Python, základní kameny až skály II

Tomáš Svoboda

B4B33RPH, 2016-10-25

funkce pravé a modifikátory

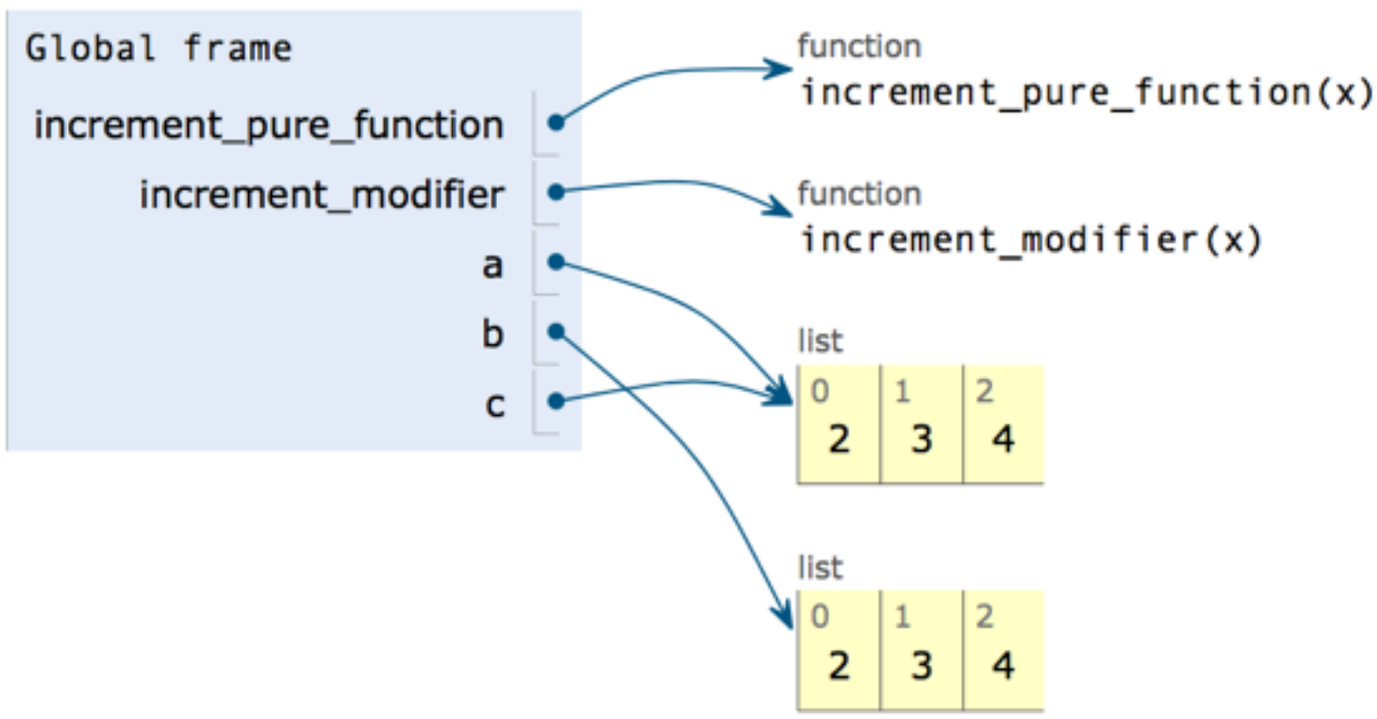
Write code in Python 3.3

(drag lower right corner to resize code editor)

```
1  def increment_pure_function(x):
2      v = []
3      for item in x:
4          v.append(item+1)
5      return(v)
6
7  def increment_modifier(x):
8      for i in range(len(x)):
9          x[i] = x[i]+1
10     return(x)
11
12 a = [1,2,3]
13 b = increment_pure_function(a)
14 print(a,',',b)
15 c = increment_modifier(a)
16 print(a,',',b,',',c)
```

Print output (drag lower right corner to resize)

