Using Git Collaboratively

`libgeometry` Repository
(GitHub)

Commits

Initial Commit

`point.py`

```
class Point:

    def __init__(self, x, y):
        self.x = x
        self.y = y
```

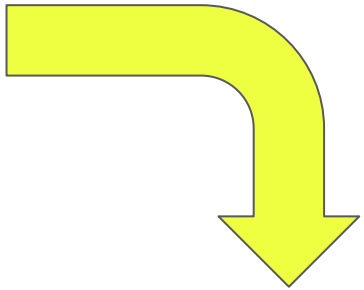
libgeometry Repository
(GitHub)

Commits

Initial Commit

point.py

```
class Point:

    def __init__(self, x, y):
        self.x = x
        self.y = y
```



Repository
(Developer A's computer)

Commits

Initial Commit

point.py

```
class Point:

    def __init__(self, x, y):
        self.x = x
        self.y = y
```

libgeometry Repository
(GitHub)

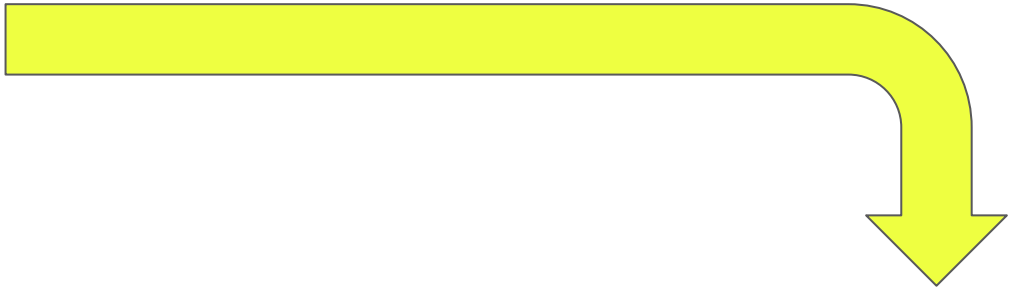
Commits

Initial Commit

point.py

```
class Point:

    def __init__(self, x, y):
        self.x = x
        self.y = y
```

git clone



libgeometry Repository
(Developer A's computer)

Commits

Initial Commit

point.py

```
class Point:

    def __init__(self, x, y):
        self.x = x
        self.y = y
```

libgeometry Repository
(Developer B's computer)

Commits

Initial Commit

point.py

```
class Point:

    def __init__(self, x, y):
        self.x = x
        self.y = y
```

`libgeometry` Repository
(GitHub)

Commits

Initial Commit

`point.py`

```
class Point:

    def __init__(self, x, y):
        self.x = x
        self.y = y
```

`libgeometry` Repository
(Developer A's computer)

Commits

Initial Commit

Added __str__

`point.py`

```
class Point:

    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __str__(self):
        ...
```

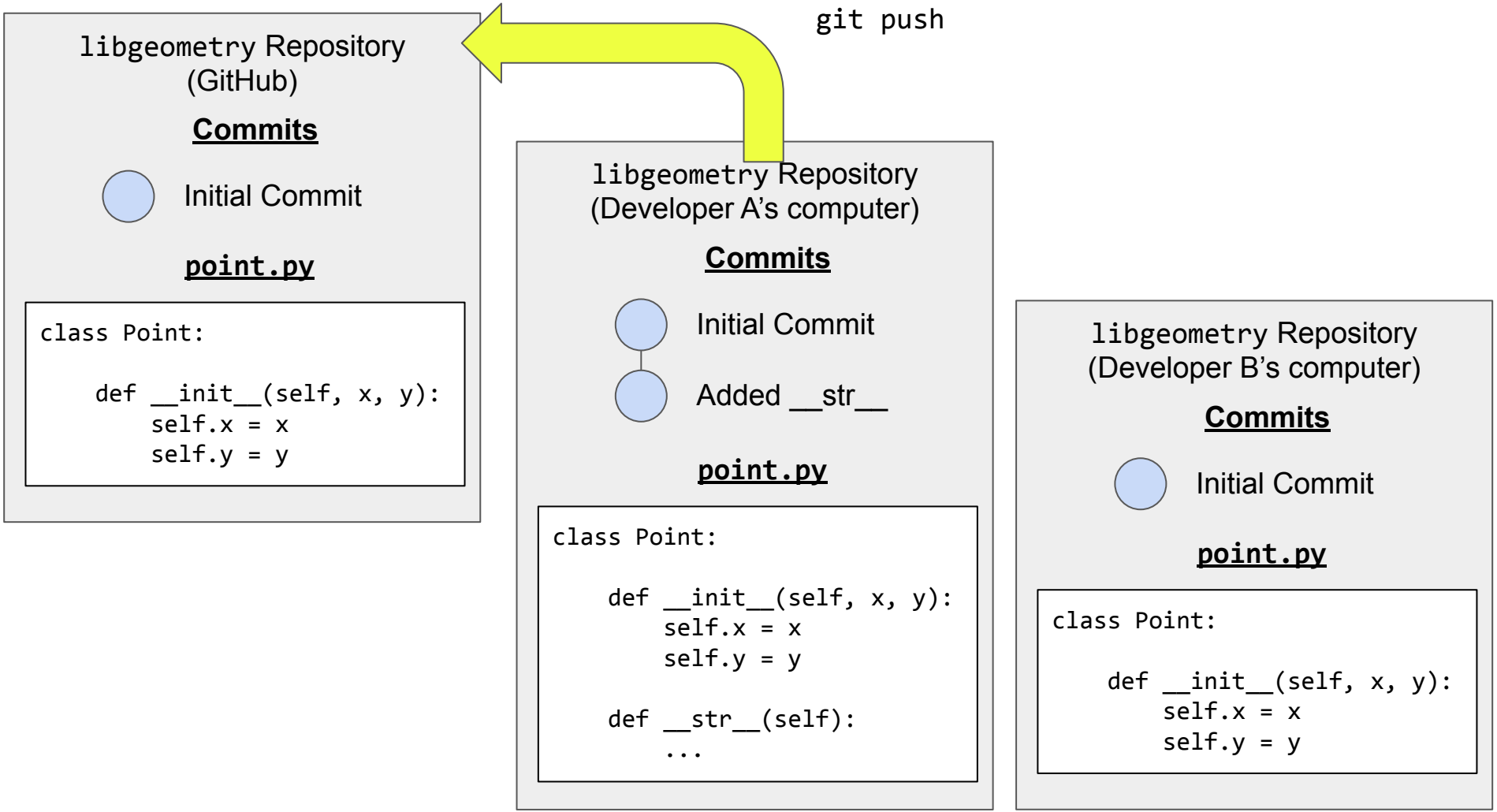
`libgeometry` Repository
(Developer B's computer)

