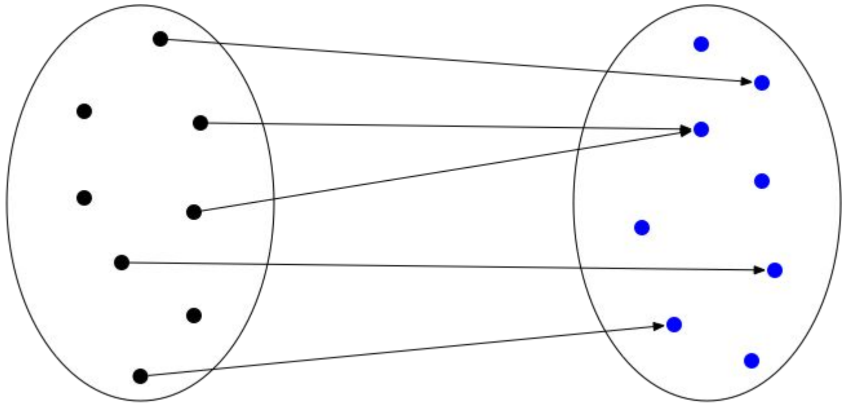
OOP via Python: Session 03

Stephen Leach, Oct 2021

A Brief Terminological Digression …

Maths Functions

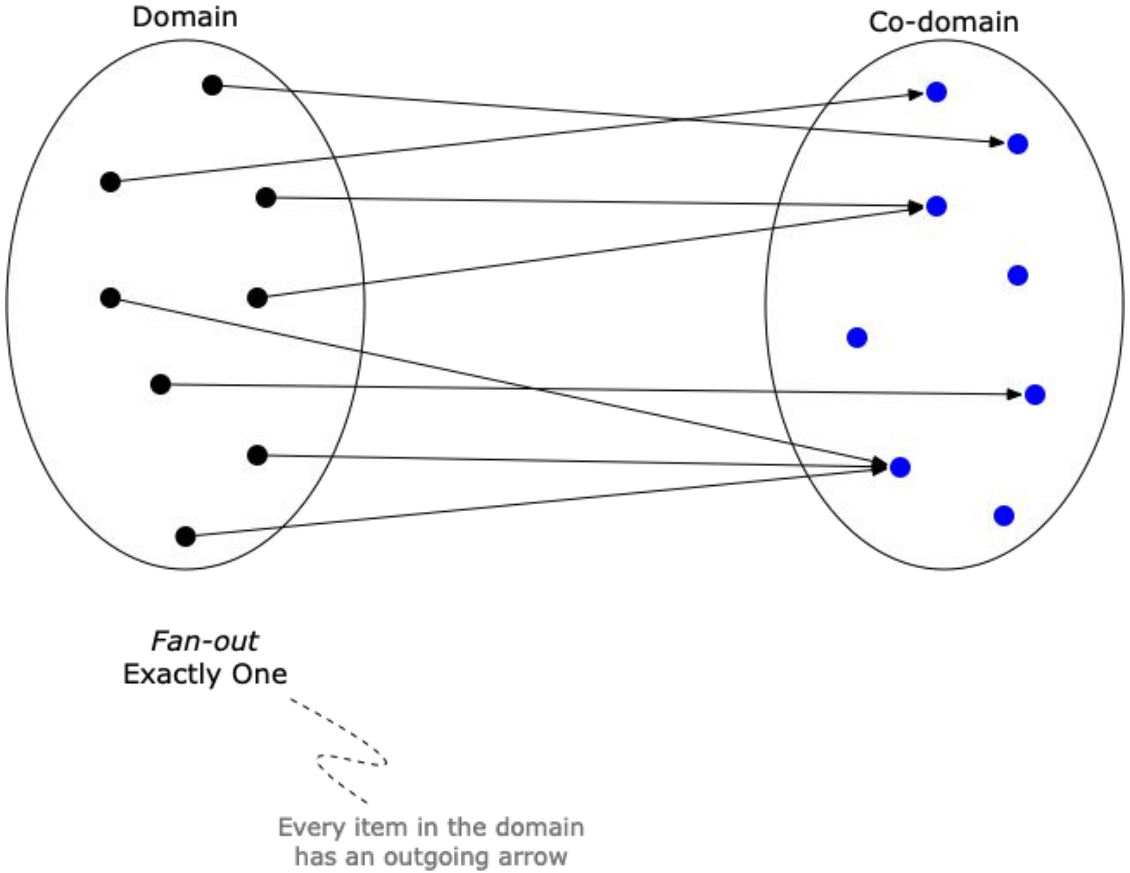


Fan-out
One or Zero

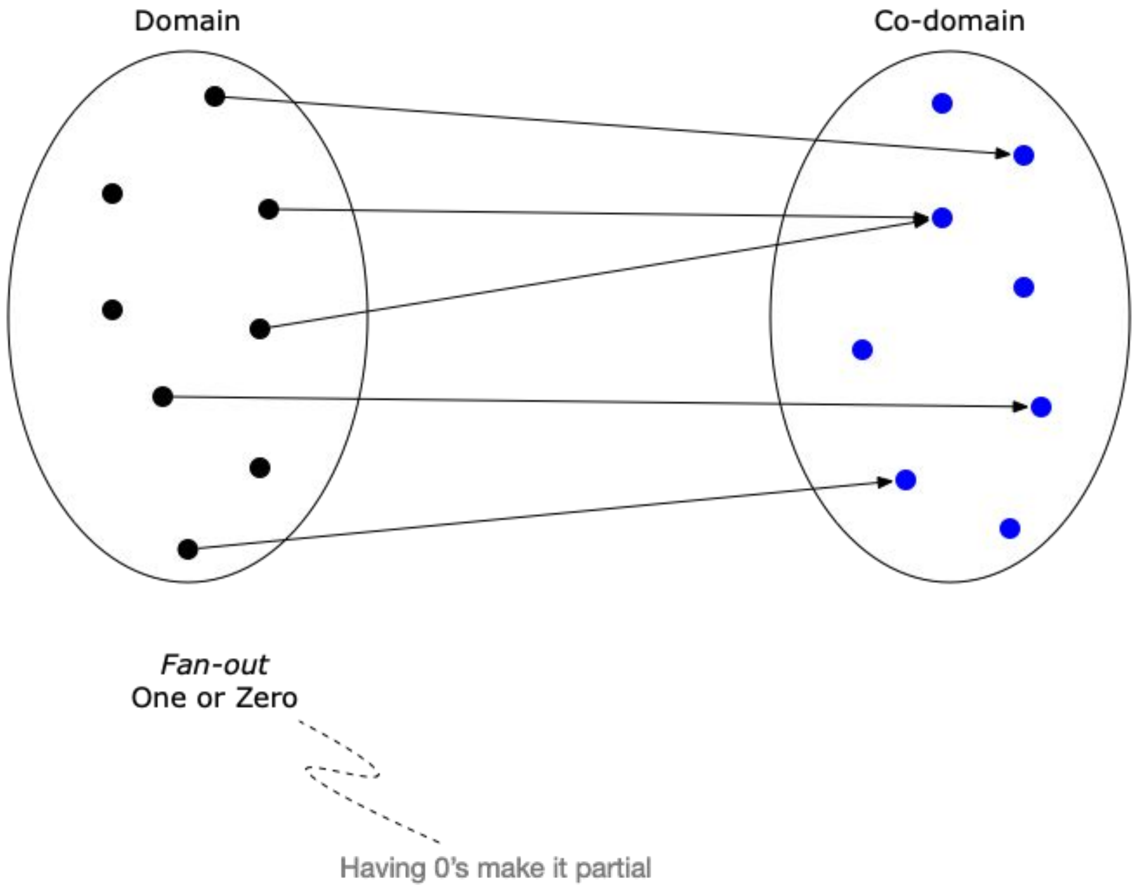
Fan-in
Any number

0/1 is what makes it a *function*

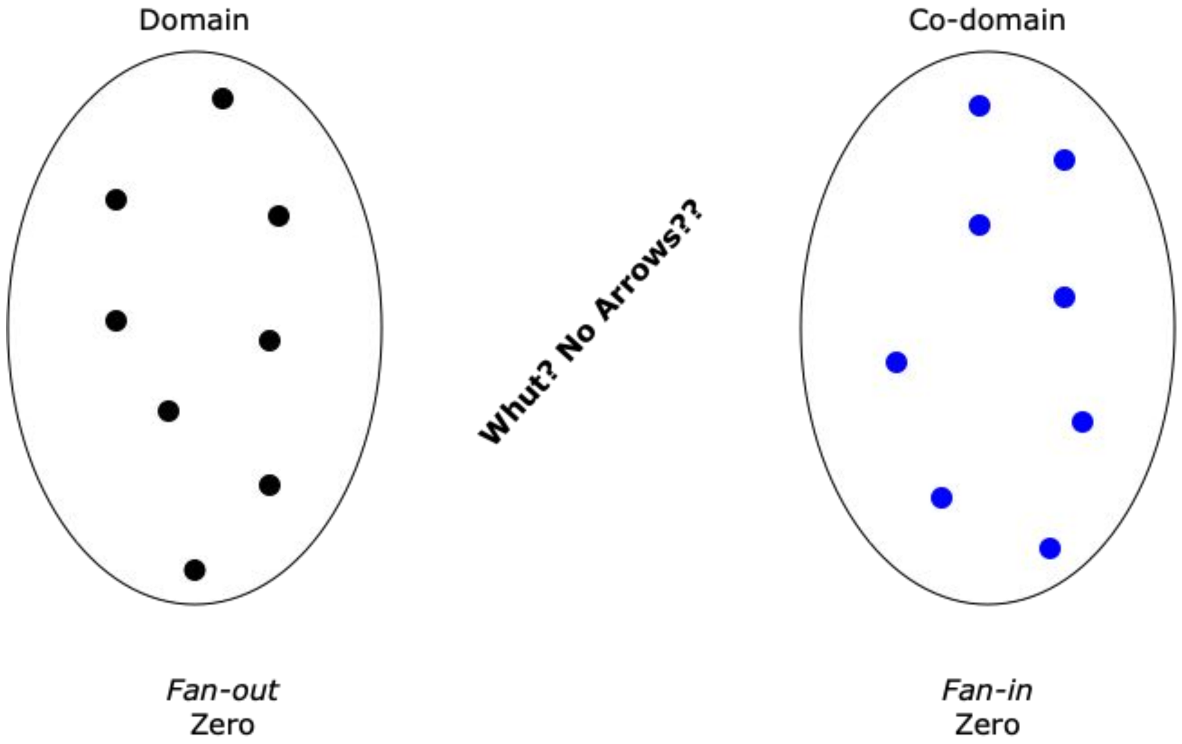
Total


Domain
Co-domain
Fan-out
Exactly One
Every item in the domain
has an outgoing arrow

Partial


Domain
Co-domain
Fan-out
One or Zero
Having 0's make it partial

Undefined



Back to our scheduled programming ....

Method Availability

For an object *obj* at any one point in time:

- *obj*.domethod(*x*, *y*) is **total** if it is safe to call regardless of the values of *x* & *y*
- *obj*.domethod(*x*, *y*) is **partial** if it is safe to call on at least some values of *x* & *y*
- *obj*.domethod(*x*, *y*) is **unavailable** if it throws exceptions regardless of *x* & *y*

add_page & can_add_page

```python
def can_add_page(self, page: int) -> bool:
    return self.is_empty() or page == self._stop

def add_page(self, page: int):
    if self.is_empty():
        self._start = page
        self._stop = page + 1
    elif self._stop == page:
        self._stop += 1
    else:
        raise Exception(f"Trying to add non-consecutive page: {page}")
```