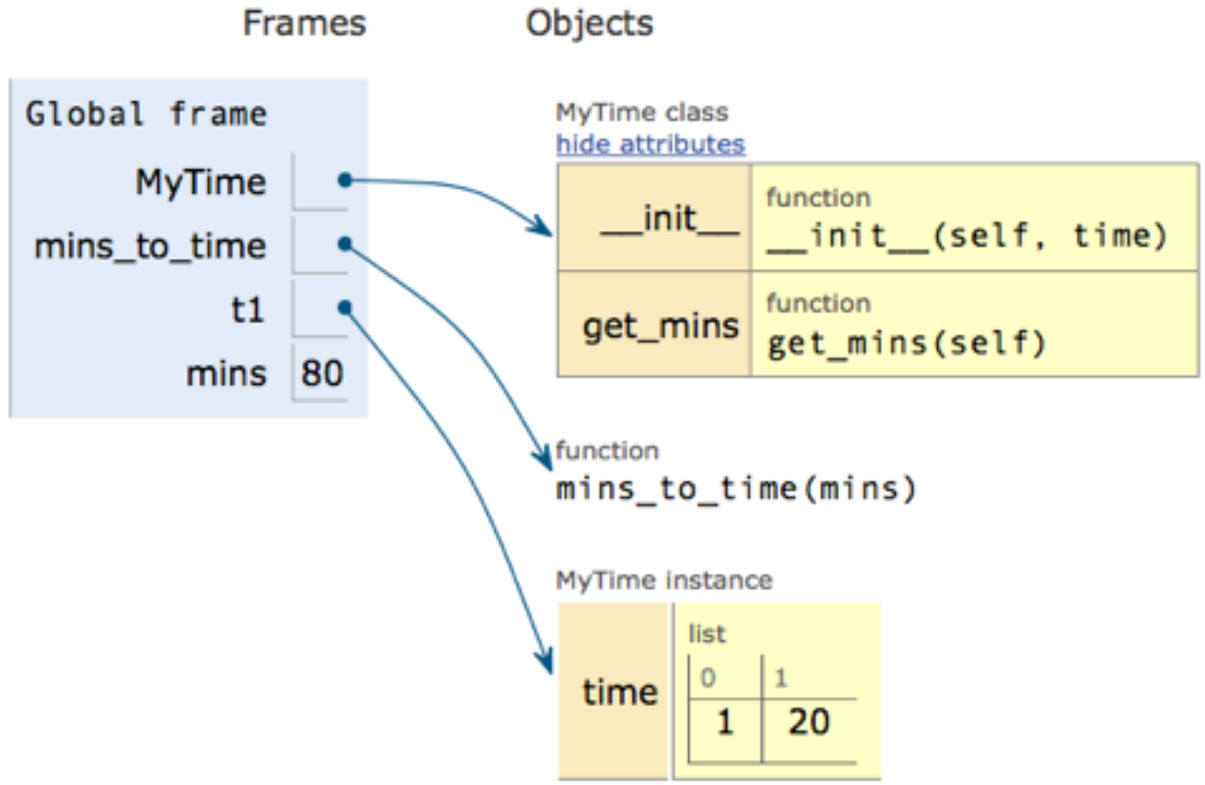
```
[1, 2, 3] , [2, 3, 4]
[2, 3, 4] , [2, 3, 4] , [2, 3, 4]
```



objekty, třídy a tak

Write code in Python 3.3 (drag lower right corner to resize code editor)

```
1  class MyTime:
2      def __init__(self,time=None):
3          self.time = time
4
5      def get_mins(self):
6          return(self.time[0]*60+self.time[1])
7
8  def mins_to_time(mins):
9      return([mins//60,mins%60])
10
11 t1 = MyTime([1,20])
12 mins = t1.get_mins()
13 time_vec = mins_to_time(mins)
```



