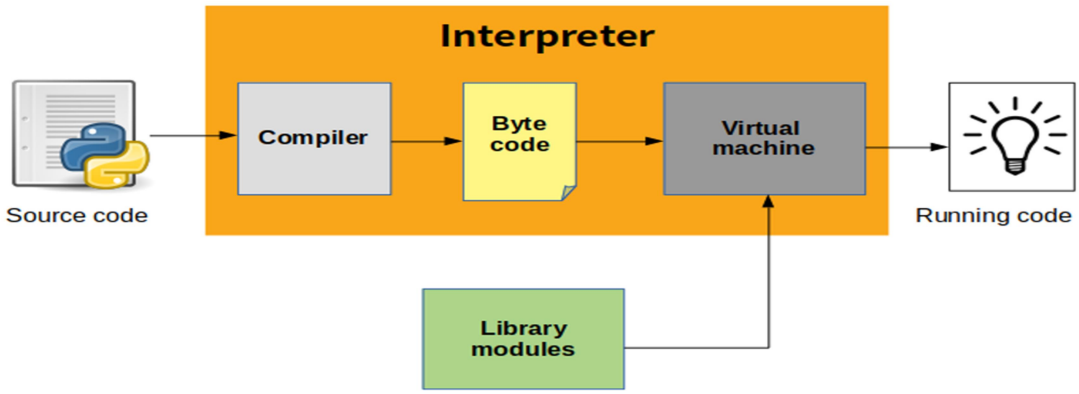
1> How is Python interpreted?



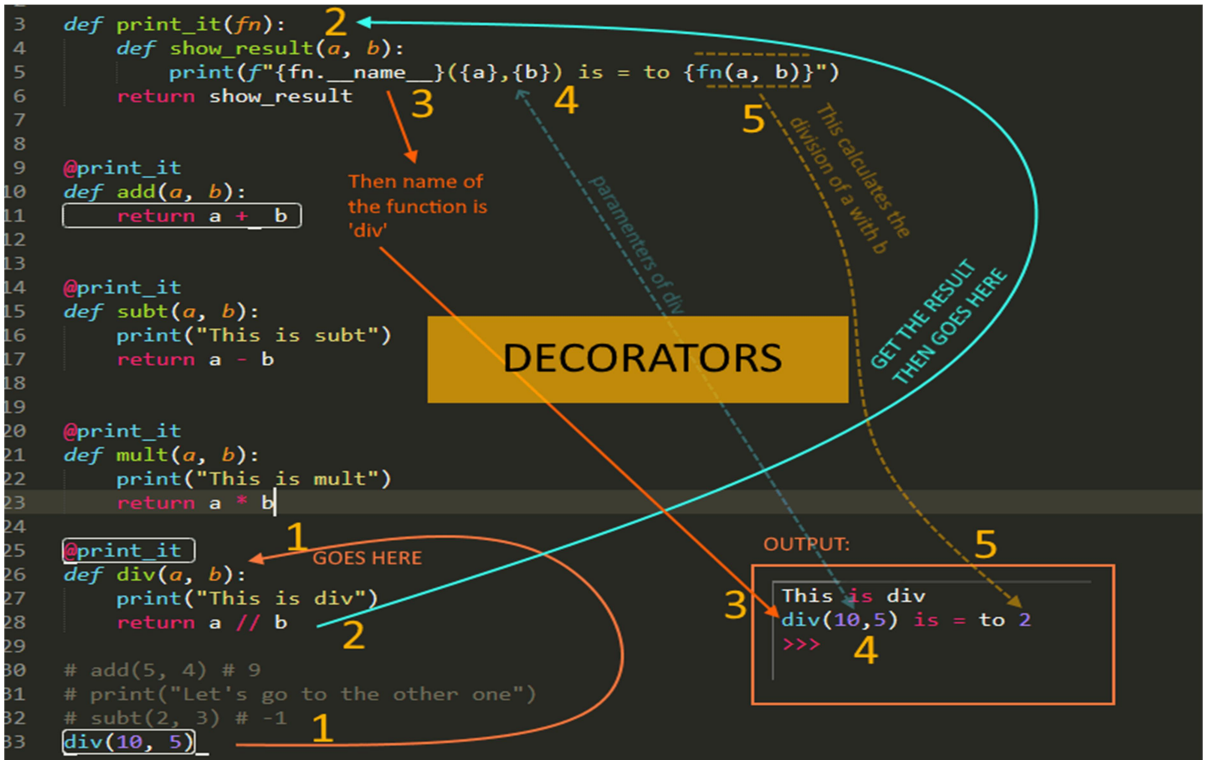
The **python code** you write is compiled into python bytecode, which creates file with extension **.pyc** . The bytecode compilation happened internally, and almost completely hidden from developer. Compilation is simply a translation step, and byte code is a lower-level, and **platform-independent** , representation of your source code. Roughly, each of your source statements is translated into a group of byte code instructions. This byte code translation is performed to speed execution **byte code** can be run much quicker than the original source code statements.

