Python3 basics

Michele La Manna
Dept. of Information Engineering
University of Pisa
michele.lamanna@phd.unipi.it
Version: 2022-03-14

PYTHON3

What is Python?

Python is an **easy to learn**, powerful programming language.

It has a **simple but effective** approach to object-oriented programming.

Python is an **ideal language for scripting** and rapid application development in many areas on **most platforms**.

Major Perks:

- ✓ Easy to read;
- ✓ Easy to learn;
- ✓ Intuitive string manipulation;
- ✓ Already installed on UNIX;
- ✓ No semicolon (;) at the end of an instruction!

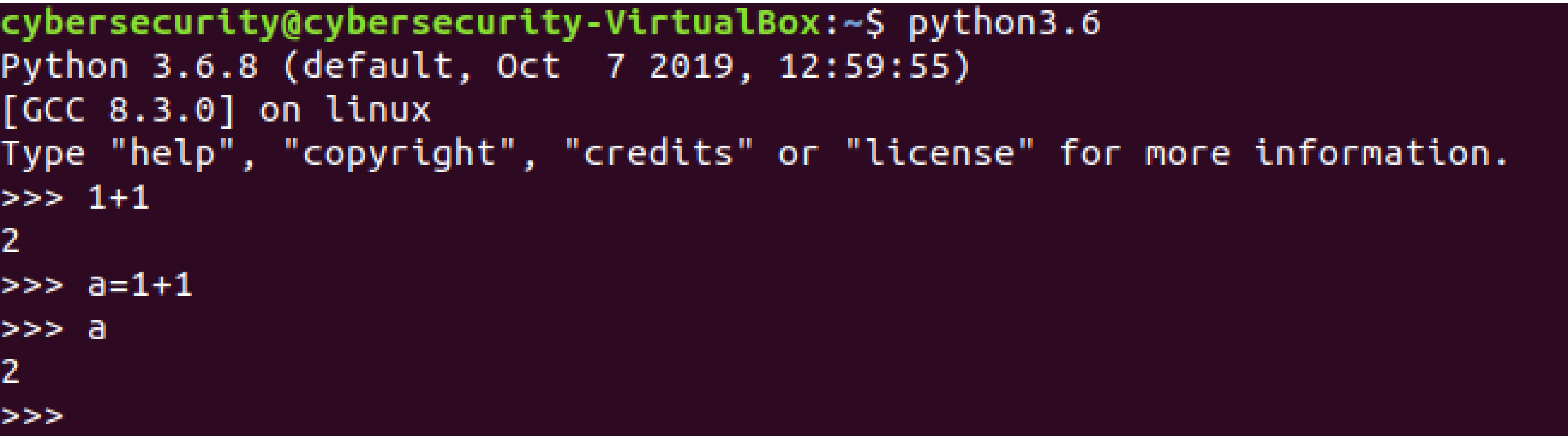
Major Cons:

- × Must indent code;
- × Slower than C/C++;
- × Beware of your python version!

Python3.6

```
sudo apt-get install python3.6
```

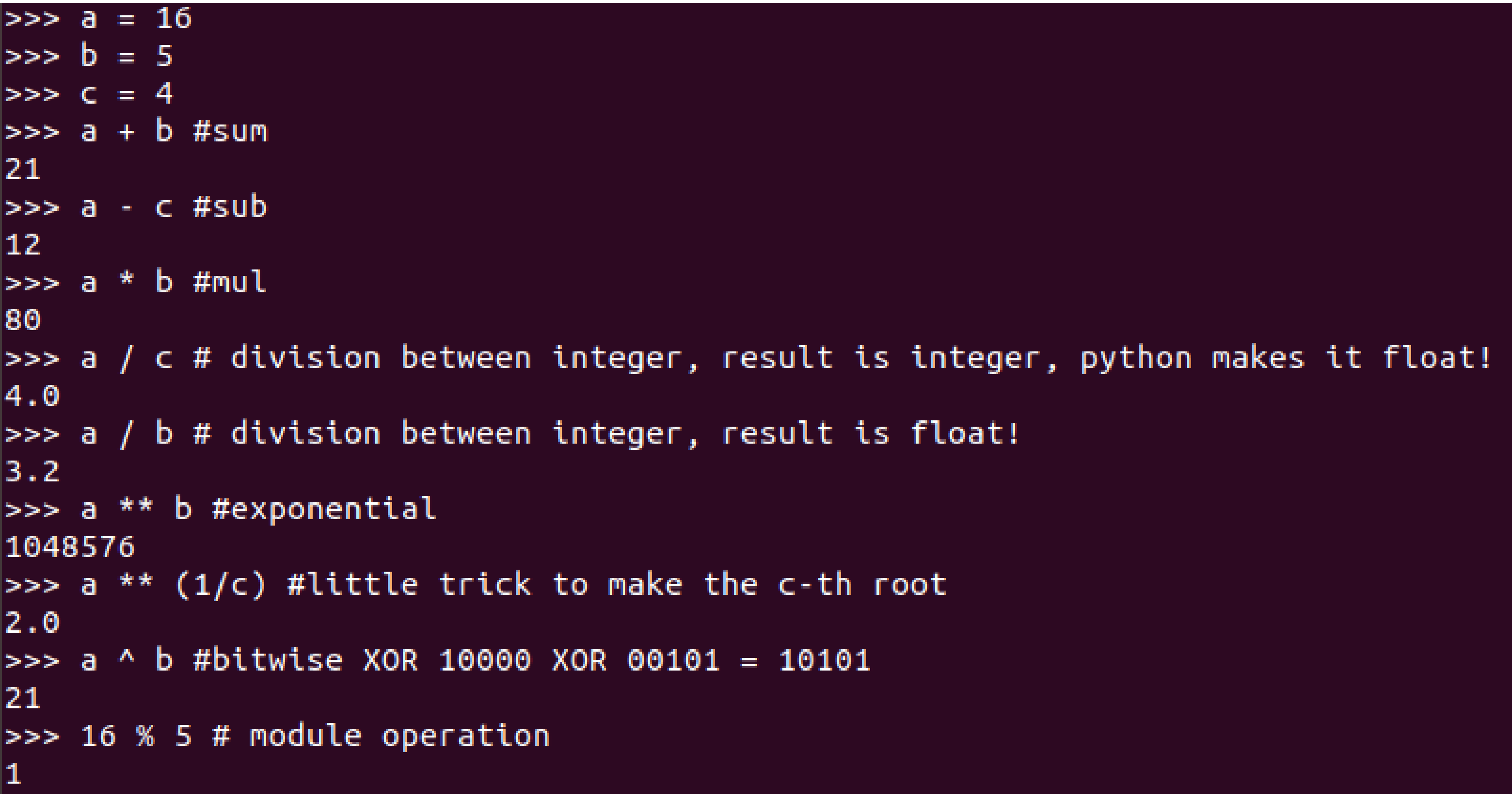
You can use python directly from the terminal
Invoking python:



```
cybersecurity@cybersecurity-VirtualBox:~$ python3.6
Python 3.6.8 (default, Oct  7 2019, 12:59:55)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 1+1
2
>>> a=1+1
>>> a
2
>>>
```

