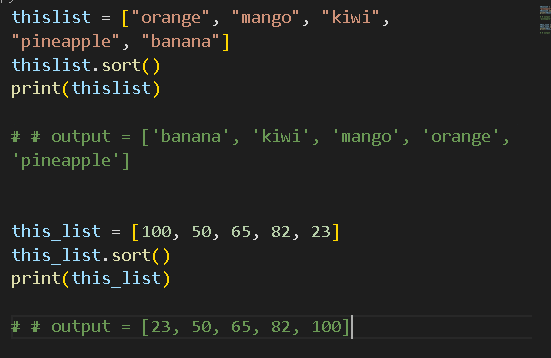
****

* **The expression can also contain conditions, not like a filter, but as a way to manipulate the outcome :**

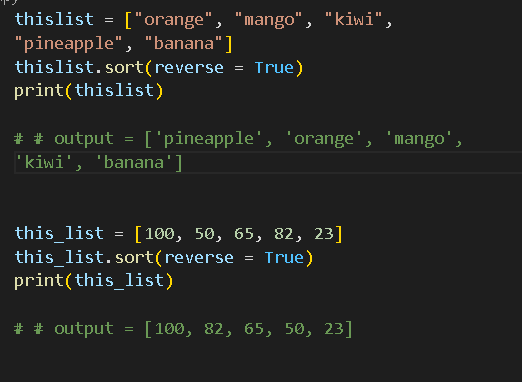
****

1. **Python - Sort Lists :** 
   1. **Sort List Alphanumerically :**

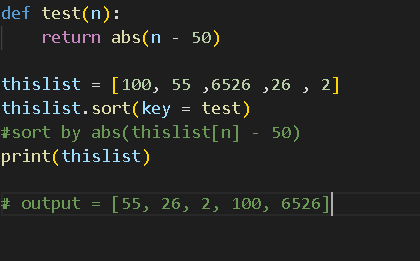
* **List objs have a sort() to sort the list alphanumerically, ascending :**

****

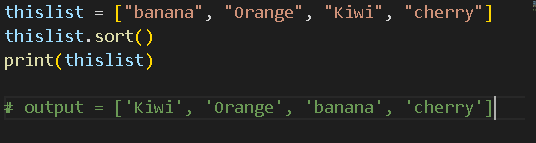
* 1. **Sort Descending :**
* **To sort descending , use keyword reverse = True :**

****

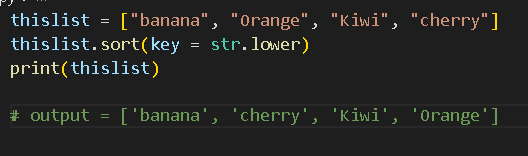
* 1. **Customize Sort Function :**
* **Customize the function by using keyword key = function :**

****

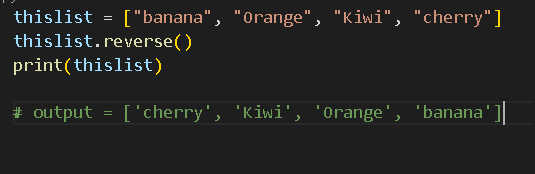
* 1. **Case insensitive sort :**
* **sort() is case sensitive -> resulting in all capital letter being sorted before lower case letters :**

****

* **Using key = function -> we can sort whatever we want :**

****

* 1. **Reverse Order : (đảo ngược)**
* **Want to reverse the current sorting order of the elements :**

****

1. **Python - Copy Lists :** 
   1. **Copy Lists :**

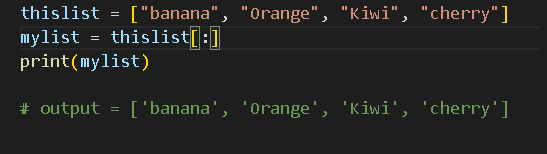
* **Use method copy() to copy a list :**