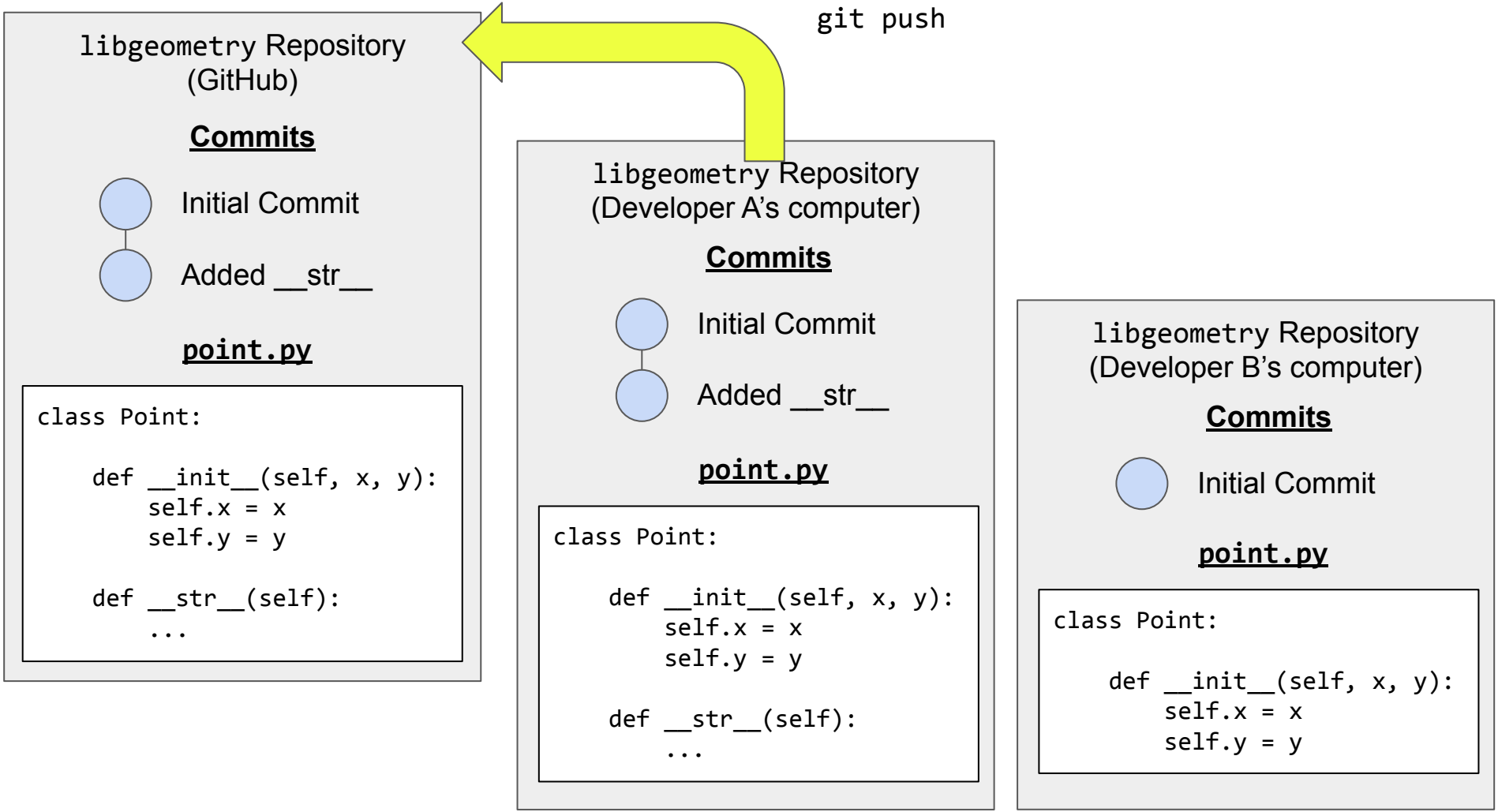
Commits

Initial Commit

`point.py`

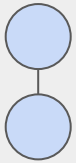
```
class Point:

    def __init__(self, x, y):
        self.x = x
        self.y = y
```