As you can see, you cannot specify the type of the variables for example:
`int a=2;`
Python automatically knows the type of the variable based on the assignment.

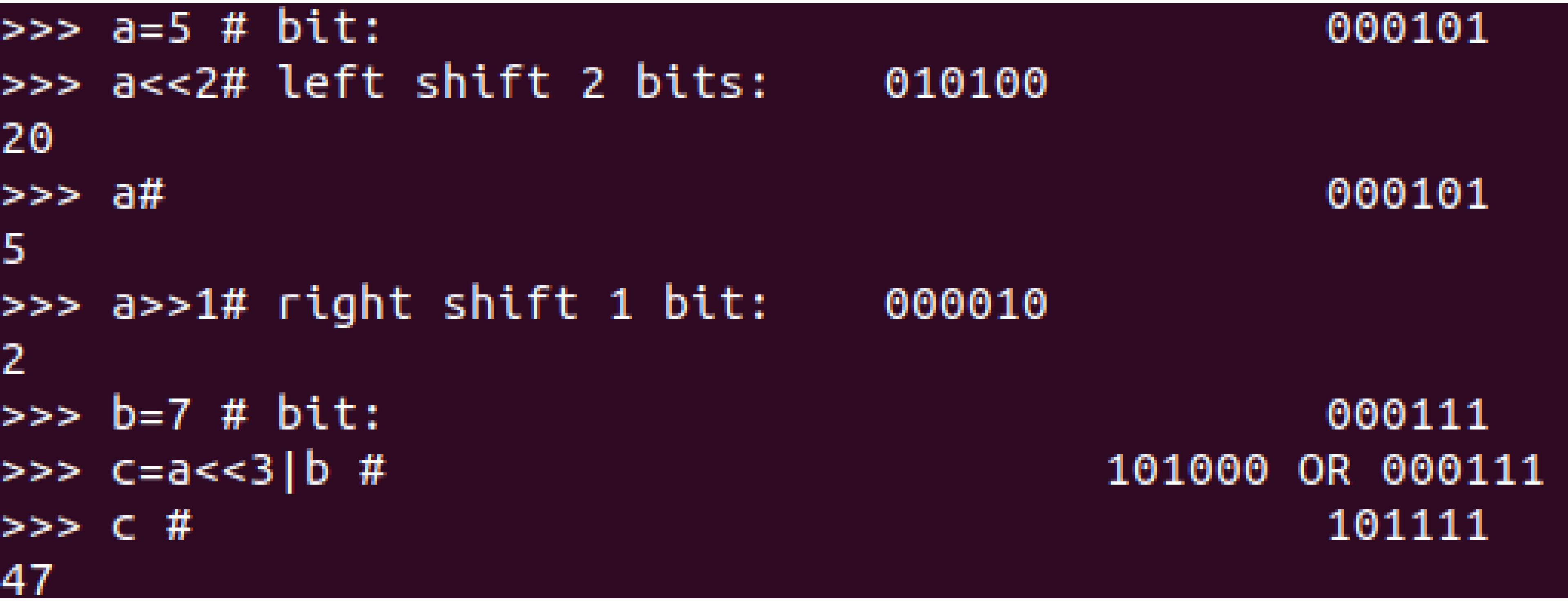
Basic math operations



```
>>> a = 16
>>> b = 5
>>> c = 4
>>> a + b #sum
21
>>> a - c #sub
12
>>> a * b #mul
80
>>> a / c # division between integer, result is integer, python makes it float!
4.0
>>> a / b # division between integer, result is float!
3.2
>>> a ** b #exponential
1048576
>>> a ** (1/c) #little trick to make the c-th root
2.0
>>> a ^ b #bitwise XOR 10000 XOR 00101 = 10101
21
>>> 16 % 5 # module operation
1
```

