**F-string**: It directly converts all different data types into print able form along with the string in the *print()* function!

This is demonstrated with the following example:

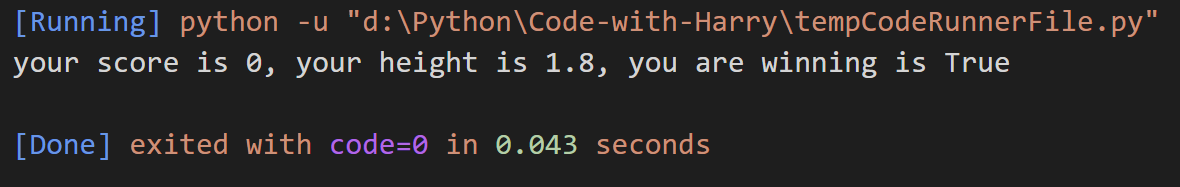
score=0

height=1.8

isWinning=True

print(f"your score is {score}, your height is {height}, you are winning is {isWinning}")

The output is as follows:



You can clearly see there is no need to convert each data type into string and then print it.

*lower()* function: The lower() function converts all uppercase characters in a string into lowercase characters and returns it.

**Syntax of String lower()**

The syntax of lower() method is:

string.lower()

**lower() Parameters()**

lower() method doesn't take any parameters.

*count()* function: The count() function returns the number of times a specified value appears in the string.

Syntax

*string*.count(*value, start, end*)

Parameter Values

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| *value* | Required. A String. The string to value to search for |
| *start* | Optional. An Integer. The position to start the search. Default is 0 |
| *end* | Optional. An Integer. The position to end the search. Default is the end of the string |

*randit()* function: It is an inbuilt function of the *random module*in Python. The random module gives access to various useful functions and one of them being able to generate random numbers, which is *randint()*.

|  |  |
| --- | --- |
| [random()](https://www.w3schools.com/PYTHON/ref_random_random.asp) | Returns a random float number between 0 and 1 |

For more information on *random* module: follow the following link

https://www.askpython.com/python-modules/python-random-module-generate-random-numbers-sequences

**Data Structures in Python:**

## List

Lists are used to store multiple items in a single variable.

Lists are one of 4 built-in data types in Python used to store collections of data, the other 3 are [Tuple](https://www.w3schools.com/python/python_tuples.asp), [Set](https://www.w3schools.com/python/python_sets.asp), and [Dictionary](https://www.w3schools.com/python/python_dictionaries.asp), all with different qualities and usage.

Lists are created using square brackets:

### **Example**

Create a List:

thislist = ["apple", "banana", "cherry"]  
print(thislist)

Some useful function which can be used as the *list* datatype are:

list.**append**(*x*)

Add an item to the end of the list. Equivalent to a[len(a):] = [x].

list.**extend**(*iterable*)

Extend the list by appending all the items from the iterable. Equivalent to a[len(a):] = iterable.

list.**insert**(*i*, *x*)

Insert an item at a given position. The first argument is the index of the element before which to insert, so a.insert(0, x) inserts at the front of the list, and a.insert(len(a), x) is equivalent to a.append(x).

list.**remove**(*x*)

Remove the first item from the list whose value is equal to *x*. It raises a [ValueError](https://docs.python.org/3/library/exceptions.html" \l "ValueError" \o "ValueError) if there is no such item.

list.**pop**([*i*])

Remove the item at the given position in the list, and return it. If no index is specified, a.pop() removes and returns the last item in the list. (The square brackets around the *i* in the method signature denote that the parameter is optional, not that you should type square brackets at that position. You will see this notation frequently in the Python Library Reference.)

list.**clear**()

Remove all items from the list. Equivalent to del a[:].

list.**index**(*x*[, *start*[, *end*]])

Return zero-based index in the list of the first item whose value is equal to *x*. Raises a [ValueError](https://docs.python.org/3/library/exceptions.html" \l "ValueError" \o "ValueError) if there is no such item.

The optional arguments *start* and *end* are interpreted as in the slice notation and are used to limit the search to a particular subsequence of the list. The returned index is computed relative to the beginning of the full sequence rather than the *start* argument.

list.**count**(*x*)

Return the number of times *x* appears in the list.

list.**sort**(*\**, *key=None*, *reverse=False*)

Sort the items of the list in place (the arguments can be used for sort customization, see [sorted()](https://docs.python.org/3/library/functions.html#sorted) for their explanation).

list.**reverse**()

Reverse the elements of the list in place.

list.**copy**()

Return a shallow copy of the list. Equivalent to a[:].

*split()* function:

The split() method breaks up a string at the specified separator and returns a list of strings.