git push
libgeometry Repository
(GitHub)
Commits
Initial Commit
point.py
class Point:
def __init__(self, x, y):
self.x = x
self.y = y
libgeometry Repository
(Developer A's computer)
Commits
Initial Commit
Added __str__
point.py
class Point:
def __init__(self, x, y):
self.x = x
self.y = y
def __str__(self):
...
libgeometry Repository
(Developer B's computer)
Commits
Initial Commit
point.py
class Point:
def __init__(self, x, y):
self.x = x
self.y = y


git push
libgeometry Repository
(GitHub)
Commits
Initial Commit
Added __str__
point.py
class Point:
def __init__(self, x, y):
self.x = x
self.y = y
def __str__(self):
...
libgeometry Repository
(Developer A's computer)
Commits
Initial Commit
Added __str__
point.py
class Point:
def __init__(self, x, y):
self.x = x
self.y = y
def __str__(self):
...
libgeometry Repository
(Developer B's computer)
Commits
Initial Commit
point.py
class Point:
def __init__(self, x, y):
self.x = x
self.y = y

`libgeometry` Repository
(GitHub)

Commits



Initial Commit

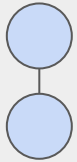
Added __str__

point.py

```
class Point:

    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __str__(self):
        ...
```

`libgeometry` Repository
(Developer A's computer)

Commits



Initial Commit

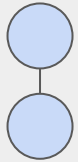
Added __str__

point.py

```
class Point:

    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __str__(self):
        ...
```

`libgeometry` Repository
(Developer B's computer)