Basic math operations

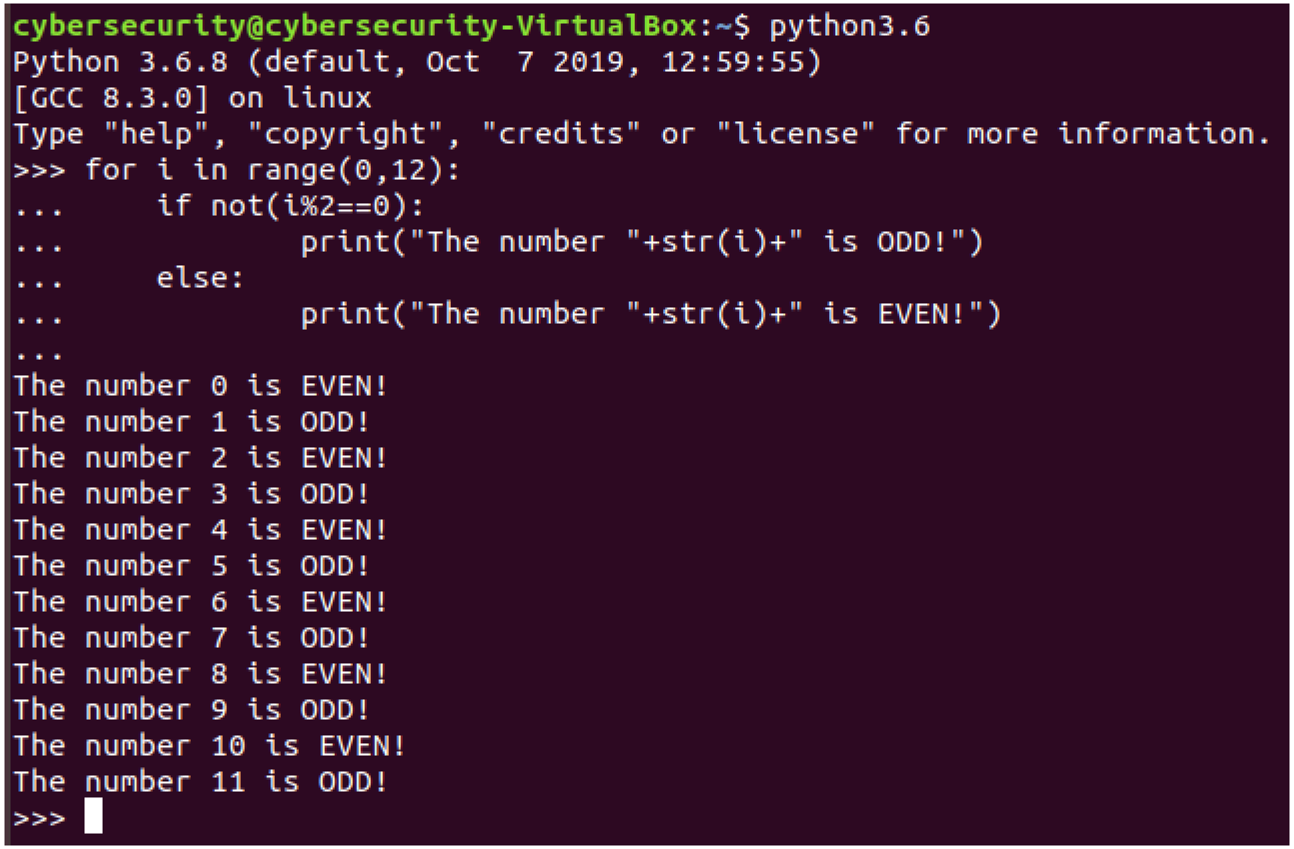
You can also manipulate bits with right or left shifts.
Then, you can combine those bits in a single number!



```
>>> a=5 # bit:                                    000101
>>> a<<2# left shift 2 bits:     010100
20
>>> a#                                            000101
5
>>> a>>1# right shift 1 bit:     000010
2
>>> b=7 # bit:                                    000111
>>> c=a<<3|b #                            101000 OR 000111
>>> c #                                           101111
47
```

Loops and conditions

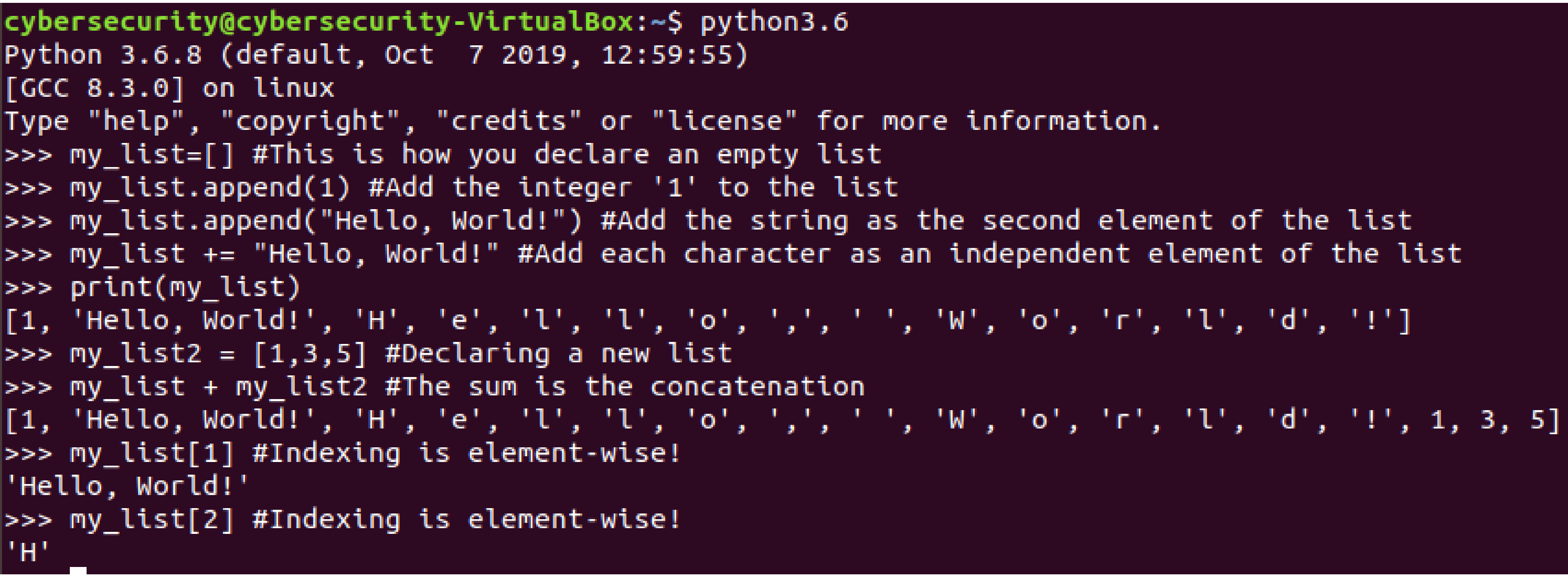
For loops and if statements do not need parenthesis.
However, you have to put a colon at the end of the statement.
Furthermore, remember to correctly indent your code!



```
cybersecurity@cybersecurity-VirtualBox:~$ python3.6
Python 3.6.8 (default, Oct  7 2019, 12:59:55)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> for i in range(0,12):
...     if not(i%2==0):
...             print("The number "+str(i)+" is ODD!")
...     else:
...             print("The number "+str(i)+" is EVEN!")
...
The number 0 is EVEN!
The number 1 is ODD!
The number 2 is EVEN!
The number 3 is ODD!
The number 4 is EVEN!
The number 5 is ODD!
The number 6 is EVEN!
The number 7 is ODD!
The number 8 is EVEN!
The number 9 is ODD!
The number 10 is EVEN!
The number 11 is ODD!
>>> 
```

