Classes in Python, Part 1

Stephen Leach, September 2021

Who is this for?

- Improvers rather than beginners
- People who aspire to a career in development
- Anyone who loves beautiful code

What is Object Oriented Programming?

- OOPS is a highly developed set of grand-sounding ideas about how code and data should relate to each other:
    - Implementation hiding ← *this matters*
    - Encapsulation ← *so does this*
    - Extensibility (aka sub-typing) ← *this too*
    - Actor-based Concurrency
- Every programming language embraces each of these ideas to a greater or lesser extent

Why Do We Care?

- Classes allow us to write libraries that provide behaviour (the 'what', the 'interface')
  - So we can write programs in terms of behaviour
- But do not expose the rest of the program to the implementation (the 'how', the 'class')
  - Which limits the extent of the damage when we change implementation
- And you can write other libraries that provide or extend the same behaviour
  - Which lets us swap libraries with no impact AND
  - Allows us to work with the common behaviour of different implementations (e.g. iterables)

Range Extraction: example from rosettacode.org

A format for expressing an ordered list of integers is to use a comma separated list of either

- individual integers
- Or a range of integers denoted by the starting integer separated from the end integer in the range by a dash, '-'. (The range includes all integers in the interval including both endpoints)
  - **The range syntax is to be used only for, and for every range that expands to more than two values.**

Example

The list of integers:

-6, -3, -2, -1, 0, 1, 3, 4, 5, 7, 8, 9, 10, 11, 14, 15, 17, 18, 19, 20

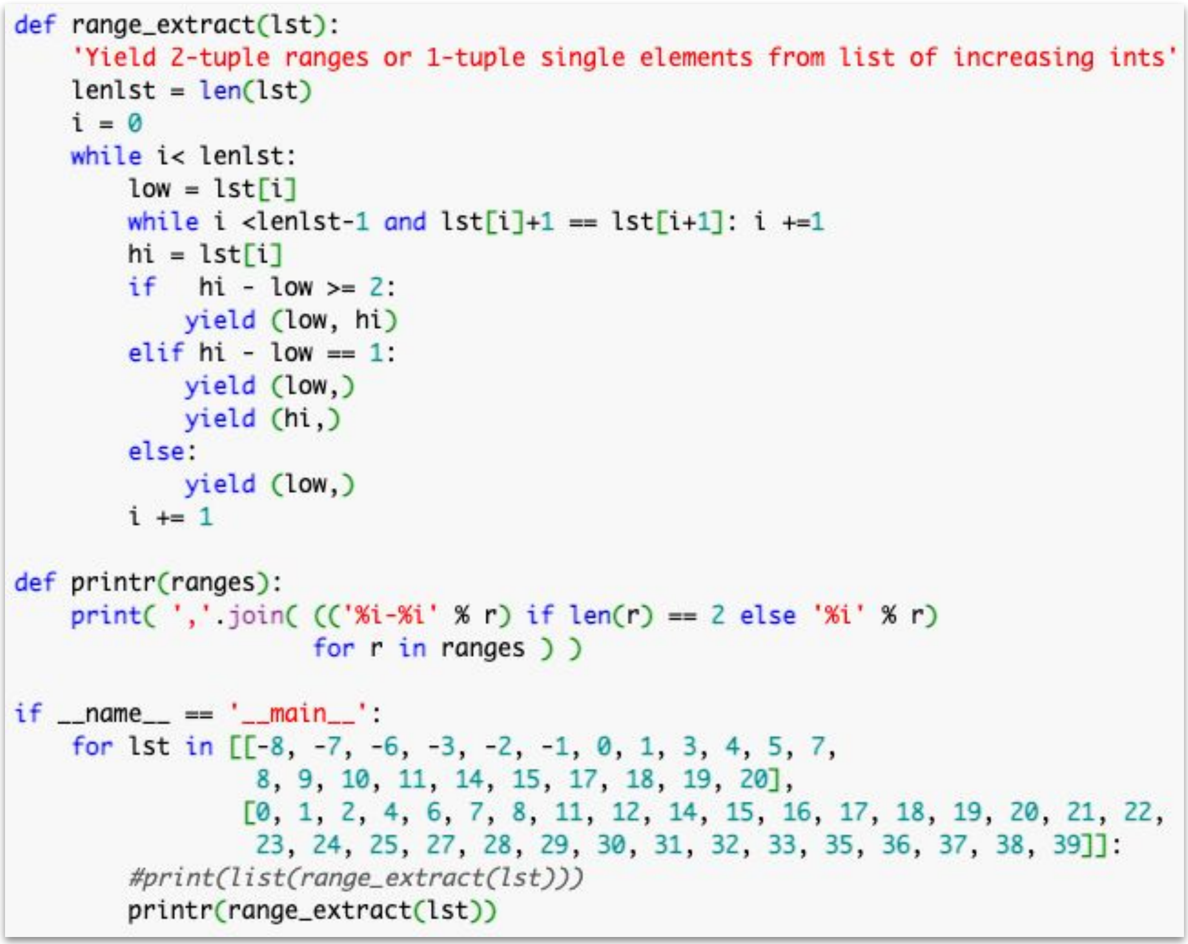
Is accurately expressed by the range expression:

