

85ª EDIÇÃO

**SEQ UFRJ**

20 a 24 de agosto



# Introdução à programação para ciência e engenharia em *Python*

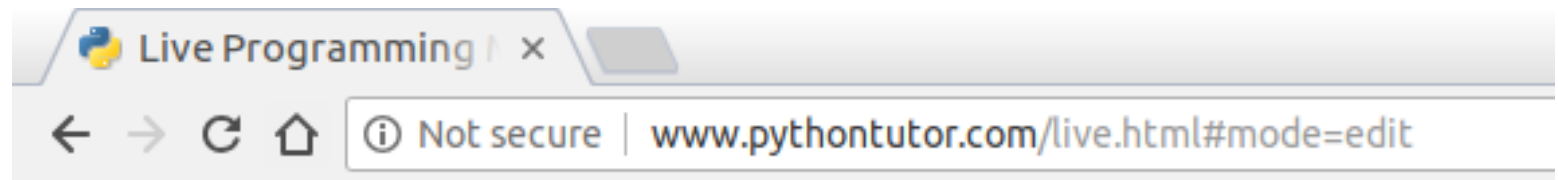
*Iuri Soter Viana Segtovich*

Parte 2: Lógica e Sintaxe

Classes (class, `__init__(self)`)

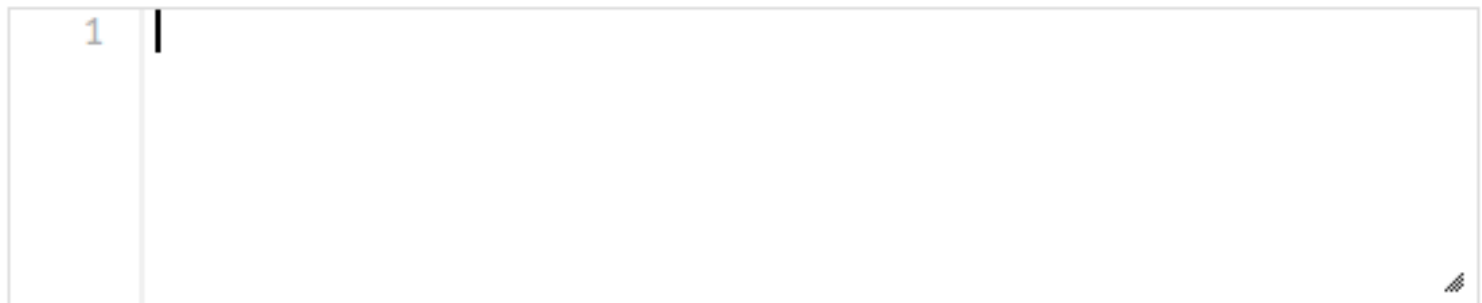
# python tutor

[www.pythontutor.com/  
live.html#mode=edit](http://www.pythontutor.com/live.html#mode=edit)



Write code in Python 3.6

(drag lower right corner to resize code editor)



→ line that has just executed

→ next line to execute

# OOP

```
1 class Employee:
2     'Common base class for all employees'
3     empCount = 0
4
5     def __init__(self, name, salary):
6         self.name = name
7         self.salary = salary
8         Employee.empCount += 1
9
10    def displayCount(self):
11        print("Total Employee %d" % Employee.empCount)
12
13    def displayEmployee(self):
14        print("Name : ", self.name, " , Salary: ", self.salary)
15
16    "This would create first object of Employee class"
17    emp1 = Employee("Zara", 2000)
18    "This would create second object of Employee class"
19    emp2 = Employee("Manni", 5000)
20    emp1.displayEmployee()
21    emp2.displayEmployee()
→ 22    print("Total Employee %d" % Employee.empCount)
```

```
Name : Zara , Salary: 2000
Name : Manni , Salary: 5000
Total Employee 2
```

Frames

Objects

Global frame

Employee  
emp1  
emp2

Employee class  
hide attributes

<code>__init__</code>	function <code>__init__(self, name, salary)</code>
<code>displayCount</code>	function <code>displayCount(self)</code>
<code>displayEmployee</code>	function <code>displayEmployee(self)</code>
<code>empCount</code>	2

