Python: Lists, Tuples, Time Complexity

Python Lists

Key Aspects:

- Sequence of items.
- Enclosed within **square brackets [].**
- Can store values of different data types.
- Variables can store references to lists.
- Their internal structure is similar to a grid, where each element is referred to using an **index**, an integer that starts at 0 for the first item and increases by 1 for each subsequent item in the list.
- They can contain **nested lists** (lists within lists).

Indices:

Starts at 0

Lists are MUTABLE

Access Items:

```
>>> a = [1, 2, 3]
>>> a[0]
1
>>>
```

<listVar>[<index>]

Change Items:

```
>>> a = [1, 2, 3]
>>> a[0] = 5
>>> a
[5, 2, 3]
```

New Value

Python Tuples

Key Aspects:

- Sequence of items.
- Enclosed within **parentheses ().**
- Can store values of different data types.
- Variables can store references to tuples.
- Their internal structure is similar to a grid, where each element is referred to using an **index**, an integer that starts at 0 for the first item and increases by 1 for each subsequent item in the tuple.
- They can contain **nested tuples** (tuples within tuples).

Indices:

Starts at 0

Tuples are IMMUTABLE

Access Items:

```
>>> a = (1, 2, 3)
>>> a[0]
1
>>>
```

<tupleVar>[<index>]

Change Items:

```
>>> a = (1, 2, 3)
>>> a[0] = 5
Traceback (most recent call last):
  File "<pyshell#12>", line 1, in <module>
    a[0] = 5
TypeError: 'tuple' object does not support item assignment
```

Error

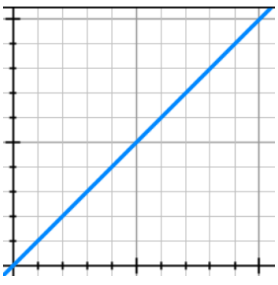
Algorithmic Time Complexity

Key Aspects:

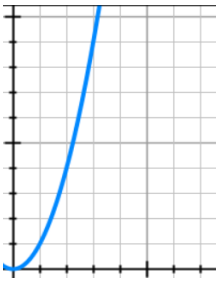
- Indication of the efficiency of algorithms.
- Execution time varies when the size of the input varies.
- Some algorithms handle large inputs much more efficiently than others (faster).
- To denote the execution time, we can analyze the best-case, the average-case, and the worst-case scenario.
- **Big O notation** is used to denote the upper bound of the execution time of an algorithm as the size of the input grows.
- Big O Time Complexities:

Constant: O(1)	**Logarithmic**: O(log(n))	**Linear**: O(n)
Log-linear: O(nlog(n))	**Polynomial**: O(n^c)	**Exponential**: O(c^n)

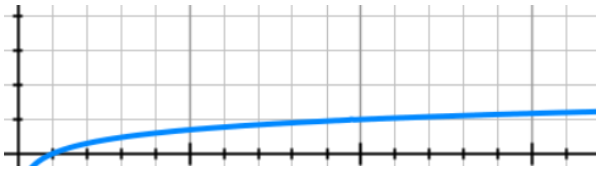
Orders of Growth:



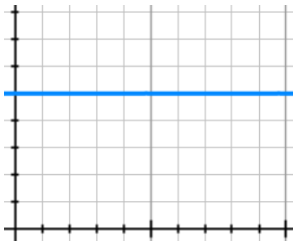
Linear



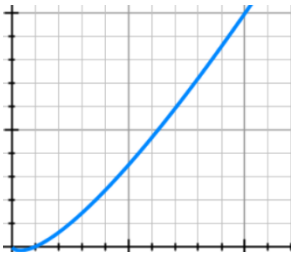
Quadratic



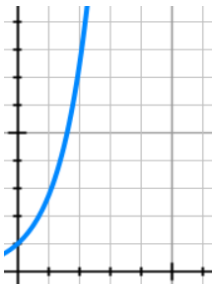
Logarithmic



Constant



nlog(n)



Exponential