Gradual Soundness: Lessons from Static Python

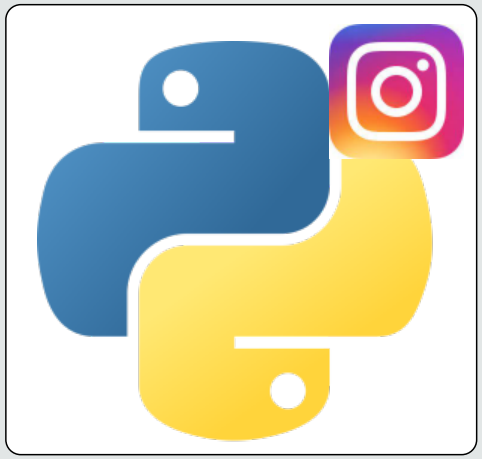
Kuang-Chen Lu **Ben Greenman** Carl Meyer Dino Viehland
Aniket Panse Shriram Krishnamurthi


BROWN

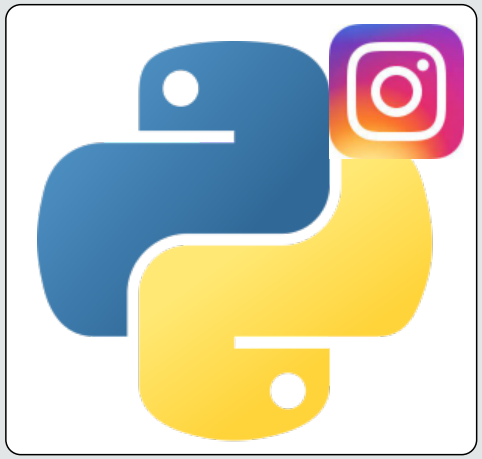
Meta

‹Programming› 2023

Static Python



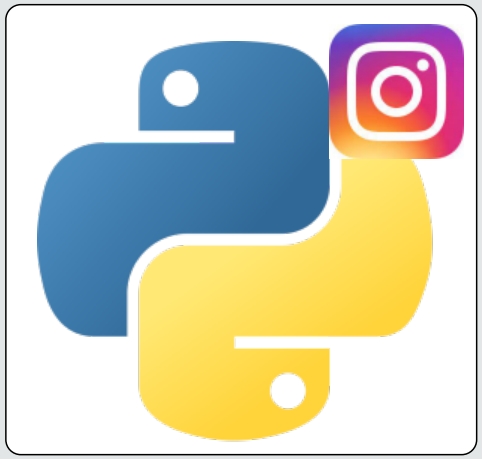
Static Python



Enhanced Python, by Instagram

+2 years running **in production**

Static Python



Enhanced Python, by Instagram

+2 years running **in production**

Gradually typed
... for some value of gradual

What is Gradual Typing?

What is Gradual Typing?

Idea: combine the best parts of typed and untyped code

What is Gradual Typing?

Idea: combine the best parts of typed and untyped code

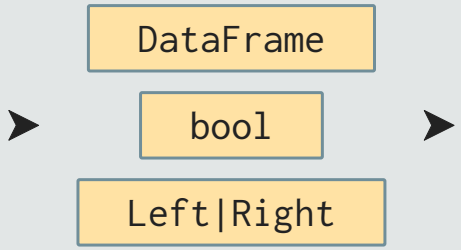
so many parameters!

```
# Python code

def join(d0,d1,sort,how):
  ....
```

What is Gradual Typing?

Idea: combine the best parts of typed and untyped code

```
# Python code

def join(d0,d1,sort,how):
  ....
```



What is Gradual Typing?

Idea: combine the best parts of typed and untyped code

```
# Python code

def join(d0,d1,sort,how):
  ....
```

DataFrame

bool

Left|Right

```
# Python + Types
def join(d0:DataFrame,
         d1:DataFrame,
         sort:bool,
         how:Left|Right)
    -> DataFrame:
  ....
```

What is Gradual Typing?

Idea: combine the best parts of typed and untyped code

```
# Python code

def join(d0,d1,sort
  ....
```

Great!

But, **what happens** when

typed code and untyped code

interact?

Are types sound?

```
# Python + Types
oin(d0:DataFrame,
    d1:DataFrame,
    sort:bool,
    how:Left|Right)
> DataFrame:
.
```

What is Gradual Typing?

What is Gradual Typing?

```
# Python + Types

def join(d0:DataFrame,
         d1:DataFrame,
         sort:bool,
         how:Left|Right)
    -> DataFrame:
  ....
```

A1.

A2.

A3.

What is Gradual Typing?

```
# Python + Types

def join(d0:DataFrame,
         d1:DataFrame,
         sort:bool,
         how:Left|Right)
    -> DataFrame:
  ....
```

A1. Optional static checks, nothing at run-time

A2.

A3.

What is Gradual Typing?

```
# Python + Types

def join(d0:DataFrame,
         d1:DataFrame,
         sort:bool,
         how:Left|Right)
    -> DataFrame:
  ....
```

A1. Optional static checks, nothing at run-time

A2.

A3.

What is Gradual Typing?

```
# Python + Types

def join(d0:DataFrame,
         d1:DataFrame,
         sort:bool,
         how:Left|Right)
    -> DataFrame:
  ....
```

A1. Optional static checks, nothing at run-time

How to debug? `join(42, "hola", ...)`

A2.

A3.

What is Gradual Typing?

```
# Python + Types

def join(d0:DataFrame,
         d1:DataFrame,
         sort:bool,
         how:Left|Right)
    -> DataFrame:
  ....
```

A1. Optional static checks, nothing at run-time

How to debug? `join(42, "hola", ...)`

A2. Static types + contracts

A3.

What is Gradual Typing?

```
# Python + Types

def join(d0:DataFrame,
         d1:DataFrame,
         sort:bool,
         how:Left|Right)
    -> DataFrame:
  ....
```

A1. Optional static checks, nothing at run-time

How to debug? `join(42, "hola", ...)`

A2. Static types + contracts

A3.

What is Gradual Typing?

```
# Python + Types

def join(d0:DataFrame,
         d1:DataFrame,
         sort:bool,
         how:Left|Right)
    -> DataFrame:
  ....
```

A1. Optional static checks, nothing at run-time

How to debug? `join(42, "hola", ...)`

A2. Static types + contracts

Performance? `join(huge0, huge1, ...)`

A3.

