Object-Oriented Programmin via Python, Part 2

Stephen Leach, Oct 2021

Types = Set of Behaviours

- At the heart of object-oriented programming is the equivalence of types and the set-of-behaviours that an object responds to.

- "If it walks like a duck and it quacks like a duck, then it ~~must be~~ **is** a duck"

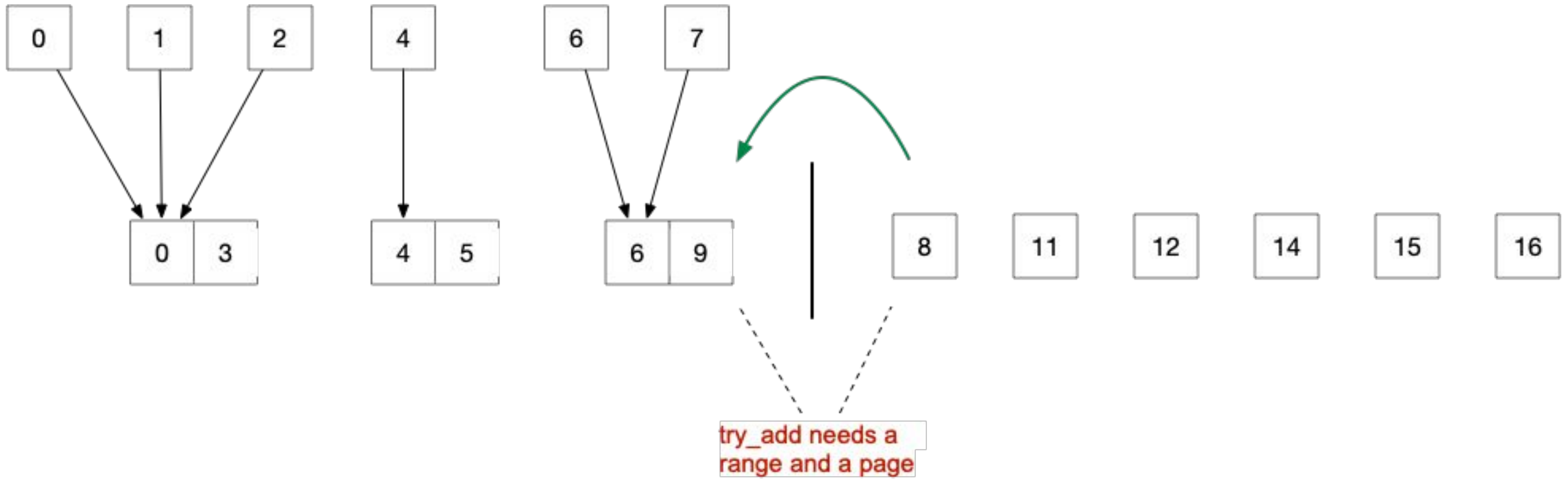
Overlapping Types

- This immediately leads to the idea of overlapping types
    - Types that share some but not all methods have an overlap of shared methods
    - If one type C includes *all* the methods of another type P then C extends P
- We will use this to solve the initialisation problem in range_extract

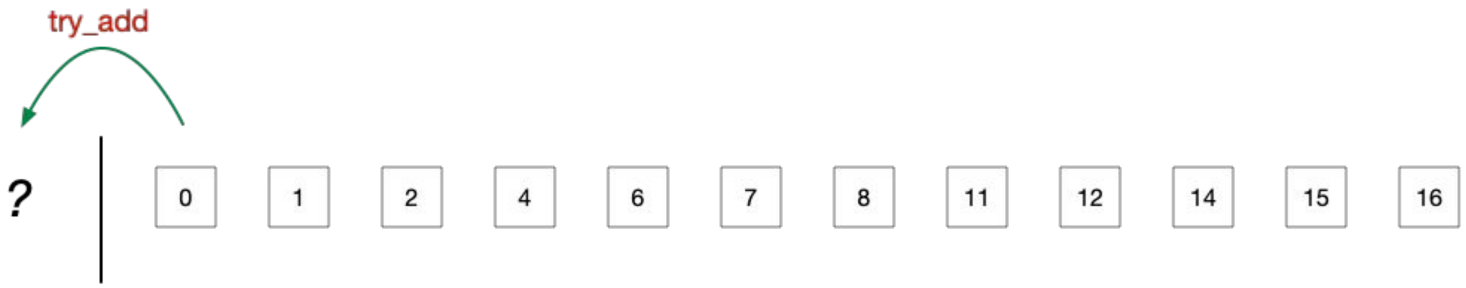
- Static typing - determining ahead of time of the set-of-behaviours that a variable is guaranteed to be able to respond to.