The **.pyc file** , created in compilation step, is then executed by appropriate virtual machines. The Virtual Machine just a big loop that iterates through your **byte code** instructions, one by one, to carry out their operations. The **Virtual Machine** is the runtime engine of Python and it is always present as part of the Python system, and is the component that truly runs the **Python scripts** . Technically, it's just the last step of what is called the Python interpreter.

2>>What are Python Decorators?

Decorators are a very powerful and useful tool in Python since it allows programmers to modify the behaviour of a function or class. Decorators allow us to wrap another function in order to extend the behaviour of the wrapped function, without permanently modifying it

As stated above the decorators are used to modify the behaviour of function or class. In Decorators, functions are taken as the argument into another function and then called inside the wrapper function.



3>> What is the difference between list and tuple?

Tuples and Lists: Key differences & similarity

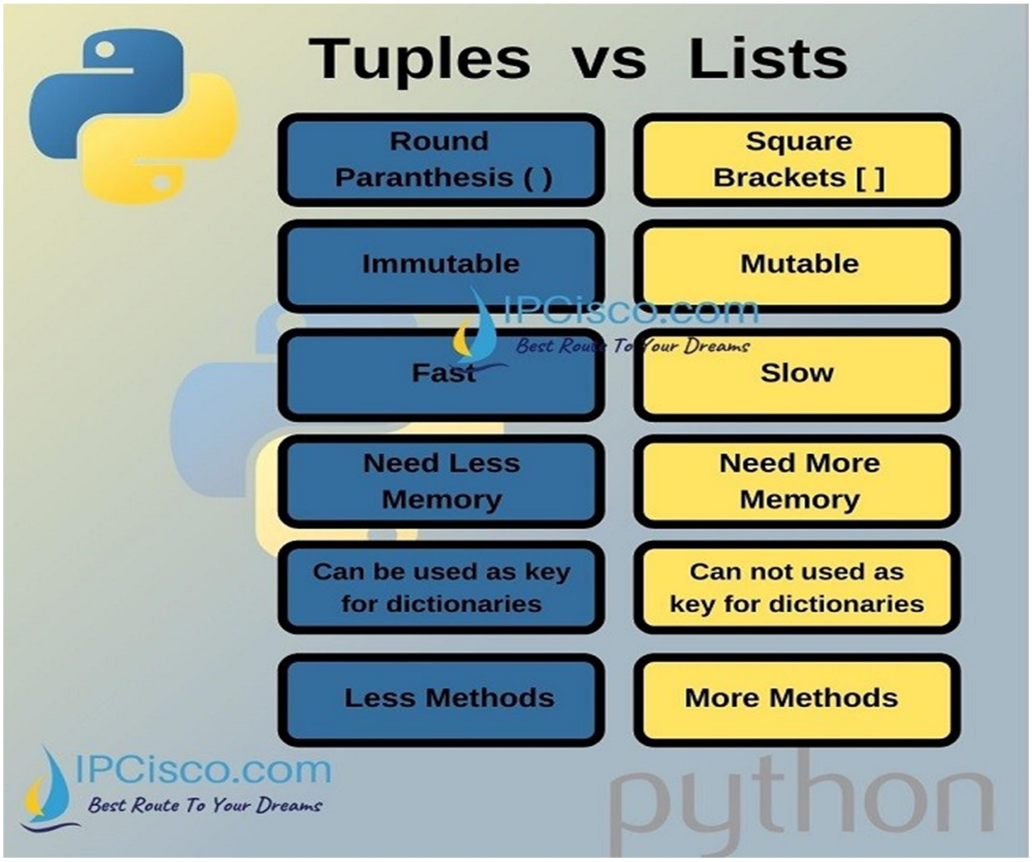
- They are both used to store collection of data
- They are both heterogeneous data types means that you can store any kind of data type
- They are both ordered means the order in which you put the items are kept.
- They are both sequential data types so you can iterate over the items contained.