What is Gradual Typing?

```
# Python + Types

def join(d0:DataFrame,
         d1:DataFrame,
         sort:bool,
         how:Left|Right)
    -> DataFrame:
  ....
```

A1. Optional static checks, nothing at run-time

How to debug? `join(42, "hola", ...)`

A2. Static types + contracts

Performance? `join(huge0, huge1, ...)`

A3. Progressive static types + tags

Today!

A3. Progressive static types + tags

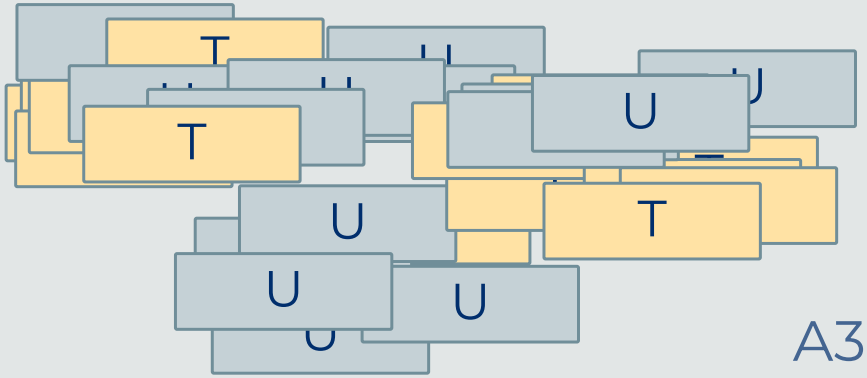
Experience @ Instagram Web Server

A3. Progressive static types + tags

Experience @ Instagram Web Server

+500 modules with **sound types**
(upgraded from Pyre)

+30k T — U interactions

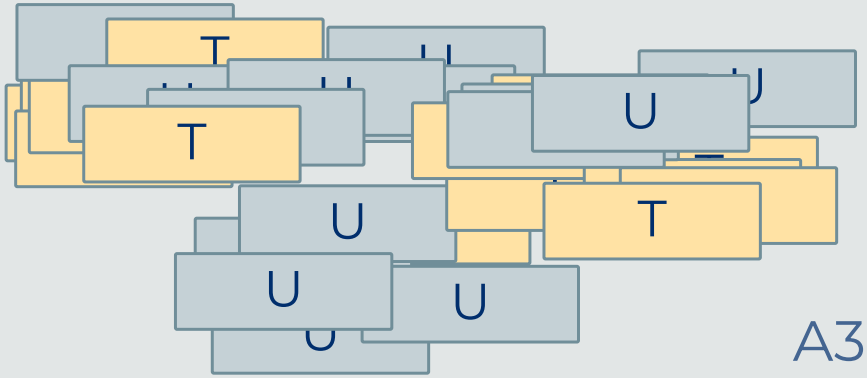


A3. Progressive static types + tags

Experience @ Instagram Web Server

+500 modules with **sound types**
(upgraded from Pyre)

+30k T — U interactions



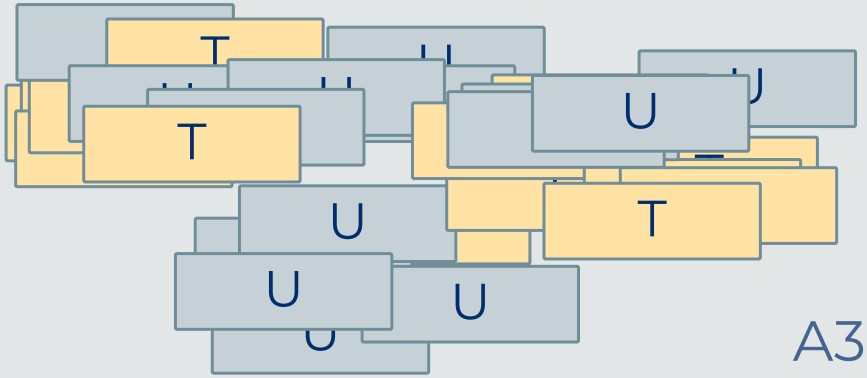
3.9% increase in CPU efficiency

A3. Progressive static types + tags

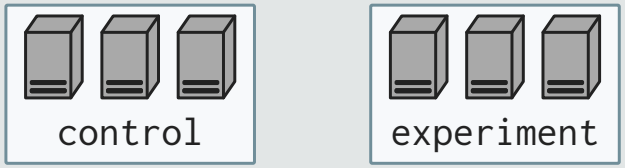
Experience @ Instagram Web Server

+500 modules with **sound types**
(upgraded from Pyre)

+30k T — U interactions



3.9% increase in CPU efficiency

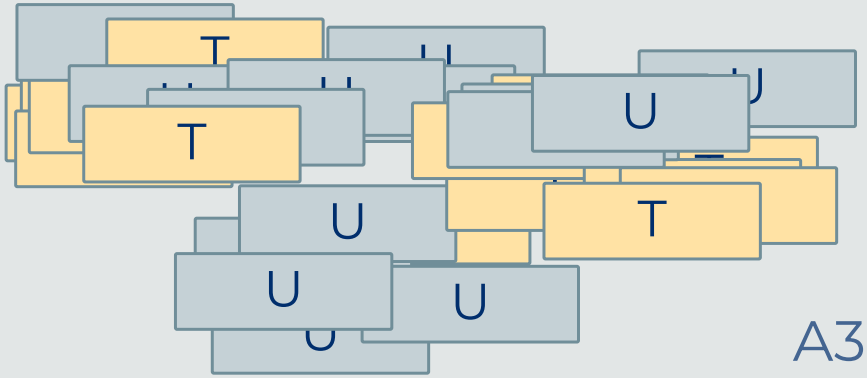


A3. Progressive static types + tags

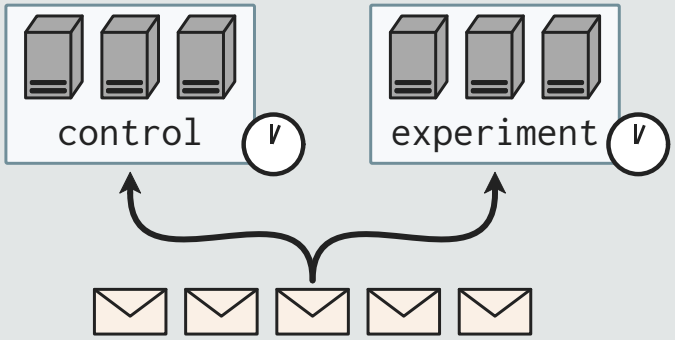
Experience @ Instagram Web Server

+500 modules with **sound types**
(upgraded from Pyre)

+30k T — U interactions



3.9% increase in CPU efficiency



A3. Progressive static types + tags

How is Static Python so Fast?