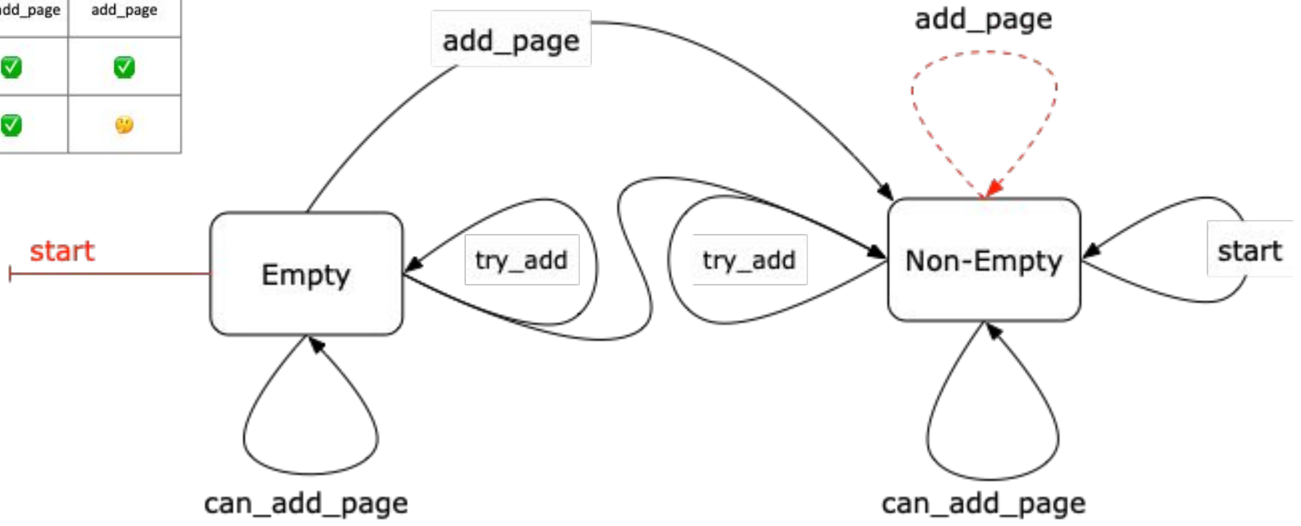
Objects that have the same pattern

Availability	**start**	**try_add**	**can_add_page**	**add_page**	**...**
Empty RangeOfPages	❌	✅	✅	✅	...
Non-Empty RangeOfPages	✅	✅	✅	🤔	...

State/ Phase Grid

Transition Diagram

	start	try_add	can_add_page	add_page
Empty	❌	✅	✅	✅
Non-Empty	✅	✅	✅	🤔



How to Call a Method?

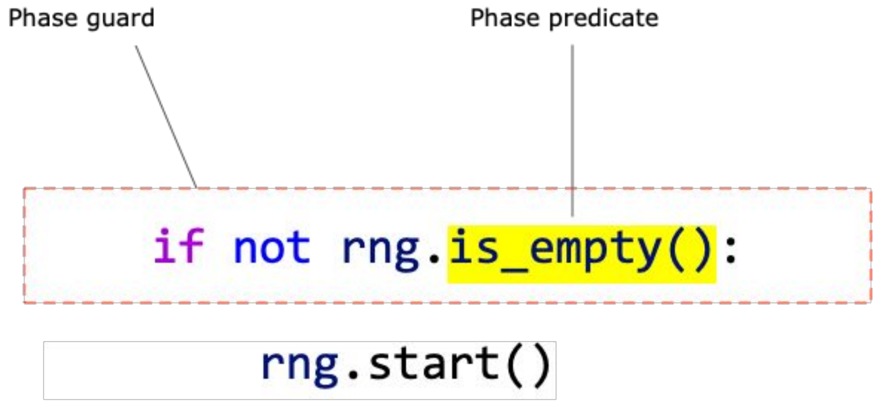
How to call a method …

… that is always total?

```
rng.can_add(99)
```

How to call a method …

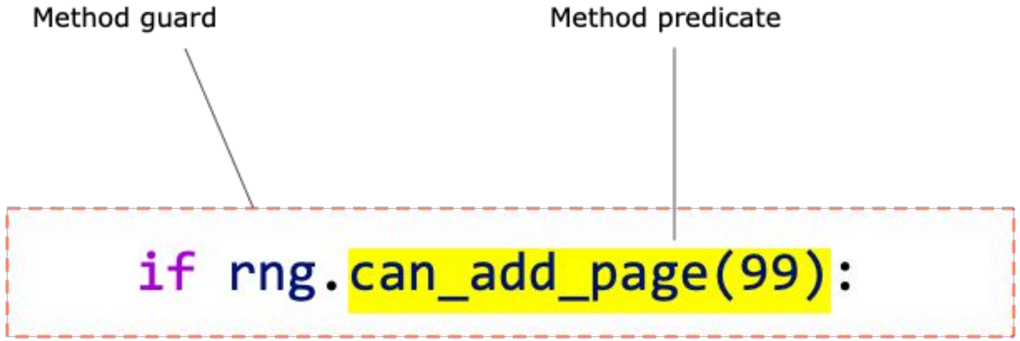
… that is either total or unavailable?



```
if not rng.is_empty():
    rng.start()
```

How to call a method ...

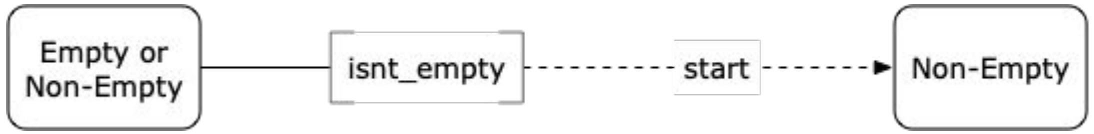
... that is partial?