Lists

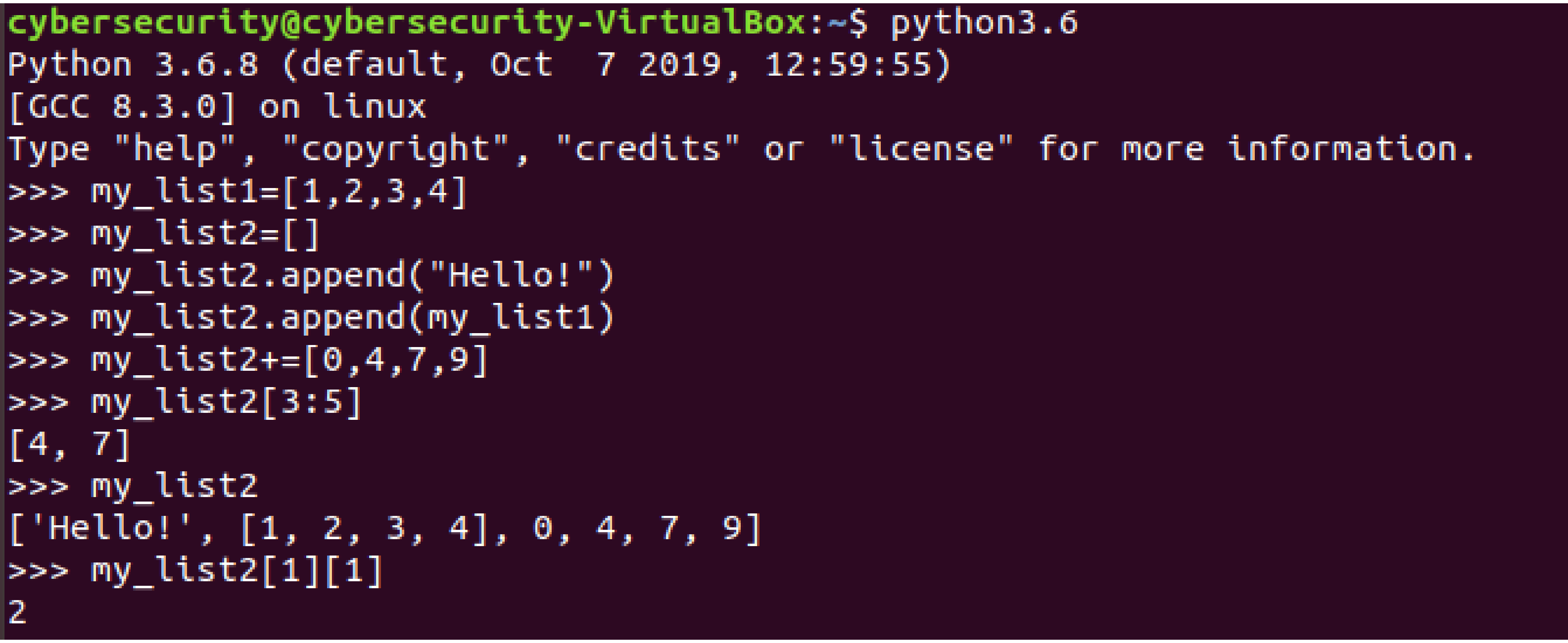
The most useful structure in python is the List.
Think of a List in python as an array in C++, but much more flexible!
A List can contain elements of the same type, but also different elements!



```
cybersecurity@cybersecurity-VirtualBox:~$ python3.6
Python 3.6.8 (default, Oct  7 2019, 12:59:55)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> my_list=[] #This is how you declare an empty list
>>> my_list.append(1) #Add the integer '1' to the list
>>> my_list.append("Hello, World!") #Add the string as the second element of the list
>>> my_list += "Hello, World!" #Add each character as an independent element of the list
>>> print(my_list)
[1, 'Hello, World!', 'H', 'e', 'l', 'l', 'o', ',', ' ', 'W', 'o', 'r', 'l', 'd', '!']
>>> my_list2 = [1,3,5] #Declaring a new list
>>> my_list + my_list2 #The sum is the concatenation
[1, 'Hello, World!', 'H', 'e', 'l', 'l', 'o', ',', ' ', 'W', 'o', 'r', 'l', 'd', '!', 1, 3, 5]
>>> my_list[1] #Indexing is element-wise!
'Hello, World!'
>>> my_list[2] #Indexing is element-wise!
'H'
```