Step 0. Better Compiler & Runtime

Step 0. Better Compiler & Runtime

https://github.com/facebookincubator/cinder

Step 0. Better Compiler & Runtime

https://github.com/facebookincubator/cinder

Cinder Runtime

V Tables
Method-based JIT
...

Step 0. Better Compiler & Runtime

https://github.com/facebookincubator/cinder

Cinder Runtime

V Tables
Method-based JIT
…

Type-Aware Bytecode

`CALL_FUNCTION`	Python default
`INVOKE_METHOD`	V Table lookup
`INVOKE_FUNCTION`	direct call

Step 1. Fast Soundness Checks

Step 1. Fast Soundness Checks

```
avg(nums)
```

```
def avg(ns:chklist[int]) -> int:
  ....
```

Step 1. Fast Soundness Checks

```
avg(nums)
```

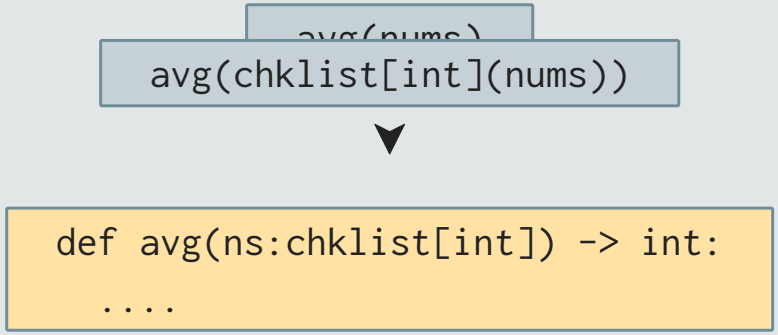
```
def avg(ns:chklist[int]) -> int:
  ....
```

Q. How to enforce soundness?

A. **Tag check**

Is `nums` an instance of `chklist[int]` ?

Step 1. Fast Soundness Checks



Q. How to enforce soundness?

A. **Tag check**

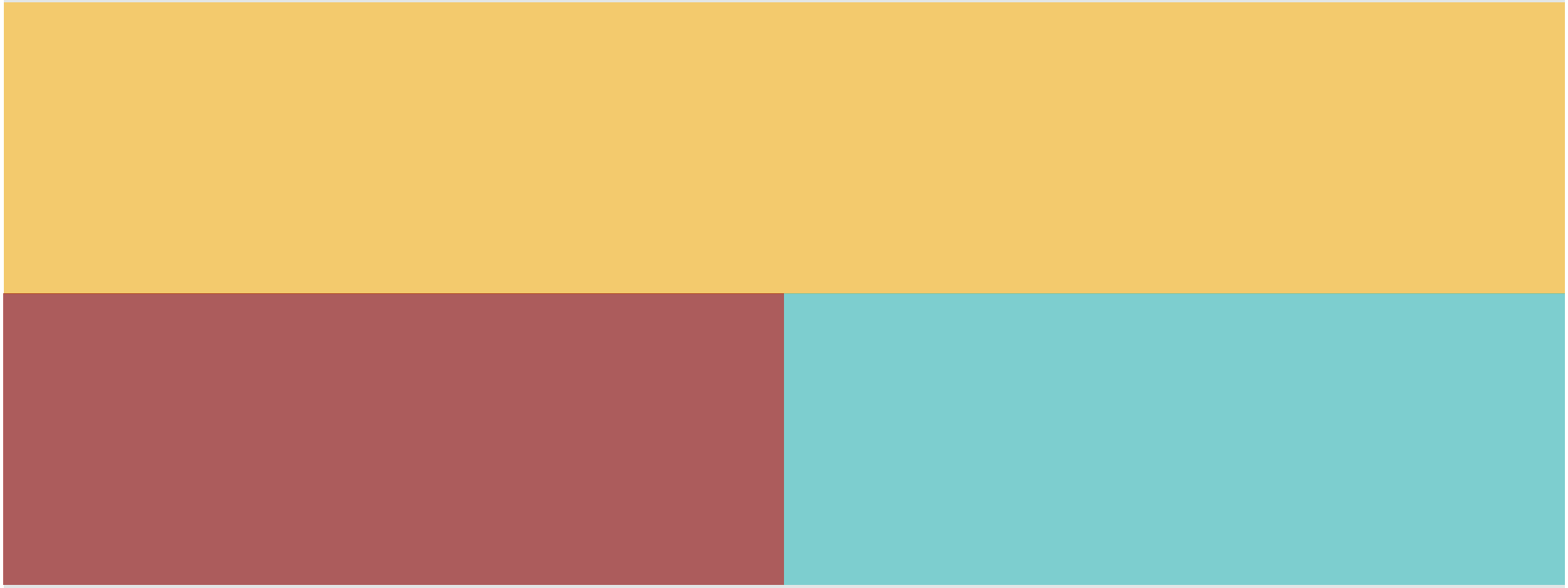
Is `nums` an instance of `chklist[int]` ?