****

* 1. **Use the list method :**
* **Another way to make a copy : list()**

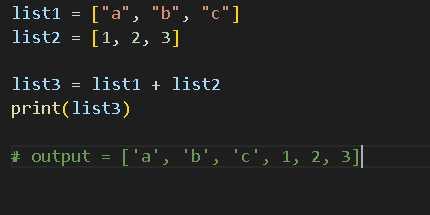
****

* 1. **Use the slice operator :**
* **Make a copy of a list with : operator :**

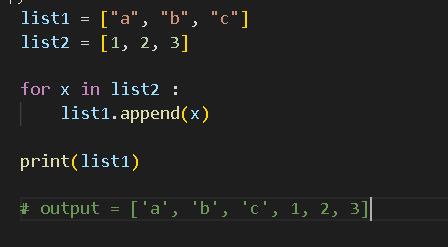
****

1. **Python - Join Lists :** 
   1. **Join two lists :**

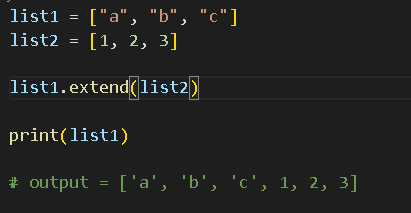
* **Use + operator to join 2 lists together :**

****

* **Another way : append all items :**

****

* **Use extend() method -> add elements from 1 list to another list :**

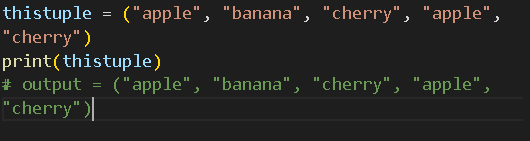
****

1. **Python - List Methods :** 
   1. **List Methods :**

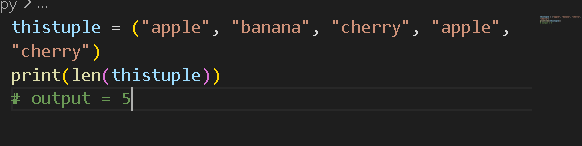
| **append()** | **Add an element at the end of the list** |
| --- | --- |
| **clear()** | **Remove all the elements from the list** |
| **copy()** | **Return a copy of the list** |
| **count()** | **Return the number of elements with the specified value** |
| **extend()** | **Add the elements of a list(or any iterable), to the end of the current list** |
| **index()** | **Return the index of the first element with the specified value** |
| **insert()**  **insert(n, “..”)** | **Add an element at the specified position** |
| **pop()** | **Remove the element at the specified position** |
| **remove()** | **Remove the item with the specified value** |
| **reverse()**  **(đảo ngược)** | **Reverse the order of the list** |
| **sort()** | **Sort the list** |

1. **Python - Tuples : (“....”)**
   1. **Tuple :**

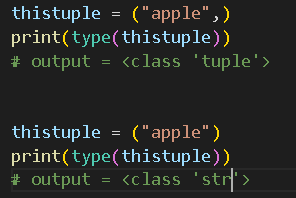
* **Used to store multiple items in a single variable.**
* **A collection which is ordered and unchangeable.**
* **Tuple items are indexed, the first : [0] , the second : [1]**
* **The tuples are ordered -> the items have a defined order and will not change.**
* **The tuples are unchangeable -> We cannot change, add, or remove items after the tuple has been created.**
  1. **Allow duplicates :**
* **Since the tuples are indexed -> they can have items with the same values.**

****

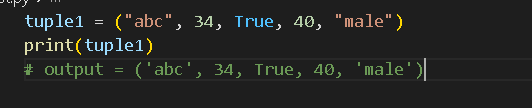
* 1. **Tuple Length :**
* **To determine how many items a tuple has : len()**

****

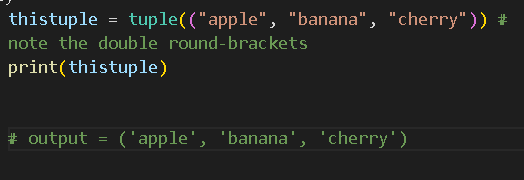
* 1. **Create tuple with 1 item :**
* **Create a tuple with 1 item -> have to add a comma after the item**

****

* 1. **Tuple items - Data Types :**
* **Tuple can contain different data types :**

****

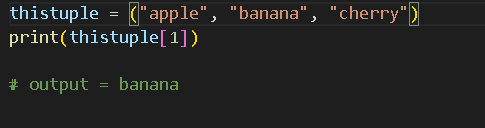
* 1. **The tuple() constructor :**
* **Use tuple() constructor to make a tuple :**