Commits

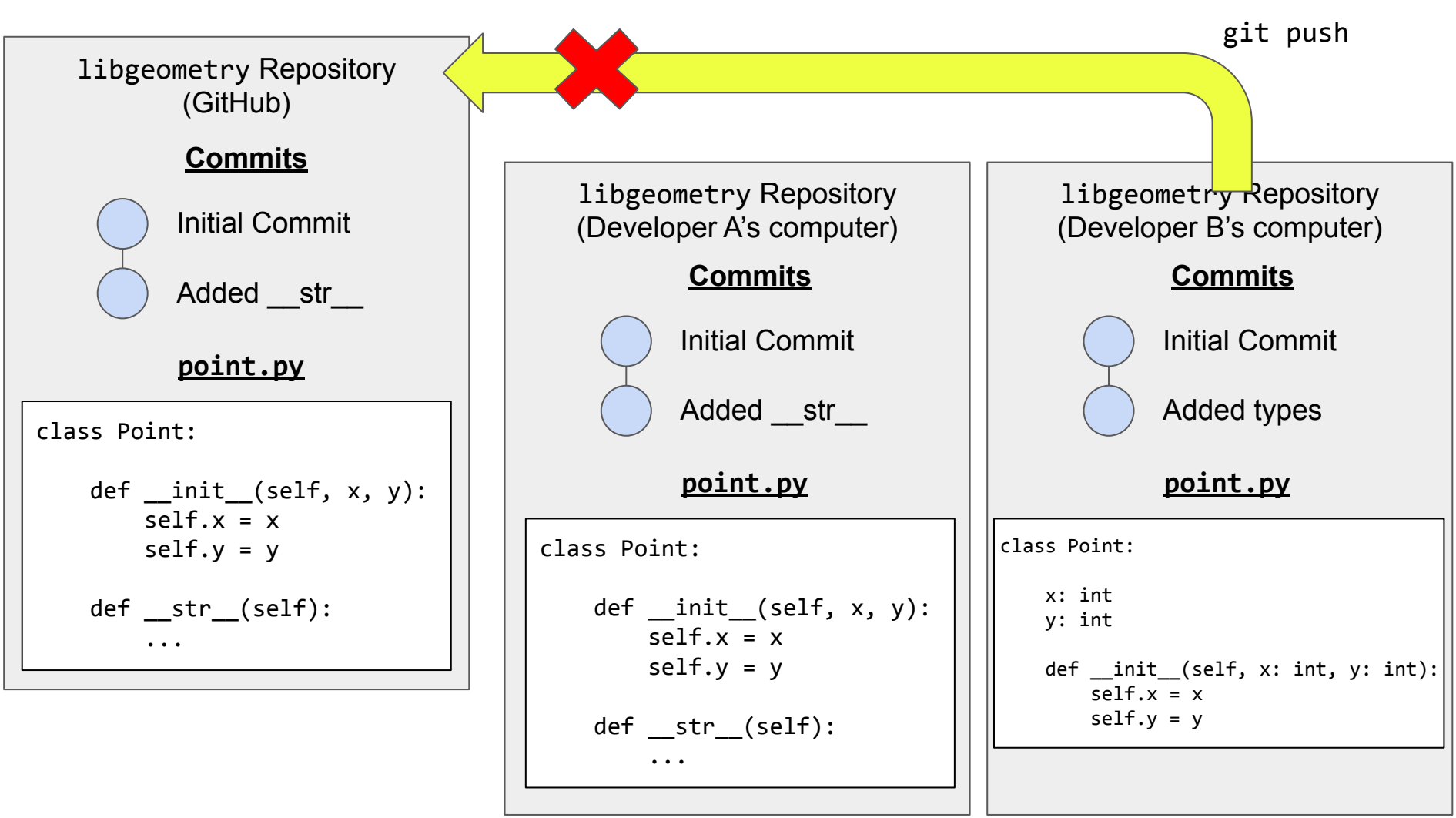


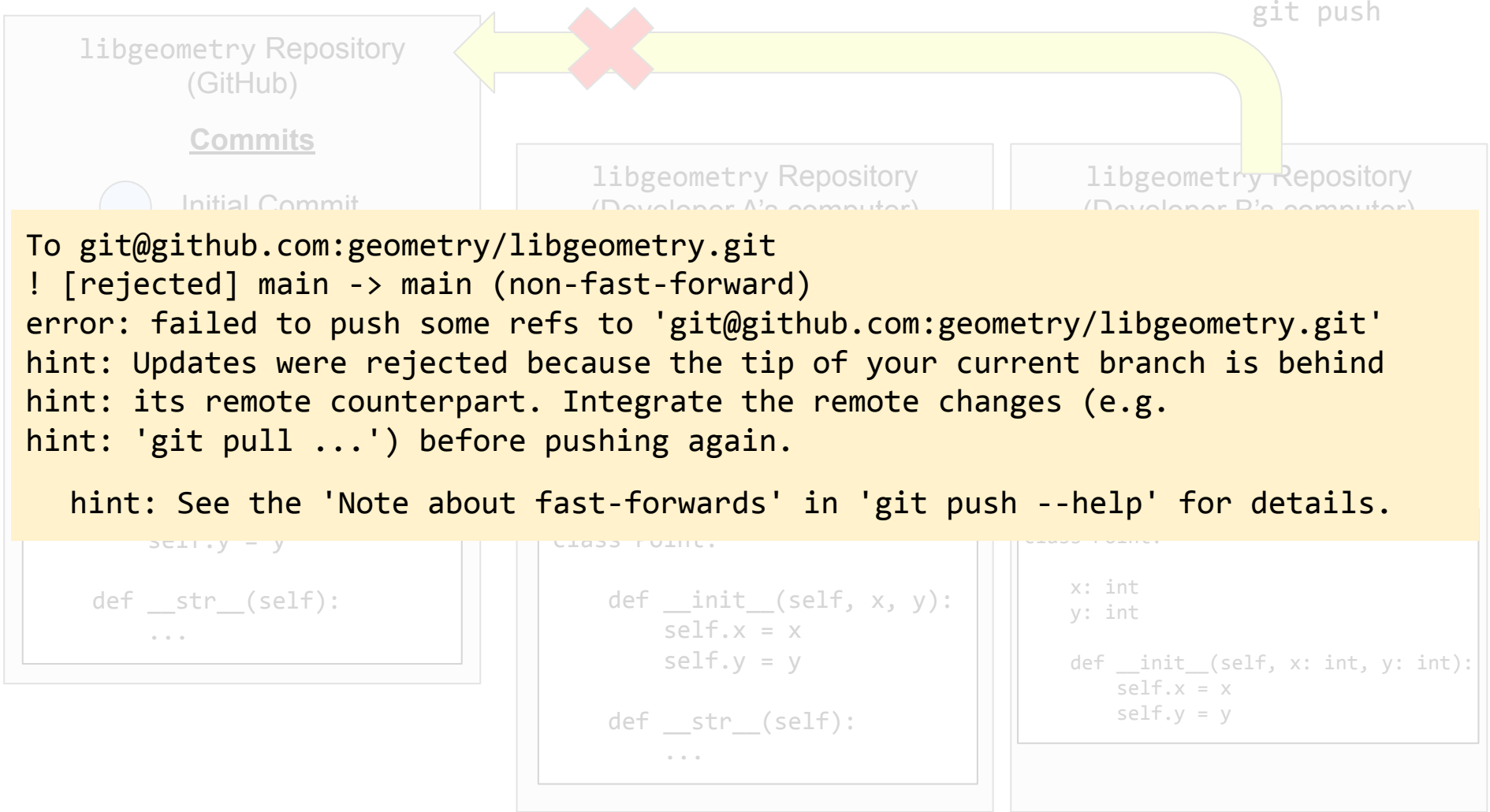
Initial Commit

Added types

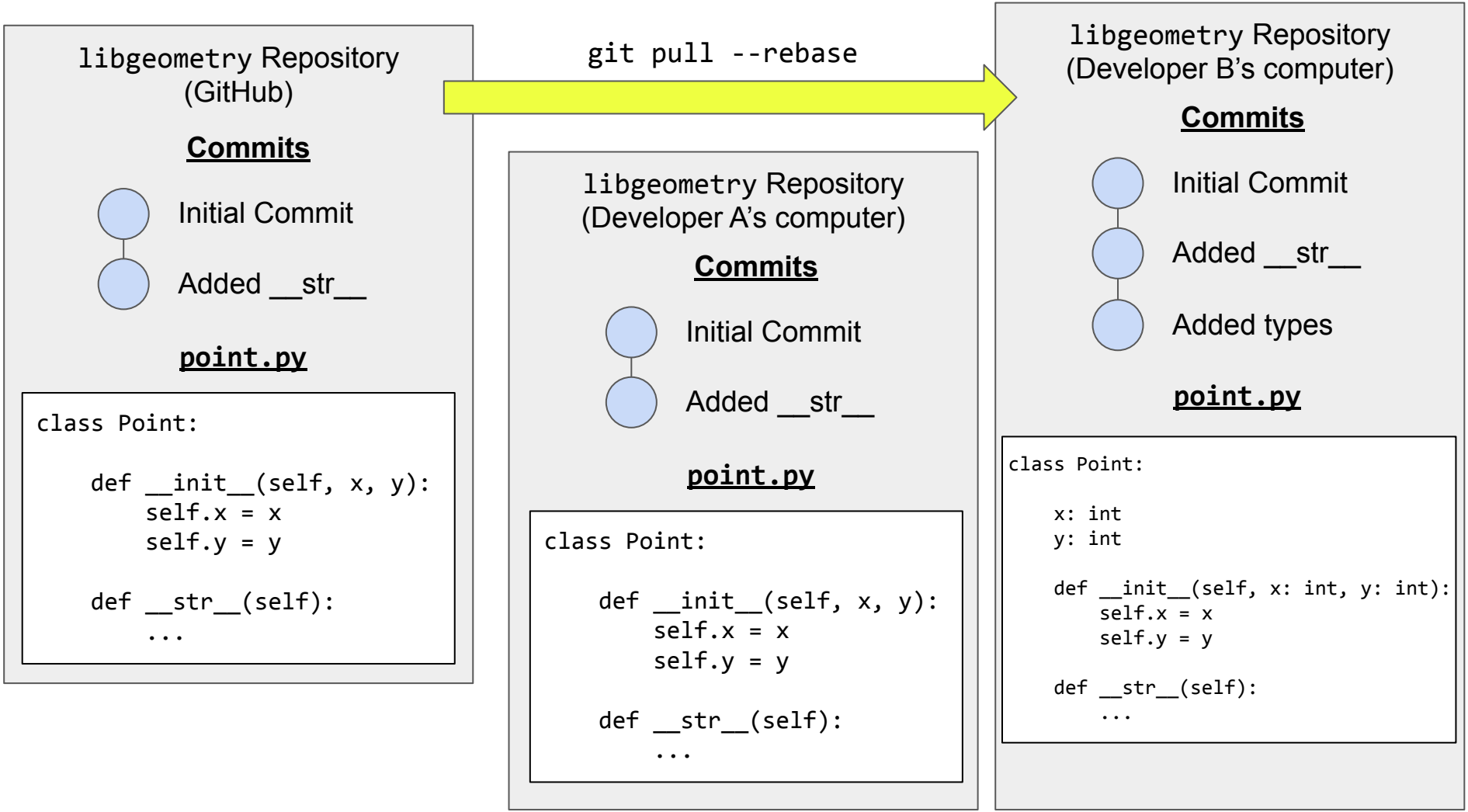
point.py