-6,-3-1,3-5,7-11,14,15,17-20

Example: Range of Pages

- e.g. Pages 120-124
- We could represent it as a pair: `( 120, 124 )`
- What's good about this?
- What's not so good?

Procedural



```python
def range_extract(lst):
    'Yield 2-tuple ranges or 1-tuple single elements from list of increasing ints'
    lenlst = len(lst)
    i = 0
    while i< lenlst:
        low = lst[i]
        while i <lenlst-1 and lst[i]+1 == lst[i+1]: i +=1
        hi = lst[i]
        if   hi - low >= 2:
            yield (low, hi)
        elif hi - low == 1:
            yield (low,)
            yield (hi,)
        else:
            yield (low,)
        i += 1

def printr(ranges):
    print( ','.join( (('%i-%i' % r) if len(r) == 2 else '%i' % r)
                     for r in ranges ) )

if __name__ == '__main__':
    for lst in [[-8, -7, -6, -3, -2, -1, 0, 1, 3, 4, 5, 7,
                 8, 9, 10, 11, 14, 15, 17, 18, 19, 20],
                [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]]:
        #print(list(range_extract(lst)))
        printr(range_extract(lst))
```

As a Class

```
class RangeOfPages:

   def __init__(self, start, length=1):
       self._start = start
       self._stop = start + length
```

What Have We Accomplished?

```
>>> r = RangeOfPages(120, 4)
>>> r._start
120
>>> r._stop
125
```

Add a Method

```
class RangeOfPages:

    def __init__(self, start, length=1):
        self._start = start
        self._stop = start + length

    def start(self):
        return self._start
```

What Have We Accomplished?

```
>>> r = RangeOfPages(120, 4)
>>> r.start()
120
>>>
```

```
def start(self):
    return self._start
```

Another Method

```python
class RangeOfPages:

    def __init__(self, start, length=1):
        self._start = start
        self._stop = start + length

    def start(self):
        return self._start

    def stop(self):
        return self._stop
```

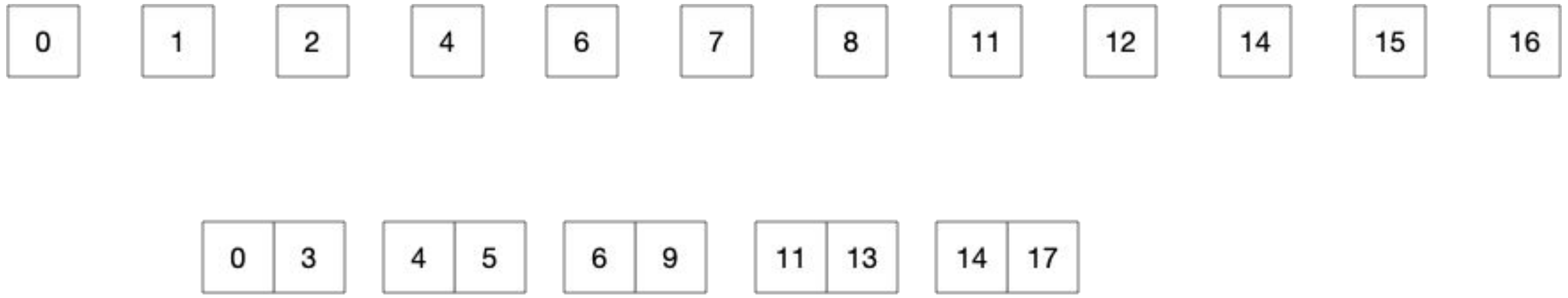
What Have We Accomplished?

```
>>> r = RangeOfPages(120, 4)
>>>
>>> r.start()
120
>>>
>>> r.stop()
125
>>>
```