****

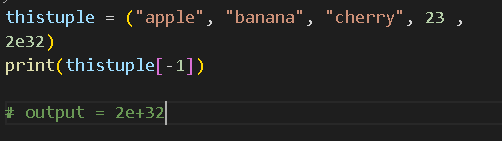
* 1. **Python Collection : (Arrays)**

1. **Python - Access Tuple Items :** 
   1. **Access Tuple Items :**

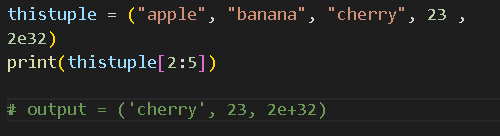
* **Access tuple items by referring to the index number, inside square brackets :**

****

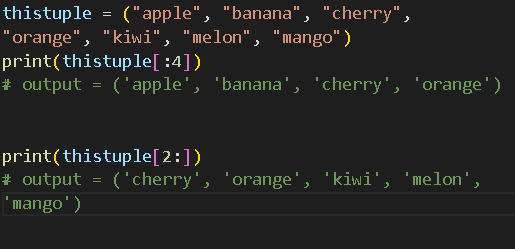
* **Negative indexing : -1 = the last item.**

****

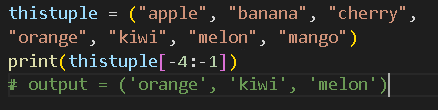
* 1. **Range of indexes :**
* **The first item has index 0.**
* **[n,m] : n included and m not included.**

****

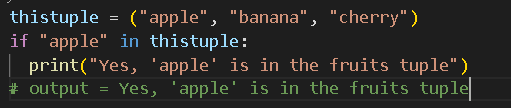
* **By leaving out the start or the end, the range will start or go on the position specified.**

****

* **Range of negative indexes :**

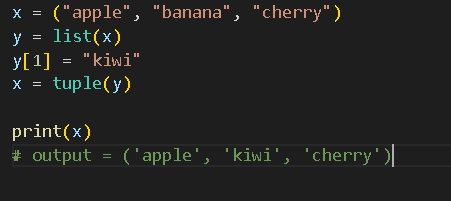
****

* **Check if item exists :**

****

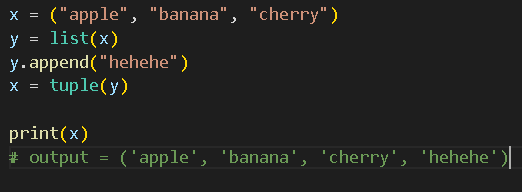
1. **Python - Update Tuples :** 
   1. **Change tuple value :**

* **We can convert the tuple into a list -> Change the list -> convert the list back into a tuple.**

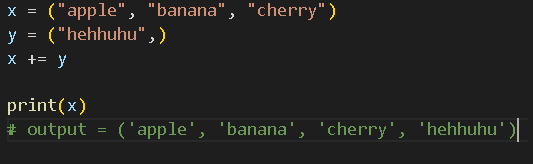
****

* 1. **Add item :**

1. **Convert into a list : Similar to Change tuple value :**

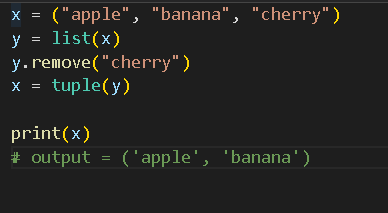
****

1. **Add tuple to a tuple : Create a new tuple with item(s) and add it to the existing tuple :**

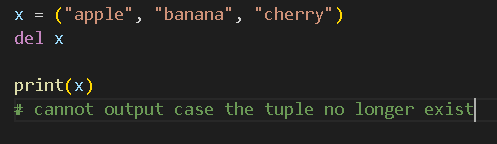
****

* 1. **Remove Items :**

1. **Remove specified item(s) :**

****

1. **Delete all items :**

****

1. **Python - Unpack Tuples :** 
   1. **Unpacking a Tuple :**

* **We can allow to extract the value back into variable -> unpacking :**