

Python, základní kameny až skály II

Tomáš Svoboda

B4B33RPH, 2016-10-25

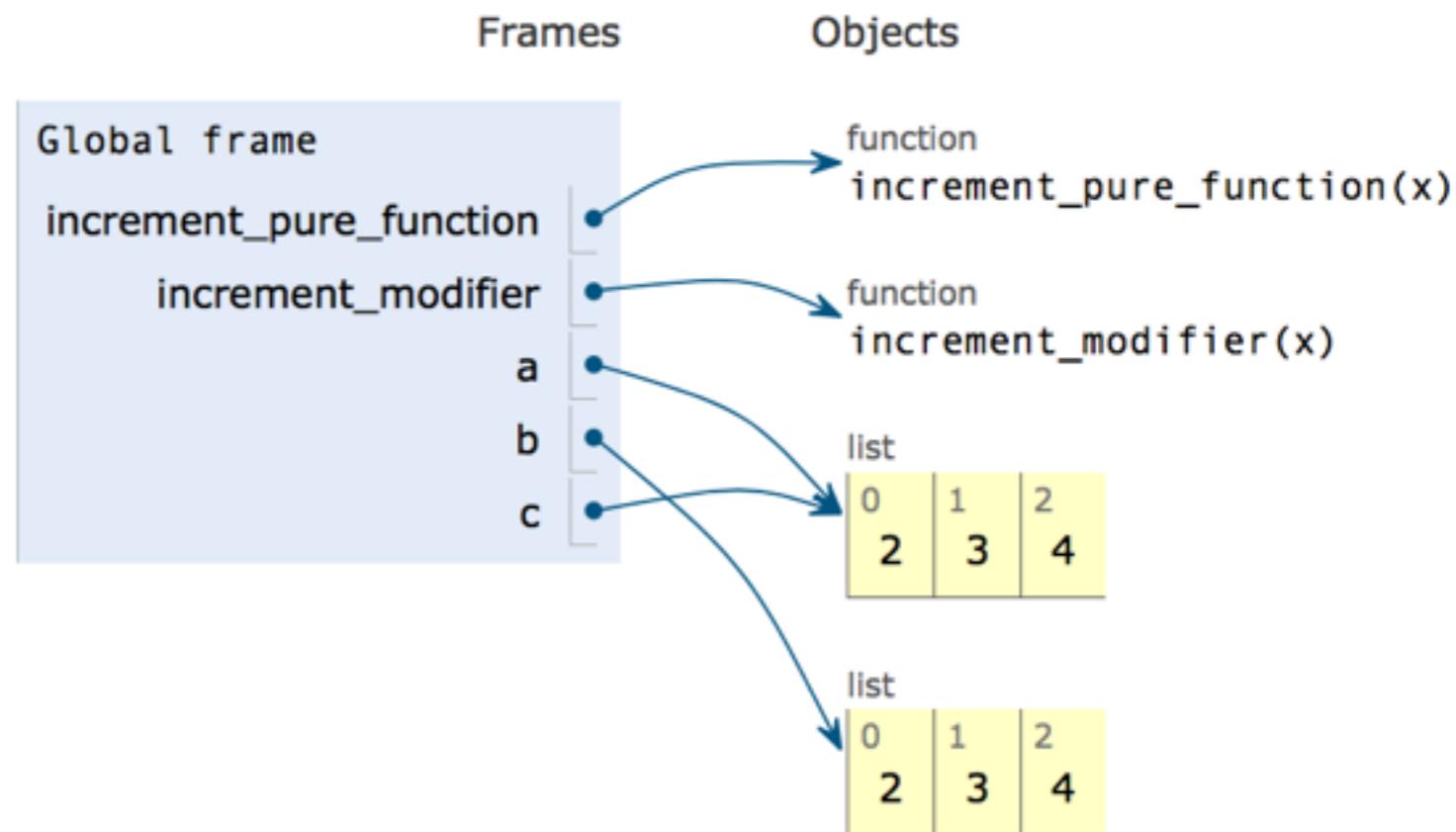
funkce pravé a modifikátory

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)