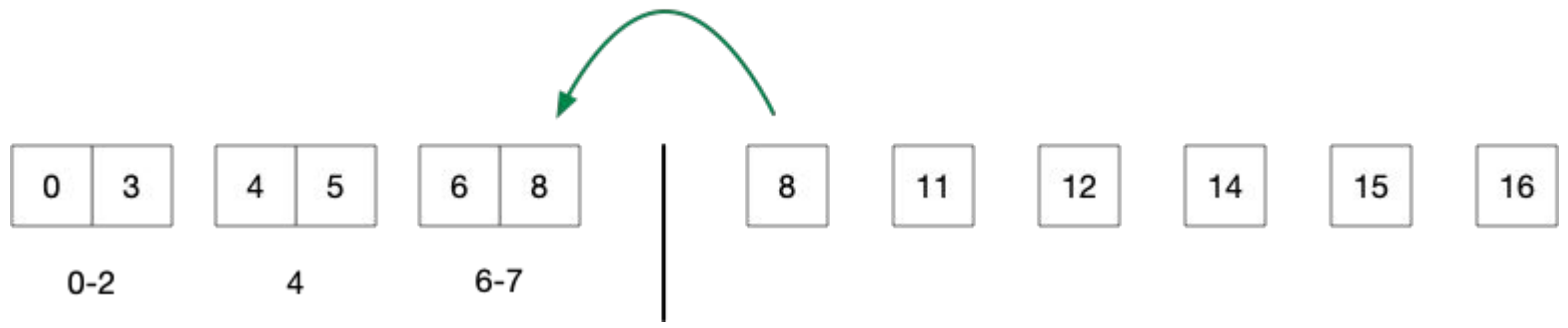
Exercise: Add a Method to Return a Count