visualisation

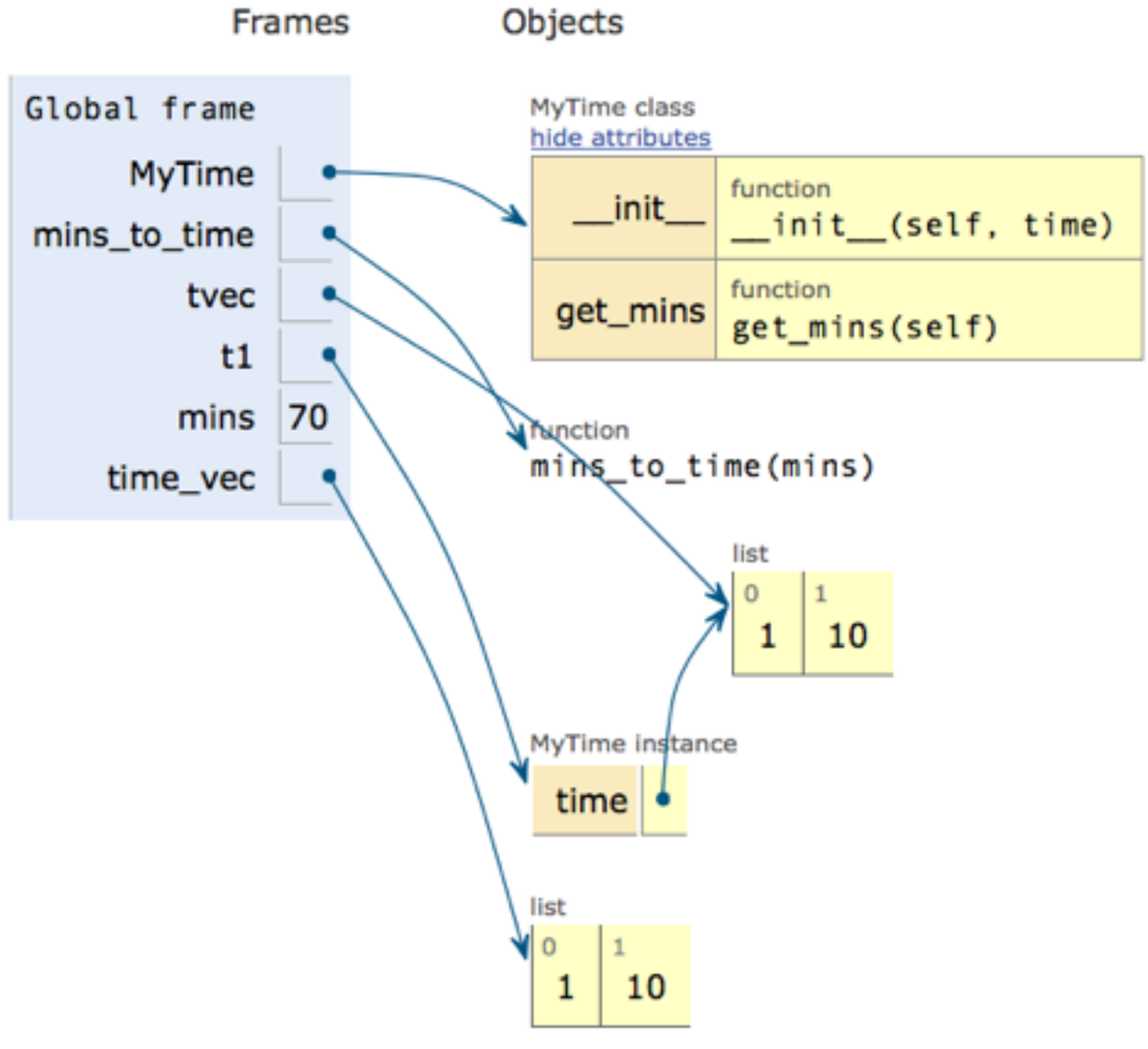
ale pozor …

Write code in Python 3.3 (drag lower right corner to resize code editor)

```
1  class MyTime:
2      def __init__(self,time=None):
3          self.time = time
4
5      def get_mins(self):
6          return(self.time[0]*60+self.time[1])
7
8  def mins_to_time(mins):
9      return([mins//60,mins%60])
10
11 tvec = [1,20]
12 t1 = MyTime(tvec)
13 tvec[1] = 10
14 mins = t1.get_mins()
```

