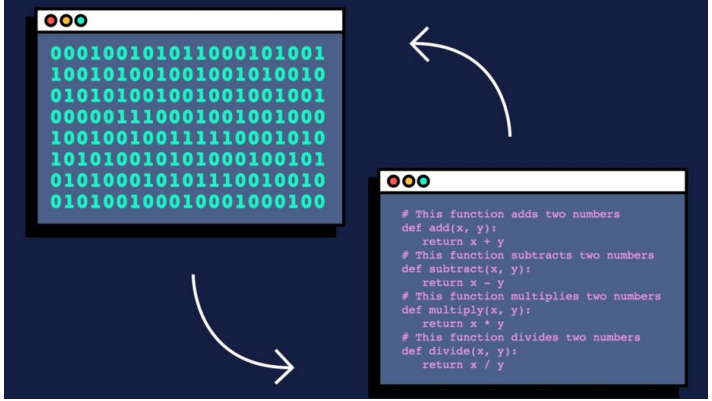
Pyoctonion Python Package

Kalpa Thudewaththage & Charith Atapattu

What is a programming language?



- Programming is giving a set of instructions to a computer to execute.
- Computers “understand” in binary — strings of 1s and 0s.
- Programming languages allow us to translate human language in to the 1s and 0s what computer can understand.
- Programming languages fall into two different classifications — low-level and high-level.

Why Python?

Low-Level vs. High-Level Programming Languages

Assembly Code	Python

fib:
movl $1, %eax
xorl %ebx, %ebx
.fib_loop:
cmpl $1, %edi
jbe .fib_done
movl %eax, %ecx
addl %ebx, %eax
movl %ecx, %ebx
subl $1, %edi
jmp .fib_loop
.fib_done:
ret
	# Function for nth Fibonacci number
	def Fibonacci(n):
	if n < 0:
	print("Incorrect input")
	elif n == 0:
	return 0
	elif n == 1 or n == 2:
	return 1
	else:
	return Fibonacci(n-1) + Fibonacci(n-2)
	
	print(Fibonacci(9))

Have Issues with High-level languages?

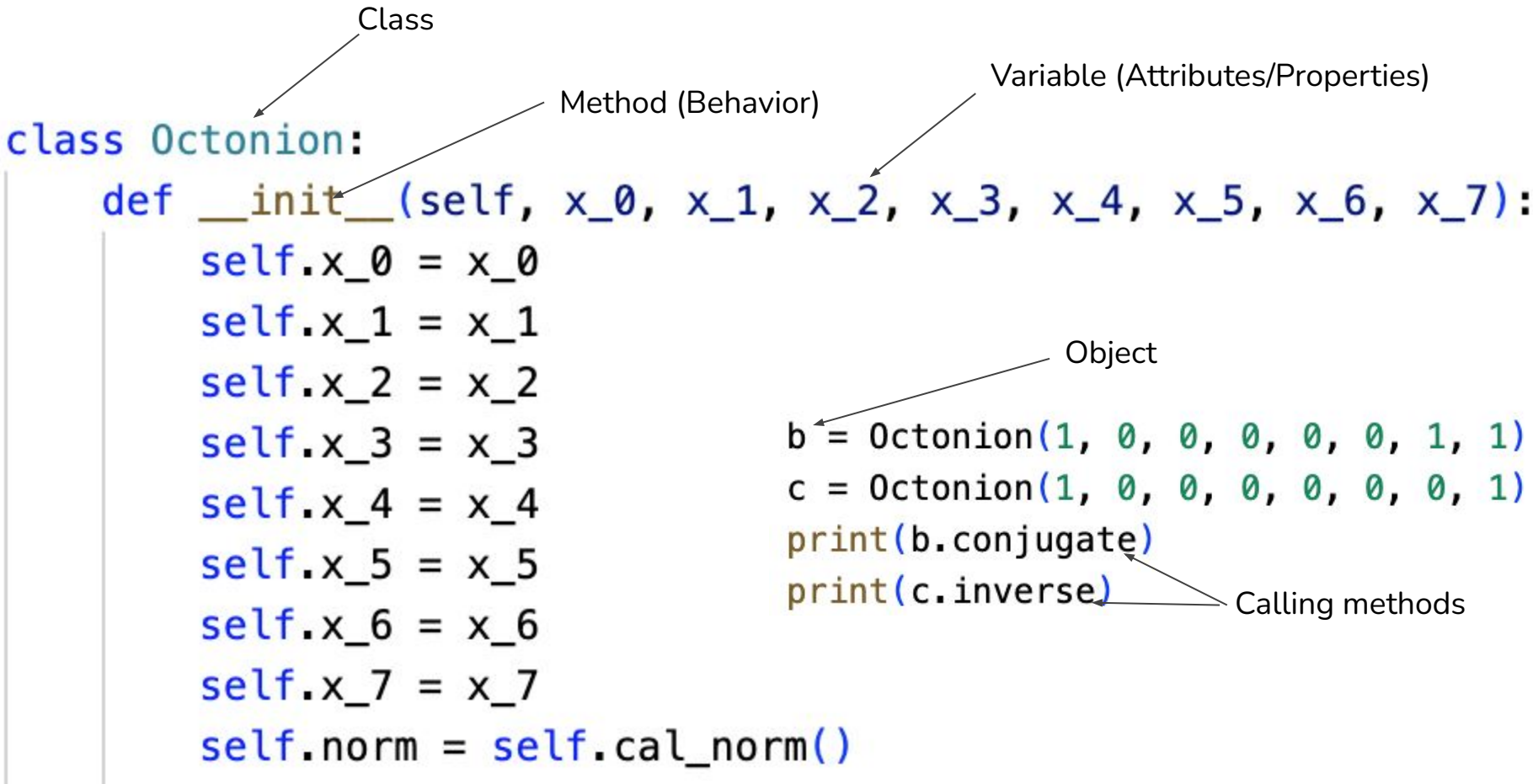
What do you think? What is good between High-level vs Low-level languages?