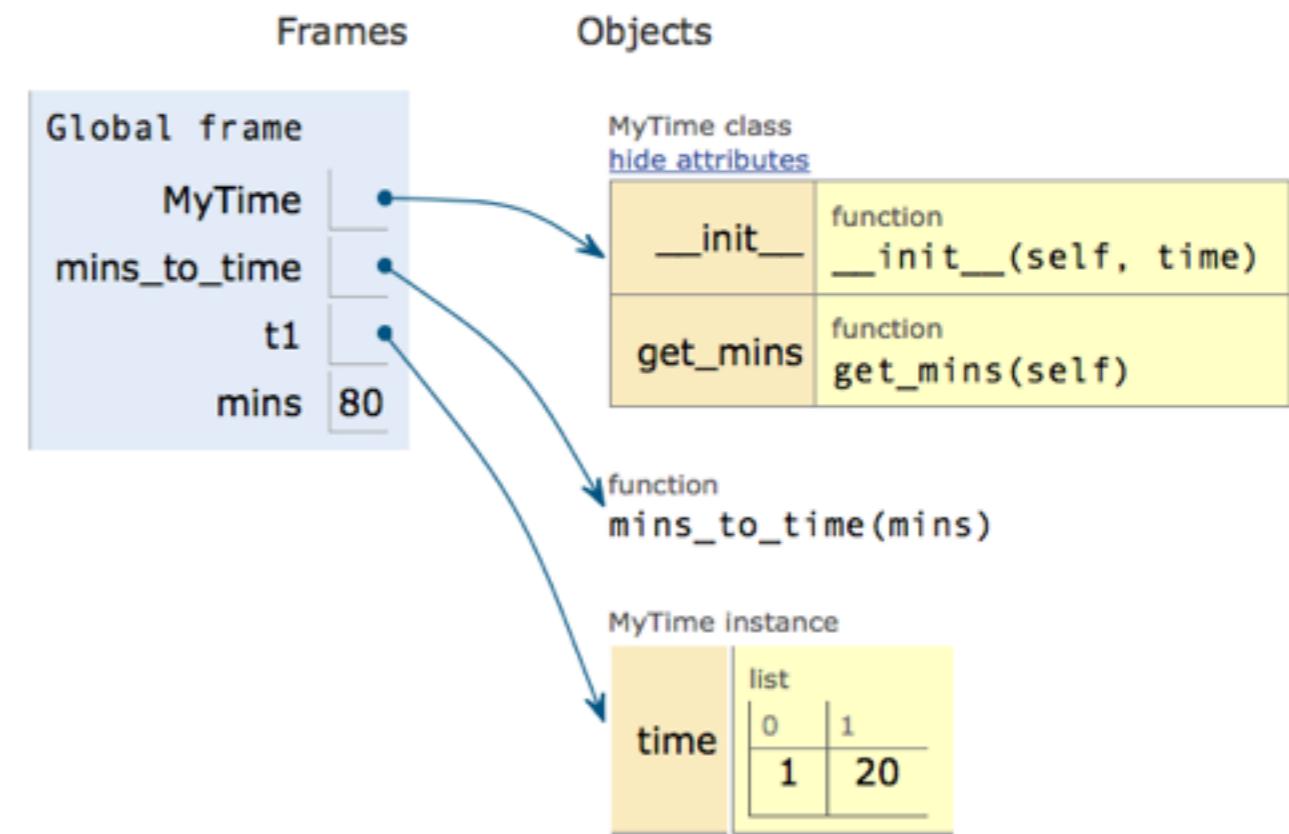
```
[1, 2, 3] , [2, 3, 4]
[2, 3, 4] , [2, 3, 4] , [2, 3, 4]
```



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()
```

Frames Objects

Global frame

MyTime	function __init__(self, time)
mins_to_time	function get_mins(self)
tvec	list
t1	list
mins	70
time_vec	list

MyTime class
hide attributes

__init__ function __init__(self, time)

get_mins function get_mins(self)

function mins_to_time(mins)

list

MyTime instance

time

list

line that has just executed

next line to execute

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

```
graph TD
    subgraph Global_Frame [Global frame]
        direction TB
        MyTime[MyTime]
        mins_to_time[mins_to_time]
        tvec[tvec]
        t1[t1]
        mins[mins]
        time_vec[time_vec]
    end

    subgraph Objects [Objects]
        direction TB
        MyTime_Class[MyTime class  
hide attributes]
        MyTime_Class_Methods[__init__ function  
get_mins function]
        Function_MinutsToTime[function mins_to_time(mins)]
        List_Tvec[tvec]
        Instance_T1[MyTime instance  
time]
        List_TimeVec[time_vec]
    end

    MyTime --> MyTime_Class_Methods
    mins_to_time --> Function_MinutsToTime
    tvec --> List_Tvec
    t1 --> Instance_T1
    mins --> List_TimeVec
    time_vec --> List_TimeVec

    MyTime_Class_Methods --> MyTime
    Function_MinutsToTime --> Function_MinutsToTime
    List_Tvec --> List_Tvec
    Instance_T1 --> Instance_T1
    List_TimeVec --> List_TimeVec
```

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])
```