line that has just executed
next line to execute

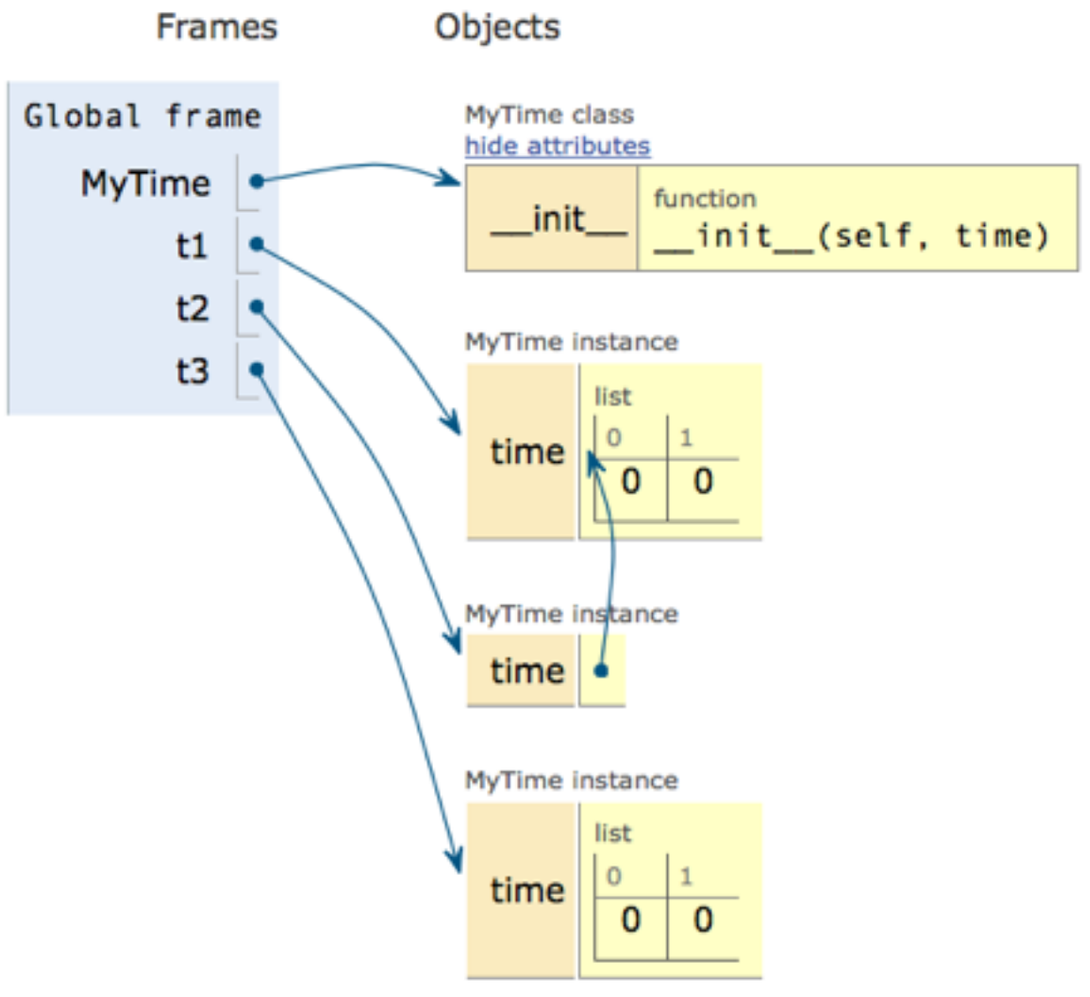
<< First | < Back | Done running (16 steps) | Forward > | Last >>



a ještě větší pozor na implicitní parametry

Write code in Python 3.3 (drag lower right corner to resize code editor)

```
1  class MyTime:
2      def __init__(self,time=[0,0]):
3          self.time = time
4
5  t1 = MyTime()
6  t2 = MyTime()
7
8  t3 = MyTime([0,0])
9
```