Employee instance

<code>name</code>	"Zara"
<code>salary</code>	2000

Employee instance

<code>name</code>	"Manni"
<code>salary</code>	5000

# Built-In Class Attributes

```
1 class Employee:
2     'Common base class for all employees'
3     empCount = 0
4
5     def __init__(self, name, salary):
6         self.name = name
7         self.salary = salary
8         Employee.empCount += 1
9
10    def displayCount(self):
11        print("Total Employee %d" % Employee.empCount)
12
13    def displayEmployee(self):
14        print("Name : ", self.name, ", Salary: ", self.salary)
15
16    print("Employee.__doc__:", Employee.__doc__)
17    print("Employee.__name__:", Employee.__name__)
18    print("Employee.__module__:", Employee.__module__)
19    print("Employee.__bases__:", Employee.__bases__)
→ 20    print("Employee.__dict__:", Employee.__dict__)
```

```

Employee.__doc__: Common base class for all emplo
Employee.__name__: Employee
Employee.__module__: __main__
Employee.__bases__: (<class 'object'>,)
Employee.__dict__: {'__module__': '__main__', '__

```

Frames

Objects

Global frame

Employee

Employee class  
[hide attributes](#)

__init__	function __init__(self, name, salary)
displayCount	function displayCount(self)
displayEmployee	function displayEmployee(self)
empCount	0

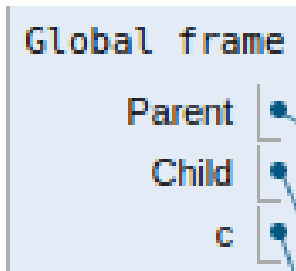
# inheritance

```
1  #!/usr/bin/python
2
3  class Parent:          # define parent class
4      parentAttr = 100
5      def __init__(self):
6          print("Calling parent constructor")
7
8      def parentMethod(self):
9          print('Calling parent method')
10
11     def setAttr(self, attr):
12         Parent.parentAttr = attr
13
14     def getAttr(self):
15         print("Parent attribute :", Parent.parentAttr)
16
17 class Child(Parent):    # define child class
18     def __init__(self):
19         print("Calling child constructor")
20
21     def childMethod(self):
22         print('Calling child method')
23
24 c = Child()             # instance of child
25 c.childMethod()         # child calls its method
26 c.parentMethod()        # calls parent's method
27 c.setAttr(200)          # again call parent's method
→ 28 c.getAttr()           # again call parent's method
```

Calling child constructor  
Calling child method  
Calling parent method  
Parent attribute : 200