- Items of both types can be accessed by an integer index operator, provided in square brackets, [index]

- The key difference between the tuples and lists is that while the **tuples are immutable objects** the **lists are mutable**. This means that tuples cannot be changed while the lists can be modified.

  - As lists are mutable, Python needs to allocate an extra memory block in case there is a need to extend the size of the list object after it is created. In contrary, as tuples are immutable and fixed size, Python allocates just the minimum memory block required for the data.

  - As a result, tuples are **more memory efficient** than the lists.



4>>> list vs tuple vs set

All are heterogeneous data type(non-homogeneous)

LIST	TUPLE	DICTIONARY	SET
Allows duplicate members	Allows duplicate members	No duplicate members	No duplicate members
Changeble	Not changeable	Changeable, indexed	Cannot be changed, but can be added, non -indexed
Ordered	Ordered	Unordered	Unordered
Square bracket [ ]	Round brackets ( )	Curly brackets{ }	Curly brackets{ }

5>> How are argument s passed by value or by reference?

Python uses a mechanism, which is known as "**Call-by-Object**", sometimes also called "**Call by Object Reference**" or "**Call by Sharing**"

If you pass immutable arguments like integers, strings or tuples to a function, the passing acts like **Call-by-value**. It's different, if we pass mutable arguments.

All **parameters (arguments**) in the Python language are **passed by reference**. It means if you change what a parameter refers to within a function, the change also reflects back in the calling function.

By geeks for geeks….>>

You might want to punch something after reading ahead, so brace yourself. Python's argument passing model is **neither "Pass by Value" nor "Pass by Reference" but it is "Pass by Object Reference".**
The paradigms of "Pass by value", "Pass by Reference" and "Pass by object Reference"

Python utilizes a system, which is known as "Call by Object Reference" or "Call by assignment". In the event that you pass arguments like whole numbers, strings or tuples to a function, the passing is like call-by-value because you can not change the value of the immutable objects being passed to the function. Whereas passing mutable objects can be considered as call by reference because when their values are changed inside the function, then it will also be reflected outside the function.

6>> **What is the difference between Xrange and range?**

- **range()** – This returns a range object (a type of iterable).
- **xrange()** – This function returns the **generator object** that can be used to display numbers only by looping. The only particular range is displayed on demand and hence called "**lazy evaluation**".

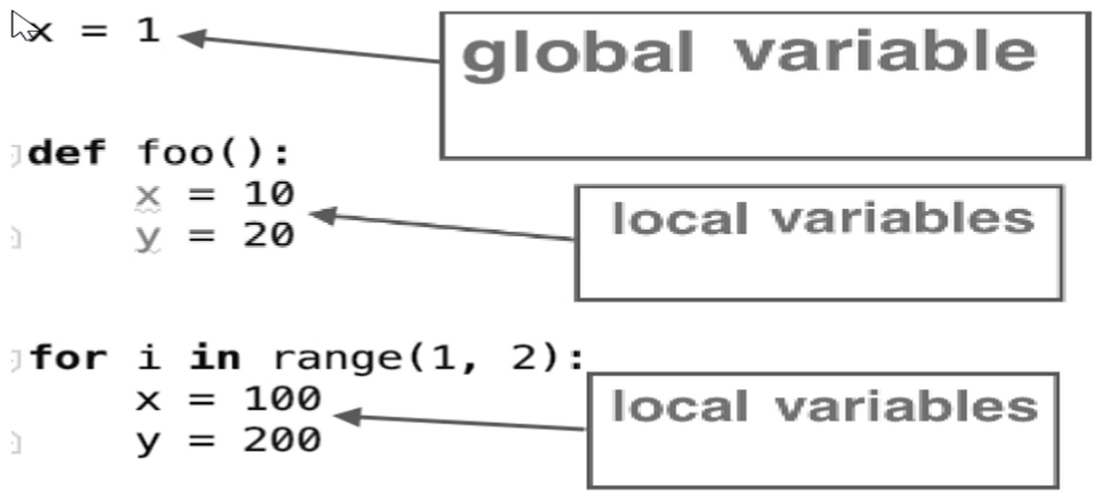
7>> Mention what are the rules for local and global variables in python?