implicitní parametry detailněji

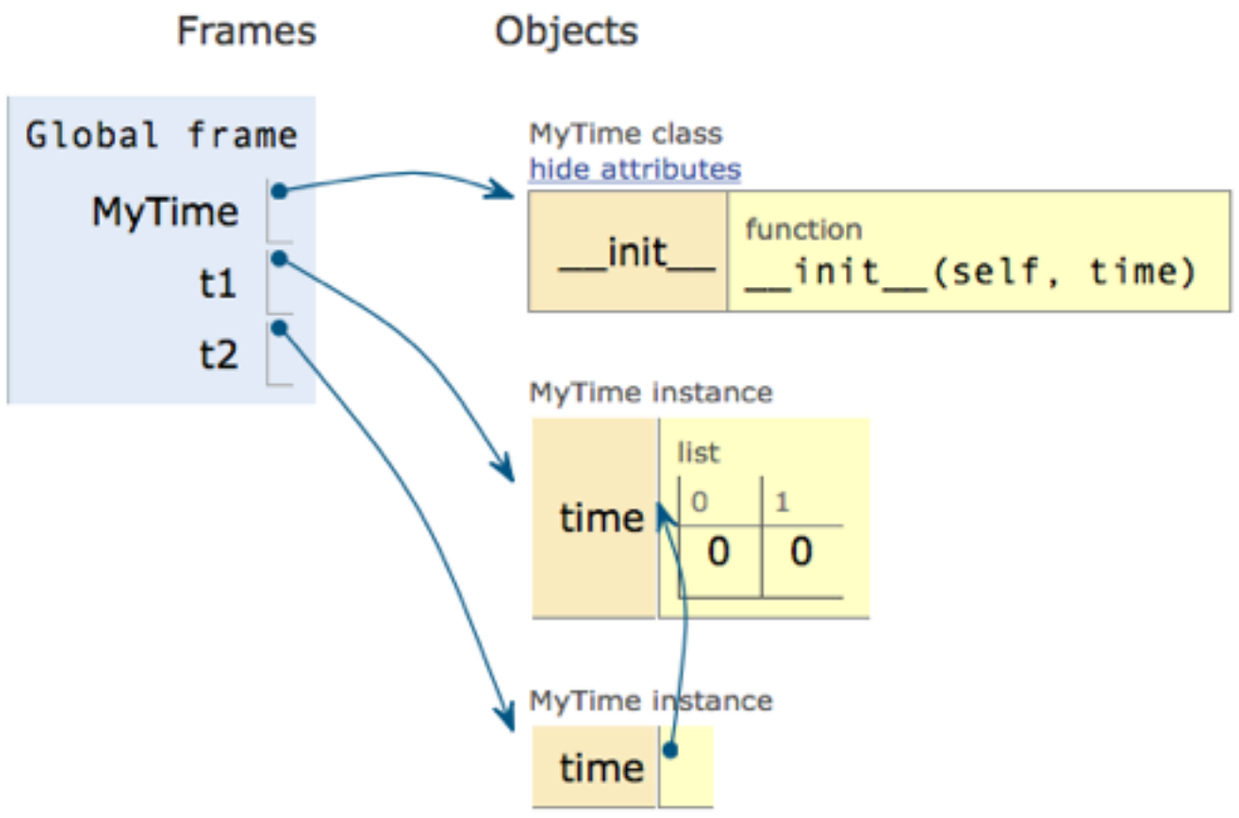
Write code in Python 3.3

(drag lower right corner to resize code editor)

```
1  class MyTime:
2      def __init__(self,time=[0,0]):
3          self.time = time
4
5  t1 = MyTime()
6  t2 = MyTime()
7  print(id(t1.time))
8  print(id(t2.time))
9  print(t1.time is t2.time)
10
```

Print output (drag lower right corner to resize)

```
140145092713432
140145092713432
True
```



