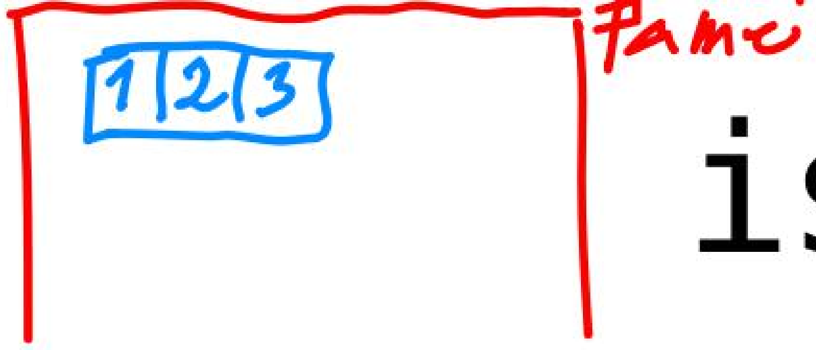


is vs ==

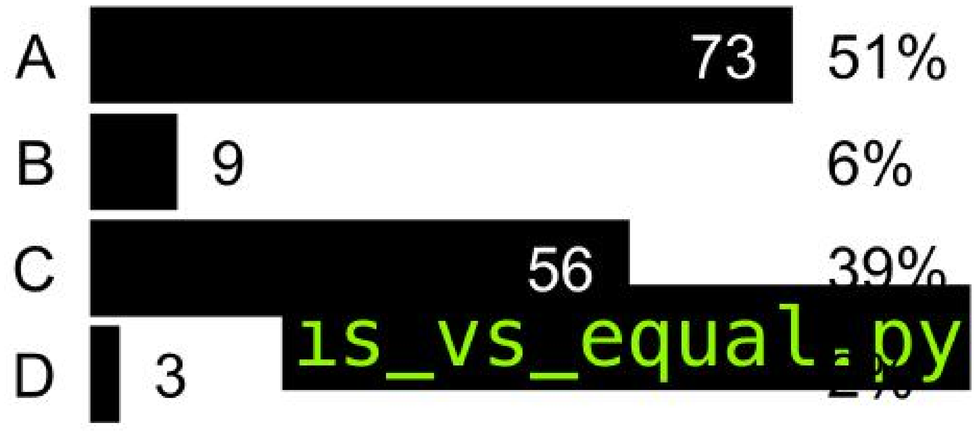
```
a = [1,2,3]
b = [1,2,3]
c = a

d1 = a==b
d2 = a is b
d3 = a==c, a is c
```

A: `d1 True, d2 False`

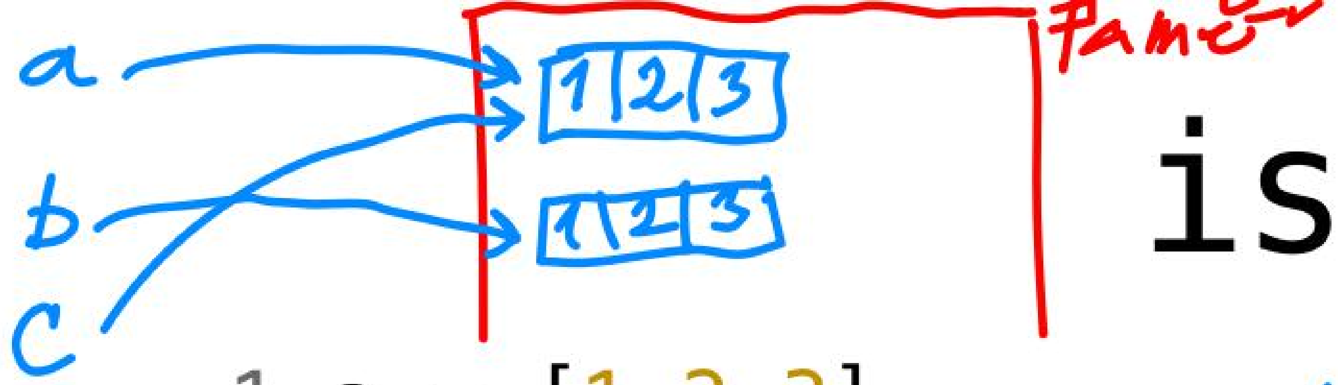
B: `d1 False, d2 False`

C: `d1 True, d2 True`

D: chyba



`is_vs_equal.py`



is vs ==

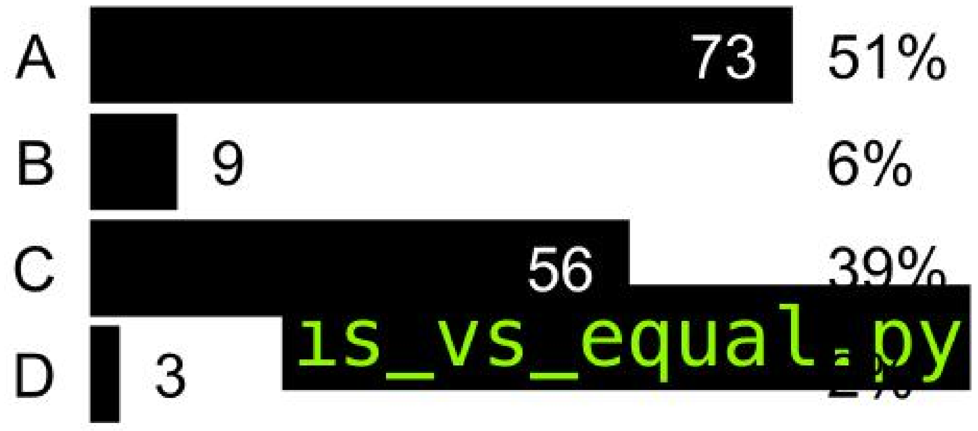
```
a = [1,2,3]
b = [1,2,3]   list([1,2,3])
c = a

d1 = a==b   True
d2 = a is b   False
d3 = a==c, a is c
```

[✓ A:] d1 True, d2 False

B: d1 False, d2 False

C: d1 True, d2 True

D: chyba



is_vs_equal.py

is vs ==

```
a = [1,2,3]
b = [1,2,3]
c = a

d1 = a==b
d2 = a is b
d3 = a==c, a is c
```

A: d1 True, d2 False

B: d1 False, d2 False

C: d1 True, d2 True

D: chyba

d3 bude:

A: True, False

B: True, True

C: False, False

D: skončí chybou

is_vs_equal.py

is vs ==

```python
a = [1,2,3]
b = [1,2,3]
c = a

d1 = a==b
d2 = a is b
d3 = a==c, a is c
```

A: `d1 True, d2 False`

B: `d1 False, d2 False`

C: `d1 True, d2 True`

D: chyba

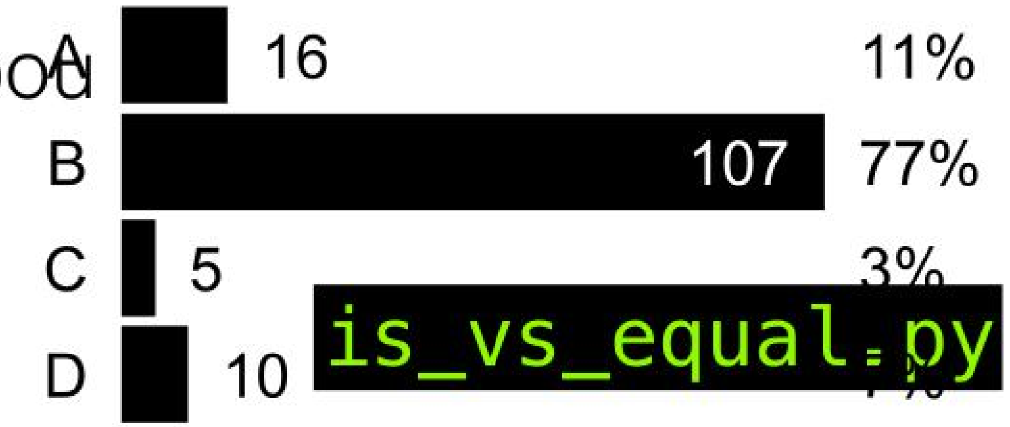
`d3` bude:

A: `True, False`

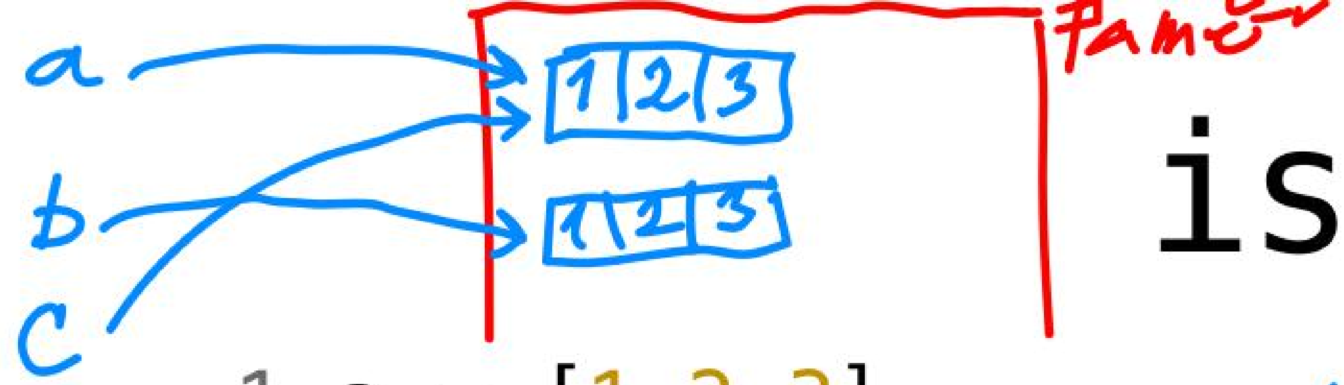
B: `True, True`

C: `False, False`

D: skončí chybou



`is_vs_equal.py`



is vs ==

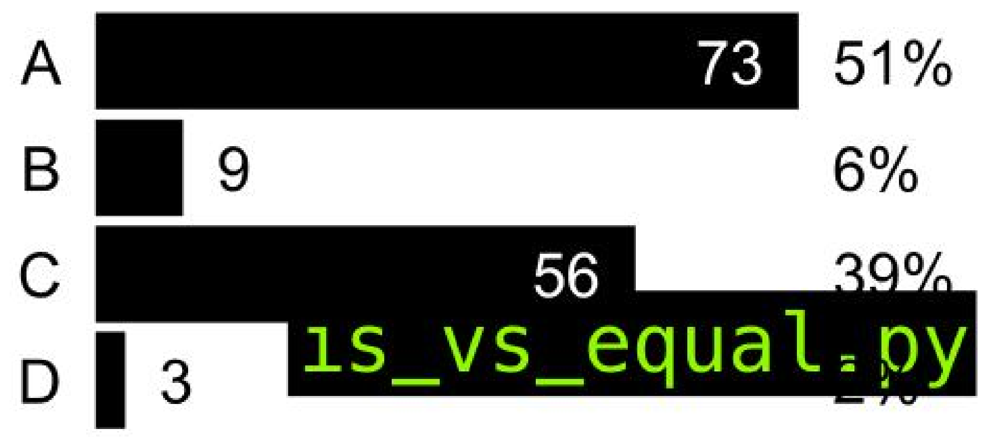
```
a = [1,2,3]
b = [1,2,3]   list([1,2,3])
c = a

d1 = a==b   True
d2 = a is b   False
d3 = a==c, a is c
```

[✓ A: d1 True, d2 False]

B: d1 False, d2 False

C: d1 True, d2 True

D: chyba



is_vs_equal.py

is vs ==

```
a = [1,2,3]
b = [1,2,3]
c = a

d1 = a==b
d2 = a is b
d3 = a==c, a is c
```

A: `d1 True, d2 False`

B: `d1 False, d2 False`

C: `d1 True, d2 True`

D: chyba

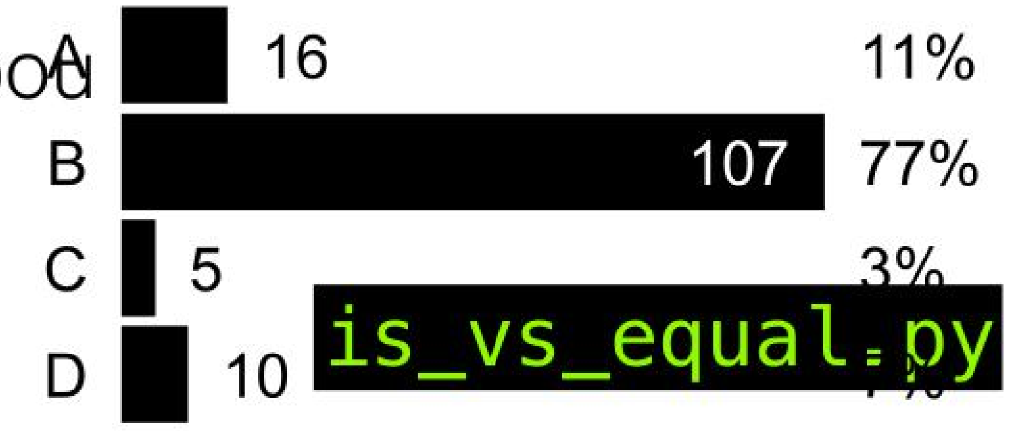
`d3` bude:

A: `True, False`

B: `True, True`

C: `False, False`

D: skončí chybou



`is_vs_equal.py`

pointers

```
a = [1,2,3]
b = a
a[1] = 9
d0 = a == [1,9,3]
d1 = b == [1,2,3]
d2 = b == [1,9,3]
d3 = b is a
```

pointers

```
a = [1,2,3]
b = a
a[1] = 9
d0 = a == [1,9,3]
d1 = b == [1,2,3]
d2 = b == [1,9,3]
d3 = b is a
```

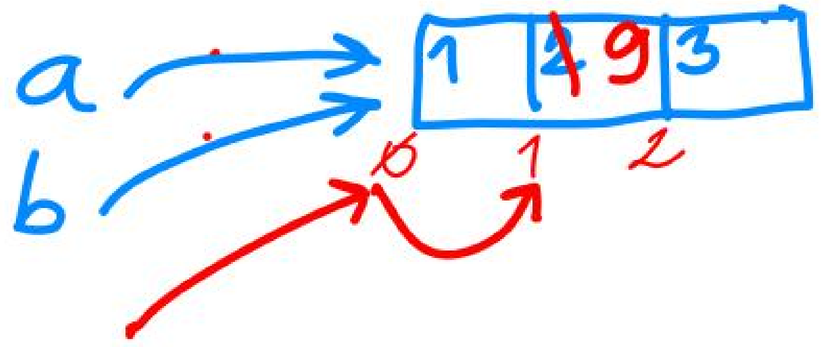
Proměnné `d0, d1, d2, d3` budou:

A: `True, True, False, False`

B: `True, False, True, True`

C: `True, False, False, True`

D: `True, False, True, False`



pointers

```
a = [1,2,3]
b = a
a[1] = 9
d0 = a == [1,9,3] T
d1 = b == [1,2,3] F
d2 = b == [1,9,3] T
d3 = b is a T
```

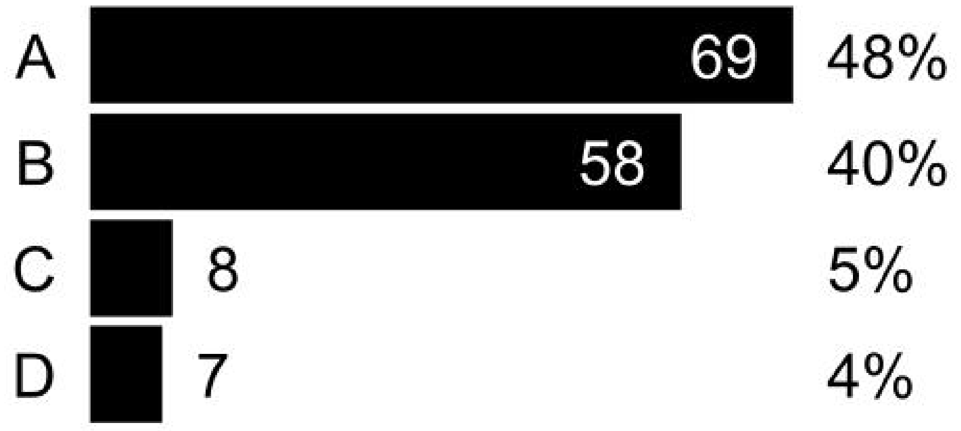
Proměnné `d0, d1, d2, d3` budou:

A: `True, True, False, False`

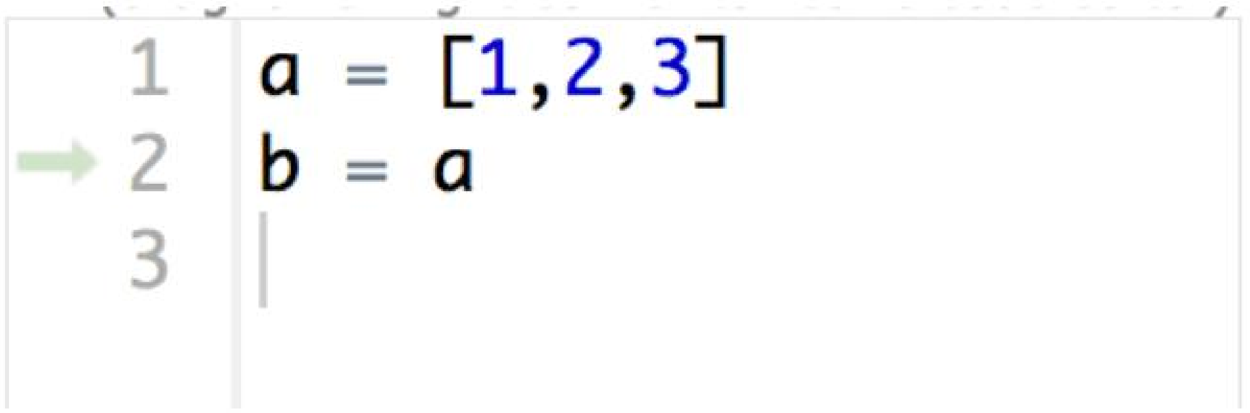
[✓ B]: `True, False, True, True`

C: `True, False, False, True`

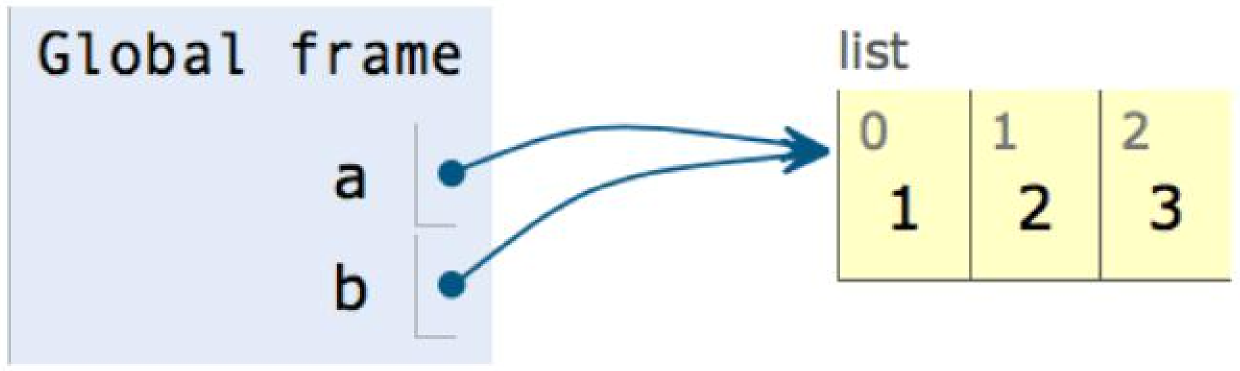
D: `True, False, True, False`



pointers



```python
a = [1,2,3]
b = a
```

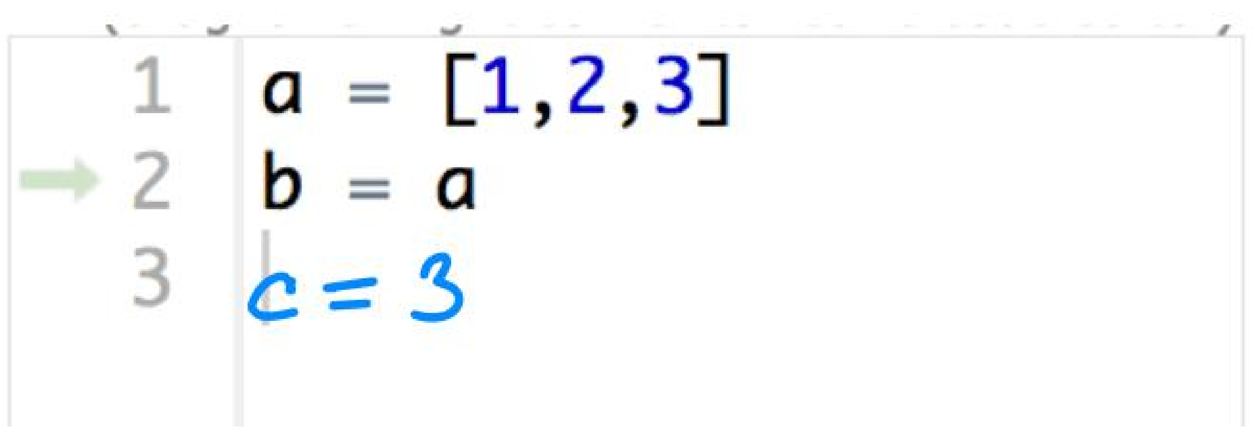


visualisation

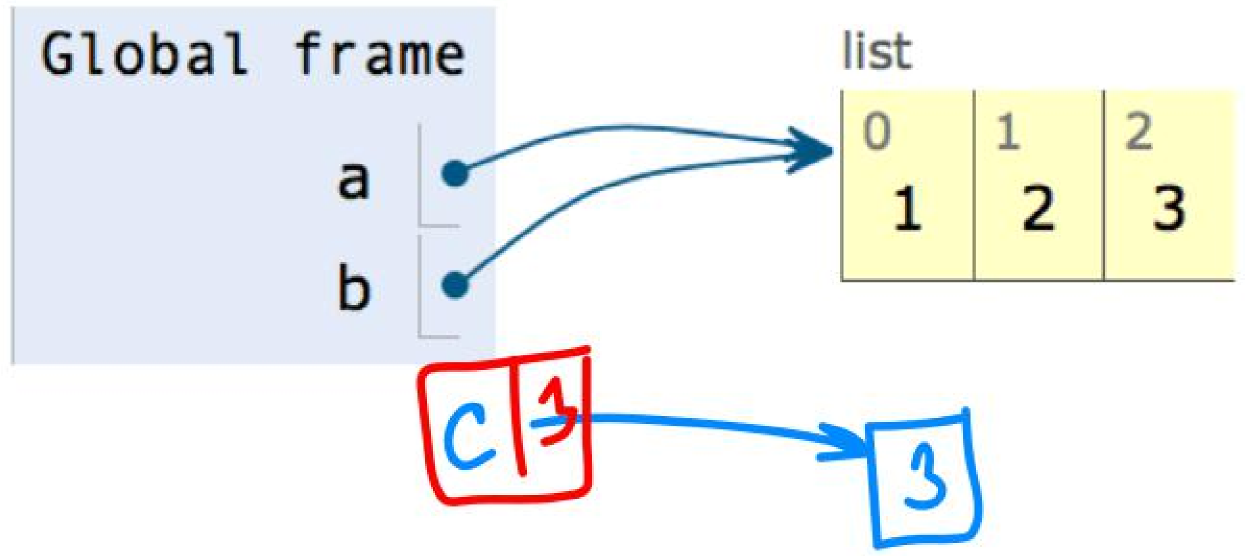
`is` vs ==

pythontutor.com

pointers



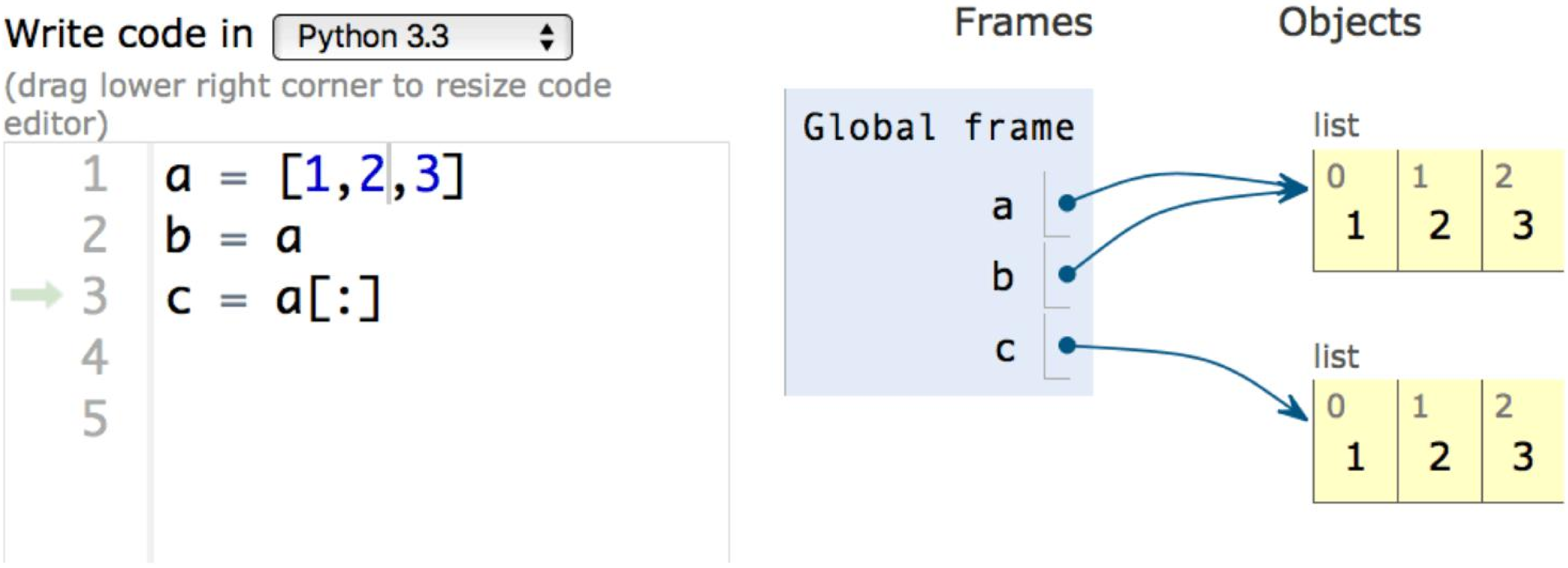
```
a = [1,2,3]
b = a
c = 3
```



visualisation

is vs ==

making copy, a[:], rychle, ale …

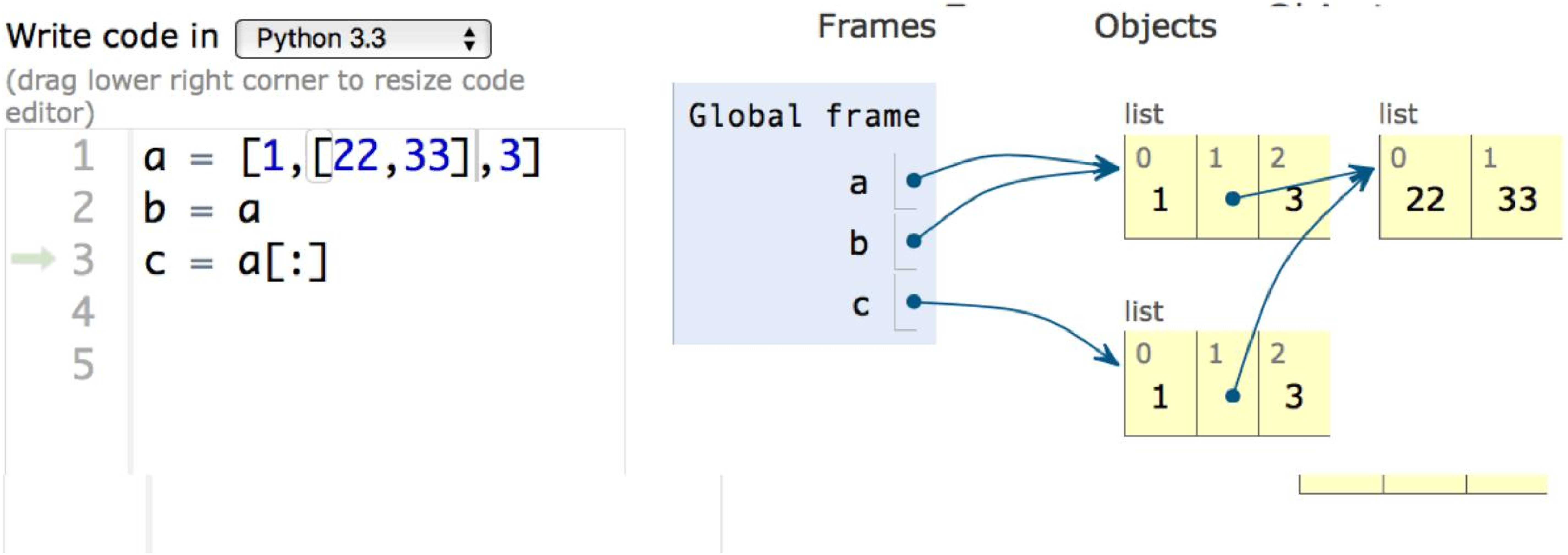


Write code in Python 3.3

(drag lower right corner to resize code editor)

```
a = [1,2,3]
b = a
c = a[:]
```

making copy, a[:], rychle, ale …

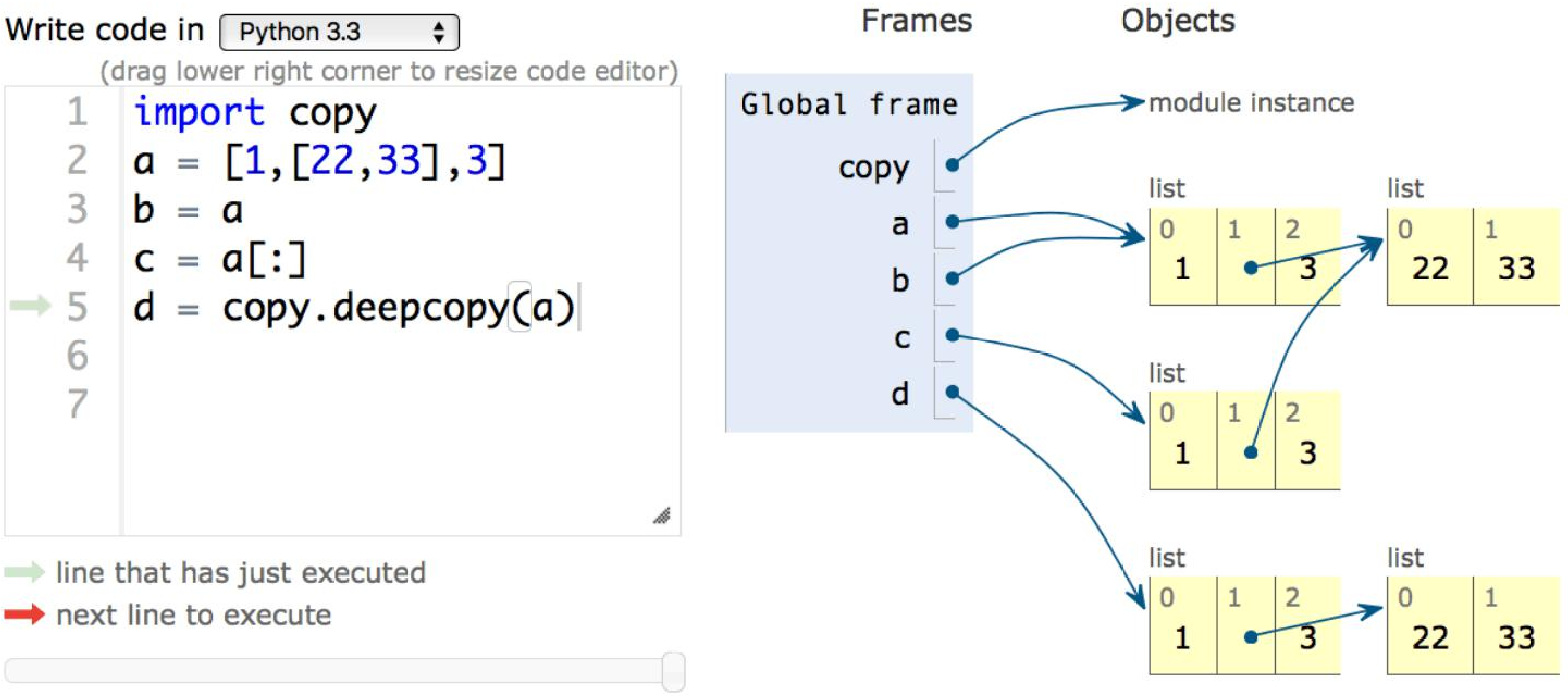

Write code in Python 3.3

(drag lower right corner to resize code editor)

```
a = [1,[22,33],3]
b = a
c = a[:]
```

import copy and go deep

http://docs.python.org/3.4/library/copy.html



Write code in Python 3.3

(drag lower right corner to resize code editor)

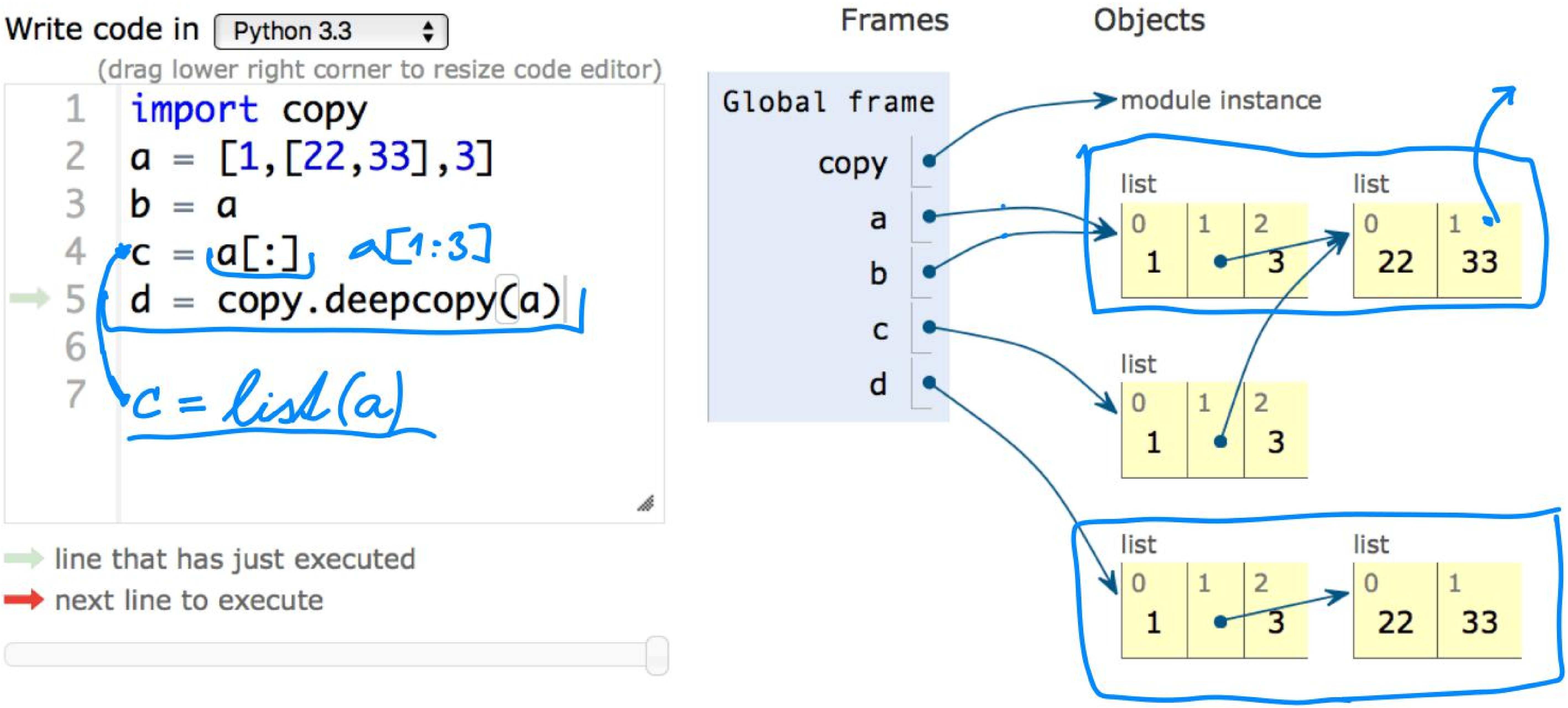
```
import copy
a = [1,[22,33],3]
b = a
c = a[:]
d = copy.deepcopy(a)
```

🟩 line that has just executed

🟥 next line to execute

`import copy` and go deep

http://docs.python.org/3.4/library/copy.html

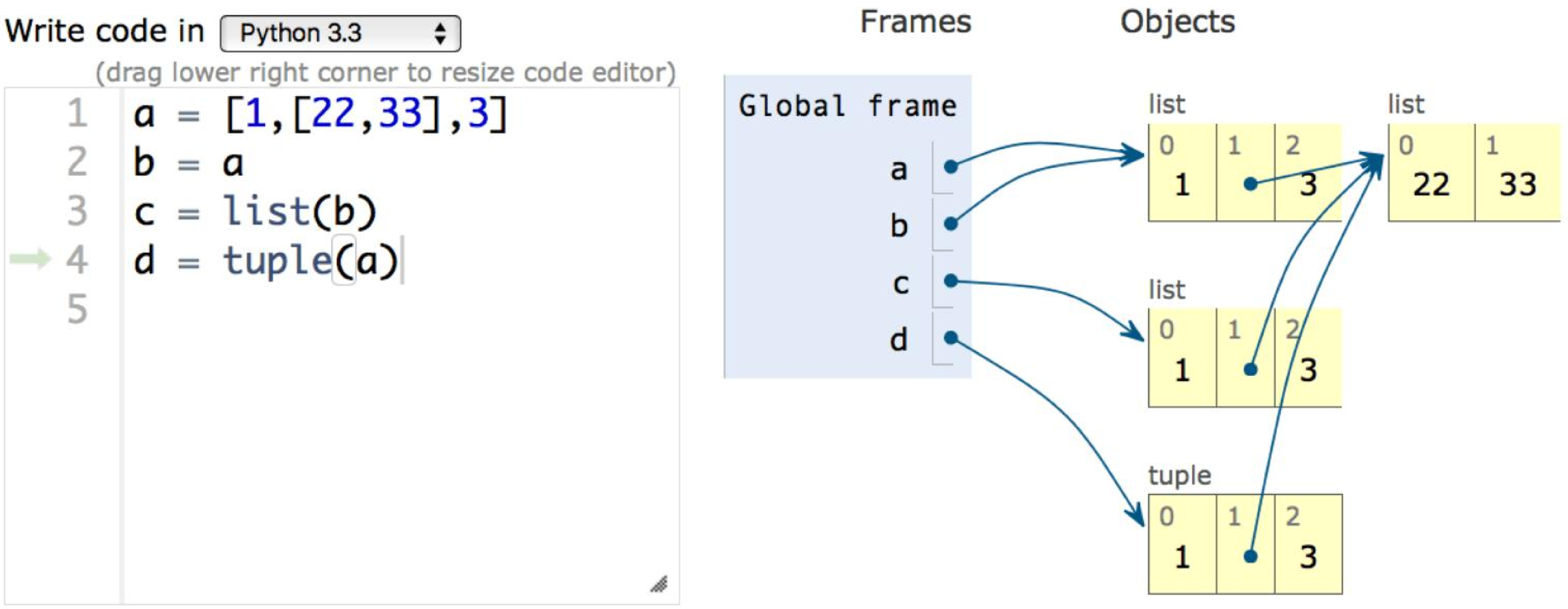


Write code in Python 3.3

(drag lower right corner to resize code editor)

```
import copy
a = [1,[22,33],3]
b = a
c = a[:]    a[1:3]
d = copy.deepcopy(a)

c = list(a)
```