Lists

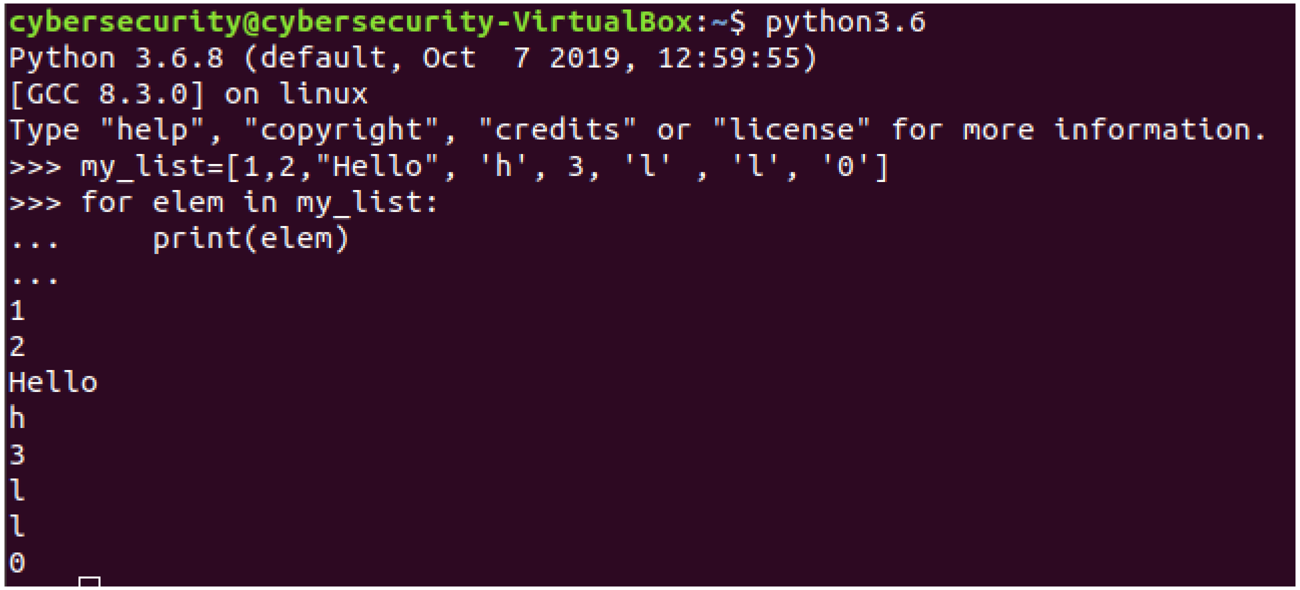
Lists can also be nested!



```
cybersecurity@cybersecurity-VirtualBox:~$ python3.6
Python 3.6.8 (default, Oct  7 2019, 12:59:55)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> my_list1=[1,2,3,4]
>>> my_list2=[]
>>> my_list2.append("Hello!")
>>> my_list2.append(my_list1)
>>> my_list2+=[0,4,7,9]
>>> my_list2[3:5]
[4, 7]
>>> my_list2
['Hello!', [1, 2, 3, 4], 0, 4, 7, 9]
>>> my_list2[1][1]
2
```

For and lists

Take a look!

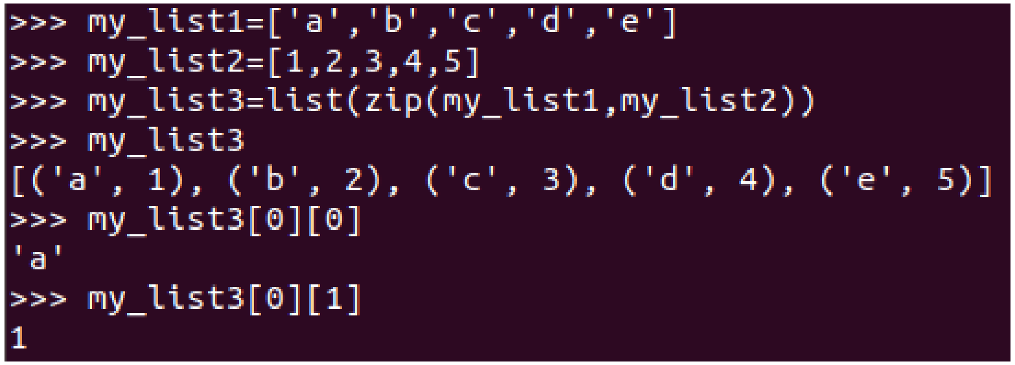


```
cybersecurity@cybersecurity-VirtualBox:~$ python3.6
Python 3.6.8 (default, Oct  7 2019, 12:59:55)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> my_list=[1,2,"Hello", 'h', 3, 'l' , 'l', '0']
>>> for elem in my_list:
...     print(elem)
...
1
2
Hello
h
3
l
l
0
```

This lines means
`for` (each element named) `elem in`(side the list named) `my_list:`
        `print("elem")`
Elem is not an index, but it assumes the *value* of the elements inside the list!