✔ Fast! No traversal, no wrapper

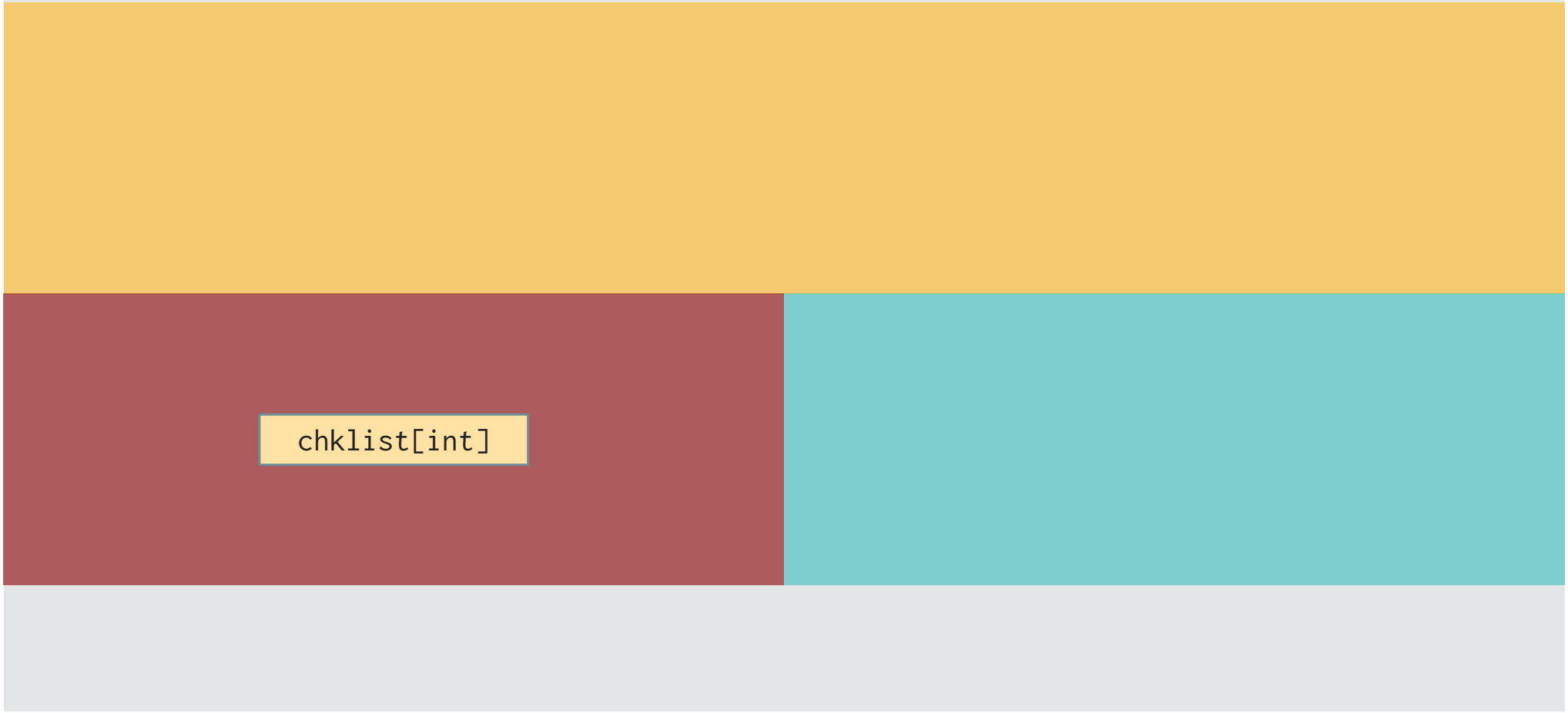
■ Rejects built-in Python lists

Step 2. Progressive Types

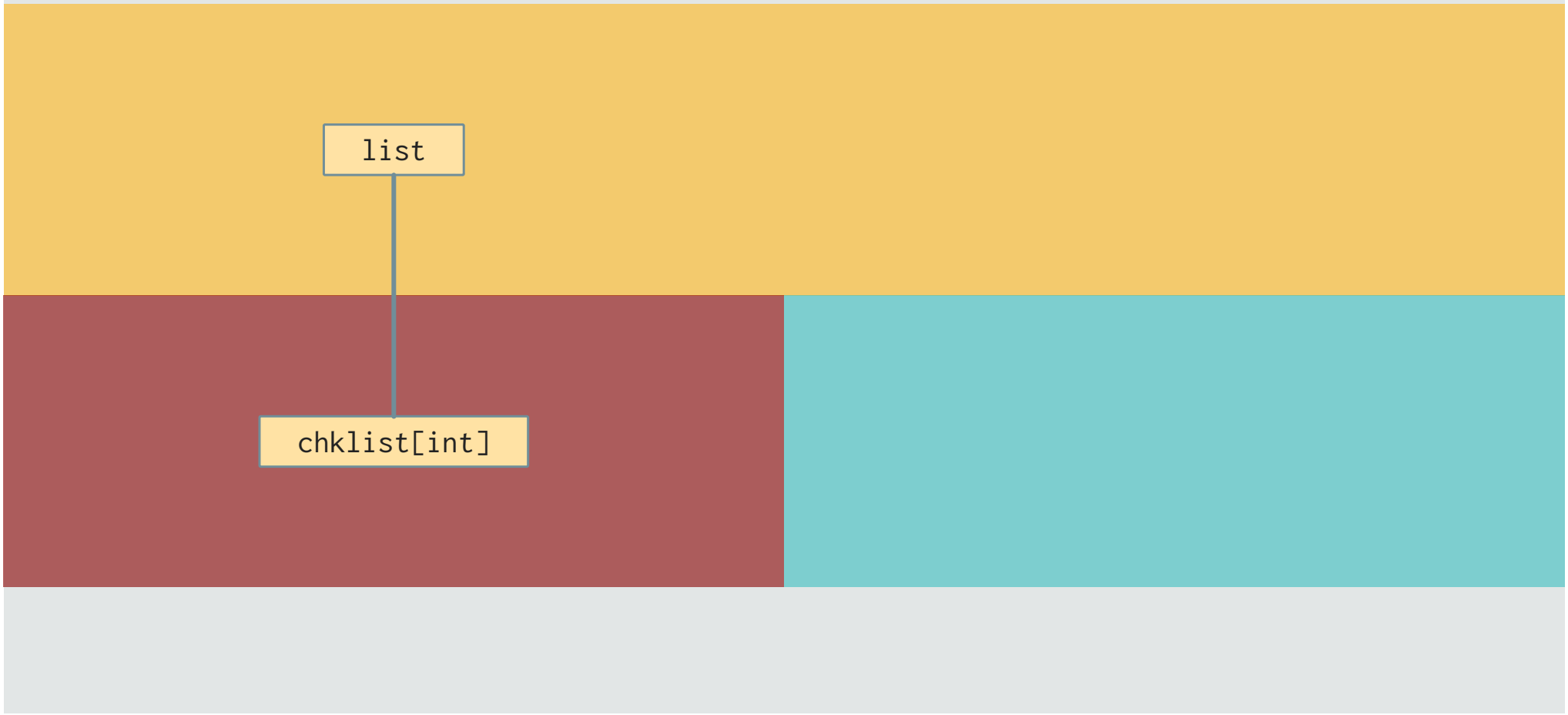
Step 2. Progressive Types



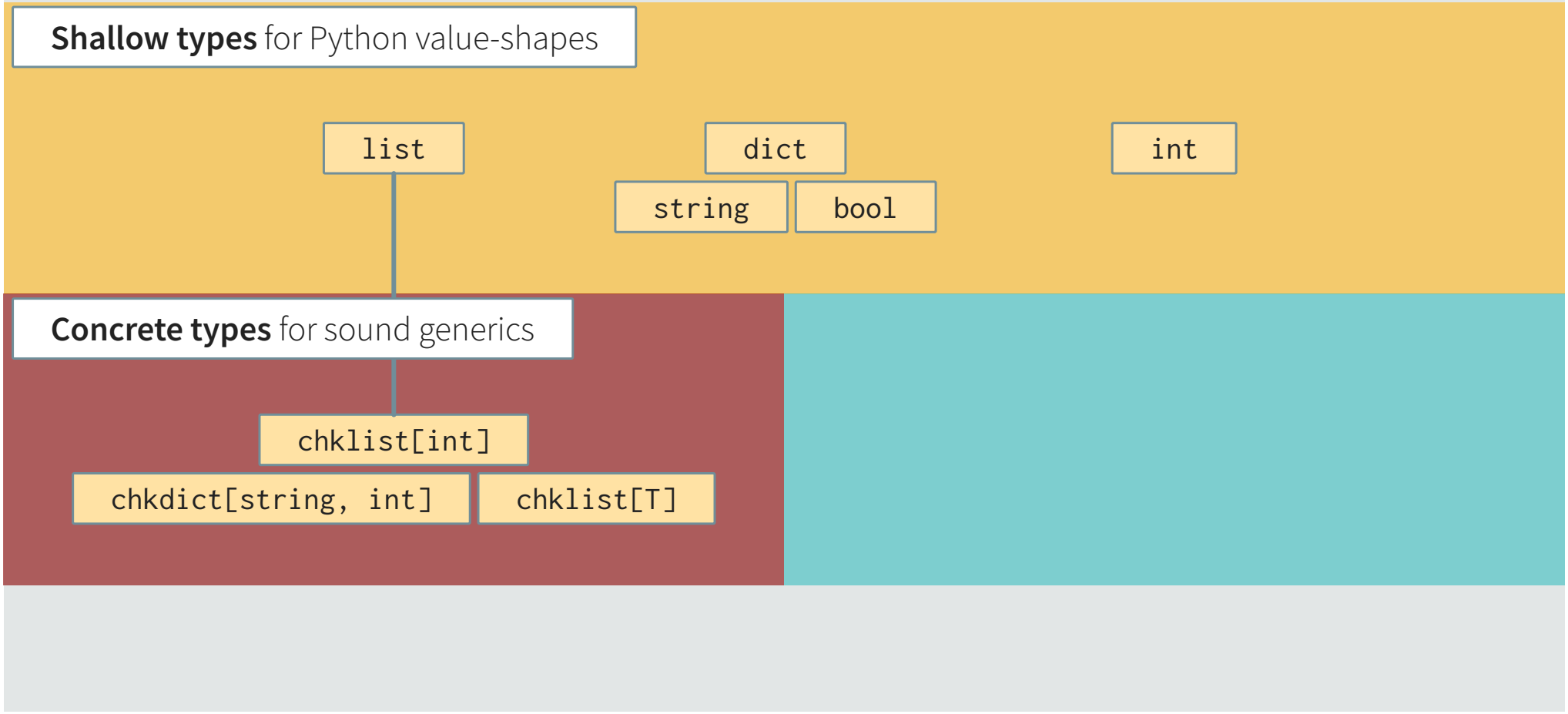
Step 2. Progressive Types



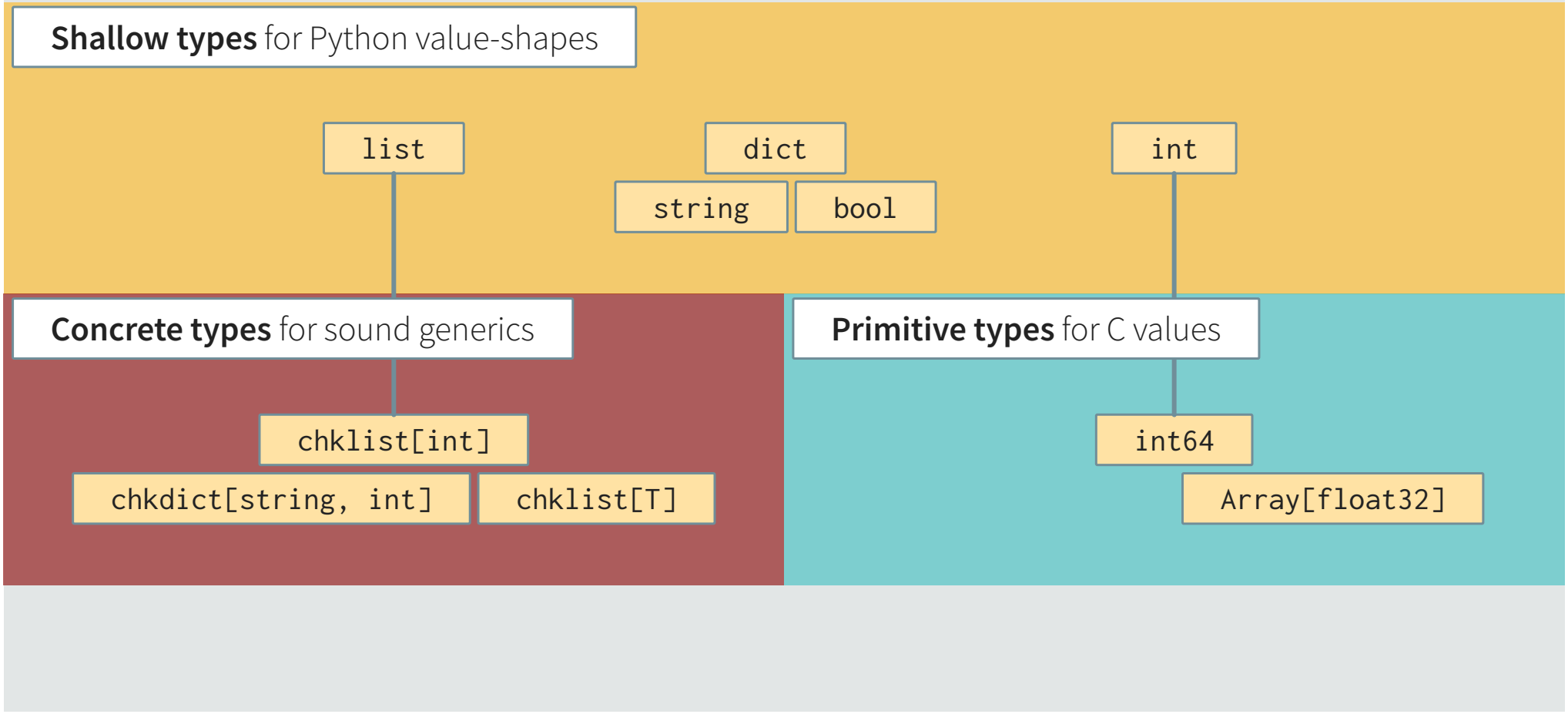
