Introdução à programação para ciência e engenharia em *Python*

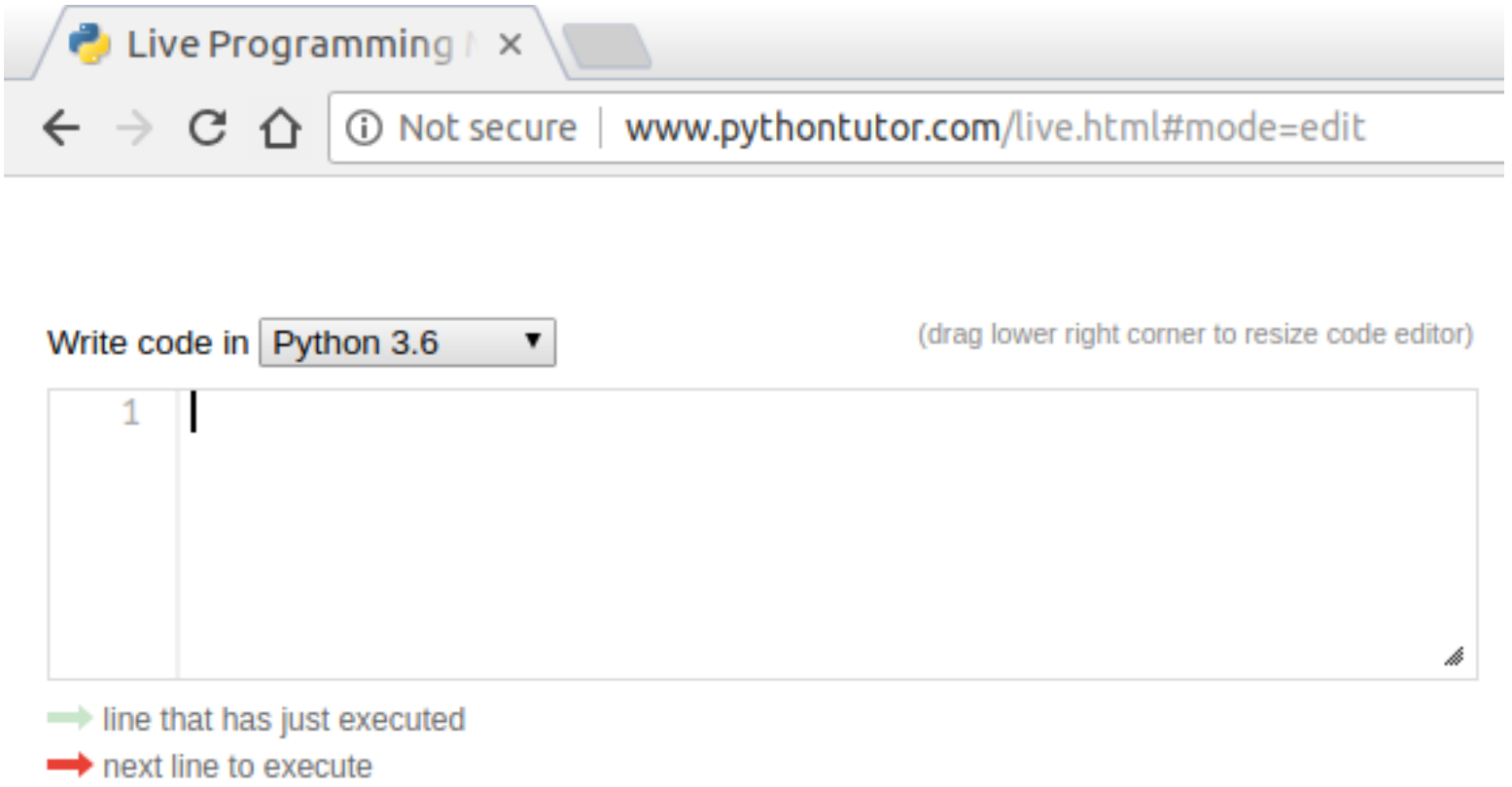
Iuri Soter Viana Segtovich

Parte 2: Lógica e Sintaxe

python tutor

www.pythontutor.com/

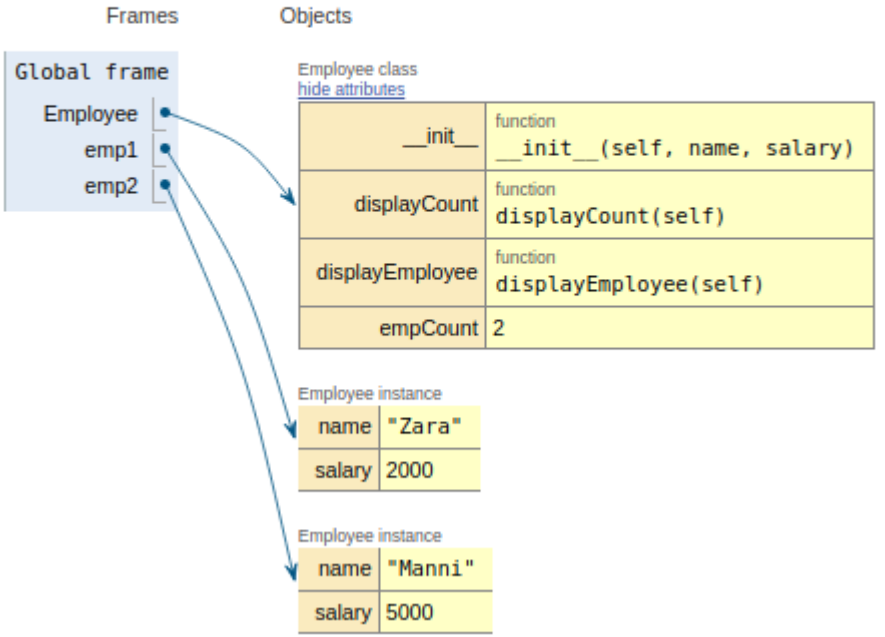
live.html#mode=edit



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)
```

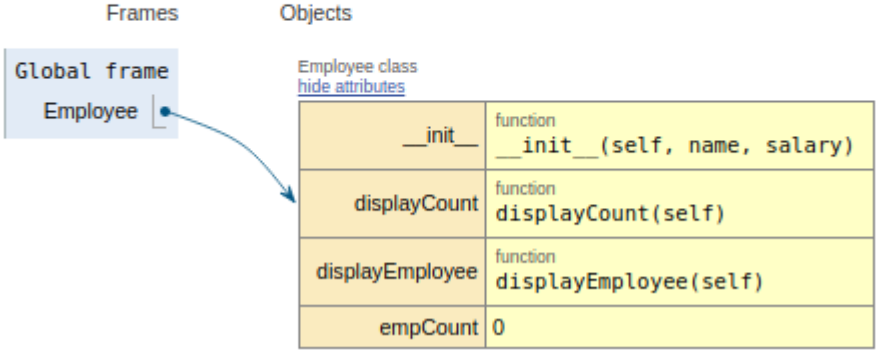
```
Name :  Zara , Salary:  2000