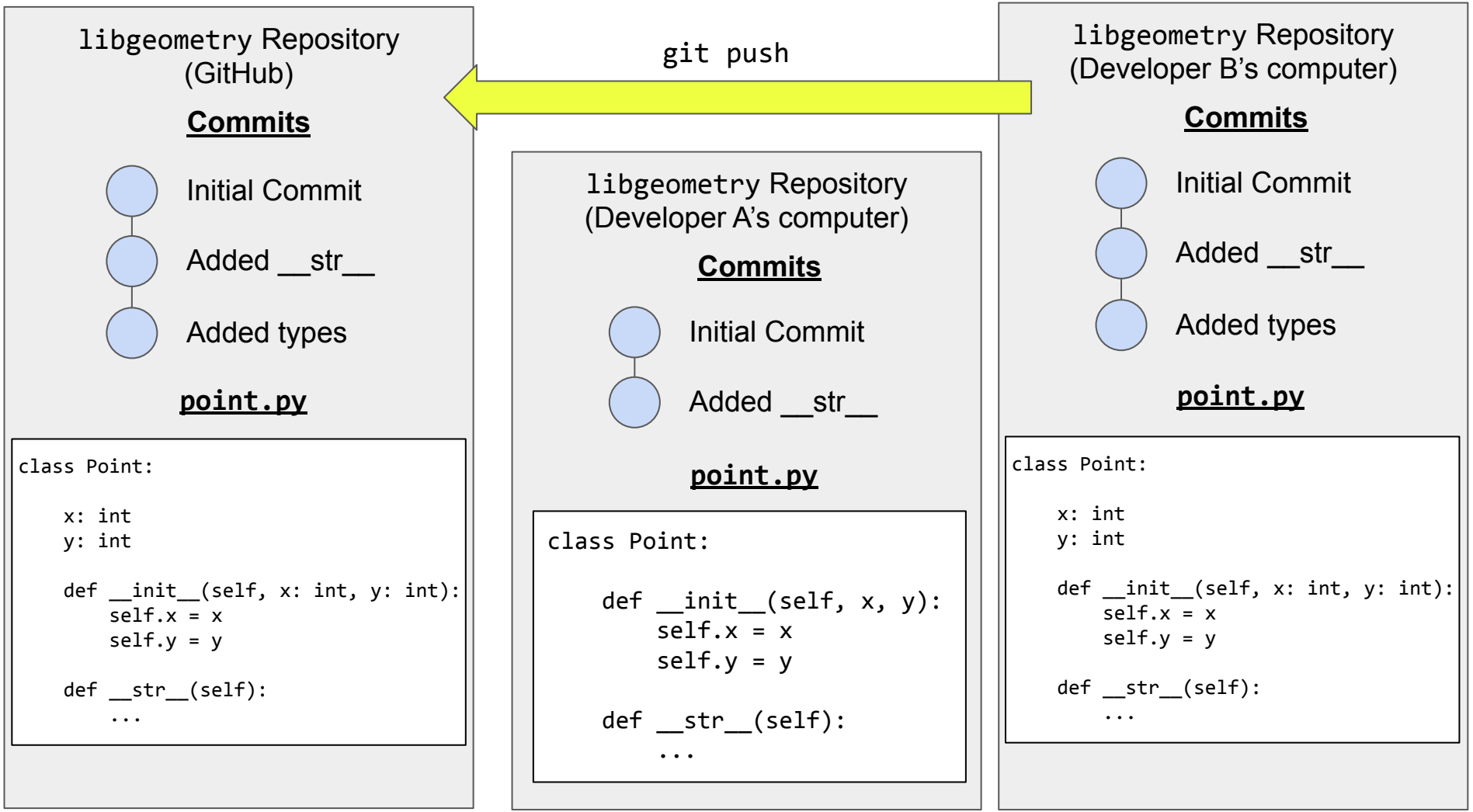
```
class Point:

    x: int
    y: int

    def __init__(self, x: int, y: int):
        self.x = x
        self.y = y
```

