1 What is python:

Python is **an interpreter,**

High-level,

General-purpose programming language.

Created by Guido van Rossum and first released in 1991,

2 Features:

- design philosophy that emphasizes code readability
- dynamic type system
- Automatic memory management.
- multiple programming paradigms, including object-oriented, imperative, functional and procedural
- large and comprehensive standard library

3 History:

- Python 1.0 reached in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce
- Python 2.0 was released on 16 October 2000. cycle-detecting garbage collector (in addition to reference counting) for memory management and support for Unicode.
- python 2.7 support is now closed and will be out of service by 2020.
- Python 3.0 was released on 3 December 2008. python 3.7 is latest and most optimized version.

4 New advancement in python 3.7:

- Easier access to debuggers through a new breakpoint() built-in
- Python 3.7 is fast
- time precision to nano sec
- multi-threading with asyncio

5 various python varieties :

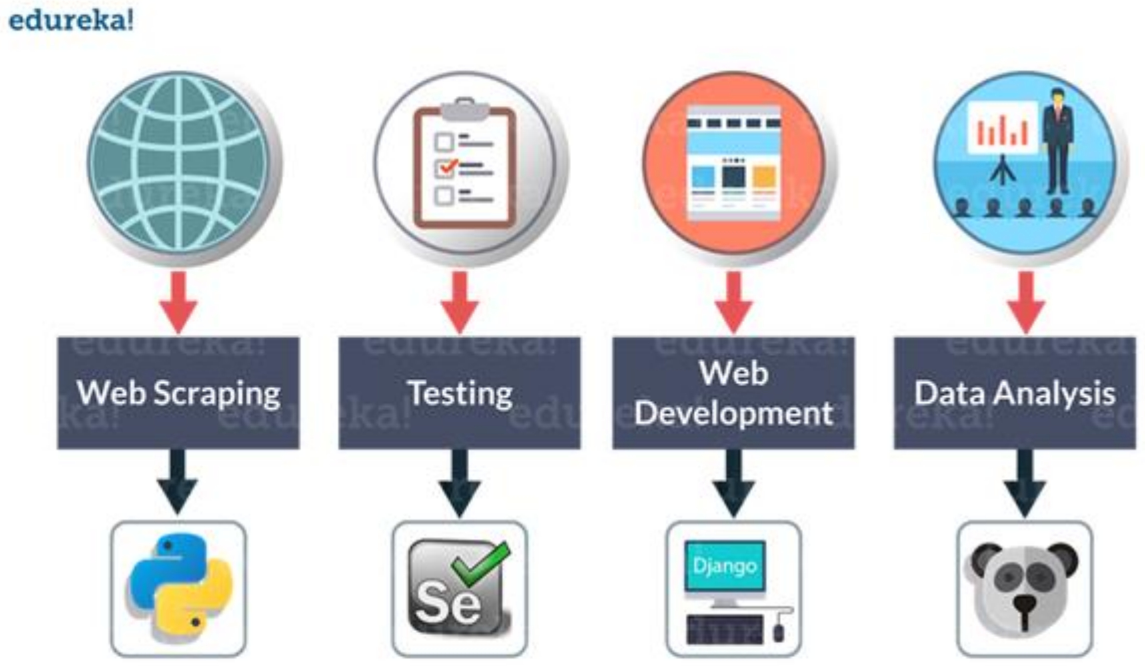
- Cpython: python with C, code converts into C
- Jpython: python with JAVA, code converts into JAVA
- ipython: interactive python, like jupiter