Step 2. Progressive Types



Step 2. Progressive Types

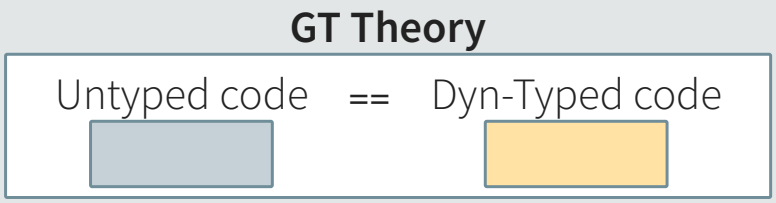


Step 2. Progressive Types



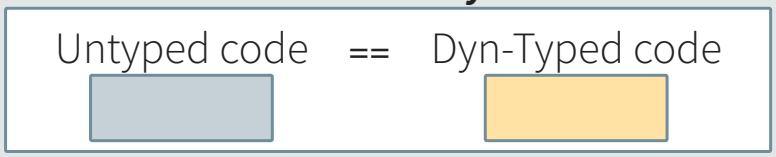
Step 3. Limited Dyn Type

Step 3. Limited Dyn Type

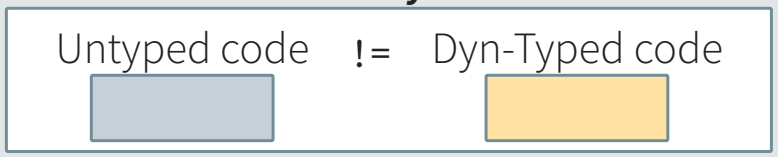


Step 3. Limited Dyn Type