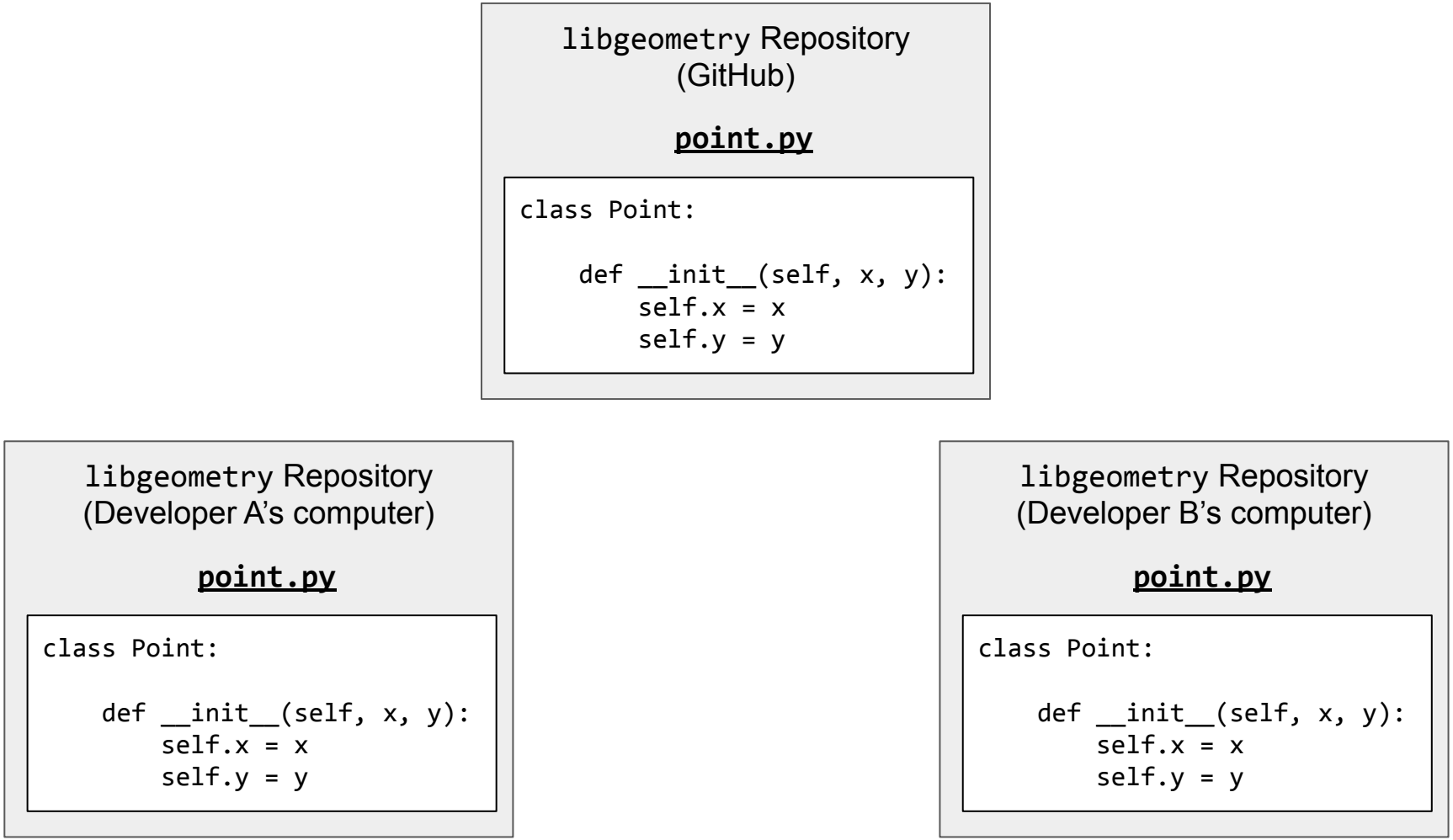

git push
libgeometry Repository
(GitHub)
Commits
Initial Commit
Added __str__
point.py
class Point:
def __init__(self, x, y):
self.x = x
self.y = y
def __str__(self):
...
libgeometry Repository
(Developer A's computer)
Commits
Initial Commit
Added __str__
point.py
class Point:
def __init__(self, x, y):
self.x = x
self.y = y
def __str__(self):
...
libgeometry Repository
(Developer B's computer)
Commits
Initial Commit
Added types
point.py
class Point:
x: int
y: int
def __init__(self, x: int, y: int):
self.x = x
self.y = y