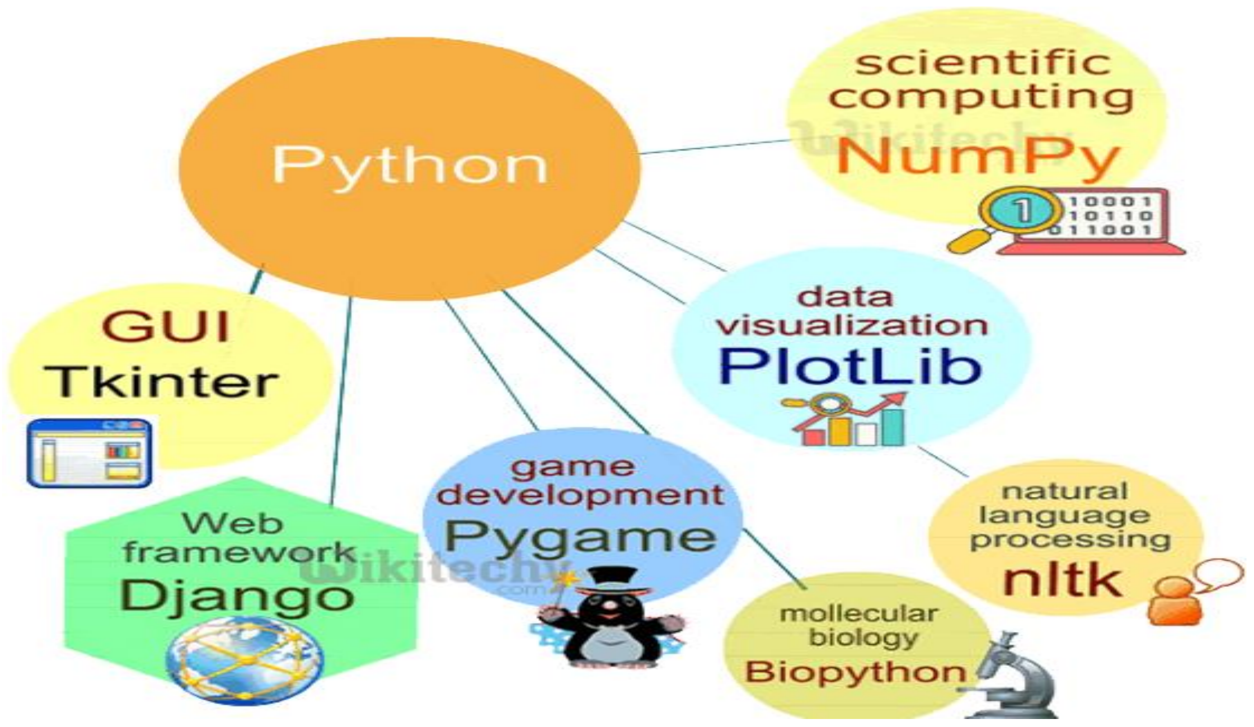
6 Write better code:

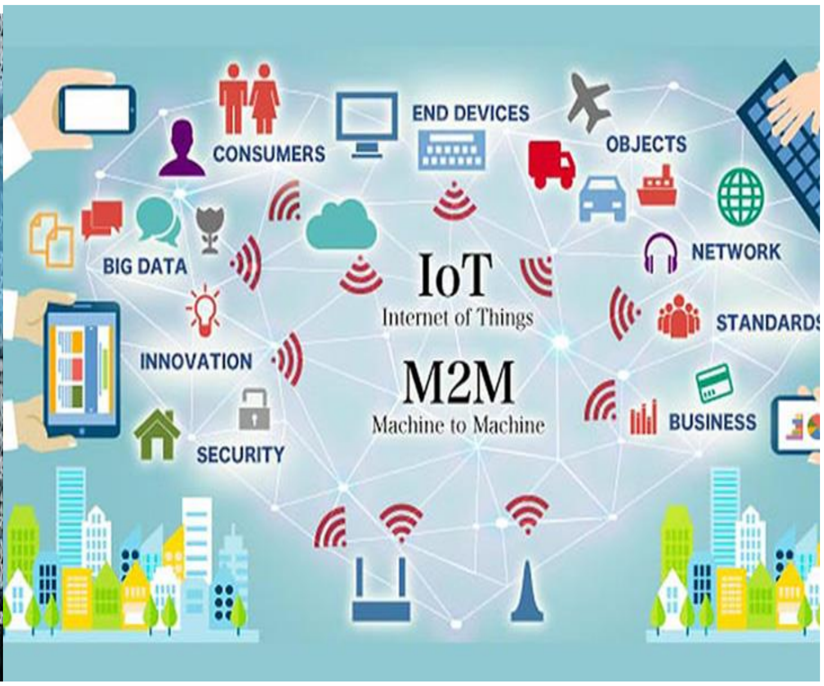
- PEP8 standard
- https://www.python.org/dev/peps/pep-0008/
- Single Leading Underscore: _var
- Single Trailing Underscore: var_
- Double Leading Underscore[dunder]: __var
- Double Leading and Trailing Underscore: __var__
- Single Underscore: _

Pattern	Example	Meaning
Single Leading Underscore	_var	Naming convention indicating a name is meant for internal use. Generally not enforced by the Python interpreter (except in wildcard imports) and meant as a hint to the programmer only.
Single Trailing Underscore	var_	Used by convention to avoid naming conflicts with Python keywords.
Double Leading Underscore	__var	Triggers name mangling when used in a class context. Enforced by the Python interpreter.
Double Leading and Trailing Underscore	__var__	Indicates special methods defined by the Python language. Avoid this naming scheme for your own attributes.
Single Underscore	_	Sometimes used as a name for temporary or insignificant variables (“don’t care”). Also: The result of the last expression in a Python REPL.