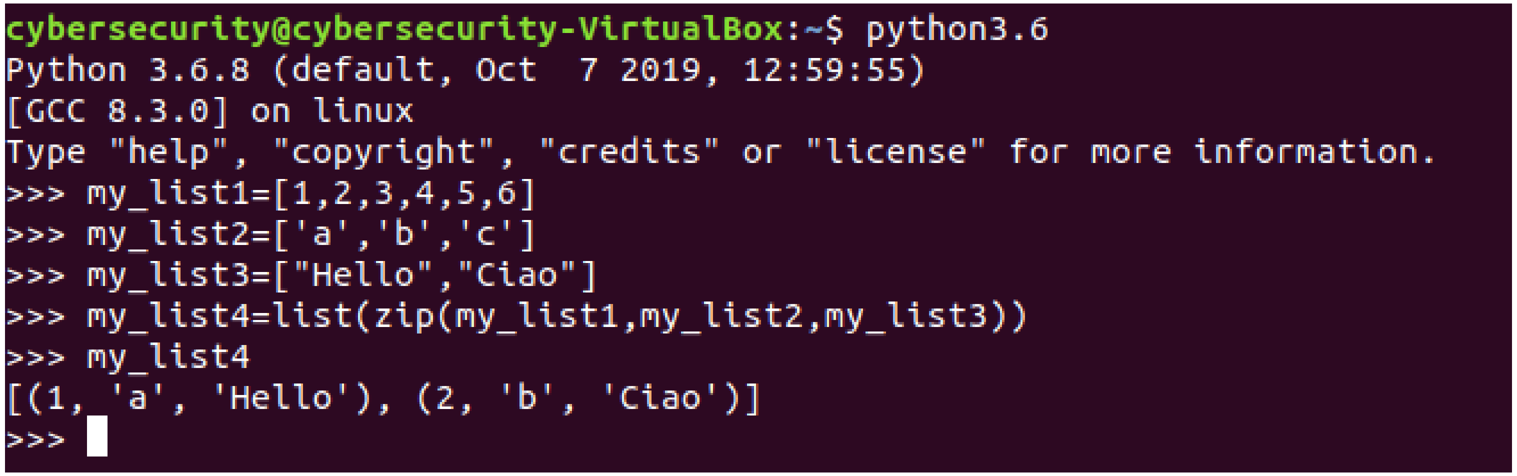
zip() command

Merges zero or more equal-length lists:



```
>>> my_list1=['a','b','c','d','e']
>>> my_list2=[1,2,3,4,5]
>>> my_list3=list(zip(my_list1,my_list2))
>>> my_list3
[('a', 1), ('b', 2), ('c', 3), ('d', 4), ('e', 5)]
>>> my_list3[0][0]
'a'
>>> my_list3[0][1]
1
```

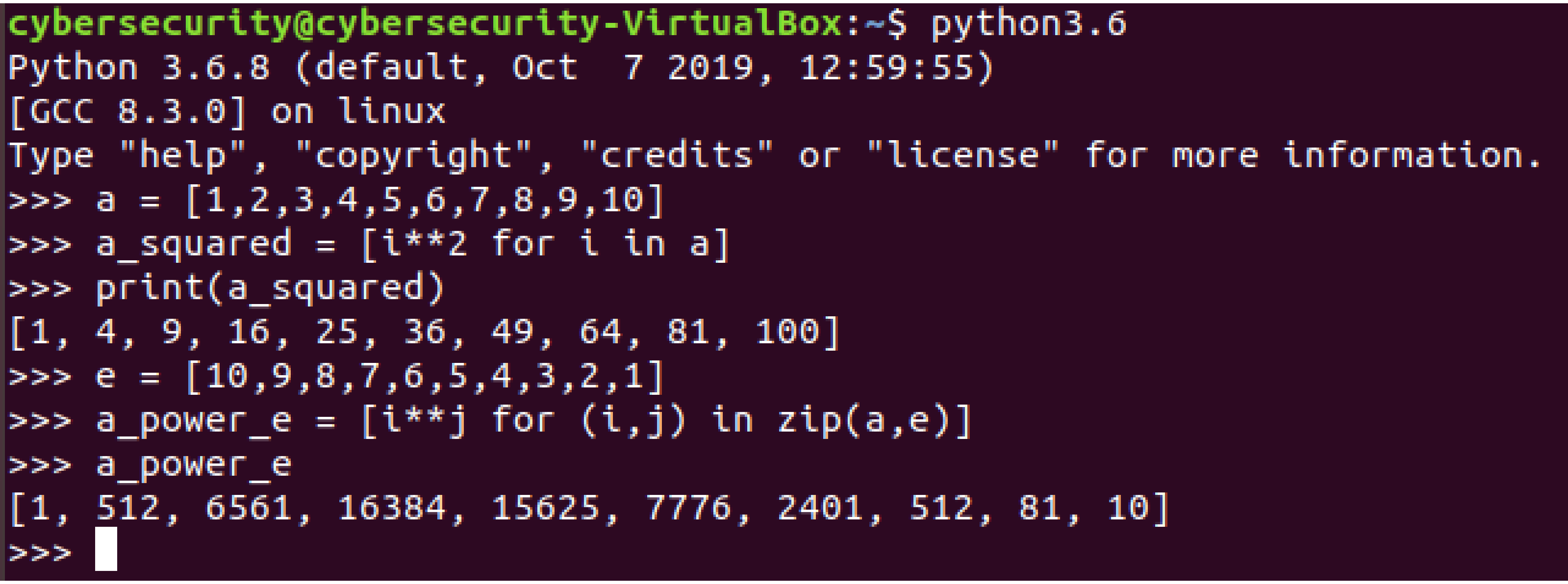
What if lists have different lengths?



```
cybersecurity@cybersecurity-VirtualBox:~$ python3.6
Python 3.6.8 (default, Oct  7 2019, 12:59:55)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> my_list1=[1,2,3,4,5,6]
>>> my_list2=['a','b','c']
>>> my_list3=["Hello","Ciao"]
>>> my_list4=list(zip(my_list1,my_list2,my_list3))
>>> my_list4
[(1, 'a', 'Hello'), (2, 'b', 'Ciao')]
>>> 
```