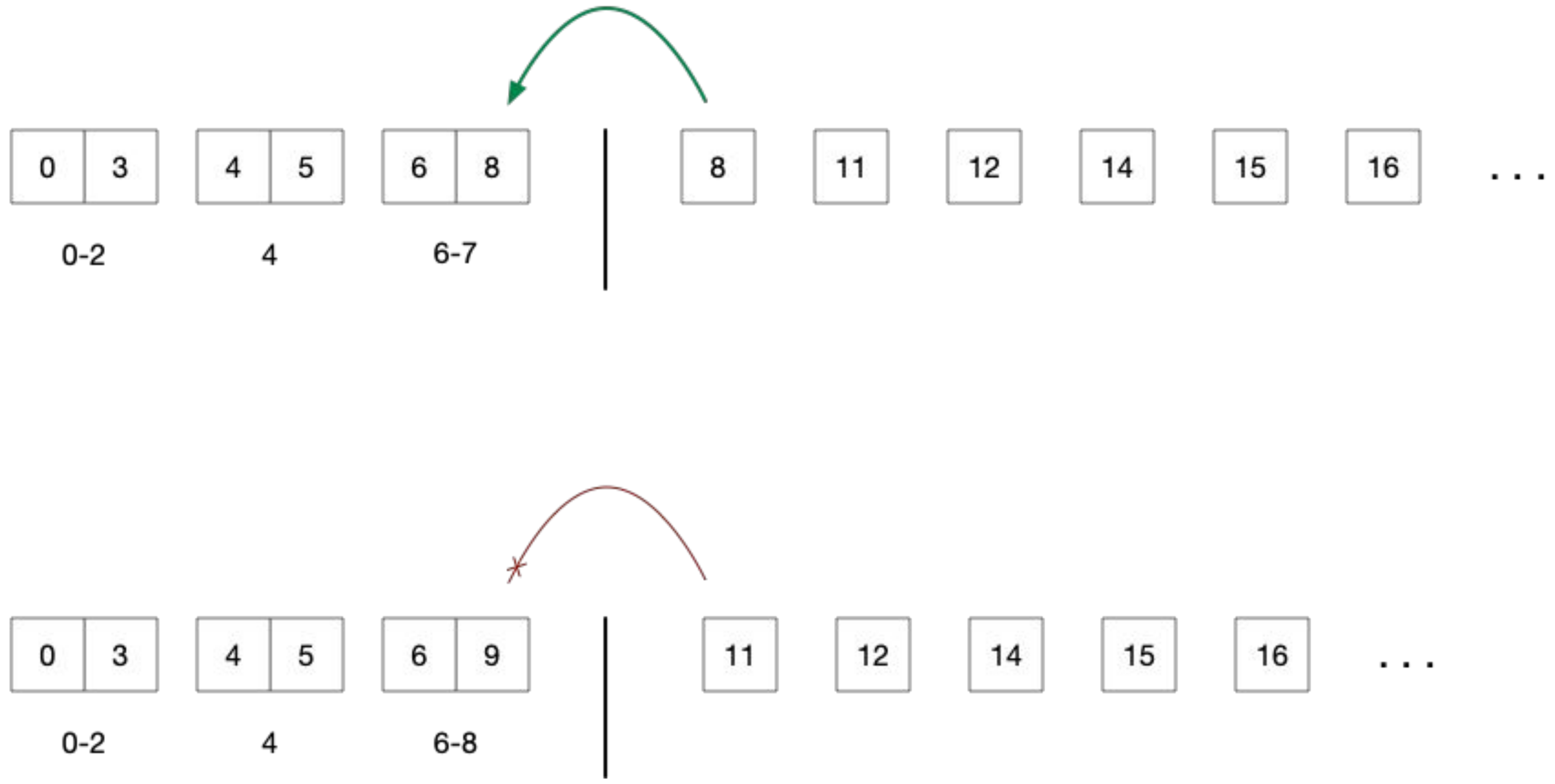
```
>>> r = RangeOfPages(120, 4)
>>> r.count()
4
```