In Python, variables that are only referenced inside a function are implicitly global. If a variable is assigned a value anywhere within the function's body, it's assumed to be a local unless explicitly declared as global.

Though a bit surprising at first, a moment's consideration explains this. On one hand, requiring `global` for assigned variables provides a bar against unintended side-effects. On the other hand, if `global` was required for all global references, you'd be using `global` all the time. You'd have to declare as global every reference to a built-in function or to a component of an imported module. This clutter would defeat the usefulness of the `global` declaration for identifying side-effects.

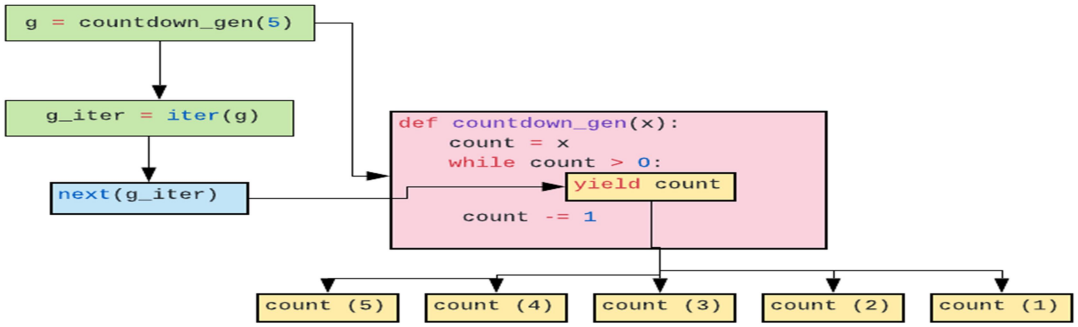
Global variables are those which are not defined inside any function and have a global scope whereas **local variables** are those which are defined inside a function and its scope is limited to that function only. In other words, we can say that local variables are accessible only inside the function in which it was initialized whereas the global variables are accessible throughout the program and inside every function.

Local variables are those which are initialized inside a function and belong only to that particular function. It cannot be accessed anywhere outside the function. Let's see how to create a local variable.

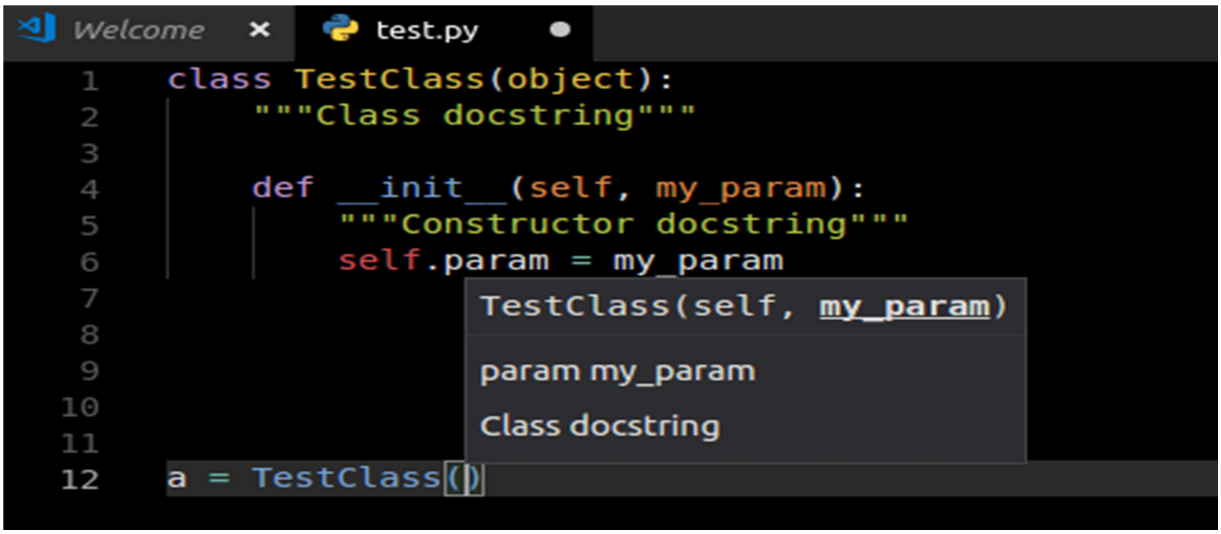


8>>**What are generator s in Python?**

Generator-Function : A generator-function is defined like a normal function, but whenever it needs to generate a value, it does so with the yield keyword rather than return. If the body of a def contains yield, the function automatically becomes a generator function.