- Static type checking - automatic verification that variables will only be asked to respond to guaranteed behaviours

Loop steady-state (invariant)



Loop initialisation?



Confusing Logic

```
def pages_to_ranges( L ):
    sofar = []
    for i in L:
        if not( sofar and sofar[-1].try_add(i) ):
            sofar.append(RangeOfPages(i))
    return sofar
```

Rewrite to track sofar[-1]

```
def pages_to_ranges( pages_list ):
   sofar = []
   last_in_sofar = None
   for p in pages_list:
       if not last_in_sofar or not last_in_sofar.try_add(p):
           last_in_sofar = RangeOfPages(p)
           sofar.append(last_in_sofar)
   return sofar
```

What behaviours does last_in_sofar support?

- bool(last_in_sofar)   … but only to make it safe to call try_add
- last_in_sofar.try_add(page)

Only need is try_add

```python
class TryAddRefusnik:
    def try_add(self, page):
        return False

def pages_to_ranges( pages_list ):
    sofar = []
    last_in_sofar = TryAddRefusnik()
    for p in pages_list:
        if not last_in_sofar.try_add(p):
            last_in_sofar = RangeOfPages(p)
            sofar.append(last_in_sofar)
    return sofar
```

Duck Typing to Type Hints

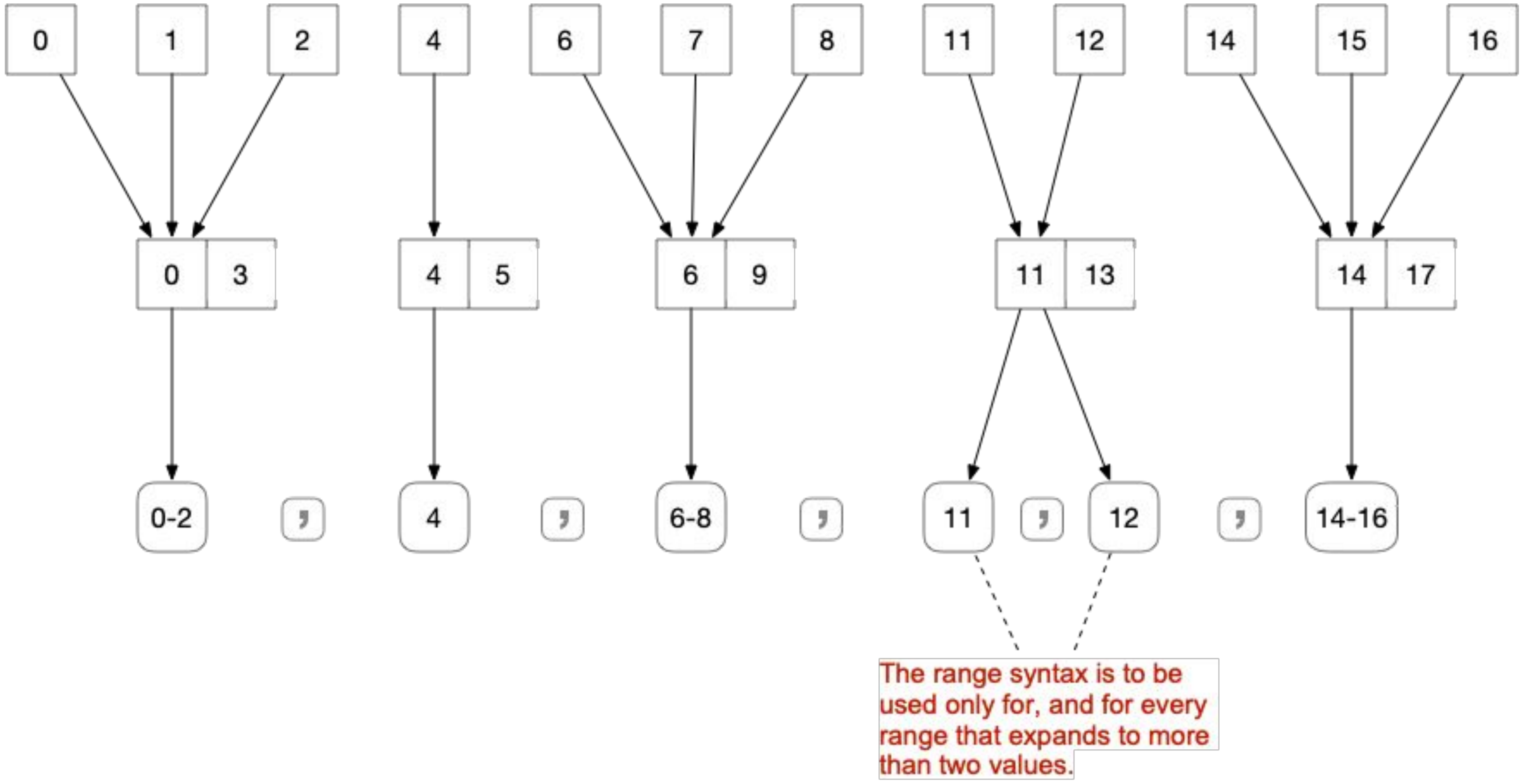
```
from typing import List
from typing_extensions import Protocol

class TryAddable(Protocol):
   def try_add(self, page: int) -> bool:
       pass
        .
        .
        .

def pages_to_ranges( pages_list: List[int] ) -> List[RangeOfPages]:
   sofar: List[RangeOfPages] = []
   last_in_sofar: TryAddable = TryAddRefusnik()
   for p in pages_list:
       if not last_in_sofar.try_add(p):
           rng: RangeOfPages = RangeOfPages(p)
           last_in_sofar = rng
           sofar.append(rng)
   return sofar
```