Frames

Objects



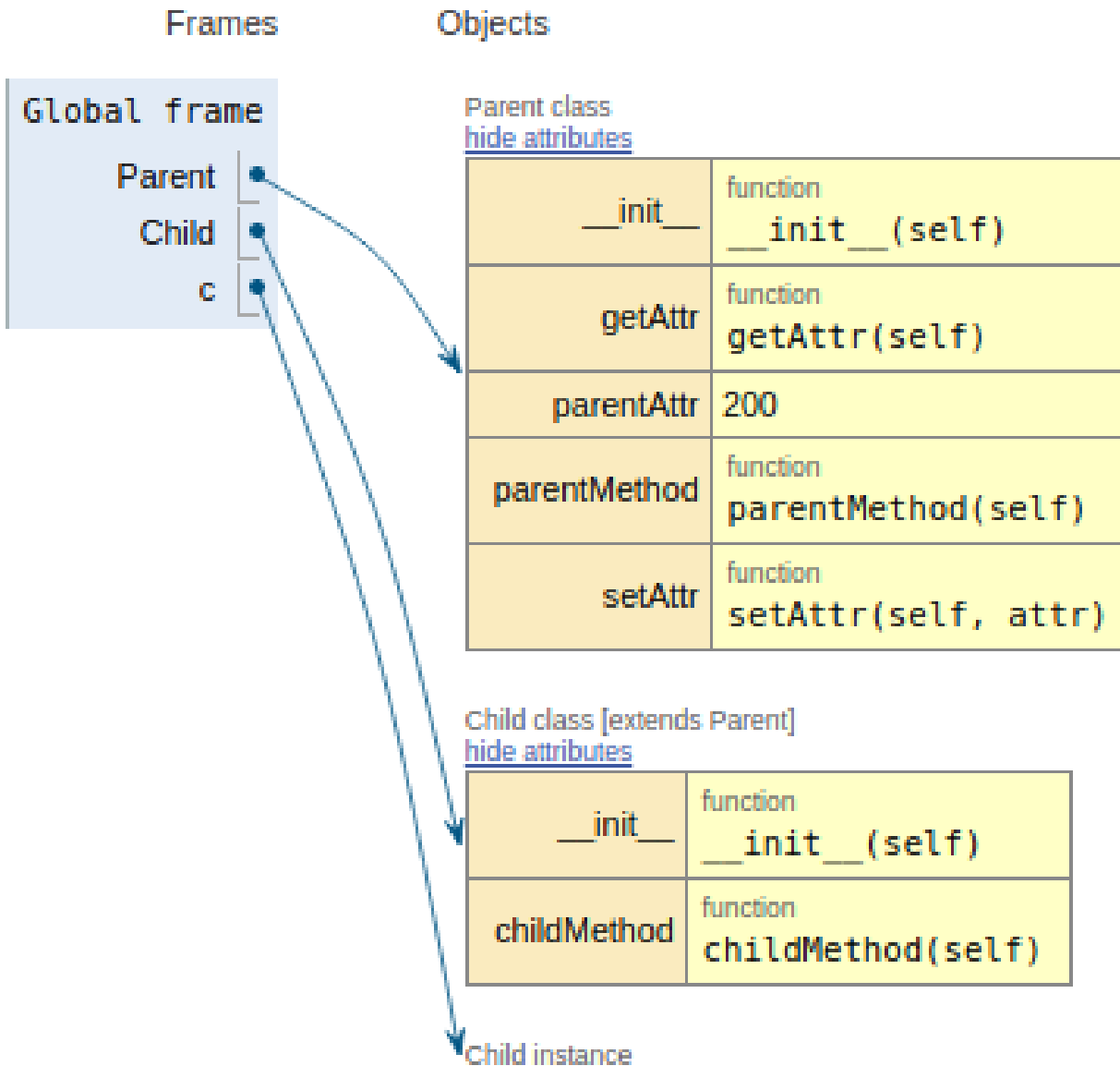
Parent class  
hide attributes

__init__	function __init__(self)
getAttr	function getAttr(self)
parentAttr	200
parentMethod	function parentMethod(self)
setAttr	function setAttr(self, attr)