pozor na mělkost kopií

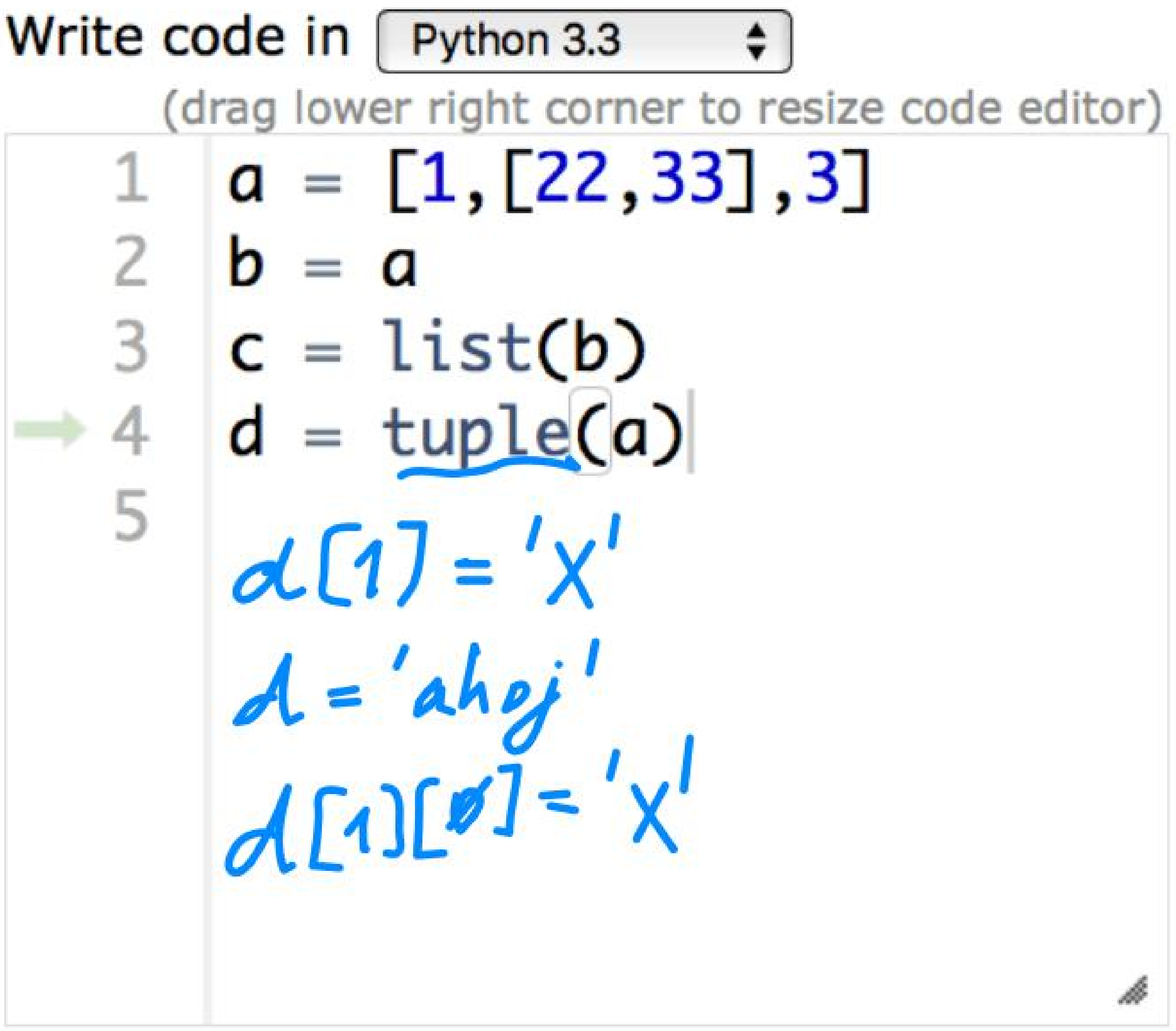


Write code in Python 3.3

(drag lower right corner to resize code editor)

```python
a = [1,[22,33],3]
b = a
c = list(b)
d = tuple(a)
```

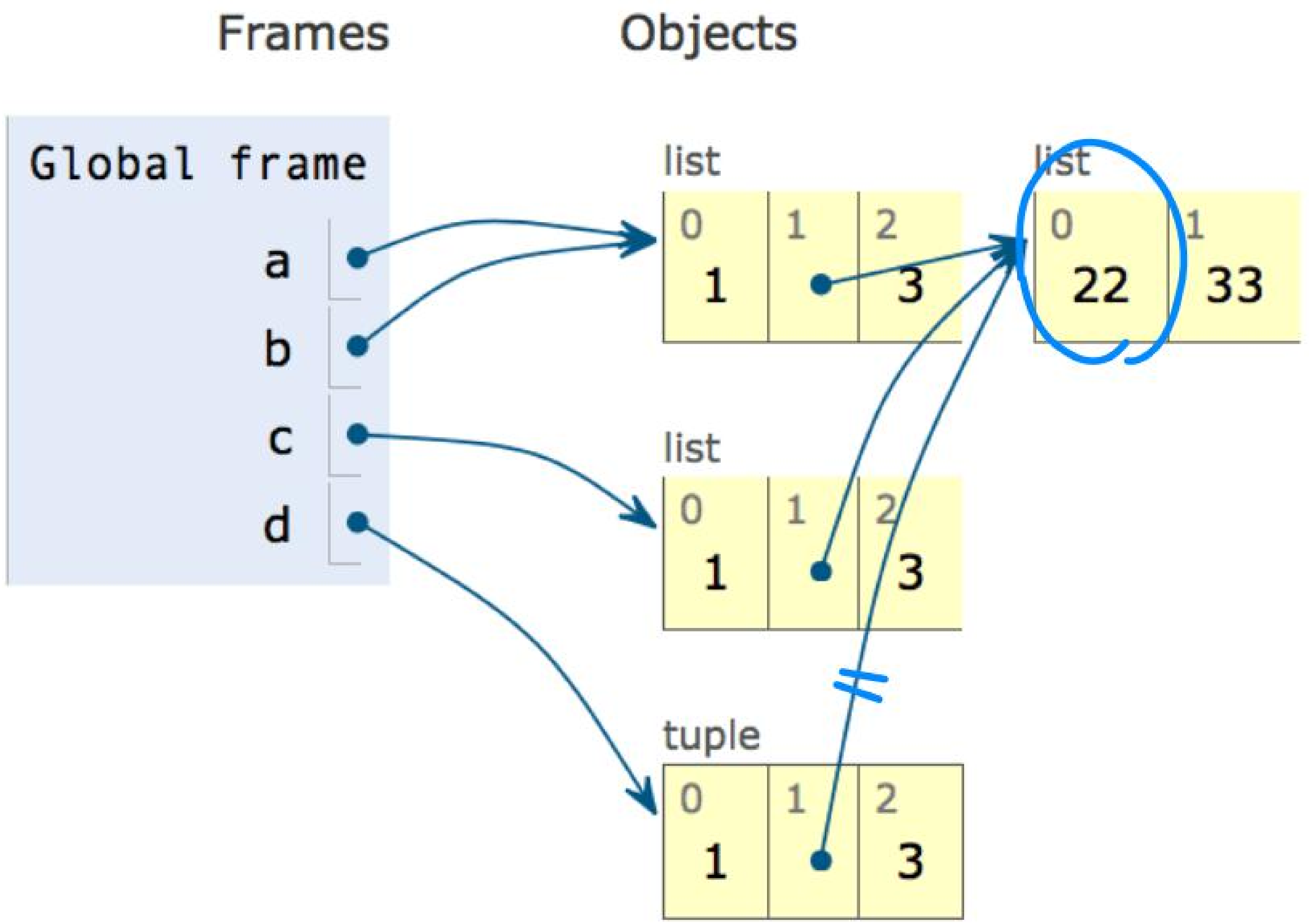
pozor na mělkost kopií



Write code in Python 3.3

(drag lower right corner to resize code editor)

```
a = [1,[22,33],3]
b = a
c = list(b)
d = tuple(a)

d[1] = 'X'
d = 'ahoj'
d[1][0] = 'X'
```



Funkce: ryzí vs. modifikátory

pure modifier

Funkce: ryzí vs. modifikátory

```python
def increment_pure_function(x):
    v = []
    for item in x:
        v.append(item+1)
    return(v)

def increment_modifier(x):
    for i in range(len(x)):
        x[i] = x[i]+1
    return(x)

if __name__ == "__main__":
    a = [1,2,3]
    b1 = increment_pure_function(a)
    d0 = a == b1
    b2 = increment_modifier(a)
    d1 = b1 == b2
    d2 = a == b1
```

```python
def increment_pure_function(x):
    v = []
    for item in x:
        v.append(item+1)
    return(v)

def increment_modifier(x):
    for i in range(len(x)):
        x[i] = x[i]+1
    return(x)

if __name__ == "__main__":
    a = [1,2,3]
    b1 = increment_pure_function(a)
    d0 = a == b1
    b2 = increment_modifier(a)
    d1 = b1 == b2
    d2 = a == b1
```

Hodnoty `d0, d1, d2` budou:

A: `False, True, True`

B: `False, True, False`

C: `False, False, False`

- Použij funkci ryzí, pokud je tvé srdce ryzí (a nechceš hledat špatně odhalitelné chyby)
- Pokud návrh vede na potřebu funkce modifkátoru, zvaž objektový návrh.
- Nikdy nevracej data, která modifikuješ a hlavně, necht' volání modikátoru nemá pravou stranu!

- Použij funkci ryzí, pokud je tvé srdce ryzí (a nechceš hledat špatně odhalitelné chyby)

- Pokud návrh vede na potřebu funkce modifkátoru, zvaž objektový návrh.

  a.append(item)

- Nikdy nevracej data, která modifikuješ a hlavně, nechť volání modikátoru nemá pravou stranu!

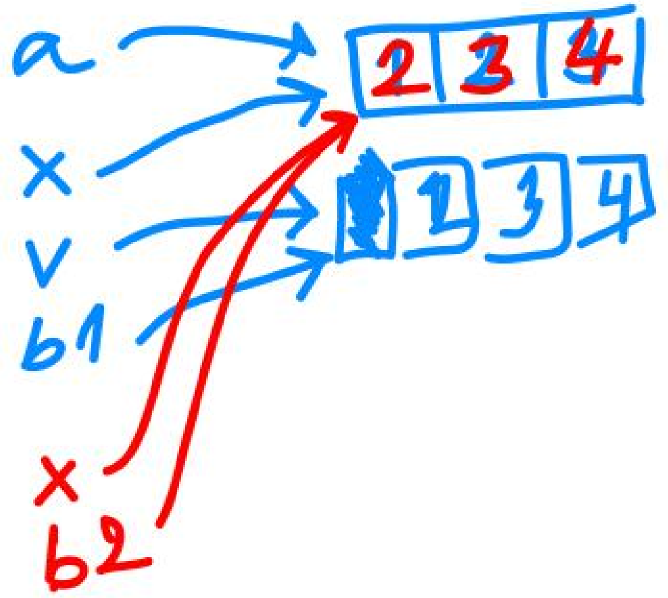
```python
def increment_pure_function(x):
    v = []   v=list()
    for item in x:
        v.append(item+1)
    return(v)

def increment_modifier(x):
    for i in range(len(x)):
        x[i] = x[i]+1
    return(x)

if __name__ == "__main__":
    a = [1,2,3]
    b1 = increment_pure_function(a)
    d0 = a == b1   F
    b2 = increment_modifier(a)
    d1 = b1 == b2   T
    d2 = a == b1   T
```

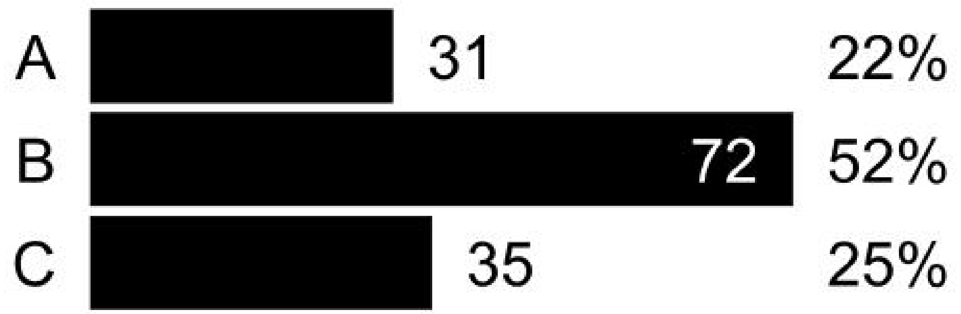
Hodnoty **d0, d1, d2** budou:

[✓ A: False, True, True]

B: False, True, False

C: False, False, False





- Použij funkci ryzí, pokud je tvé srdce ryzí (a nechceš hledat špatně odhalitelné chyby)

- Pokud návrh vede na potřebu funkce modifkátoru, zvaž objektový návrh.

  a.append(item)

- Nikdy nevracej data, která modifikuješ a hlavně, nechť volání modikátoru nemá pravou stranu!