For, lists and zip

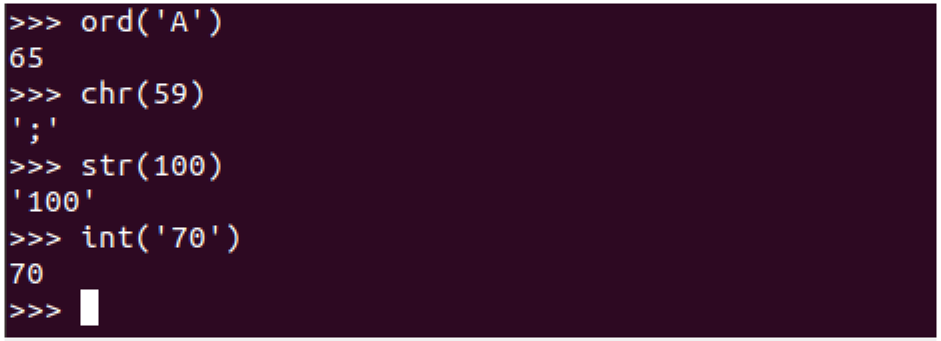
You can build lists using for loops also in a single line!
Check out the example below (list comprehension):
Are you able to understand what's going on?



```
cybersecurity@cybersecurity-VirtualBox:~$ python3.6
Python 3.6.8 (default, Oct  7 2019, 12:59:55)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> a = [1,2,3,4,5,6,7,8,9,10]
>>> a_squared = [i**2 for i in a]
>>> print(a_squared)
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
>>> e = [10,9,8,7,6,5,4,3,2,1]
>>> a_power_e = [i**j for (i,j) in zip(a,e)]
>>> a_power_e
[1, 512, 6561, 16384, 15625, 7776, 2401, 512, 81, 10]
>>> 
```

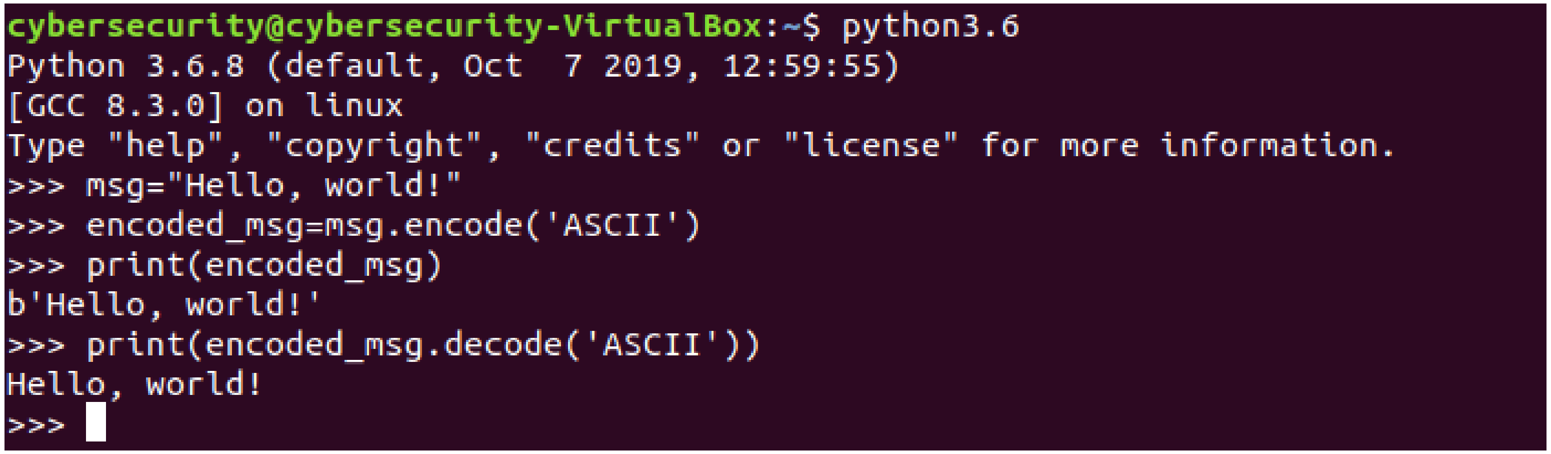
Conversion commands

`ord('<char>')` converts a single character into its ASCII integer value, `chr(<int>)` does the opposite.



```
>>> ord('A')
65
>>> chr(59)
';'
>>> str(100)
'100'
>>> int('70')
70
>>> 
```

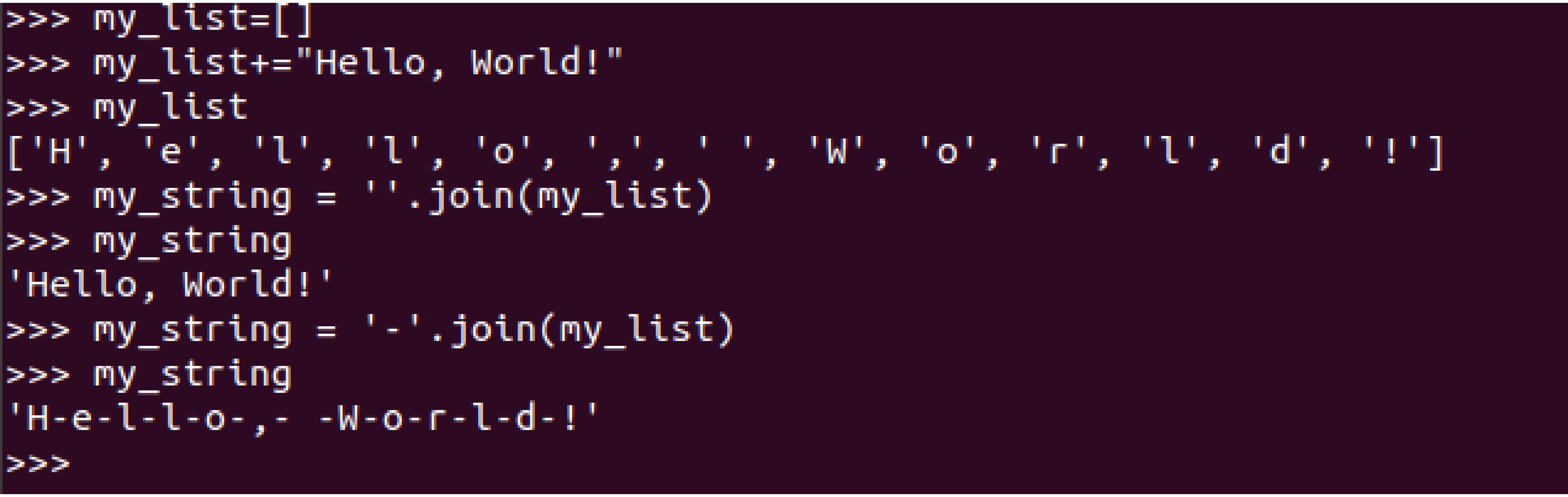
For strings, use the functions `encode('<format>')`, `decode('<format>')`



```
cybersecurity@cybersecurity-VirtualBox:~$ python3.6
Python 3.6.8 (default, Oct  7 2019, 12:59:55)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> msg="Hello, world!"
>>> encoded_msg=msg.encode('ASCII')
>>> print(encoded_msg)
b'Hello, world!'
>>> print(encoded_msg.decode('ASCII'))
Hello, world!
>>> 
```