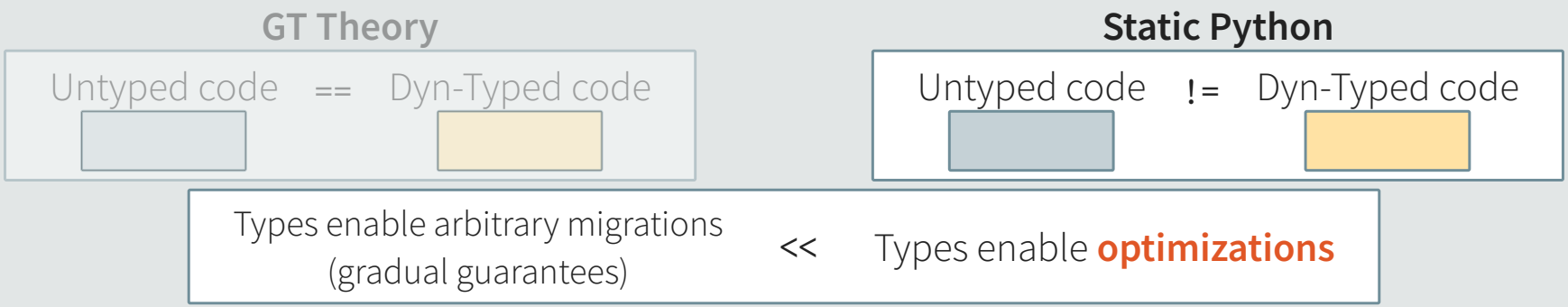
GT Theory



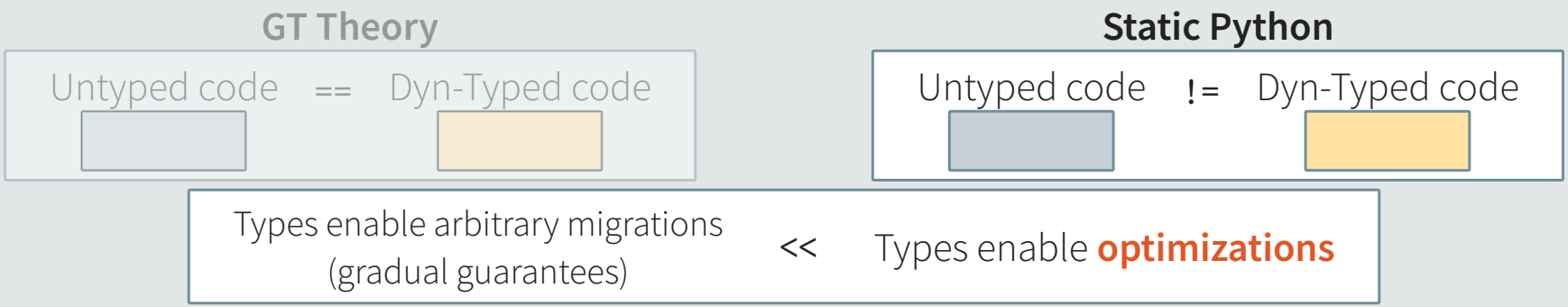
Static Python

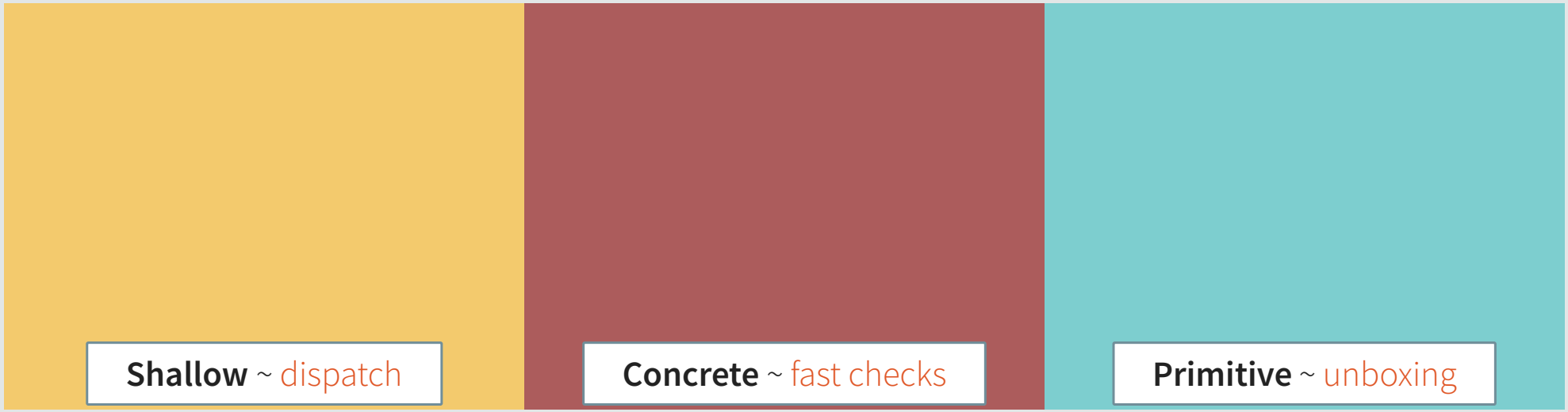


Step 3. Limited Dyn Type

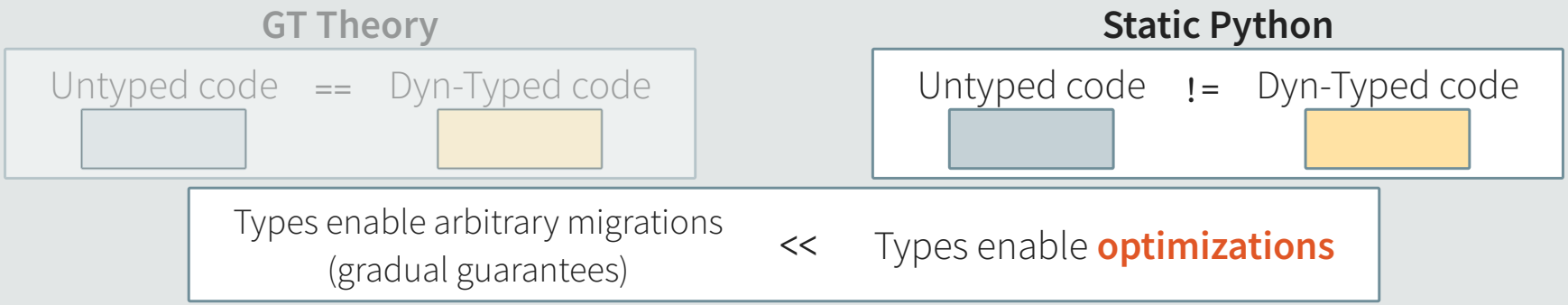


Step 3. Limited Dyn Type





Step 3. Limited Dyn Type




```
class A:
 def f(self)->int:
```

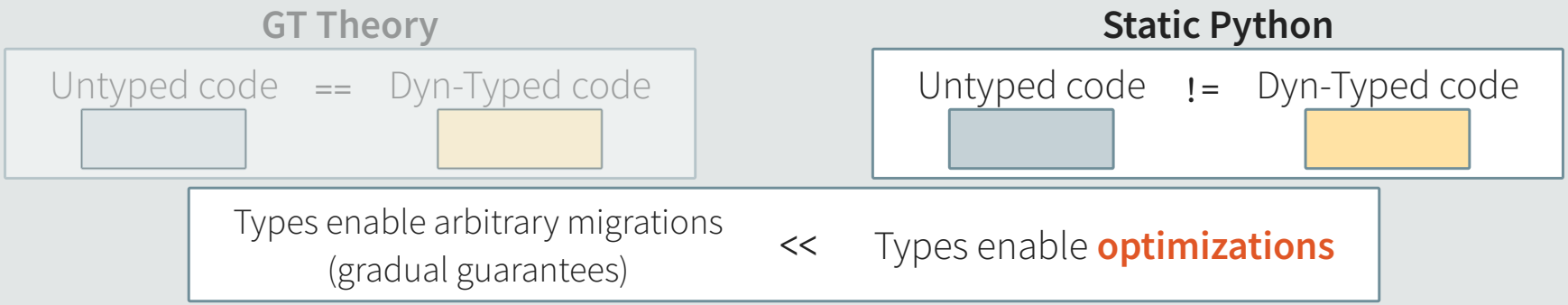
```
class B(A):
 def f(self):
   # Type Error
```

Shallow ~ dispatch

Concrete ~ fast checks

Primitive ~ unboxing

Step 3. Limited Dyn Type




```
class A:
 def f(self)->int:
```

```
class B(A):
 def f(self):
  # Type Error
```