- Použij funkci ryzí, pokud je tvé srdce ryzí (a nechceš hledat špatně odhalitelné chyby)
- Pokud návrh vede na potřebu funkce modifkátoru, zvaž objektový návrh.

a.append(item)

- Nikdy nevracej data, která modifikuješ a hlavně, nechť volání modikátoru nemá pravou stranu!

Nebezpečí implicitních parametrů

```python
def fn(x=[0,0]):
    x[0] = x[0]+1
    return x+[1]

a = fn()
b = fn()
```

```python
def fn(x=[0,0]):
    x[0] = x[0]+1
    return x+[1]

a = fn()
b = fn()
```

Hodnoty proměnných `a, b` budou:

A: `[1,0,1], [1,0,1]`

B: `[2,1], [2,1]`

C: `[1,0,1], [2,0,1]`

D: dojde k chybě za běhu programu

```python
def fn(x=[0,0]):
    x[0] = x[0]+1
    return x+[1]

a = fn()
b = fn()
```

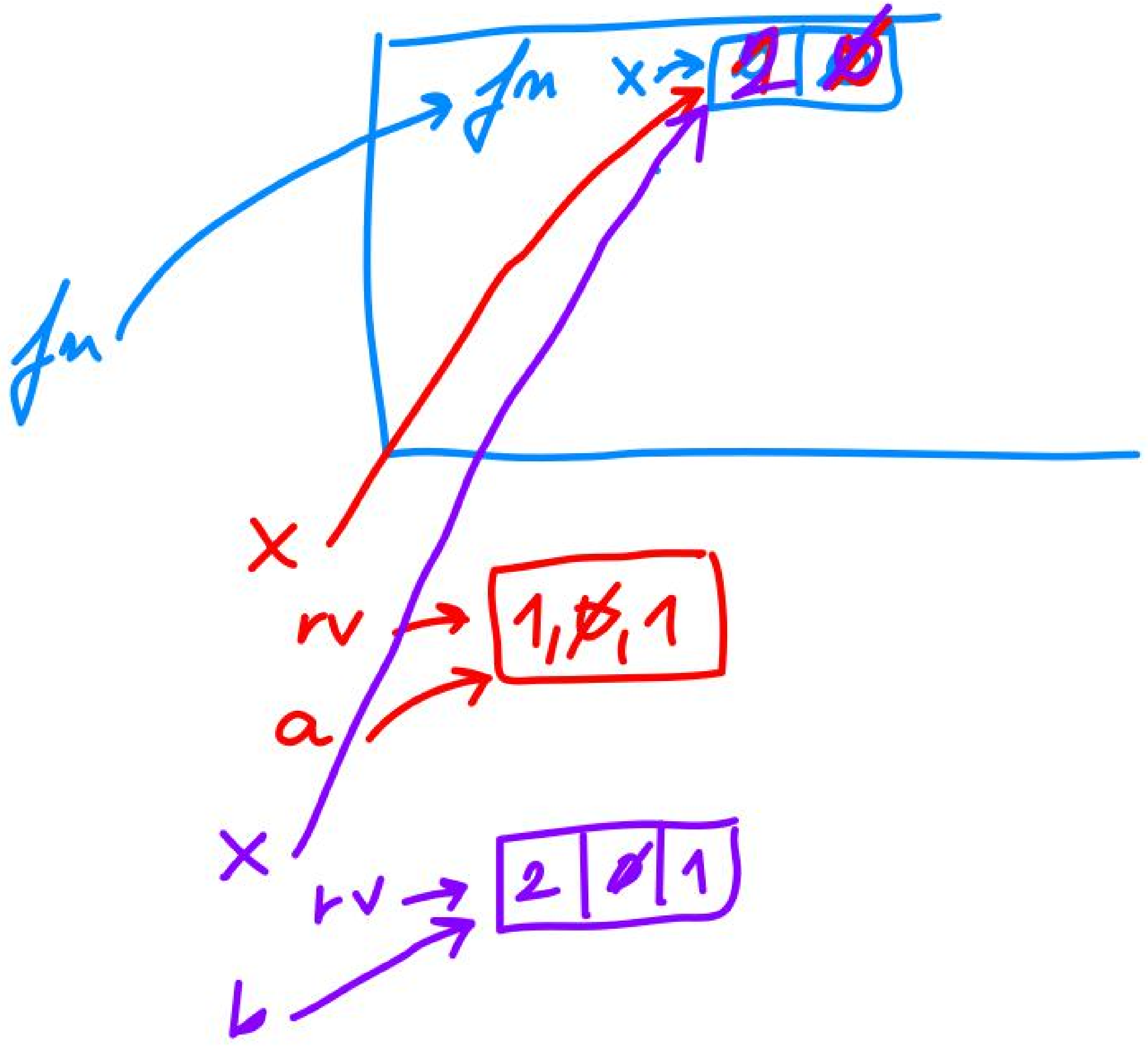
Hodnoty proměnných **a, b** budou:

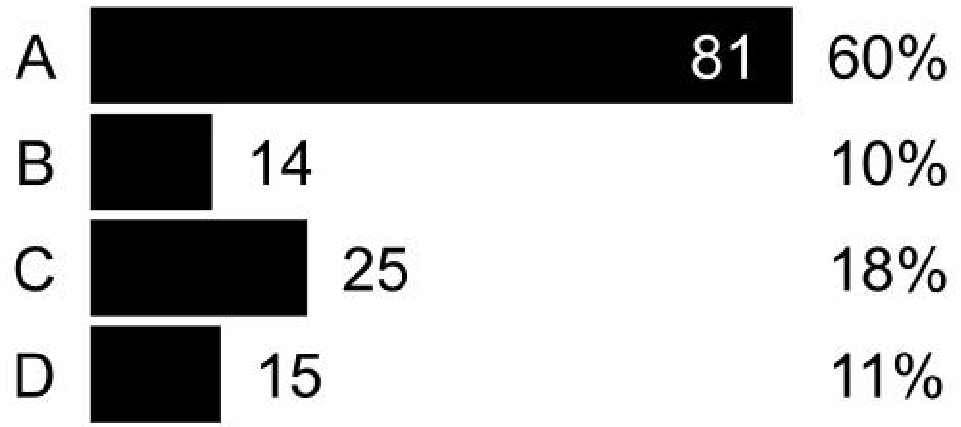
A: `[1,0,1], [1,0,1]`

B: `[2,1], [2,1]`

[✓ C: `[1,0,1], [2,0,1]`]

D: dojde k chybě za běhu programu





```python
def fn(x=[0,0]):
    x[0] = x[0]+1
    return x+[1]

a = fn()
b = fn()
c = fn([0,0])
```

```python
def fn(x=[0,0]):
    x[0] = x[0]+1
    return x+[1]

a = fn()
b = fn()
c = fn([0,0])
```

Hodnota proměnné `c` bude:

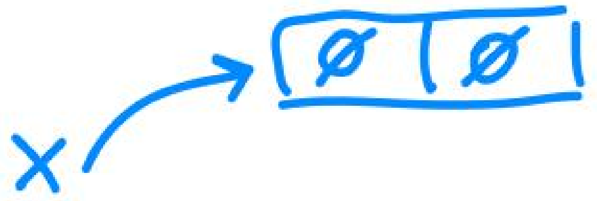
A: `[1,0,1]`

B: `[2,0,1]`

C: `[3,0,1]`

D: dojde k chybě za běhu programu

```python
def fn(x=[0,0]):
    x[0] = x[0]+1
    return x+[1]

a = fn()
b = fn()
c = fn([0,0])
```



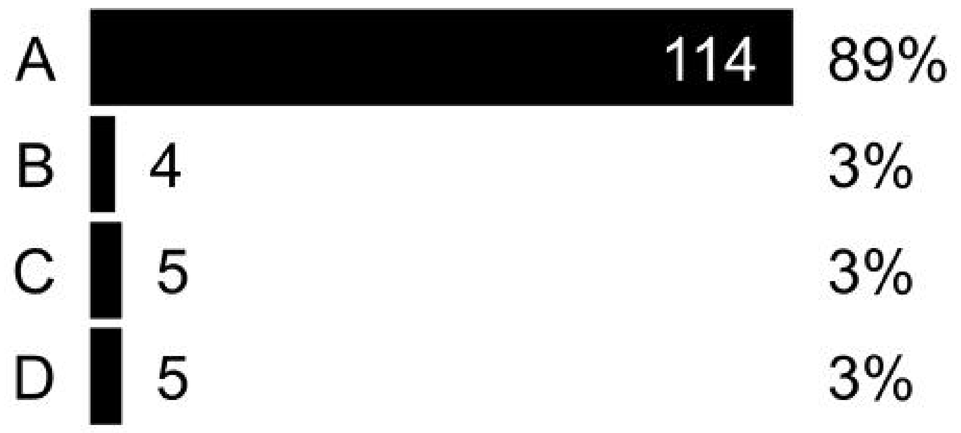
Hodnota proměnné `c` bude:

A: `[1,0,1]`

B: `[2,0,1]`

C: `[3,0,1]`

D: dojde k chybě za běhu programu



```python
def fn(x=[0,0]):
    x[0] = x[0]+1
    return x+[1]

a = fn()
b = fn()
c = fn([0,0])
```

Hodnota proměnné `c` bude:

A: `[1,0,1]`

B: `[2,0,1]`

C: `[3,0,1]`

D: dojde k chybě za běhu programu

`implicit_mutable_parameters_simple.py`

https://docs.python-guide.org/writing/gotchas/

```python
def fn(x=[0,0]):
    x[0] = x[0]+1
    return x+[1]

a = fn()
b = fn()
c = fn([0,0])
```

```
fn (x=None):
  if x==None:
    x=[0,0]
  ⋮
```

Hodnota proměnné `c` bude:

A: `[1,0,1]`

B: `[2,0,1]`

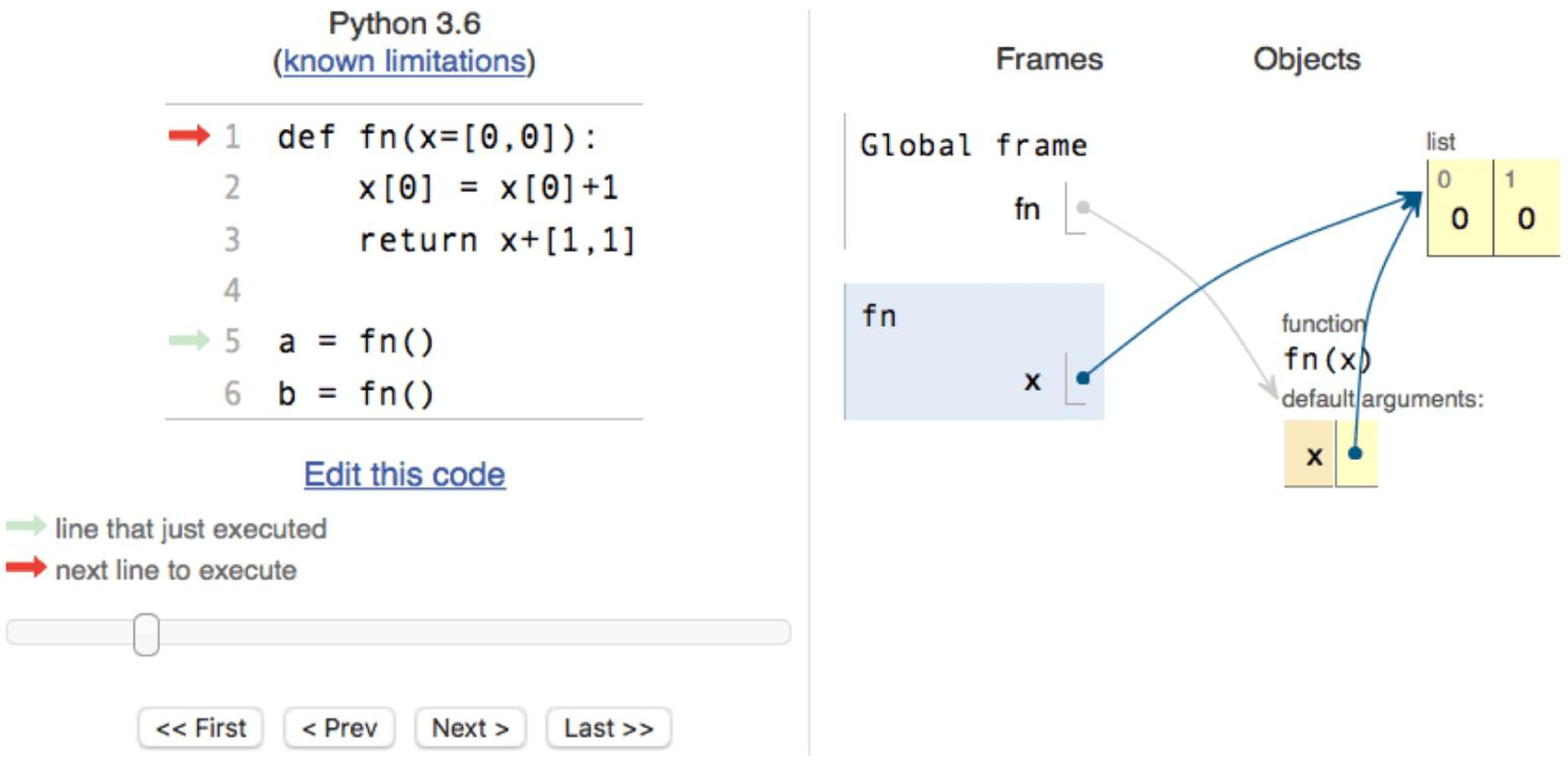
C: `[3,0,1]`

D: dojde k chybě za běhu programu

`implicit_mutable_parameters_simple.py`

https://docs.python-guide.org/writing/gotchas/

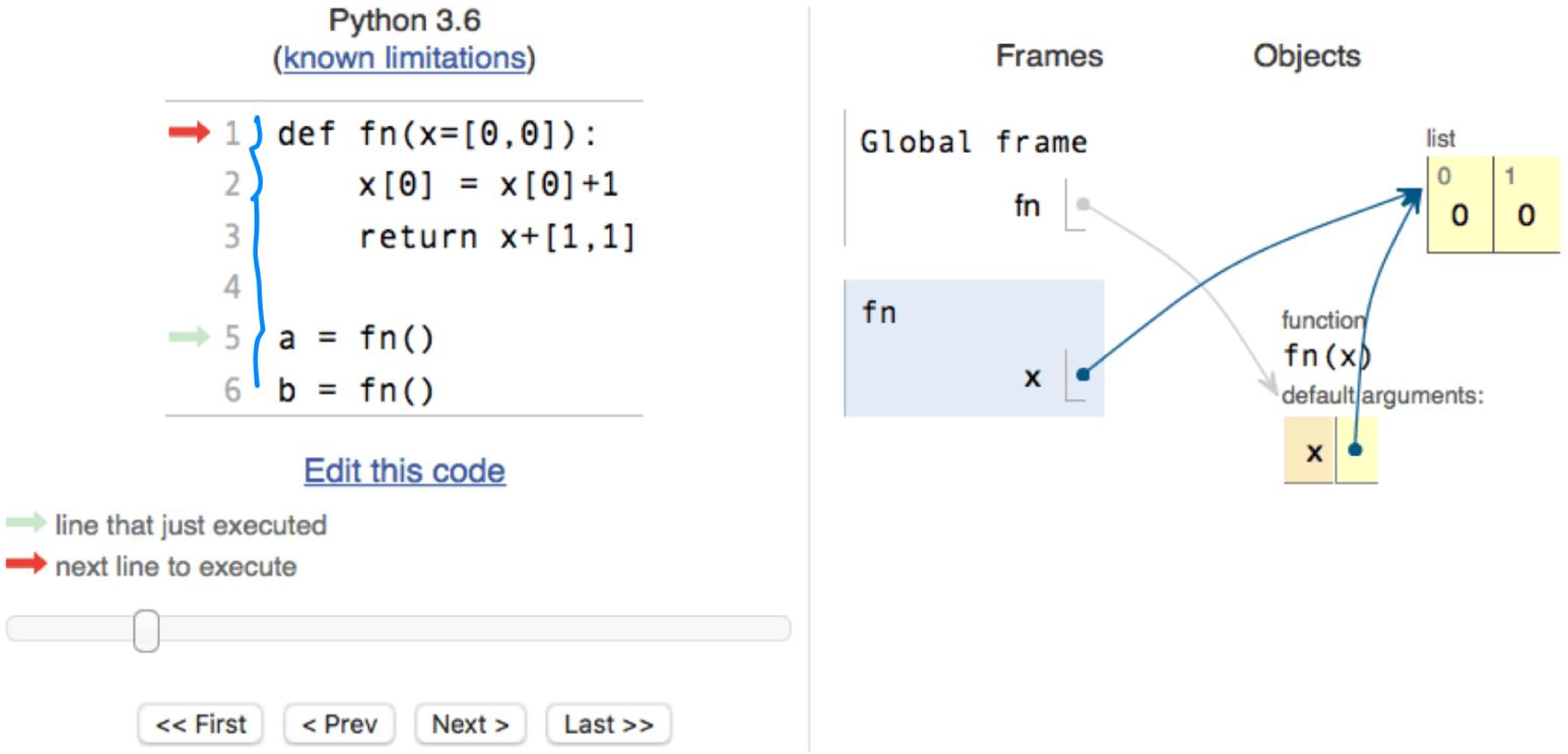
Vyzkoušejte/odkrokujte v pythontutor.com



Python 3.6
(known limitations)

```
def fn(x=[0,0]):
    x[0] = x[0]+1
    return x+[1,1]

a = fn()
b = fn()
```

Vyzkoušejte/odkrokujte v pythontutor.com



Python 3.6
(known limitations)

```
def fn(x=[0,0]):
    x[0] = x[0]+1
    return x+[1,1]

a = fn()
b = fn()
```

```python
class MyTime:
    def __init__(self, time=[0,0]):
        self.time = time

    def increment_time(self, inc):
        for i in range(len(self.time)):
            self.time[i] = self.time[i] + inc[i]

    def __eq__(self,other):
        return self.time == other.time

if __name__ == "__main__":
    t1 = MyTime()
    t2 = MyTime()
    d0 = t1 == t2
    t2.increment_time([2,2])
    d1 = t1 == t2
    print(d0, d1)
```

```python
class MyTime:
    def __init__(self, time=[0,0]):
        self.time = time

    def increment_time(self, inc):
        for i in range(len(self.time)):
            self.time[i] = self.time[i] + inc[i]

    def __eq__(self,other):
        return self.time == other.time

if __name__ == "__main__":
    t1 = MyTime()
    t2 = MyTime()
    d0 = t1 == t2
    t2.increment_time([2,2])
    d1 = t1 == t2
    print(d0, d1)
```

Hodnoty **`d0, d1`** budou:

A: `True, True`

B: `True, False`

C: `False, False`

```python
class MyTime:
    def __init__(self, time=[0,0]):
        self.time = time

    def increment_time(self, inc):
        for i in range(len(self.time)):
            self.time[i] = self.time[i] + inc[i]

    def __eq__(self,other):
        return self.time == other.time

if __name__ == "__main__":
    a = [0,0]
    t1 = MyTime(a)
    t2 = MyTime(a)
    d0 = t1 == t2
    t2.increment_time([2,2])
    d1 = t1 == t2
    print(d0, d1)
```

```python
class MyTime:
    def __init__(self, time=[0,0]):
        self.time = time

    def increment_time(self, inc):
        for i in range(len(self.time)):
            self.time[i] = self.time[i] + inc[i]

    def __eq__(self,other):
        return self.time == other.time

if __name__ == "__main__":
    a = [0,0]
    t1 = MyTime(a)
    t2 = MyTime(a)
    d0 = t1 == t2
    t2.increment_time([2,2])
    d1 = t1 == t2
    print(d0, d1)
```

Hodnoty `d0, d1` budou:

A: `True, True`

B: `True, False`

C: `False, False`

```python
class MyTime:
    def __init__(self, time=[0,0]):
        self.time = time

    def increment_time(self, inc):
        for i in range(len(self.time)):
            self.time[i] = self.time[i] + inc[i]

    def __eq__(self,other):
        return self.time == other.time

if __name__ == "__main__":
    t1 = MyTime([0,0])
    t2 = MyTime([0,0])
    d0 = t1 == t2
    t2.increment_time([2,2])
    d1 = t1 == t2
    print(d0, d1)
```

```python
class MyTime:
    def __init__(self, time=[0,0]):
        self.time = time

    def increment_time(self, inc):
        for i in range(len(self.time)):
            self.time[i] = self.time[i] + inc[i]

    def __eq__(self,other):
        return self.time == other.time

if __name__ == "__main__":
    t1 = MyTime([0,0])
    t2 = MyTime([0,0])
    d0 = t1 == t2
    t2.increment_time([2,2])
    d1 = t1 == t2
    print(d0, d1)
```

Hodnoty **d0, d1** budou:

A: **True, True**

B: **True, False**

C: **False, False**

```python
class MyTime:
    def __init__(self, time=[0,0]):
        self.time = time

    def increment_time(self, inc):
        for i in range(len(self.time)):
            self.time[i] = self.time[i] + inc[i]

    def __eq__(self,other):
        return self.time == other.time

if __name__ == "__main__":
    t1 = MyTime([0,0])
    t2 = MyTime([0,0])
    d0 = t1 == t2
    t2.increment_time([2,2])
    d1 = t1 == t2
    print(d0, d1)
```

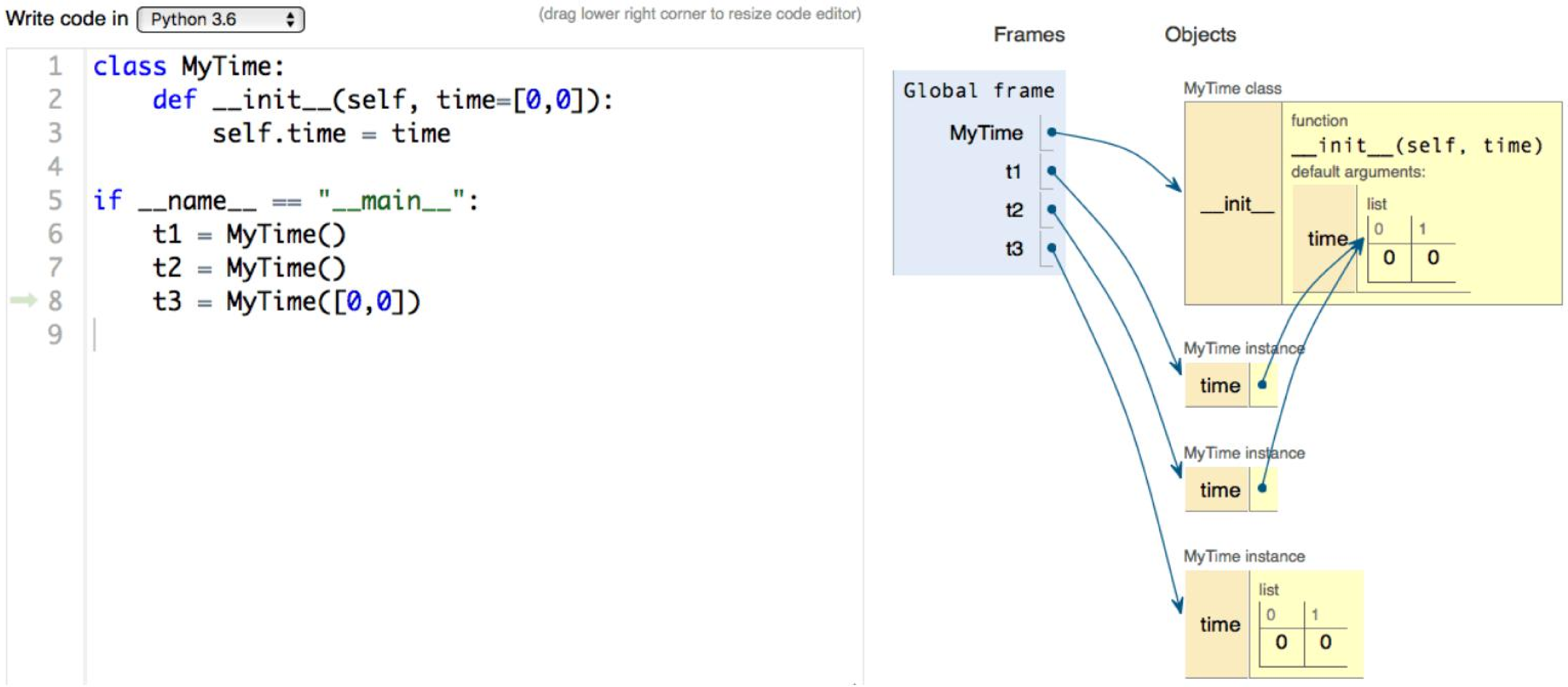
Hodnoty **d0, d1** budou:

A: **True, True**

B: **True, False**

C: **False, False**

vyzkoušejte si sami

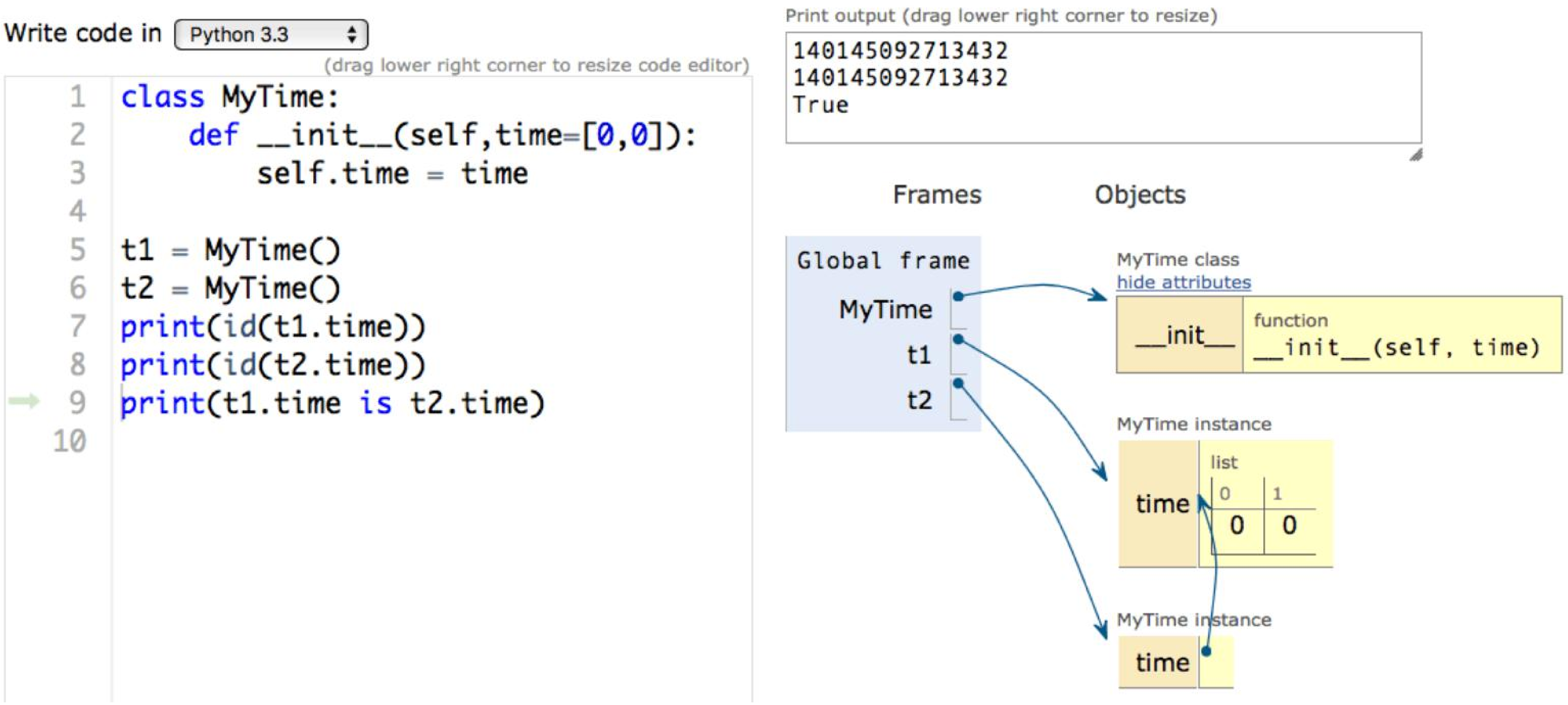


Write code in Python 3.6

(drag lower right corner to resize code editor)

```python
class MyTime:
    def __init__(self, time=[0,0]):
        self.time = time

if __name__ == "__main__":
    t1 = MyTime()
    t2 = MyTime()
    t3 = MyTime([0,0])
```

implicitní parametry detailněji



Write code in Python 3.3

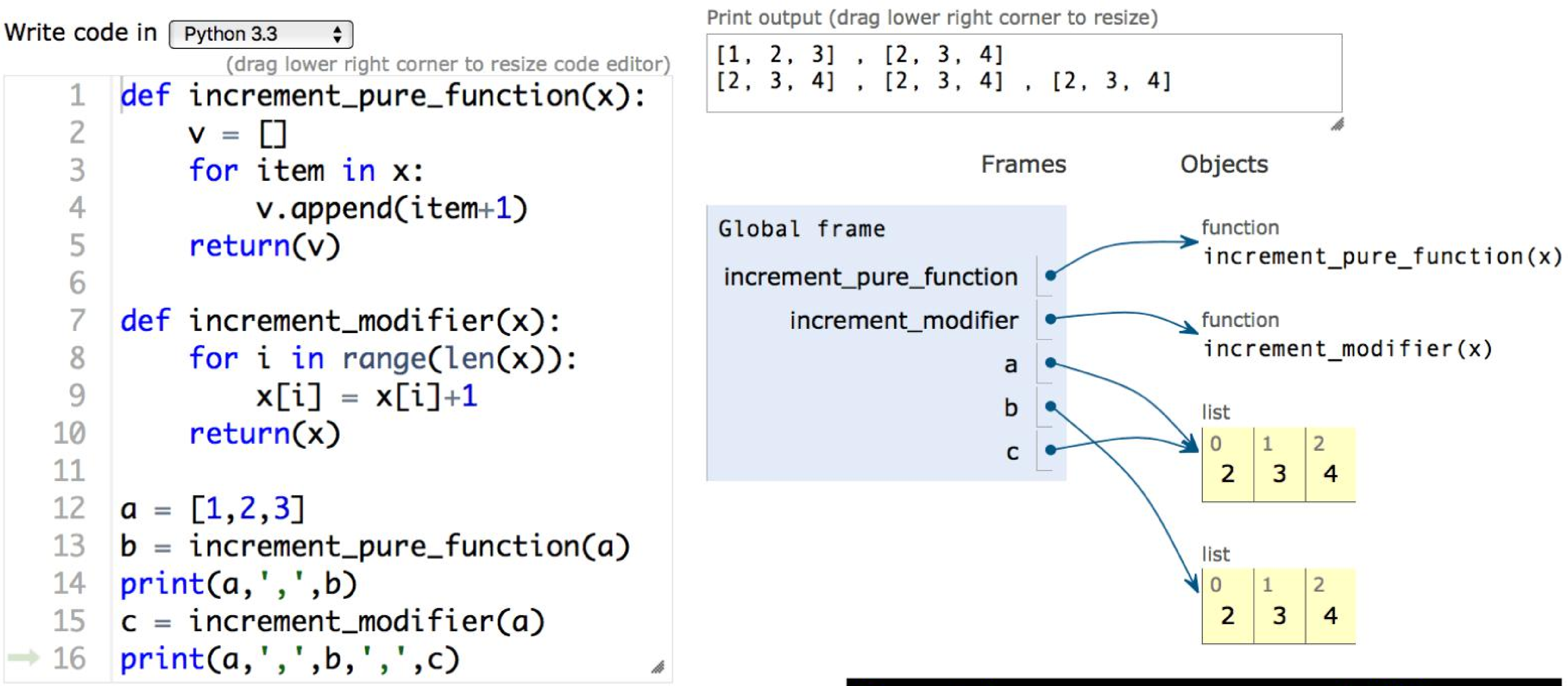
(drag lower right corner to resize code editor)

```python
class MyTime:
    def __init__(self,time=[0,0]):
        self.time = time

t1 = MyTime()
t2 = MyTime()
print(id(t1.time))
print(id(t2.time))
print(t1.time is t2.time)
```

Print output (drag lower right corner to resize)

```
140145092713432
140145092713432
True
```

funkce pravé a modifikátory



Write code in Python 3.3

(drag lower right corner to resize code editor)

```python
def increment_pure_function(x):
    v = []
    for item in x:
        v.append(item+1)
    return(v)

def increment_modifier(x):
    for i in range(len(x)):
        x[i] = x[i]+1
    return(x)

a = [1,2,3]
b = increment_pure_function(a)
print(a,',',b)
c = increment_modifier(a)
print(a,',',b,',',c)
```

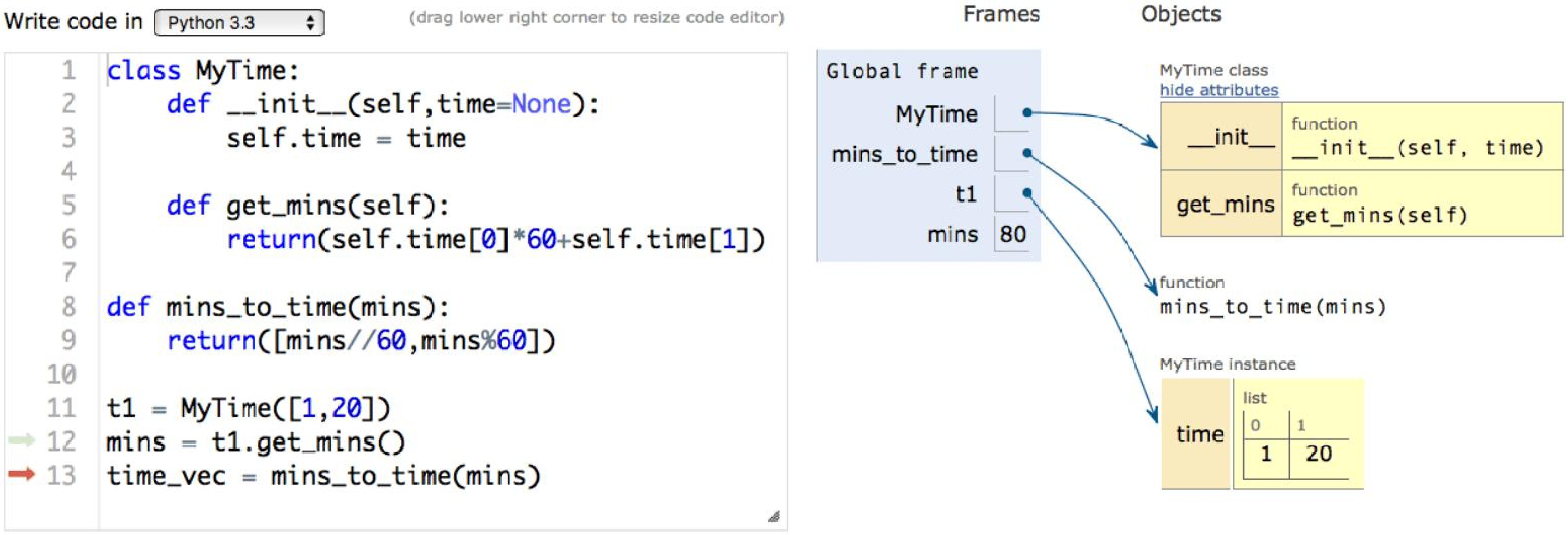
Print output (drag lower right corner to resize)

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

`function_pure_vs_modifier.py`

mytime.py

objekty, třídy a tak

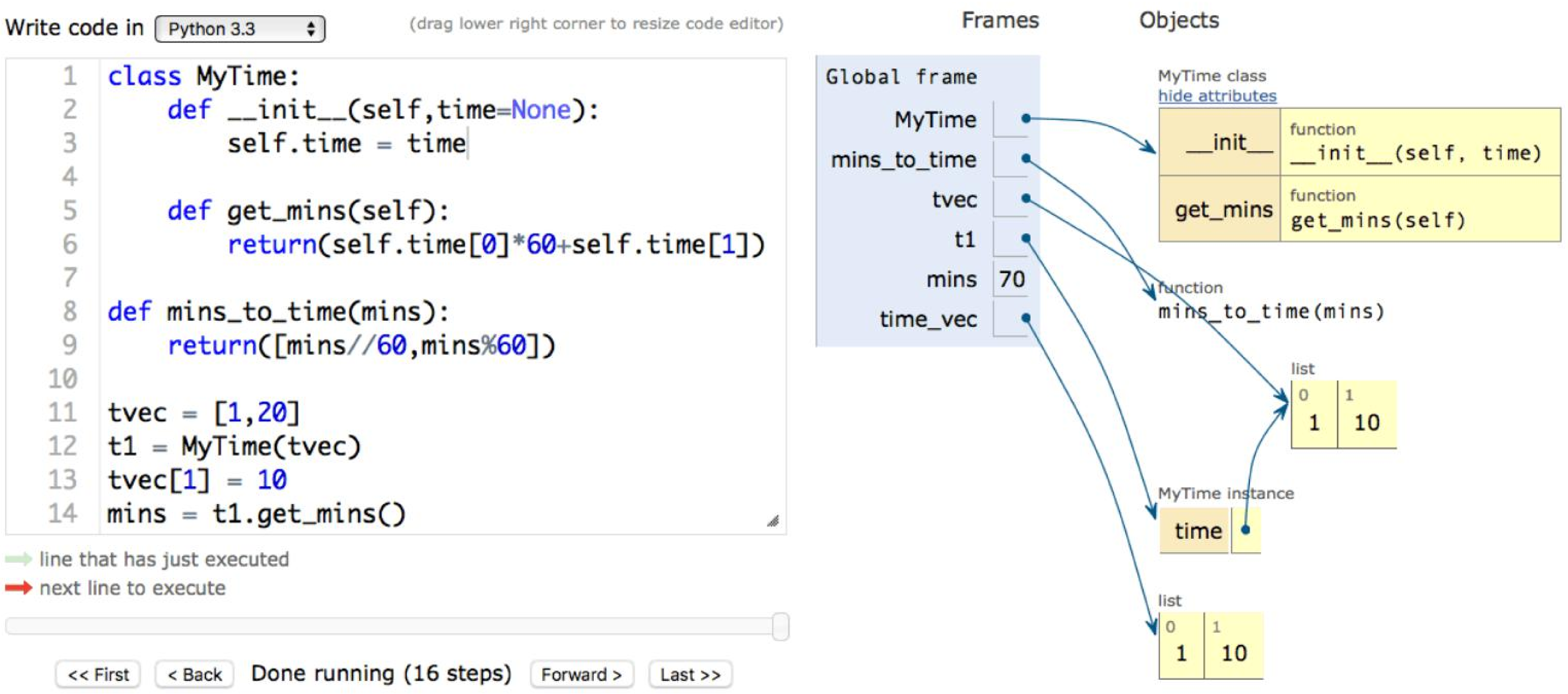


Write code in Python 3.3 (drag lower right corner to resize code editor)

```
class MyTime:
    def __init__(self,time=None):
        self.time = time

    def get_mins(self):
        return(self.time[0]*60+self.time[1])

def mins_to_time(mins):
    return([mins//60,mins%60])

t1 = MyTime([1,20])
mins = t1.get_mins()
time_vec = mins_to_time(mins)
```

visualisation

ale pozor ...