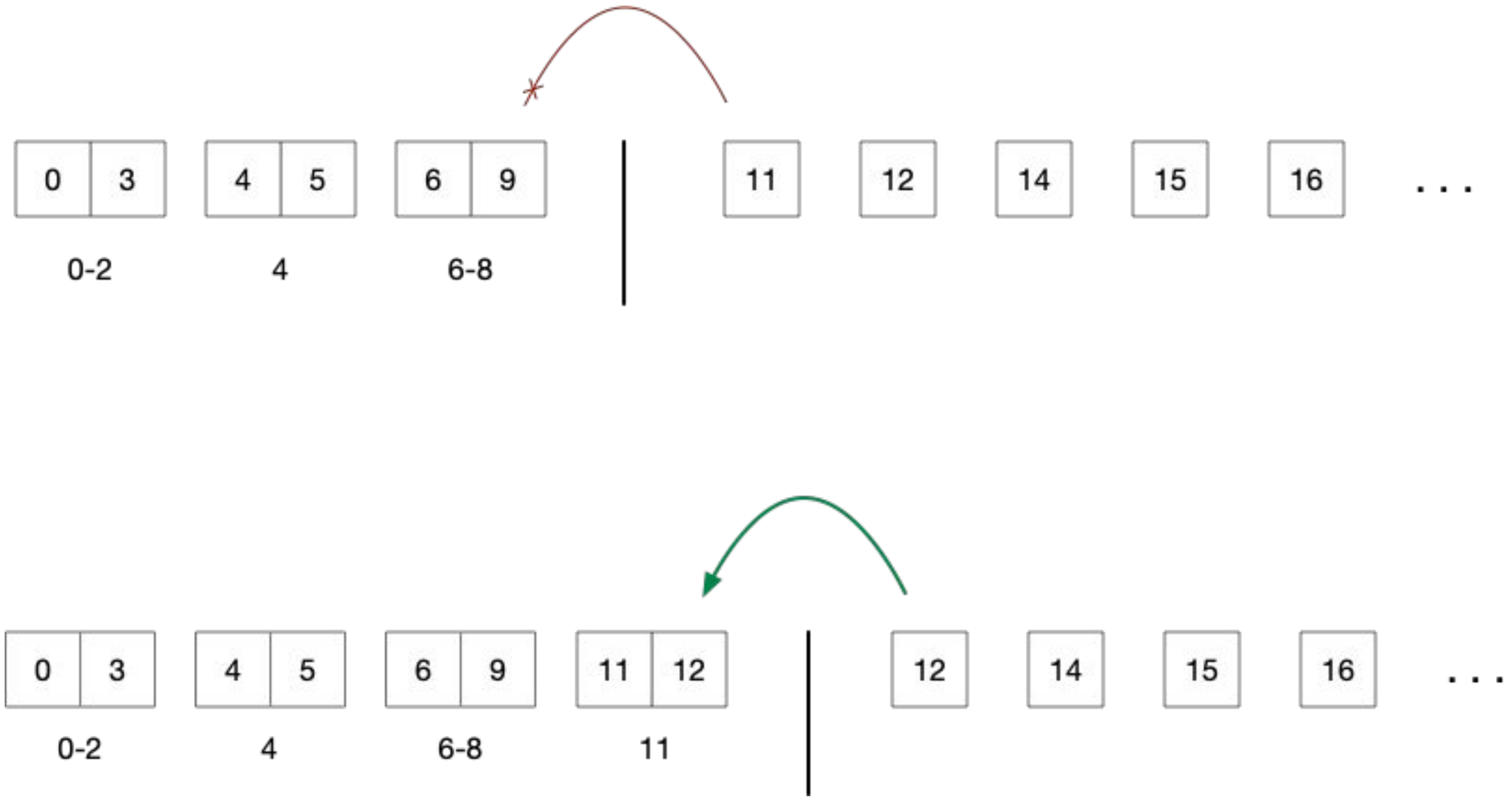
From Pages to Ranges



Half-and-Half




0
3
4
5
6
8
0-2
4
6-7
8
11
12
14
15
16
...
0
3
4
5
6
9
0-2
4
6-8
11
12
14
15
16
...


0
3
4
5
6
9
0-2
4
6-8
11
12
14
15
16
. . .
0
3
4
5
6
9
11
12
0-2
4
6-8
11
12
14
15
16
. . .

Exercise: Add a Method to Try to Add a Page

```
>>> r = RangeOfPages(120)
>>> print(r.try_add(121))
True
>>> print(r.count())
2
```

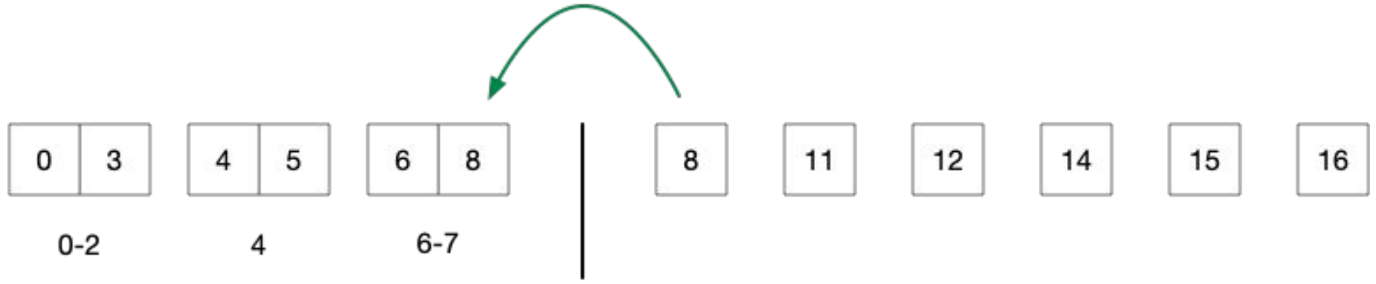
My Answer

```python
class RangeOfPages:

    def try_add(self, page):
        is_next = page == self._stop
        if is_next:
            self._stop += 1
        return is_next
```

Pages to Ranges

```
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
```



What Did We Learn?

.....