dict - tuples as keys

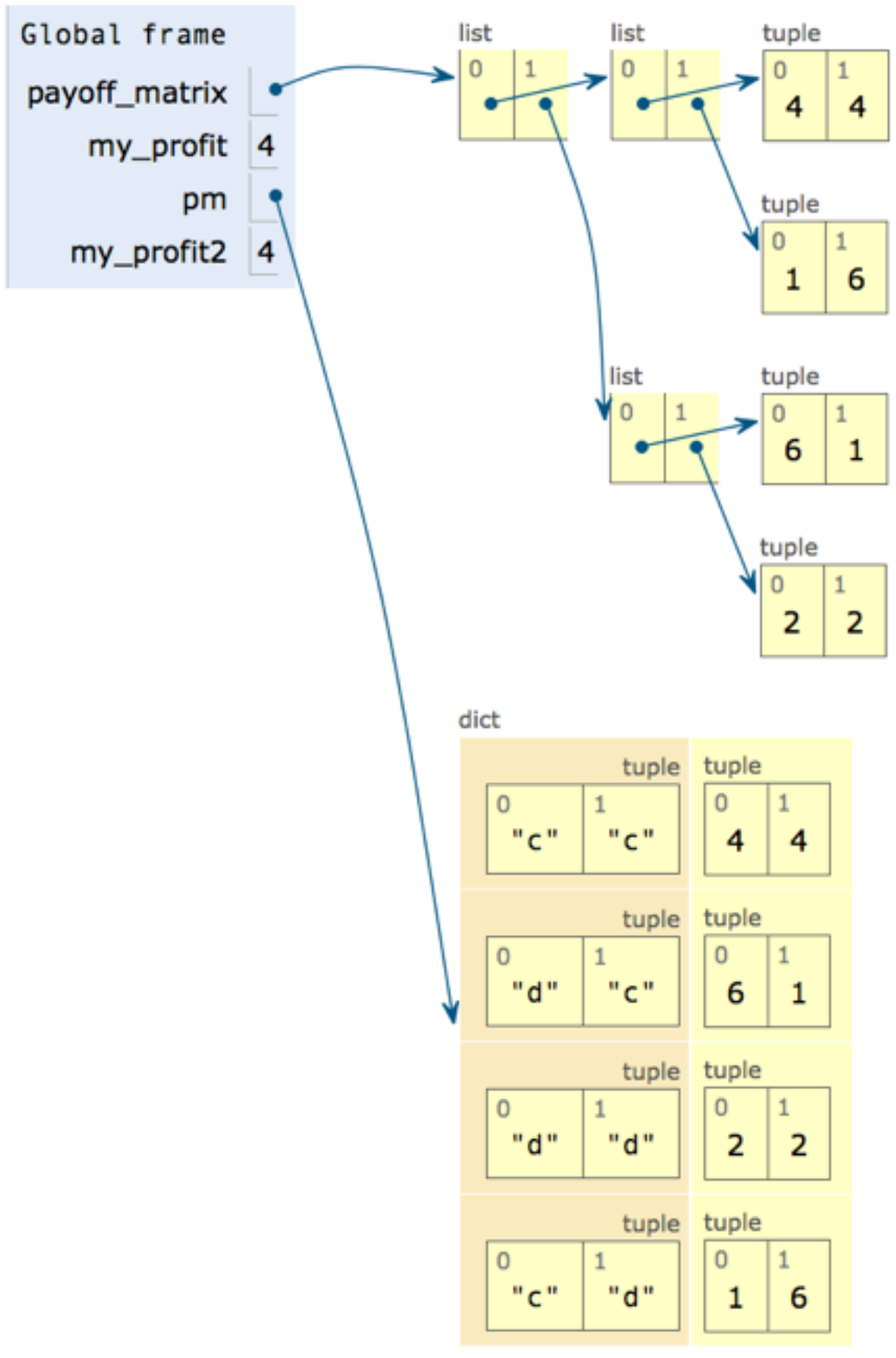
Write code in Python 3.3 (drag lower right corner to resize code editor)

```
1  payoff_matrix = [ [(4,4),(1,6)] , [(6,1),(2,2)] ]
2  # cooperate, cooperate, mine
3  my_profit = payoff_matrix[0][0][0]
4
5  pm = {}
6  pm['c','c'] = (4,4)
7  pm['d','d'] = (2,2)
8  pm['c','d'] = (1,6)
9  pm['d','c'] = (6,1)
10 my_profit2 = pm['c','c'][0]
11
```

line that has just executed

next line to execute

<< First | < Back | Done running (8 steps) | Forward > | Last >>



dictionary loops …

```
1  pm = {}
2  pm['c','c'] = (4,4)
3  pm['d','d'] = (2,2)
4  pm['c','d'] = (1,6)
5  pm['d','c'] = (6,1)
6
7  for key in pm:
8      print(key, pm[key])
9
10 for key,value in pm.items():
11     print(key, value)
```

skládání objektů, dědění

- ukážeme si na příkladu hráče piškvorek (tic-tac-toe)

- live-coding-session

skládání

```
14  class MyPlayer:
15      def __init__(self, mine_sym, opponent_sym, empty_sym):
16          """
17              :param mine_sym: string my symbol
18              :param opponent_sym: string opponent symbol
19              :param empty_sym: string empty symbol
20              :param win_length: int number of own symbols in a row
21              :return:
22          """
23          self.m = mine_sym
24          self.o = opponent_sym
25          self.empty = empty_sym
26          self.pf = playfield.PlayField(empty_sym=self.empty)
27
28      def play(self, field):...
36
37      def find_best_move(self, moves):...
40
```

dědění

```
14  class MyPlayer:
15      def __init__(self, mine_sym, opponent_sym, empty_sym):...
27
28      def play(self, field):...
36
37      def find_best_move(self, moves):...
40
41
```

```
42  class RandomPlayerSimple(MyPlayer):
43      '''a simple implementation but not the fastest I'm afraid'''
44      def find_best_move(self, moves):
45          return random.choice(moves)
46
```