9>> **What is docstring in Python?**



A Python docstring is a string used to document a Python module, class, function or method, so programmers can understand what it does without having to read the details of the implementation.

Also, it is a common practice to generate online (html) documentation automatically from docstrings.

Docstrings must be defined with three double-quotes. No blank lines should be left before or after the docstring. The text starts in the next line after the opening quotes. The closing quotes have their own line (meaning that they are not at the end of the last sentence).

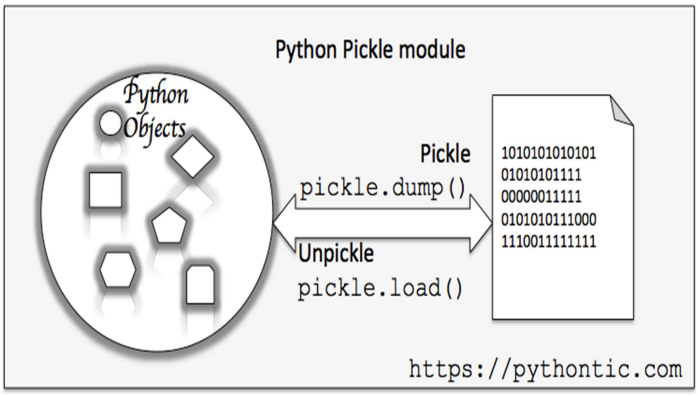
10>> What is PEP 8?

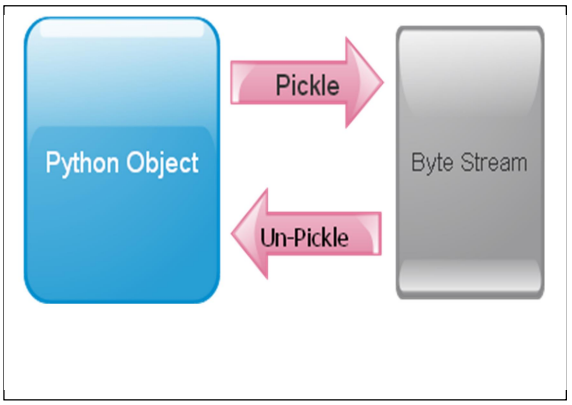
PEP 8, sometimes spelled PEP8 or PEP-8, is **a document that provides guidelines and best practices on how to write Python code**. It was written in 2001 by Guido van Rossum, Barry Warsaw, and Nick Coghlan. The primary focus of PEP 8 is to improve the readability and consistency of Python code.

11>> What is pickling and unpickling?

The pickle module is used for implementing binary protocols for serializing and de-serializing a Python object structure.

- Pickling: It is a process where a Python object hierarchy is converted into a byte stream.
- Unpickling: It is the inverse of Pickling process where a byte stream is converted into an object hierarchy.





12>>Explain how you can generate random number s in Python?

Python defines a set of functions that are used to generate or manipulate random numbers through the **random module.**
Functions in the random module rely on a pseudo-random number generator function **random()**, which generates a random float number between 0.0 and 1.0. These particular type of functions is used in a lot of games, lotteries, or any application requiring a random number generation.

1. import random>>>>> n = random. random() >>>>>>print(n)
2. import random>>>>>> n = random. randint(0,22) >>>>>>>>>>>print(n)
3. import random>>>> randomlist = [] for i in range(0,5):>> n = random. randint(1,30) >>>randomlist.

13>>>>**What is the purpose of the pass statement in Python**?

in Python programming, the `pass` statement is a null statement. The difference between a comment and a `pass` statement in Python is that while the interpreter ignores a comment entirely, `pass` is not ignored. However, nothing happens when the pass is executed. It results in no operation (NOP).

n Python programming, the `pass` statement is a null statement. The difference between a comment and a `pass` statement in Python is that while the interpreter ignores a comment entirely, `pass` is not ignored. However, nothing happens when the pass is executed. It results in no operation (NOP).