Write code in Python 3.3

(drag lower right corner to resize code editor)

```
class MyTime:
    def __init__(self,time=None):
        self.time = time

    def get_mins(self):
        return(self.time[0]*60+self.time[1])

def mins_to_time(mins):
    return([mins//60,mins%60])

tvec = [1,20]
t1 = MyTime(tvec)
tvec[1] = 10
mins = t1.get_mins()
```

běžte a programujte!

- http://pythontutor.com/visualize.html#mode=edit
- http://openbookproject.net/thinkcs/python/english3e/index.html

Python, základní kameny až skály II

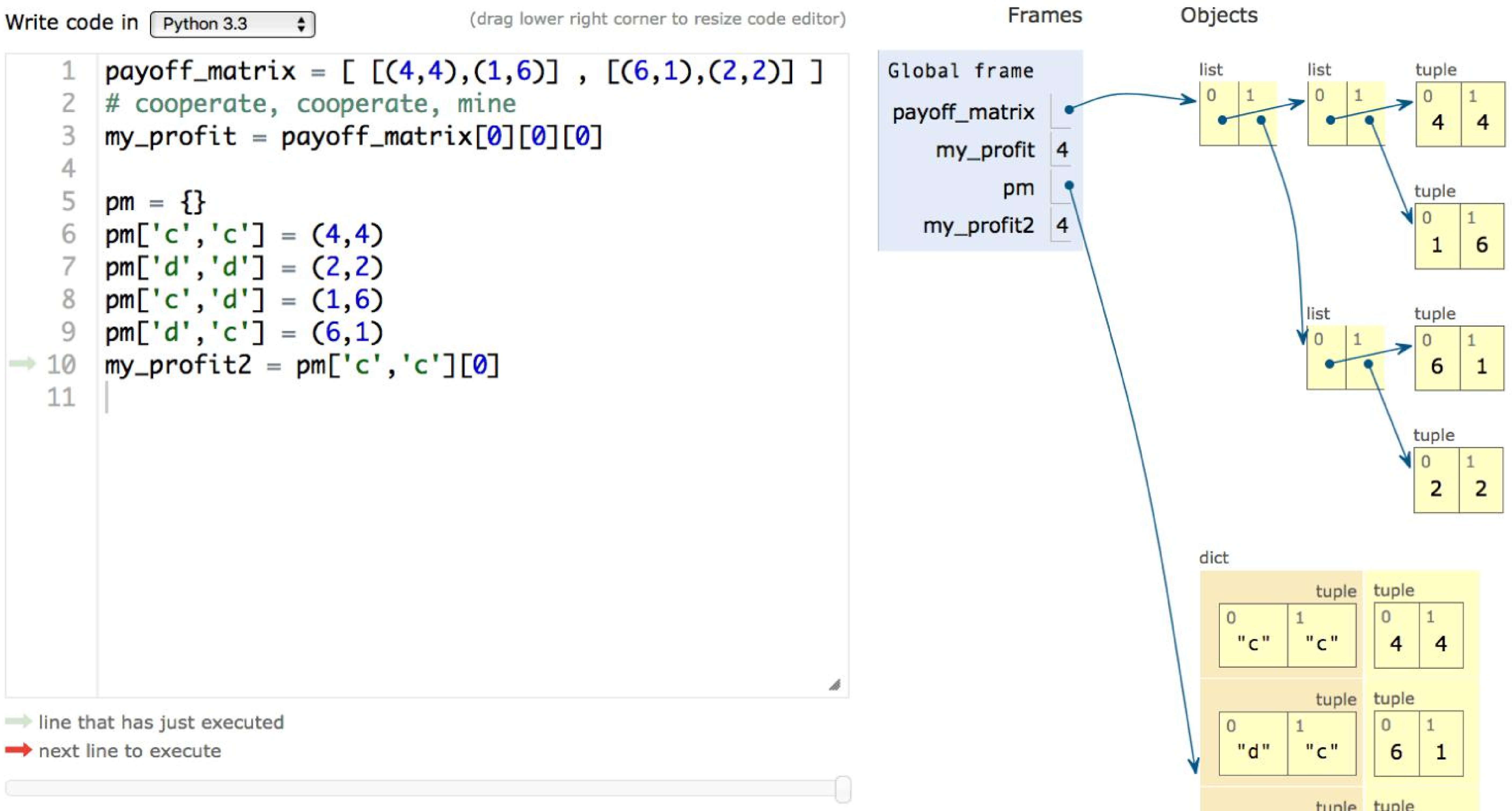
Tomáš Svoboda
B4B33RPH, 2020-10-13

slovníky, dictionary, `dict()`

```python
d = {}
d[1] = 'a'
d[0] = 'b'
d[2] = 'c'

print('for key in d:')
for key in d:
    print(key, d[key])

print('for key, value in d.items():')
for key, value in d.items():
    print(key, value)

print('for key in sorted(d.keys()):')
for key in sorted(d.keys()):
    print(key, d[key])
```

dicts.py

dict - tuples as keys



Write code in Python 3.3

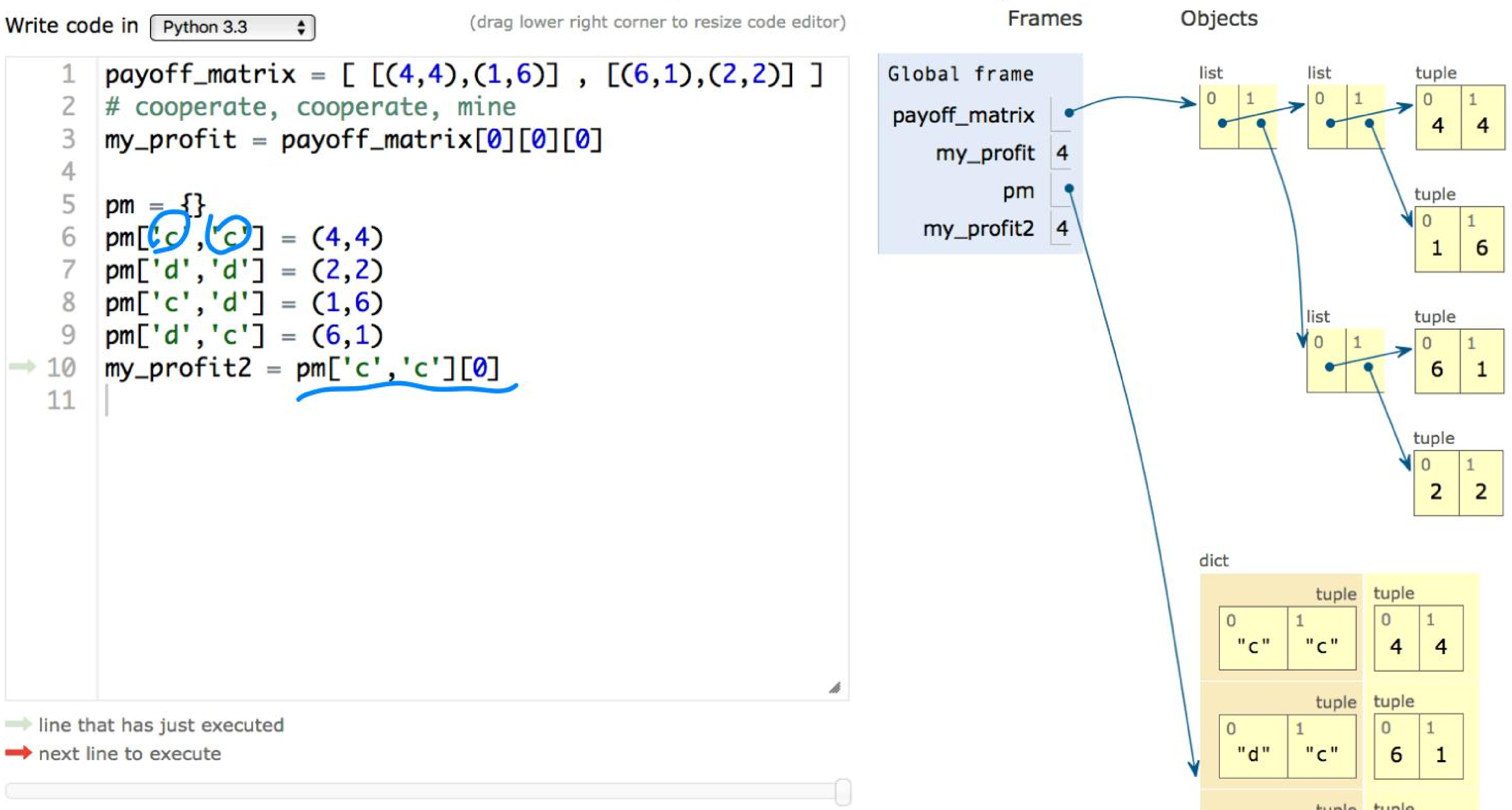
(drag lower right corner to resize code editor)

```python
payoff_matrix = [ [(4,4),(1,6)] , [(6,1),(2,2)] ]
# cooperate, cooperate, mine
my_profit = payoff_matrix[0][0][0]

pm = {}
pm['c','c'] = (4,4)
pm['d','d'] = (2,2)
pm['c','d'] = (1,6)
pm['d','c'] = (6,1)
my_profit2 = pm['c','c'][0]
```

line that has just executed

next line to execute

dict - tuples as keys



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

```
payoff_matrix = [ [(4,4),(1,6)] , [(6,1),(2,2)] ]
# cooperate, cooperate, mine
my_profit = payoff_matrix[0][0][0]

pm = {}
pm['c','c'] = (4,4)
pm['d','d'] = (2,2)
pm['c','d'] = (1,6)
pm['d','c'] = (6,1)
my_profit2 = pm['c','c'][0]

```

line that has just executed
next line to execute

dictionary loops …

```
pm = {}
pm['c','c'] = (4,4)
pm['d','d'] = (2,2)
pm['c','d'] = (1,6)
pm['d','c'] = (6,1)

for key in pm:
    print(key, pm[key])

for key,value in pm.items():
    print(key, value)
```

http://openbookproject.net/thinkcs/python/english3e/dictionaries.html

skládání objektů, dědění

- vylepšíme trochu hráče R-P-S
- ukážeme si na příkladu hráče piškvorek (tic-tac-toe)
- live-coding-session

dictionary loops ...

```
pm = {}
pm['c','c'] = (4,4)
pm['d','d'] = (2,2)
pm['c','d'] = (1,6)
pm['d','c'] = (6,1)

for key in pm:
    print(key, pm[key])

for key,value in pm.items():
    print(key, value)
```

$s = [1, 2, 3]$

$\emptyset \quad 1 \quad 2$

```
for ele in s:
    print(ele)
```

http://openbookproject.net/thinkcs/python/english3e/dictionaries.html

dictionary loops ...

	C	D
C	(4,4)	(1,6)
D		

```python
pm = {}
pm['c','c'] = (4,4)
pm['d','d'] = (2,2)
pm['c','d'] = (1,6)
pm['d','c'] = (6,1)

for key in pm:
    print(key, pm[key])

for key,value in pm.items():
    print(key, value)
```

$$s = \underset{0\ \ 1\ \ 2}{[1,2,3]}$$

```
for ele in s:
    print(ele)
```

http://openbookproject.net/thinkcs/python/english3e/dictionaries.html