Child class [extends Parent]  
hide attributes

__init__	function __init__(self)
childMethod	function childMethod(self)

Child instance





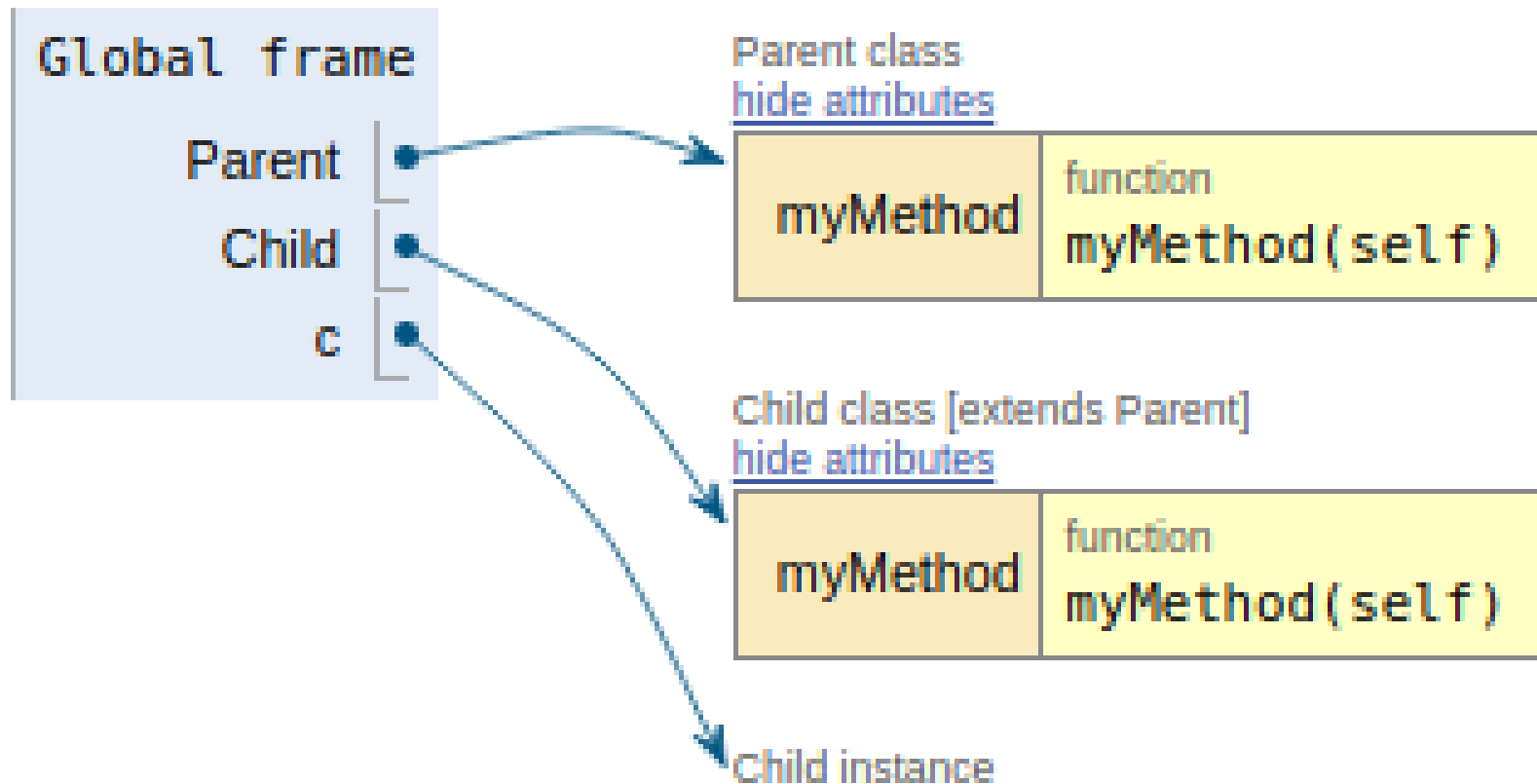
# Method override

```
1  #!/usr/bin/python
2
3  class Parent:          # define parent class
4      def myMethod(self):
5          print('Calling parent method')
6
7  class Child(Parent):   # define child class
8      def myMethod(self):
9          print('Calling child method')
10
11  c = Child()            # instance of child
→ 12  c.myMethod()         # child calls overridden method
```

## Calling child method

Frames

Objects



# Operator overload

```
1  #!/usr/bin/python
2
3  class Vector:
4      def __init__(self, a, b):
5          self.a = a
6          self.b = b
7
8      def __str__(self):
9          return 'Vector (%d, %d)' % (self.a, self.b)
10
11     def __add__(self, other):
12         return Vector(self.a + other.a, self.b + other.b)
13
14     v1 = Vector(2,10)
15     v2 = Vector(5,-2)
16     print(v1 + v2)
```

Vector (7, 8)

Frames

Objects

Global frame

Vector

v1

v2

Vector class  
[hide attributes](#)

<code>__add__</code>	function <code>__add__(self, other)</code>
<code>__init__</code>	function <code>__init__(self, a, b)</code>
<code>__str__</code>	function <code>__str__(self)</code>

Vector instance

Vector (2, 10)

a	2
b	10

Vector instance

Vector (5, -2)

a	5
b	-2

# Referências principais

[https://www.tutorialspoint.com/  
python3/  
python\\_basic\\_syntax.htm](https://www.tutorialspoint.com/python3/python_basic_syntax.htm)

[https://stackoverflow.com/  
search](https://stackoverflow.com/search)

# perguntas