Name :  Manni , Salary:  5000
Total Employee 2
```



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__)
```

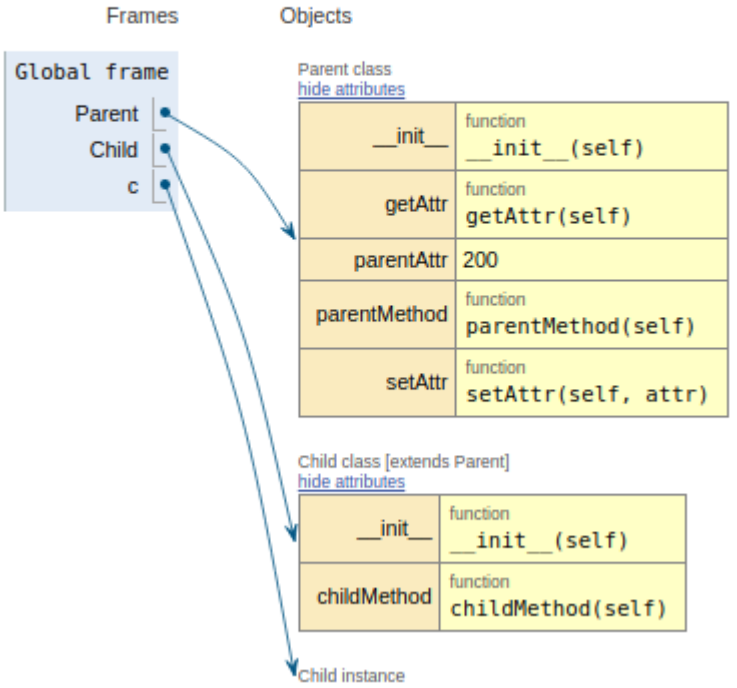
```
Employee.__doc__: Common base class for all emplo
Employee.__name__: Employee
Employee.__module__: __main__
Employee.__bases__: (<class 'object'>,)
Employee.__dict__: {'__module__': '__main__', '__
```