### Example

text = 'Python is a fun programming language'

# split the text from space

print(text.split(' '))

# Output: ['Python', 'is', 'a', 'fun', 'programming', 'language']

[Run Code](https://www.programiz.com/python-programming/online-compiler)

## Syntax of String split()

The syntax of split() is:

str.split(separator, maxsplit)

## split() Parameters

The split() method takes a maximum of 2 parameters:

* **separator** (optional)- Delimiter at which splits occur. If not provided, the string is splitted at whitespaces.
* **maxsplit** (optional) - Maximum number of splits. If not provided, there is no limit on the number of splits.

## split() Return Value

The split() method returns a list of strings.

## Example 1: How split() works in Python?

text= 'Love thy neighbor'

# splits at space

print(text.split())

grocery = 'Milk, Chicken, Bread'

# splits at ','

print(grocery.split(', '))

# Splits at ':'

print(grocery.split(':'))

[Run Code](https://www.programiz.com/python-programming/online-compiler)

**Output**

['Love', 'thy', 'neighbor']

['Milk', 'Chicken', 'Bread']

['Milk, Chicken, Bread']

## Example 2: How split() works when maxsplit is specified?

grocery = 'Milk, Chicken, Bread, Butter'

# maxsplit: 2

print(grocery.split(', ', 2))

# maxsplit: 1

print(grocery.split(', ', 1))

# maxsplit: 5

print(grocery.split(', ', 5))

# maxsplit: 0

print(grocery.split(', ', 0))

[Run Code](https://www.programiz.com/python-programming/online-compiler)

**Output**

['Milk', 'Chicken', 'Bread, Butter']

['Milk', 'Chicken, Bread, Butter']

['Milk', 'Chicken', 'Bread', 'Butter']

['Milk, Chicken, Bread, Butter']

## Python For Loops

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

This is less like the for keyword in other programming languages, and works more like an iterator method as found in other object-orientated programming languages.

With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc.

### **Example**

Print each fruit in a fruit list:

fruits = ["apple", "banana", "cherry"]  
for x in fruits:  
  print(x)

The for loop does not require an indexing variable to set beforehand.

. Contrast the for statement with the [''while'' loop](https://wiki.python.org/moin/WhileLoop), used when a condition needs to be checked each iteration or to repeat a block of code forever. For example:

For loop from 0 to 2, therefore running 3 times.

for x in range(0, 3):

print("We're on time %d" % (x))

While loop from 1 to infinity, therefore running forever.

x = 1

while True:

print("To infinity and beyond! We're getting close, on %d now!" % (x))

x += 1

When running the above example, you can stop the program by pressing ctrl+c at the same time. As you can see, these loop constructs serve different purposes. The for loop runs for a fixed amount of times, while the while loop runs until the loop condition changes. In this example, the condition is the boolean True which will never change, so it will run forever.

The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and stops before a specified number.

## Syntax

range(start, stop, step)

## Parameter Values

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| start | Optional. An integer number specifying at which position to start. Default is 0 |
| stop | Required. An integer number specifying at which position to stop (not included). |
| step | Optional. An integer number specifying the incrementation. Default is 1 |

## More Examples

### **Example**

Create a sequence of numbers from 3 to 5, and print each item in the sequence:

x = range(3, 6)  
for n in x:  
  print(n)

### **Example**

Create a sequence of numbers from 3 to 19, but increment by 2 instead of 1:

x = range(3, 20, 2)  
for n in x:  
  print(n)

Built-in *functions* in python: Refer the following link to know about more built-in python functions.

https://docs.python.org/3/library/functions.html

*title()* function :

str.**title**()[¶](https://docs.python.org/3/library/stdtypes.html#str.title)

Return a titlecased version of the string where words start with an uppercase character and the remaining characters are lowercase.

For example:

>>>

**>>>** 'Hello world'.title()

'Hello World'

The algorithm uses a simple language-independent definition of a word as groups of consecutive letters. The definition works in many contexts but it means that apostrophes in contractions and possessives form word boundaries, which may not be the desired result:

>>>

**>>>** "they're bill's friends from the UK".title()

"They'Re Bill'S Friends From The Uk"

The [string.capwords()](https://docs.python.org/3/library/string.html" \l "string.capwords" \o "string.capwords) function does not have this problem, as it splits words on spaces only.

Alternatively, a workaround for apostrophes can be constructed using regular expressions:

>>>

**>>> import** **re**

**>>> def** titlecase(s):

**...**  **return** re.sub(r"[A-Za-z]+('[A-Za-z]+)?",

**...**  **lambda** mo: mo.group(0).capitalize(),

**...**  s)

**...**

**>>>** titlecase("they're bill's friends.")

"They're Bill's Friends."

**Block Scope:** The block scope of python is different from other languages as C/C++ etc. Refer the **stackoverlow** to know about Python’s block scope.

<https://stackoverflow.com/questions/6167923/block-scope-in-python>

Global Constants: Refer the stack website to know more about global variables.

<https://stackoverflow.com/questions/24133929/how-do-you-declare-a-global-constant-in-python>