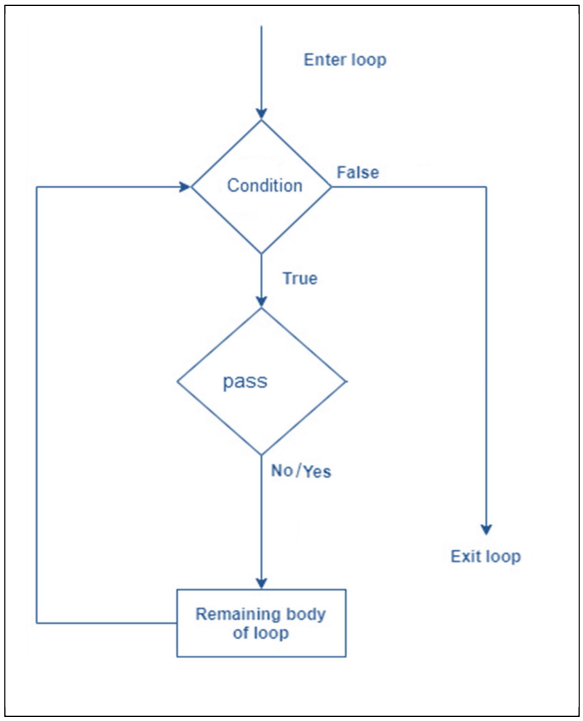
n Python programming, the `pass` statement is a null statement. The difference between a comment and a `pass` statement in Python is that while the interpreter ignores a comment entirely, `pass` is not ignored. However, nothing happens when the pass is executed. It results in no operation (NOP).

```
x = 3

if (x == 3):
    pass
else:
    print("x is not equal to 3")
```

```
C:\Users\ak111\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/ak111/PycharmPr

Process finished with exit code 0
```



14>>> What is the purpose of the PYTHONPATH environment variable?

Python's behavior is greatly influenced by its environment variables. One of those variables is PYTHONPATH. It is used to set the path for the user-defined modules so that it can be directly imported into a Python program. It is also responsible for handling the default search path for Python Modules. The PYTHONPATH variable holds a string with the name of various directories that need to be added to the sys.path directory list by Python. The primary use of this variable is to allow users to import modules that are not made installable yet. Let's try to understand the concept using an example.

15>> What is purpose of / / ,% and ** operators?

// operator = Floor division - division that results into whole number adjusted to the left in the number line

True Division vs Floor Division

The operator / is true division and the operator // returns floor division(round down after true divide).

```
In[1]: 23 // 7
Out [1]: 3
In[2]: 3 // 9
Out [2]: 0
In[2]: -4 // 3
Out [2]: -2
In[3]: 6 / 5
Out [3]: 1.2
```

Operator	Name	Description	Syntax	Example
+	Addition	Performs addition	c = a + b	a = 5, b = 5 then c = 10
-	Subtraction	Performs subtraction	c = a – b	a = 5, b = 3 then c = 2
*	Multiplication	Performs multiplication	c = a * b	a = 5, b = 5 then c = 25
/	Division	Performs division	c = a / b	a = 10, b = 5 then c = 2
%	Modulus	Performs division but returns the remainder	c = a % b	a = 15, b = 2 then c = 1
//	Floor Division	Performs division but returns the quotient in which the digits after the decimal points are removed	c = a // b	a = 15, b = 2 then c = 7
**	Exponent	Performs multiplication to power raised	c = a ** b	a = 2, b = 4 then c = 16

16 >>> How do you get a list of all the keys in a dictionary?

To get a list of all keys from a dictionary, you can simply use the dict.keys() function.
Example>>>

```
my_dict = {'name': 'TutorialsPoint', 'time': '15 years', 'location': 'India'}
key_list = list(my_dict.keys())
print(key_list)
```

output>>>

['name', 'time', 'location']

17 >> How will you check if all characters in a string are alphanumeric?

The `isalnum()` method returns True if all the characters are alphanumeric, meaning alphabet letter (a-z) and numbers (0-9)

18 >> Given the first and last names of all employees in your firm. What data type will you use to store it?

A list of first and last names is best stored as a list of dictionaries.

19>> If you are ever stuck in an infinite loop, how will you break out of it?

An infinite loop is a loop that runs indefinitely and it only stops with external intervention or when a break statement is found. You can stop an infinite loop with **CTRL + C** .