Chasing Anys

Dibri Nsofor, Ben Greenman

How can the Dynamic type (Any) inform the growth of a Type System?

: mypy

We are here studying 221 repos

Filter for the dynamic type → Identify patterns → Search for flagged patterns → 70k REPOS

```
## t = int | str | t -> t
## textbook types cannot express python idioms

def div(n, d) -> ??? :
    if d == 0:
        return "div0"
    else:
        return n/d
```

Two options:
1. Wildcard (Any)
2. Precise types Union[str, int]

Patterns Found, So Far

Untyped self references
for subclass polymorphism

Bounded typevars

```
S = TypeVar('S', bound='Shape')
class Shape:
    def move(self: S, dist: int) -> S:
        self.position += dist
        return self

class Circle(Shape):
    pass

Circle().move(4) #type: Circle
```

Unbounded type variables
not sure. Workaround for recursive types?

```
Car = TypeVar('Car')## car is unbounded
Traffic = Union[Car, List['Traffic']]
class CarObj:
    pass
def count_cars(x: Traffic, car: Car) -> int:
    if isinstance(x, List):
        x.append(car)
    return len(x)

count_cars([CarObj(), CarObj()], 5) #type: int
```

😱

Untyped dicts
for external data

Make type from config file

```
def parse_config() -> dict[str, any]:
    ...
```

Any for exceptions
laziness... does it matter?

```
def __getattr__(self, key: str) -> Any:
    raise AttributeError
```

Research Challenges

Manual search is painstaking:
Originally 320k signatures
50k distinct sigs with Any

Type stubs v. Code:
Some patterns span a block of code, but we only look at stubs.

Pattern matching with regex is too slow.
Over 3 days to search for `-> Any` in a 512 character type signature.

```
[OpExecCtx, str, Optional[str],
Optional[List[Dict[str, str]]],…] ->
Any
```