How we overcome these issues?

- Writing clear abstract code
- Choosing proper data structure
- Following good coding practice like OOP

What is OOP?

Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which can contain data and code: data in the form of fields (often known as attributes or properties), and code, in the form of procedures (often known as methods).


Class
Method (Behavior)
Variable (Attributes/Properties)
class Octonion:
def __init__(self, x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7):
self.x_0 = x_0
self.x_1 = x_1
self.x_2 = x_2
self.x_3 = x_3
self.x_4 = x_4
self.x_5 = x_5
self.x_6 = x_6
self.x_7 = x_7
self.norm = self.cal_norm()
Object
b = Octonion(1, 0, 0, 0, 0, 0, 1, 1)
c = Octonion(1, 0, 0, 0, 0, 0, 0, 1)
print(b.conjugate)
print(c.inverse)
Calling methods

What are the benefits?

1. Easy to maintain
2. High performance
3. Low coupling and high cohesion code
4. Reduce complexity of your code
5. Less memory allocation and management

Octonion vs Quaternion

Calculate inverse using Quaternion package

In [11]:

```
%%time
print(Octonion(1,2,3,4,5,6,7,8))
print(Octonion(1,3,5,7,9,2,4,6))
print(Octonion(8,7,6,5,4,3,2,1))
print(Oct_inverse(1,2,3,4,5,6,7,8))
print(Oct_inverse(1,3,5,7,9,2,4,6))
print(Oct_inverse(8,7,6,5,4,3,2,1))
```

```
[1. 2. 3. 4. 5. 6. 7. 8.]
[1. 3. 5. 7. 9. 2. 4. 6.]
[8. 7. 6. 5. 4. 3. 2. 1.]
[ 0.00490196 -0.00980392 -0.01470588 -0.01960784 -0.0245098  -0.02941176
 -0.03431373 -0.03921569]
[ 0.00452489 -0.01357466 -0.02262443 -0.03167421 -0.04072398 -0.00904977
 -0.01809955 -0.02714932]
[ 0.03921569 -0.03431373 -0.02941176 -0.0245098  -0.01960784 -0.01470588
 -0.00980392 -0.00490196]
CPU times: user 6.87 ms, sys: 6.24 ms, total: 13.1 ms
Wall time: 14.3 ms
```

Octonion vs Quaternion

Calculate inverse using Octonion package

```
%%time
from pyoctonion import Octonion
a = Octonion(1,2,3,4,5,6,7,8)
b = Octonion(1,3,5,7,9,2,4,6)
c = Octonion(8,7,6,5,4,3,2,1)

#Print Octonion values
print(a)
print(a)
print(a)
print(a.inverse)
print(b.inverse)
print(c.inverse)

```

```
+1.0000 +2.0000i +3.0000j +4.0000k +5.0000l +6.0000il +7.0000jl +8.0000kl
+1.0000 +2.0000i +3.0000j +4.0000k +5.0000l +6.0000il +7.0000jl +8.0000kl
+1.0000 +2.0000i +3.0000j +4.0000k +5.0000l +6.0000il +7.0000jl +8.0000kl
+0.0049 -0.0098i -0.0147j -0.0196k -0.0245l -0.0294il -0.0343jl -0.0392kl
+0.0045 -0.0136i -0.0226j -0.0317k -0.0407l -0.0090il -0.0181jl -0.0271kl
+0.0392 -0.0343i -0.0294j -0.0245k -0.0196l -0.0147il -0.0098jl -0.0049kl
CPU times: user 543 µs, sys: 72 µs, total: 615 µs
Wall time: 720 µs
```

Time and Space Complexity

Time complexity is a function describing the amount of time an algorithm takes in terms of the amount of input to the algorithm.

- How quickly you get output

Space complexity is a function describing the amount of memory (space) an algorithm takes in terms of the amount of input to the algorithm.

- How you can use lesser resources in the system

Example:

Imagine a classroom of 100 students in which you gave your pen to one person. You have to find that pen without knowing to whom you gave it.

Here are some ways to find the pen and what the O order is.

- $O(n^2)$: You go and ask the first person in the class if he has the pen. Also, you ask this person about the other 99 people in the classroom if they have that pen and so on,
  This is what we call $O(n^2)$.
- O(n): Going and asking each student individually is O(n).
- O(log n): Now I divide the class into two groups, then ask: “Is it on the left side, or the right side of the classroom?” Then I take that group and divide it into two and ask again, and so on. Repeat the process till you are left with one student who has your pen. This is what you mean by O(log n).

Code Review

pyOctonion Code in Github Repository,

- https://github.com/charithsiu/pyoctonion

Pypi repository code where you upload package to install using “pip” in python,

- https://pypi.org/project/pyoctonion/

Library documentation,

- https://github.com/charithsiu/pyoctonion/blob/master/README.md

Practice samples & presentations decks

- https://github.com/charithsiu/pyoctonion/tree/master/demo
- https://github.com/charithsiu/pyoctonion/tree/master/samples/
- https://github.com/charithsiu/pyoctonion/tree/master/samples/Octonionic-left-quadratic equation-(python-code)-For-Science-Seminar.ipynb
- https://github.com/charithsiu/pyoctonion/tree/master/samples/Real-Eigenvalues-for-3-by 3-octonionic-Hermitian-matrices.ipynb