The popular YouTube video sharing system is largely written in Python
Google makes extensive use of Python in it's web search system
Dropbox storage service codes both its server and client software primarily in Python
The Raspberry Pi single-board computer promotes Python as its educational language
Google
Dropbox
RaspberryPi
COMPANIES USING PYTHON
BitTorrent
NASA
NETFLIX
BitTorrent peer-to-peer file sharing system began its life as a Python Program
NASA uses Python for specific Programming Task
The NSA uses Python for cryptography and intelligence analysis
Netflix and Yelp have both documented the role of Python in their software infrastructures


edureka!
Web Scraping
Testing
Web Development
Data Analysis
Se
Django


Python
scientific computing
NumPy
data visualization
PlotLib
GUI
Tkinter
Web framework
Django
game development
Pygame
natural language processing
nltk
mollecular biology
Biopython


CONSUMERS
END DEVICES
OBJECTS
BIG DATA
IoT
Internet of Things
NETWORK
STANDARDS
INNOVATION
M2M
Machine to Machine
BUSINESS
SECURITY

7 Various IDLE:

- Pycharm
- Anaconda
- Jupiter
- Default

8_Data structure:

- List
- Tuple
- Dictionary
- Strings
- Sets

8.1 Python Dictionaries, Hashmaps, and Hash Tables

- The dictionary abstract data type is one of the most frequently used and most important data structures in computer science. Because of this importance Python features a robust dictionary implementation as one of its built-in data types (dict).
- Python even provides some useful syntactic sugar for working with dictionaries in your programs. For example, the curly-braces dictionary expression syntax ({}) and dictionary comprehensions allow you to conveniently define new dictionaries:

```
phonebook = {
    'bob': 7387,
    'alice': 3719,
    'jack': 7052,
}

squares = {x: x * x for x in range(10)}
```

8.2 list – Mutable Dynamic Arrays

Lists are a part of the core Python language. Despite their name, Python's lists are implemented as dynamic arrays behind the scenes. This means lists allow elements to be added or removed and they will automatically adjust the backing store that holds these elements by allocating or releasing memory.

Python lists can hold arbitrary elements—“everything” is an object in Python, including functions. Therefore you can mix and match different kinds of data types and store them all in a single list.

This can be a powerful feature, but the downside is that supporting multiple data types at the same time means that data is generally less tightly packed and the whole structure takes up more space as a result.

```
>>> arr = ['one', 'two', 'three']

>>> arr[0]

'one'



# Lists have a nice repr:

>>> arr

['one', 'two', 'three']



# Lists are mutable:

>>> arr[1] = 'hello'

>>> arr

['one', 'hello', 'three']



>>> del arr[1]

>>> arr

['one', 'three']
```

```
# Lists can hold arbitrary data types:

>>> arr.append(23)

>>> arr

['one', 'three', 23]
```

8.3 tuple – Immutable Containers

Tuples are a part of the Python core language. Unlike lists Python’s tuple objects are immutable, this means elements can’t be added or removed dynamically—all elements in a tuple must be defined at creation time.
Just like lists, tuples can hold elements of arbitrary data types. Having this flexibility is powerful, but again it also means that data is less tightly packed than it would be in a typed array.

```
>>> arr = 'one', 'two', 'three'

>>> arr[0]

'one'



# Tuples have a nice repr:

>>> arr

('one', 'two', 'three')



# Tuples are immutable:

>>> arr[1] = 'hello'

TypeError: "'tuple' object does not support item assignment"
```

```
>>> del arr[1]

TypeError: "'tuple' object doesn't support item deletion"



# Tuples can hold arbitrary data types:

# (Adding elements creates a copy of the tuple)

>>> arr + (23,)

('one', 'two', 'three', 23)
```

8.4 array.array

```
import array

>>> arr = array.array('f', (1.0, 1.5, 2.0, 2.5))

>>> arr[1]

1.5
```

8.5 str – Immutable Arrays of Unicode Characters

String objects are space-efficient because they’re tightly packed and specialize in a single data type. If you’re storing Unicode text you should use them.

```
>>> arr = 'abcd'

>>> arr[1]

'b'



>>> arr
```

```
'abcd'



# Strings are immutable:

>>> arr[1] = 'e'

TypeError: "'str' object does not support item assignment"

# Strings are recursive data structures:

>>> type('abc')

"<class 'str'>"

>>> type('abc'[0])

"<class 'str'>"
```

8.6 bytes – Immutable Arrays of Single Bytes

Bytes objects are immutable sequences of single bytes (integers in the range of 0 <= x <= 255). Conceptually they're similar to str objects and you can also think of them as immutable arrays of bytes.

```
>>> arr = bytes((0, 1, 2, 3))

>>> arr[1]

1



# Bytes literals have their own syntax:

>>> arr

b'\x00\x01\x02\x03'
```

```
>>> arr = b'\x00\x01\x02\x03'



# Only valid "bytes" are allowed:

>>> bytes((0, 300))

ValueError: "bytes must be in range(0, 256)"
```

9 Typecasting:

Fxn- Type():

- int to float
- int to str
- str to int
- list to tuple
- tuple to list