inheritance

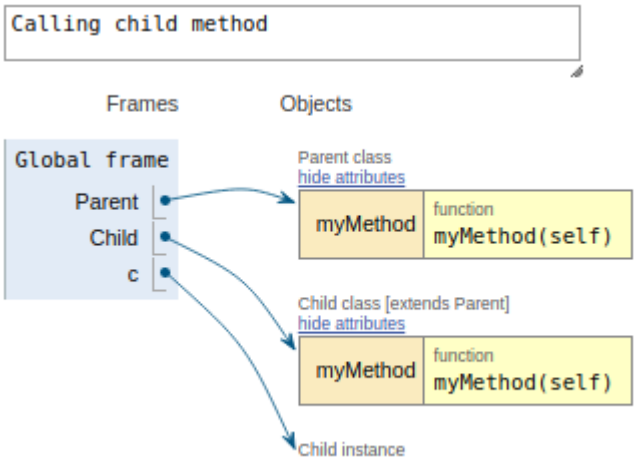
```python
#!/usr/bin/python

class Parent:        # define parent class
   parentAttr = 100
   def __init__(self):
      print("Calling parent constructor")

   def parentMethod(self):
      print('Calling parent method')

   def setAttr(self, attr):
      Parent.parentAttr = attr

   def getAttr(self):
      print("Parent attribute :", Parent.parentAttr)

class Child(Parent): # define child class
   def __init__(self):
      print("Calling child constructor")

   def childMethod(self):
      print('Calling child method')

c = Child()          # instance of child
c.childMethod()      # child calls its method
c.parentMethod()     # calls parent's method
c.setAttr(200)       # again call parent's method
c.getAttr()          # again call parent's method
```

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



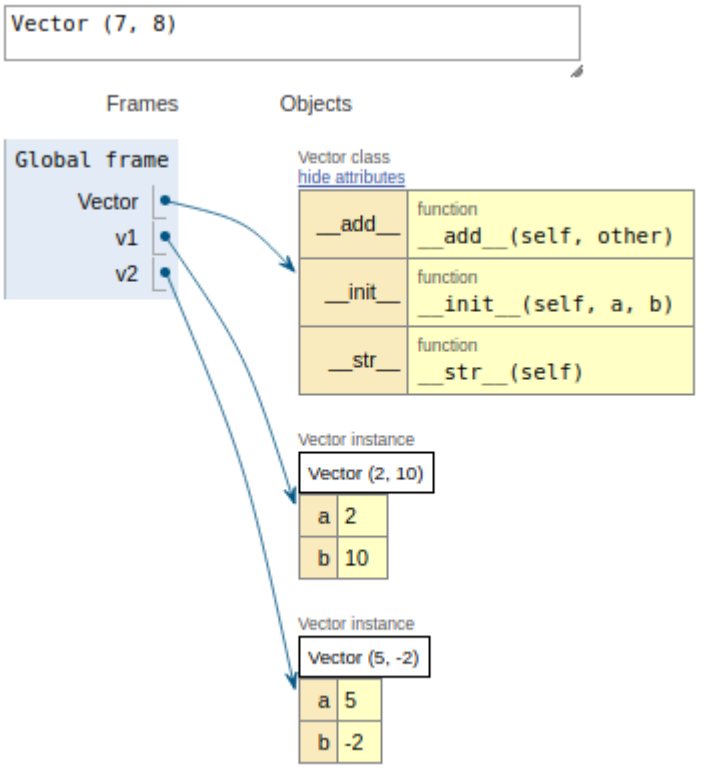
Method override

```
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
```



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)
```



Referências principais

https://www.tutorialspoint.com/

python3/

python_basic_syntax.htm

https://stackoverflow.com/

search

perguntas