Frames Objects

Global frame

MyTime	•
t1	•
t2	•
t3	•

MyTime class
hide attributes

__init__	function
__init__(self, time)	

MyTime instance

time	list
0	1
0	0

MyTime instance

time	•
------	---

MyTime instance

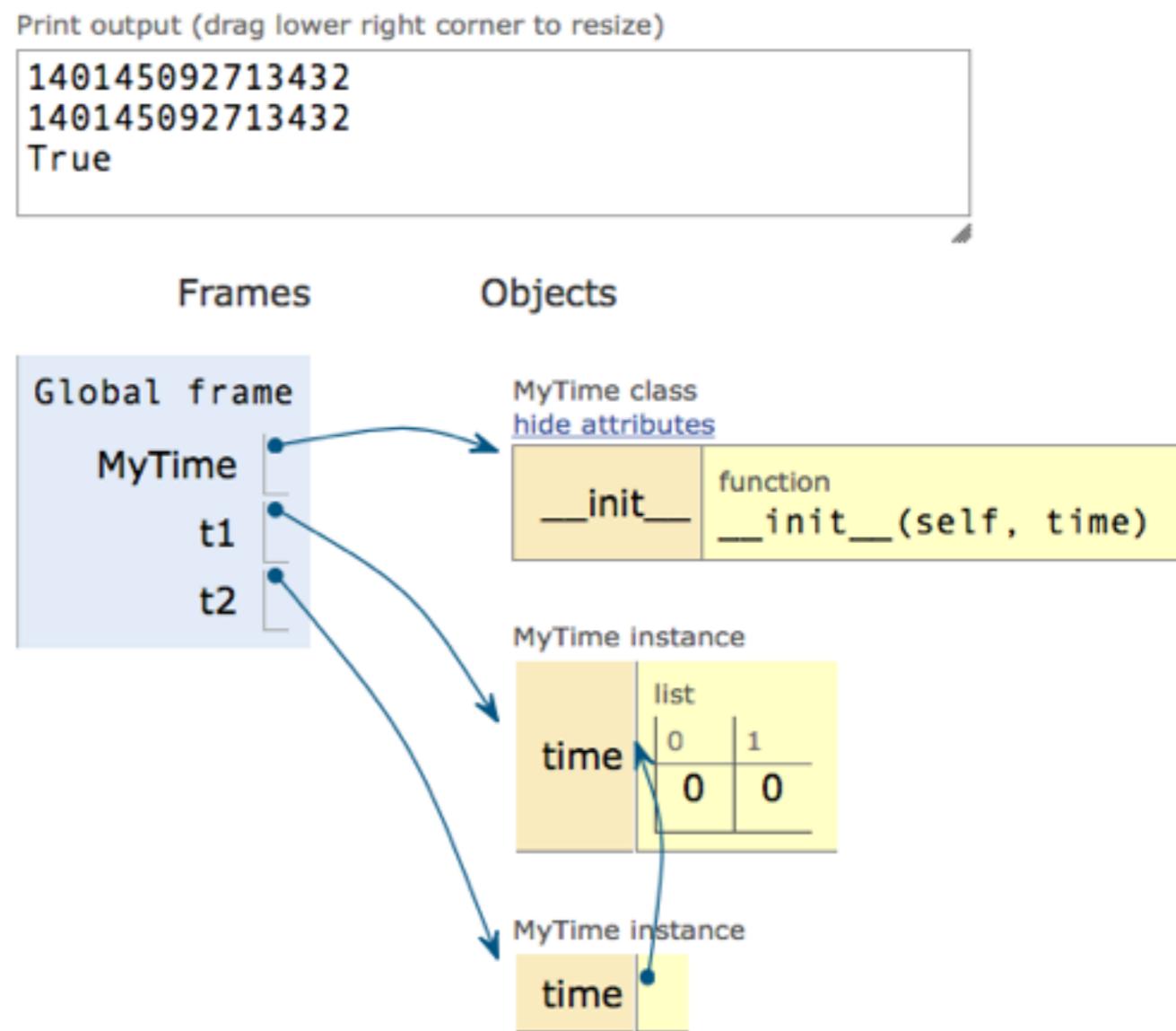
time	list
0	1
0	0

(drag lower right corner to resize code editor)