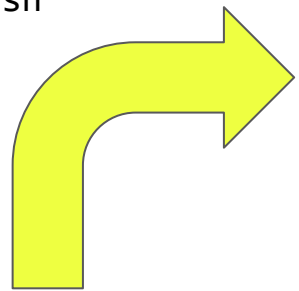

```
To git@github.com:geometry/libgeometry.git
! [rejected] main -> main (non-fast-forward)
error: failed to push some refs to 'git@github.com:geometry/libgeometry.git'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Integrate the remote changes (e.g.
hint: 'git pull ...') before pushing again.

  hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```


libgeometry Repository
(GitHub)
Commits
Initial Commit
Added __str__
point.py
class Point:
def __init__(self, x, y):
self.x = x
self.y = y
def __str__(self):
...
git pull --rebase
libgeometry Repository
(Developer A's computer)
Commits
Initial Commit
Added __str__
point.py
class Point:
def __init__(self, x, y):
self.x = x
self.y = y
def __str__(self):
...
libgeometry Repository
(Developer B's computer)
Commits
Initial Commit
Added __str__
Added types
point.py
class Point:
x: int
y: int
def __init__(self, x: int, y: int):
self.x = x
self.y = y
def __str__(self):
...


libgeometry Repository
(GitHub)
Commits
Initial Commit
Added __str__
Added types
point.py
class Point:
x: int
y: int
def __init__(self, x: int, y: int):
self.x = x
self.y = y
def __str__(self):
...
git push
libgeometry Repository
(Developer A's computer)
Commits
Initial Commit
Added __str__
point.py
class Point:
def __init__(self, x, y):
self.x = x
self.y = y
def __str__(self):
...
libgeometry Repository
(Developer B's computer)
Commits
Initial Commit
Added __str__
Added types
point.py
class Point:
x: int
y: int
def __init__(self, x: int, y: int):
self.x = x
self.y = y
def __str__(self):
...


libgeometry Repository
(GitHub)
point.py
class Point:
def __init__(self, x, y):
self.x = x
self.y = y
libgeometry Repository
(Developer A's computer)
point.py
class Point:
def __init__(self, x, y):
self.x = x
self.y = y
libgeometry Repository
(Developer B's computer)
point.py
class Point:
def __init__(self, x, y):
self.x = x
self.y = y

git push



libgeometry Repository
(GitHub)

point.py

```
class Point:

    def __init__(self, x, y):
        self._x = x
        self._y = y
```

libgeometry Repository
(Developer A's computer)

point.py

```
class Point:

    def __init__(self, x, y):
        self._x = x
        self._y = y
```

libgeometry Repository
(Developer B's computer)

point.py

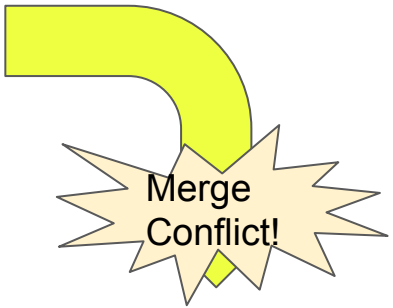
```
class Point:

    def __init__(self, x, y):
        self._px = x
        self._py = y
```

libgeometry Repository
(GitHub)

point.py

```
class Point:

    def __init__(self, x, y):
        self._x = x
        self._y = y
```

git pull



libgeometry Repository
(Developer A's computer)

point.py

```
class Point:

    def __init__(self, x, y):
        self._x = x
        self._y = y
```

libgeometry Repository
(Developer B's computer)

point.py

```
class Point:

    def __init__(self, x, y):
        self._px = x
        self._py = y
```

git pull

```
Auto-merging point.py
CONFLICT (content): Merge conflict in point.py
Automatic merge failed; fix conflicts and then commit the result.
```


git status

```
You have unmerged paths.
(fix conflicts and run "git commit")
(use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
        both modified:   point.py
```

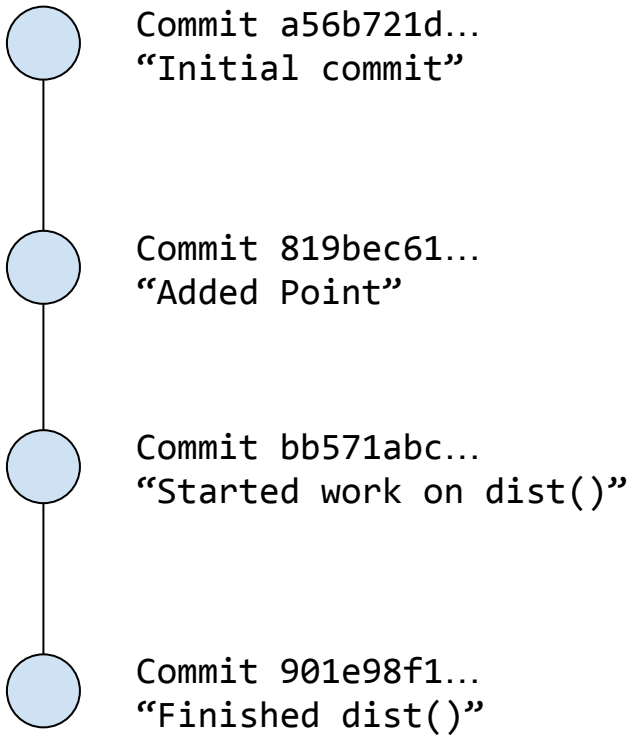
```
class Point:

    def __init__(self, x, y):
        self._x = x
        self._y = y
```

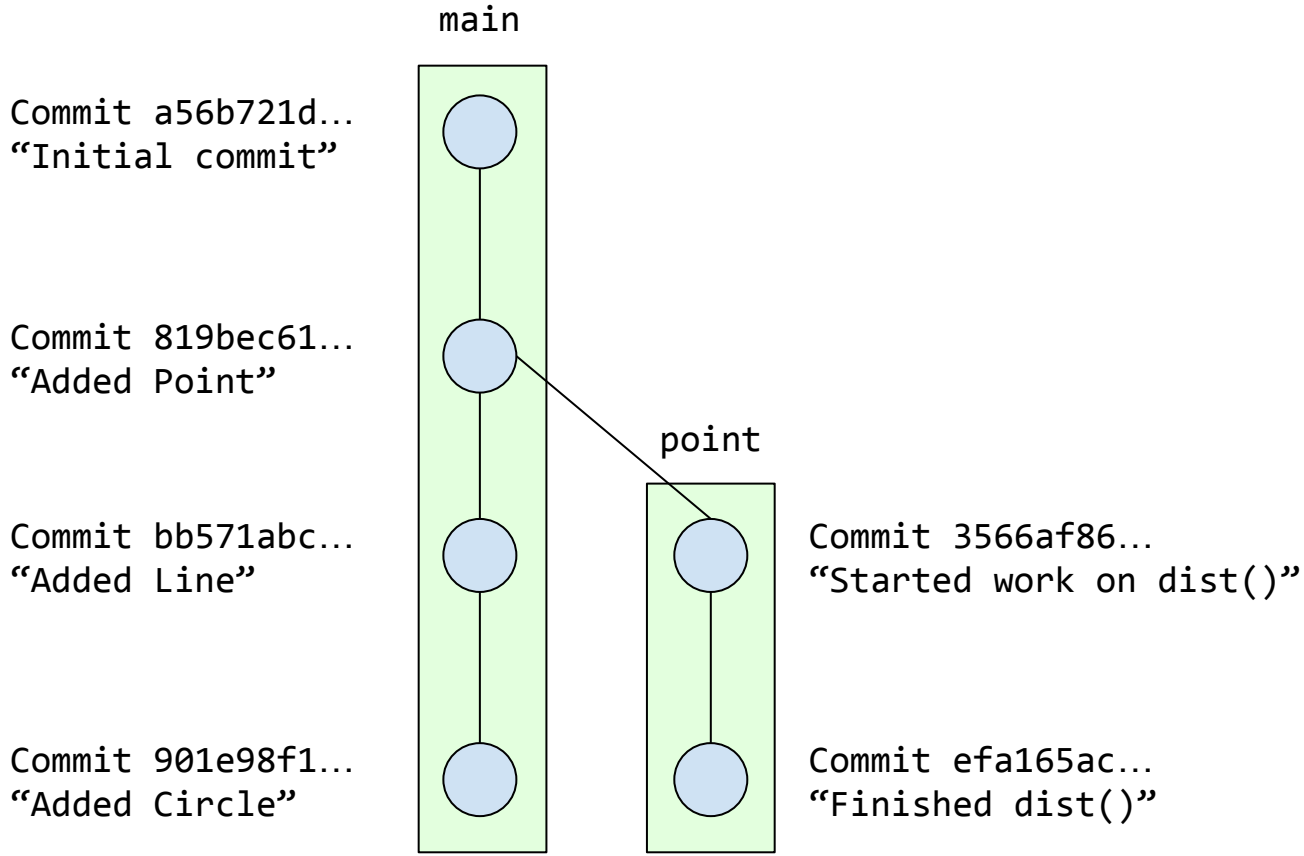
```
class Point:

    def __init__(self, x, y):
        self._px = x
        self._py = y
```

Which one do we keep?

```
class Point:

    def __init__(self, x, y):
<<<<<<< HEAD
        self._px = x
        self._py = y
=======
        self._x = x
        self._y = y
>>>>>>> main
```

Which one do we keep?


Commit a56b721d…
“Initial commit”
Commit 819bec61…
“Added Point”
Commit bb571abc…
“Started work on dist()”
Commit 901e98f1…
“Finished dist()”


main
Commit a56b721d…
“Initial commit”
Commit 819bec61…
“Added Point”
Commit bb571abc…
“Added Line”
Commit 901e98f1…
“Added Circle”
point
Commit 3566af86…
“Started work on dist()”
Commit efa165ac…
“Finished dist()”