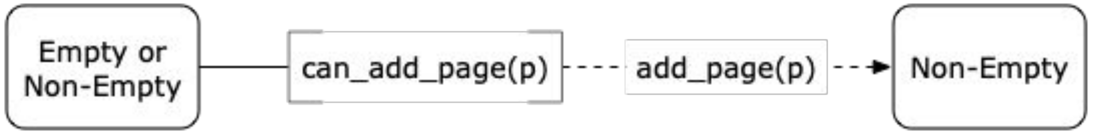
```
if rng.can_add_page(99):
```

```
        rng.add_page(99)
```

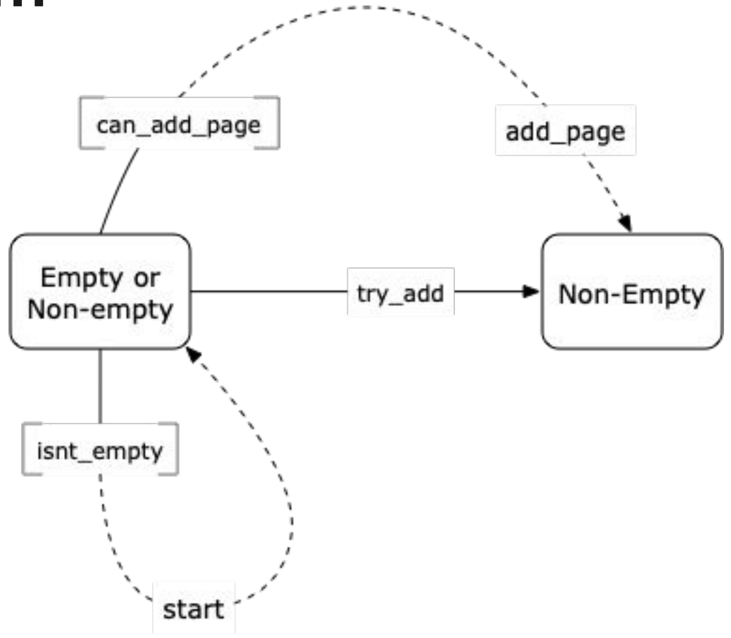
Phase Guards



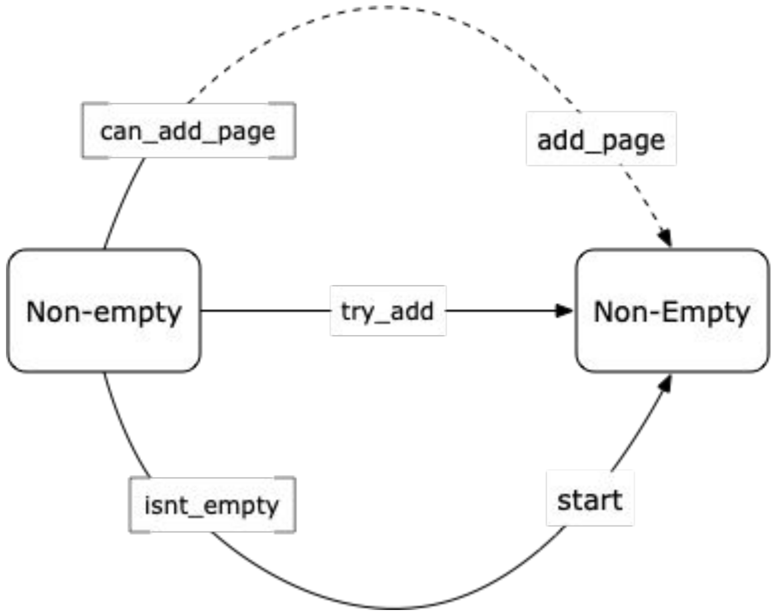
Method Guards



Redrawn Transition Diagram



When Phase is Known



All Code "Constantly Aspires to the Condition of Totality" *

- Phase guards and method guards are both ways of making methods safe - totalisation

- The other way is to design-in totality
  - Returning null e.g. re.search(), contextvar.get()
  - Returning -1 e.g. str.find(sub: str)
  - Pythonic: catching specific exceptions e.g. next() raises StopIteration
  - Return status code + default values … not Pythonic!
  - Returning Maybe<T> e.g. pymaybe

Method Guards … missing in action - why?

Indexing a list: `x[i]` aka `x.__getitem__(i)`

```
# LBYL
if x.can___getitem__(i):
    return x[i]
```

```
# LBYL - open coded
if 0 <= i < len(x):
    return x[i]
```

```
# EAFP
try:
    return x[i]
except IndexError:
    # Something else
```

What's Tricky about EAFP?

```python
def LBYL( x, i ):
    if 0 <= i < len(x):
        n = x[i]
        return f( n, g(n) )
    else:
        return None
```

```python
def EAFP( x, i ):
    try:
        n = x[i]
        return f( n, g(n) )  # WRONG!
    except IndexError:
        return None
```

Example

Example code taken from a current codebase

Exercise

- See exercise.py in [INSERT URL]

Next Session : No Trespassing!