Printing

- "The range syntax is to be used only for, and for every range that expands to more than two values."


0
1
2
4
6
7
8
11
12
14
15
16
0
3
4
5
6
9
11
13
14
17
0-2
,
4
,
6-8
,
11
,
12
,
14-16
The range syntax is to be used only for, and for every range that expands to more than two values.

str_parts

```python
def str_parts(self) -> Iterable[str]:
    if self.count() <= 2:
        yield from map(str, range(self.start(), self.stop()))
    else:
        yield f"{self.start()}-{self.finish()}"
```

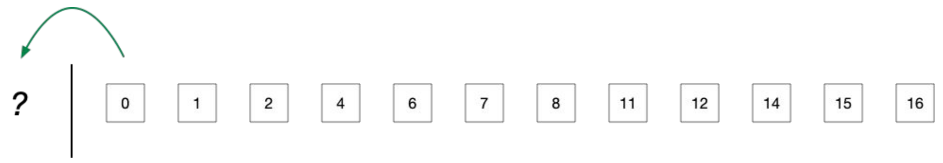
Put it all together

- https://github.com/sfkleach/oop-via-python
- https://github.com/sfkleach/oop-via-python/blob/main/session02/range_extract_02_D.py

```
$ python3 test.py
Pages  : [-8, -7, -6, -3, -2, -1, 0, 1, 3, 4, 5, 7, 8, 9, 10, 11, 14, 15, 17, 18, 19, 20]
Extract: -8--6,-3-1,3-5,7-11,14,15,17-20
Pages  : [0, 1, 2, 4, 6, 7, 8, 11, 12, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 27, 28, 29, 30, 31, 32, 33, 35, 36, 37, 38, 39]
Extract: 0-2,4,6-8,11,12,14-25,27-33,35-39
$
```

Exercise - Empty Range (30 mins)

An alternative solution for **?** is an empty range.



- Starting from https://github.com/sfkleach/oop-via-python/session02/exercise_02.py
- Extend RangeOfPages to support an empty range
- Hint: It should generate no strParts
- Question: What is the start() of an empty range? Should it be allowed?
- Stretch goal: Alter the algorithm to use the empty range
- Reflection on the different approaches

Reflections

- The concept of the empty range solved the initialisation problem
- It's a meaningful idea
- But some methods don't work on an empty RangeOfPages
- Hold on .... didn't we assert that types = set-of-supported-behaviours
- ***WTH ?!?!?!***

Stay tuned for next week's episode on ....

.... DEEP STATE

Reflections 2

- The challenge is to get the range-construction and list-addition coordinated
- A general approach to this is the Builder Pattern
    - TryAddRefusnik is a big step towards the builder pattern
    - Builders will buffer up info
    - So they have a special newInstance event which acts as a flush