Join command

If you have a **list of characters or strings**, and want them to be a single string, use: `'<glue>'.join(<list>)`

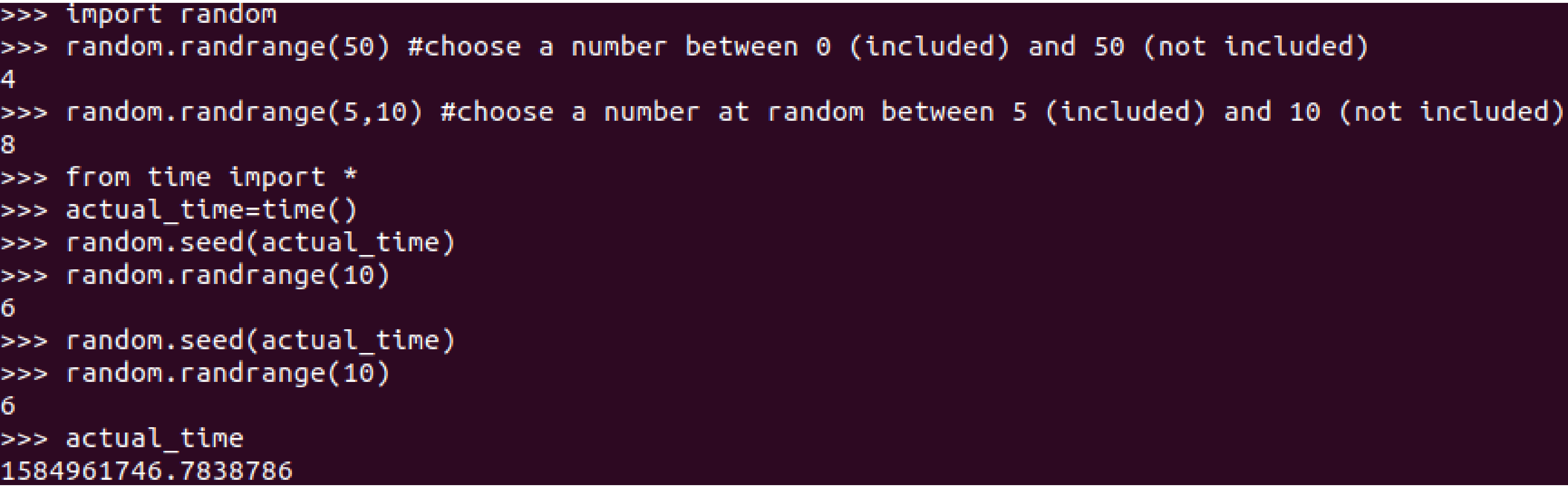


```
>>> my_list=[]
>>> my_list+="Hello, World!"
>>> my_list
['H', 'e', 'l', 'l', 'o', ',', ' ', 'W', 'o', 'r', 'l', 'd', '!']
>>> my_string = ''.join(my_list)
>>> my_string
'Hello, World!'
>>> my_string = '-'.join(my_list)
>>> my_string
'H-e-l-l-o-,- -W-o-r-l-d-!'
>>>
```

The function "glues" every character of the list together through the parameter <glue>. Of course, you can put absolutely nothing as the <glue> parameter!

modules

Python has libraries, called modules, just like any other language!
Let's see an example:



```
>>> import random
>>> random.randrange(50) #choose a number between 0 (included) and 50 (not included)
4
>>> random.randrange(5,10) #choose a number at random between 5 (included) and 10 (not included)
8
>>> from time import *
>>> actual_time=time()
>>> random.seed(actual_time)
>>> random.randrange(10)
6
>>> random.seed(actual_time)
>>> random.randrange(10)
6
>>> actual_time
1584961746.7838786
```

The module random lets you play with randomness.
One of the most useful command is `randrange()`.
The function `time.time()` returns the number of seconds elapsed since epoch (For UNIX systems: 1 jan 1970).
Timestamp are not fixed in length!

Confused? Good!

Now it’s time to cement those concepts with some practice!
Download the archive, you will find 3 exercises:

- Nuclear_threat/Easy
- Nuclear_threat/Hard
- evil_man

In each folder you will find:

- A ciphertext file;
- a code in python3 who lets you encrypt a message;
- some instructions which give you a little context, just for fun!

Your objective is to analyze the python code so that you can write a script that decrypts the ciphertext. The message contains a string between two curly brackets, for example: {This_1s_A_dem0_5tr1ng}

We will give you hints every 10 to 20 minutes!
Good luck!