**Python**

* Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.

Interpreted- An interpreted language is a type of programming language for which most of its implementations execute instructions directly and freely, without previously compiling a program into machine-language instructions.

High level- human readable

* Python is dynamically typed, this means that you don’t need to state the types of variables when you declare them or anything like that. You can do things like x=111 and then x="I'm a string" without error.
* Writing Python code is quick but running it is often slower than compiled languages. Fortunately，Python allows the inclusion of C based extensions so bottlenecks can be optimized away and often are. The [numpy](https://www.edureka.co/blog/python-numpy-tutorial/" \t "_blank) package is a good example of this, it’s really quite quick because a lot of the number crunching it does isn’t actually done by Python

**Python list, tuples, sets and dictionaries**

<https://www.w3schools.com/python/python_lists.asp>

**List** ordered Indexed allow duplicates changeable

**Tuple** ordered indexed allow duplicates unchangeable

**Dictionary** unordered indexed not allow changeable

**Set** unordered not indexed not allow can not change, but can add

**Difference between append or extend**

Append adds its argument as a single element to the end of a list. The length of the list itself will increase by one. extend iterates over its argument adding each element to the list, extending the list. The length of the list will increase by however many elements were in the iterable argument.

x=[1,2]

x.extend([3,4])

x=[1,2,3,4]

x=[1,2]

x.append([3,4])

x=[1,2,[3,4]]

**Change Tuple Values**

Once a tuple is created, you cannot change its values. Tuples are unchangeable, or immutable as it also is called.But there is a workaround. You can convert the tuple into a list, change the list, and convert the list back into a tuple.

x = ("apple", "banana", "cherry")  
y = list(x)  
y[1] = "kiwi"  
x = tuple(y)

**Why tuples are faster than list?**

Tuples are stored in a single block of memory. Tuples are immutalbe so, It doesn't require extraspace to store new objects. Lists are allocated in two blocks: the fixed one with all the Python object information and a variable sized block for the data. It is the reason creating a tuple is faster than List.

Difference between list and tuple

1. List mutable tuple immutable
2. List size large tuple smaller
3. List variable length tuple fixed length
4. Syntax diff
5. Tuples cannot be copied as its immutable.  If you run tuple(tuple\_name), it will immediately return itself. Points to same address.

For example:

names = ('Nicholas', 'Michelle', 'Alex')

copyNames = tuple(names)

print(names is copyNames)

**Python code compilation (that makes it a programming language as well)**

* In Python, the source is compiled into a much simpler form called bytecode. These are instructions similar in spirit to CPU instructions, but instead of being executed by the CPU, they are executed by software called a virtual machine
* Python source code is automatically compiled into Python byte code by the CPython interpreter. Compiled code is usually stored in PYC (or PYO) files, and is regenerated when the source is updated, or when otherwise necessary.
* In short: it run your existing Python software much faster, with no change in your source but it doesn't compile to object code the same way a C compiler would. You could take Python code and attempt to compile it into the equivalent C code using the CPython API

**PEP8**

PEP stands for Python Enhancement Proposal. It is a set of rules that specify how to format Python code for maximum readability.

**Difference between array and list**

<https://www.pythoncentral.io/the-difference-between-a-list-and-an-array/>

### What is \_\_init\_\_?

\_\_init\_\_ is a method or constructor in [Python](https://www.edureka.co/blog/python-programming-language). This method is automatically called to allocate memory when a new object/ instance of a class is created. All classes have the \_\_init\_\_ method.

**Reverse a list/tuple in python**

list[<start>:<stop>:<step>]

So, when you do a[::-1] , it starts from the end, towards the first, taking each element. So it reverses a.

a = '1232'

a[::-1]

'2321

**Python iterators, generators**

<https://www.programiz.com/python-programming/iterator>

<https://anandology.com/python-practice-book/iterators.html>

**Ternary operator**

The Ternary operator will be given as:  
[on\_true] if [expression] else [on\_false]x, y = 25, 50big = x if x < y else y

**\*args and \*\*kwargs**

\*args passes variable number of non-keyworded arguments list and on which operation of the list can be performed.

 \*\*kwargs passes variable number of keyword arguments dictionary to function on which operation of a dictionary can be performed. \*args and \*\*kwargs make the function flexible.

<https://www.programiz.com/python-programming/args-and-kwargs>

**packages and modules in python**

<https://www.learnpython.org/en/Modules_and_Packages>

You could either use the environment variable PYTHONPATH to specify additional directories to look for modules in, like this:

Another method is the sys.path.append function. You may execute it before running an import command:

**\_\_all\_\_ variable in python \_\_init\_\_ file**

<https://stackoverflow.com/questions/44834/can-someone-explain-all-in-python>

**Shallow copy and deep copy in python**

<https://www.geeksforgeeks.org/copy-python-deep-copy-shallow-copy/>

**Frozenset**

<https://www.programiz.com/python-programming/methods/built-in/frozenset>

**first class function**

First Class objects are those objects, which can be handled uniformly.

* can be passed as a parameter
* can be stored as a variable
* function can return another function
* storefunction in list, type and other data structures

**Higher-Order Functions**. Python also supports higher-order functions, meaning that functions can accept other functions as arguments and return functions to the caller.

**Python closures**

<https://www.geeksforgeeks.org/python-closures/>

**Decorators**

<https://www.geeksforgeeks.org/decorators-in-python/>

<https://www.programiz.com/python-programming/decorator>

Garbage collector

List comprehension

<https://medium.com/better-programming/list-comprehension-in-python-8895a785550b>

Treat missing values in python

<https://www.geeksforgeeks.org/working-with-missing-data-in-pandas/>

Pickling and unpickling

Pyton regex

Numpy

Shall copy and deep copy in python

Multithreading

Reverse

def rev\_str(my\_str):

length = len(my\_str)

for i in range(length - 1,-1,-1):

yield my\_str[i]

# For loop to reverse the string

# Output:

# o

# l

# l

# e

# h

for char in rev\_str("hello"):

print(char)