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
```



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

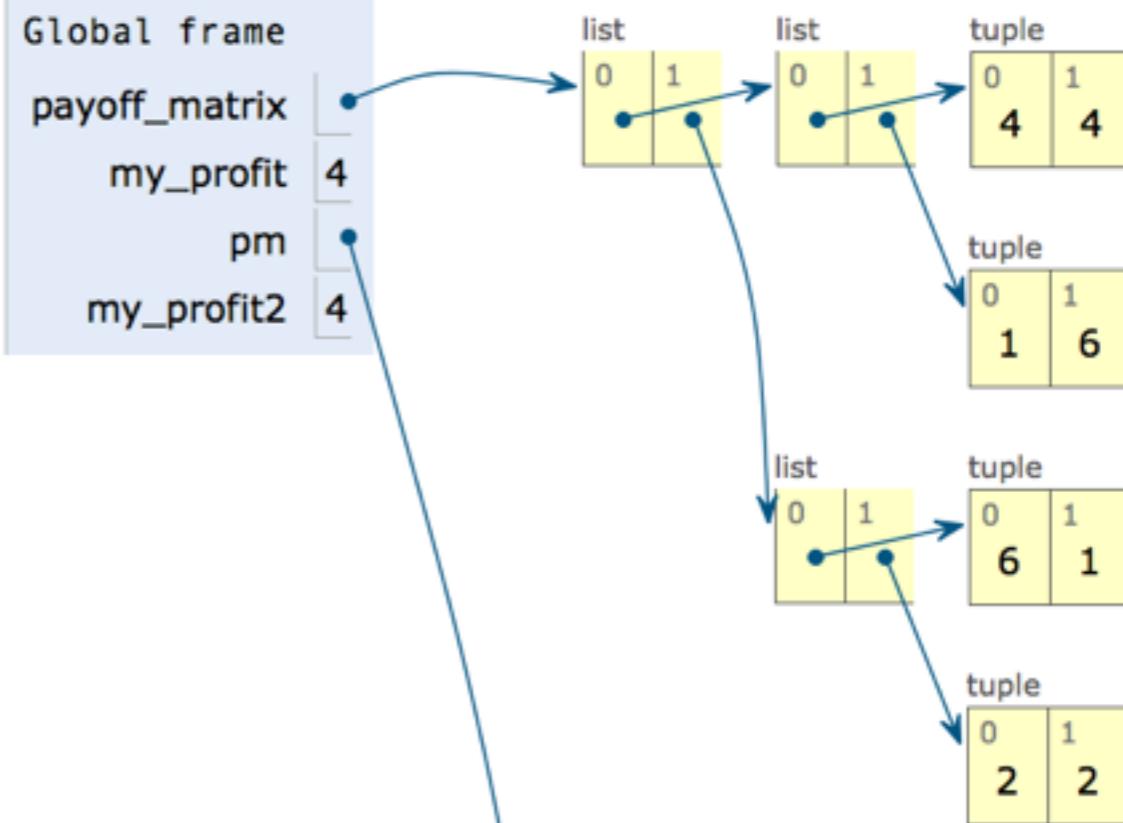
<< First < Back Done running (8 steps) Forward > Last >>

Frames

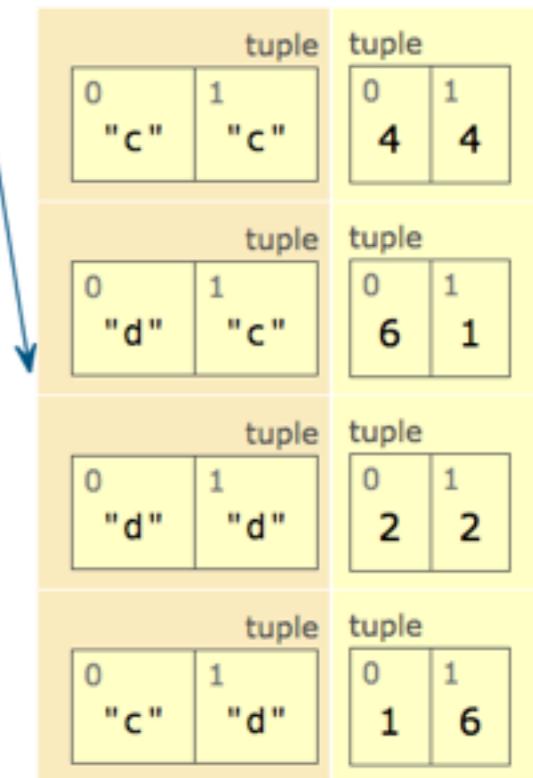
Objects

Global frame

payoff_matrix	list
my_profit	4
pm	list
my_profit2	4



dict



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
- live-coding-session

skládání

```
15
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
24         self.m = mine_sym
25         self.o = opponent_sym
26         self.empty = empty_sym
27         self.pf = playfield.PlayField(empty_sym=self.empty)
28
29     def play(self, field):...
30
31     def find_best_move(self, moves):...
```