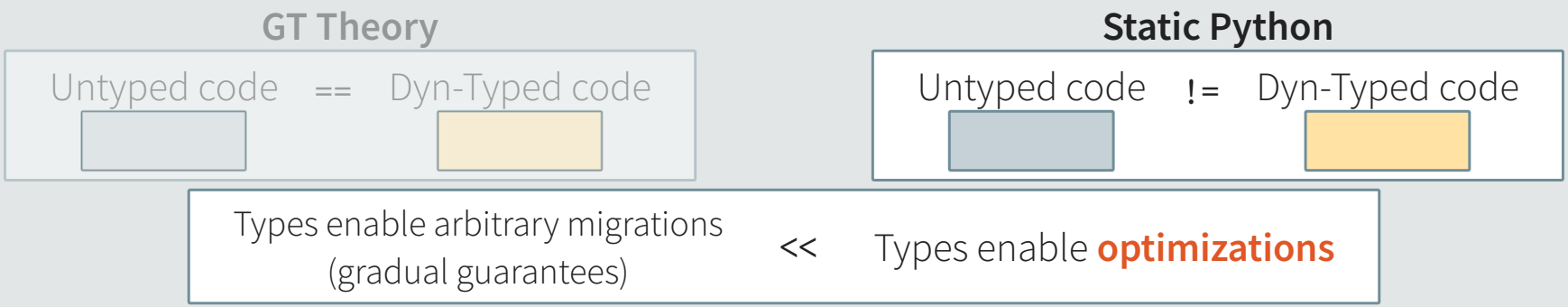
Shallow ~ dispatch

Concrete ~ fast checks

```
x:int64 = 42
y = x
# Type Error
```

Primitive ~ unboxing

Step 3. Limited Dyn Type




```
class A:
 def f(self)->int:
```

```
class B(A):
 def f(self):
   # Type Error
```

Shallow ~ dispatch

```
def avg(ns:chklist[dyn]):
  ....
```

```
avg(chklist[int](1,2))
# Runtime Error
```

Concrete ~ fast checks

```
x:int64 = 42
y = x
# Type Error
```

Primitive ~ unboxing

Step 4. Limited Scope

Step 4. Limited Scope

Focus on high-payoff **optimizations**
rather than feature-completeness

`eval` `first-class class`
`multiple inheritance` ==> defer to Python

`Callable[T0, T1]` `Setof[T]`
`Union[T0, T1, T2]` ==> defer to Pyre

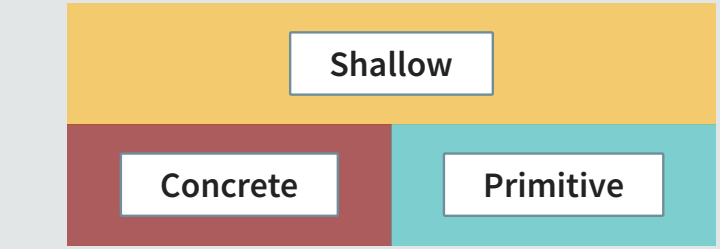
PYRE

How is Static Python so Fast?

0. Better Compiler & Runtime
1. Fast Soundness Checks
2. Progressive Types
3. Limited Dyn Type
4. Limited Overall Scope

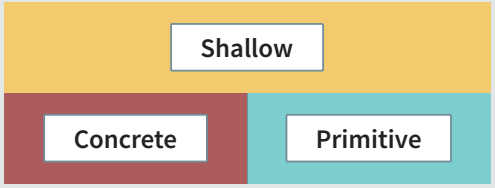
How is Static Python so Fast?

0. Better Compiler & Runtime
1. Fast Soundness Checks
2. Progressive Types
3. Limited Dyn Type
4. Limited Overall Scope

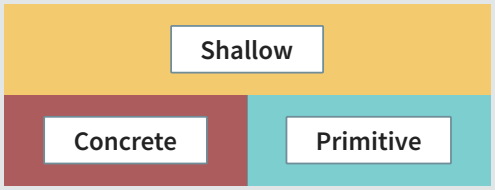


Types gradually enable optimizations
Gradual Soundness

More Experience



More Experience



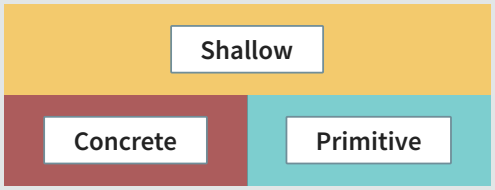
Instagram, March 2023:

959 typed modules

10 with **Concrete** (fast reads)

16 with **Primitives** (unboxed math)

More Experience



Instagram, March 2023:

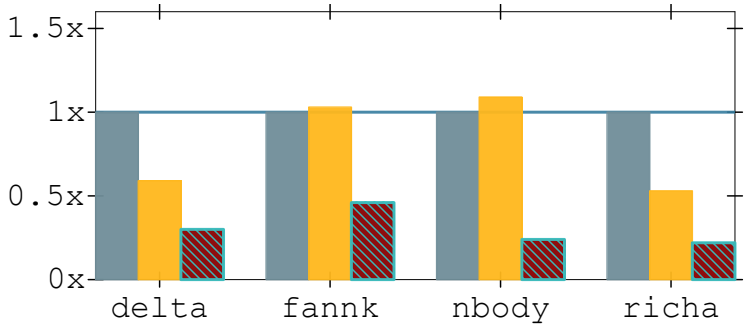
959 typed modules

10 with **Concrete** (fast reads)

16 with **Primitives** (unboxed math)

Microbenchmarks

1x = Python, **lower** is faster



Takeaways

Takeaways

τλ GT Researchers

Guarantees vs. Performance?

Qs for Concrete:

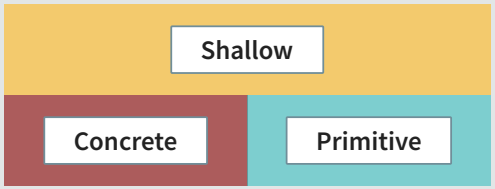
* migrating `list` to `chklist[T]` etc.
* fast tags for `Union[T0, T1, T2]`

Prior work:  

Takeaways

Practitioners

Why not your language?



Takeaways

Language Designers

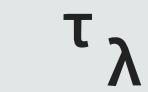
Redex model found:

5 critical soundness bugs

16 correctness issues

Takeaways

The End



New research directions

Who's next?

Model found:
5 soundness + **16** other issues

Static Python