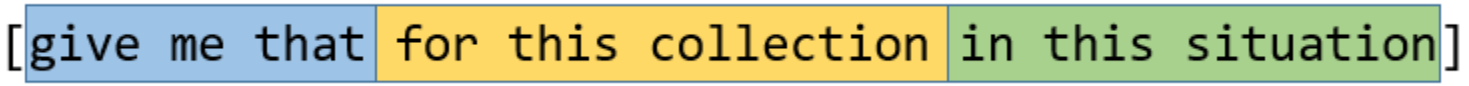
# on list | set | dict comprehension

support

# List | Set | dict comprehension




https://dbader.org/blog/list-dict-set-comprehensions-in-python
https://towardsdatascience.com/python-basics-list-comprehensions-631278f22c40

# List comprehension

- Equivalent to other list definition
- More declarative i.e. similar to mathematic formulation

```
(values) = [ (expression) for
(item) in (collection) ]
```

```
(values) = []
for (item) in (collection):
    (values).append( (expression) )
```

```
values = [expression
          for item in collection
          if condition]
```

```
values = []
for item in collection:
    if condition:
        values.append(expression)
```

https://dbader.org/blog/list-dict-set-comprehensions-in-python

# example

```
>>> even_squares = [x * x for x in range(10)
                    if x % 2 == 0]
even_squares = []
for x in range(10):
    if x % 2 == 0:
        even_squares.append(x * x)
```

```
values = [expression
          for item in collection
          if condition]
```

# examples

```
>>> sentence = 'the rocket came back from mars'
>>> vowels = [i for i in sentence if i in 'aeiou']
>>> vowels
['e', 'o', 'e', 'a', 'e', 'a', 'o', 'a’]
```

```
>>> sentence = 'The rocket, who was named Ted, came back \
... from Mars because he missed his friends.'
>>> def is_consonant(letter):
...     vowels = 'aeiou'
...     return letter.isalpha() and letter.lower() not in
vowels
>>> consonants = [i for i in sentence if is_consonant(i)]
```

# examples

```
>>> original_prices = [1.25, -9.45, 10.22, 3.78, -5.92, 1.16]
>>> prices = [i if i > 0 else 0 for i in original_prices]
>>> prices
[1.25, 0, 10.22, 3.78, 0, 1.16]
```

# Comprehension for dict

- To create a dictionary comprehension we just need to change the brackets [] to curly braces {}. Additionally, in the output expression, we need to separate key and value by a colon :.

```
prices = {"beer": 2, "fish": 5, "apple": 1}
float_prices = {key:float(value) for key, value in
prices.items()}
print(float_prices)

#output
{'beer': 2.0, 'fish': 5.0, 'apple': 1.0}
```

https://towardsdatascience.com/python-basics-list-comprehensions-631278f22c40

# Comprehension for sets

- To create a set comprehension we only need to change the brackets [] to curly braces {}.

```
numbers = [10, 10, 20, 30, 12, -20, 0, 1]
unique_squares = {number**2 for number in numbers}
print(unique_squares)

#output
{0, 1, 100, 144, 400, 900}
```

# Other examples

- A set

```
>>> { x * x for x in range(-9, 10) }
set([64, 1, 36, 0, 49, 9, 16, 81, 25, 4])
```

- A dictionary

```
>>> { x: x * x for x in range(5) }
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16}
```

# flatten

```
matrix = [[1, 2, 3], [4, 5], [6, 7, 8, 9]] # 2-D List
flatten_matrix = []
for sublist in matrix:
    for val in sublist:
        flatten_matrix.append(val)
print(flatten_matrix) # -> [1, 2, 3, 4, 5, 6, 7, 8, 9]

matrix = [[1, 2, 3], [4, 5], [6, 7, 8, 9]]
# Nested List Comprehension to flatten a given 2-D matrix
flatten_matrix = [val for sublist in matrix for val in
sublist]
print(flatten_matrix) # -> [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

https://www.geeksforgeeks.org/nested-list-comprehensions-in-python/

# nested

```
>>> cities = ['Austin', 'Tacoma', 'Topeka', 'Sacramento',
'Charlotte']
>>> temps = {city: [0 for _ in range(7)] for city in cities}
>>> temps
{
    'Austin': [0, 0, 0, 0, 0, 0, 0],
    'Tacoma': [0, 0, 0, 0, 0, 0, 0],
    'Topeka': [0, 0, 0, 0, 0, 0, 0],
    'Sacramento': [0, 0, 0, 0, 0, 0, 0],
    'Charlotte': [0, 0, 0, 0, 0, 0, 0]
}
```

https://www.geeksforgeeks.org/nested-list-comprehensions-in-python/

# nested

```
my_list = []
for x in [20, 40, 60]:
    for y in [2, 4, 6]:
        my_list.append(x * y)
# → [40, 80, 120, 80, 160, 240, 120, 240, 360]

my_list = [x * y for x in [20, 40, 60] for y in [2, 4, 6]]
# → [40, 80, 120, 80, 160, 240, 120, 240, 360]

matrix = [[item for item in range(5)] for row in range(3)]
# → [[0, 1, 2, 3, 4], [0, 1, 2, 3, 4], [0, 1, 2, 3, 4]]
```

https://www.geeksforgeeks.org/nested-list-comprehensions-in-python/

# Map: other options use …

```
>>> list(map(lambda x: x.capitalize(), ['cat', 'dog', 'cow']))
['Cat', 'Dog', 'Cow’]

>>> [x.capitalize() for x in ['cat', 'dog', 'cow']]
['Cat', 'Dog', 'Cow']
```

https://realpython.com/python-lambda/

# Filter: other option

```
>>> even = lambda x: x%2 == 0
>>> list(filter(even, range(11)))
[0, 2, 4, 6, 8, 10]

>>> [x for x in range(11) if x%2 == 0]
[0, 2, 4, 6, 8, 10]
```

https://realpython.com/python-lambda/

# Reduce

```
>>> import functools
>>> pairs = [(1, 'a'), (2, 'b'), (3, 'c')]
>>> functools.reduce(lambda acc, pair: acc + pair[0], pairs, 0)
6
```

```
>>> pairs = [(1, 'a'), (2, 'b'), (3, 'c')]
>>> sum(x[0] for x in pairs)
6
```

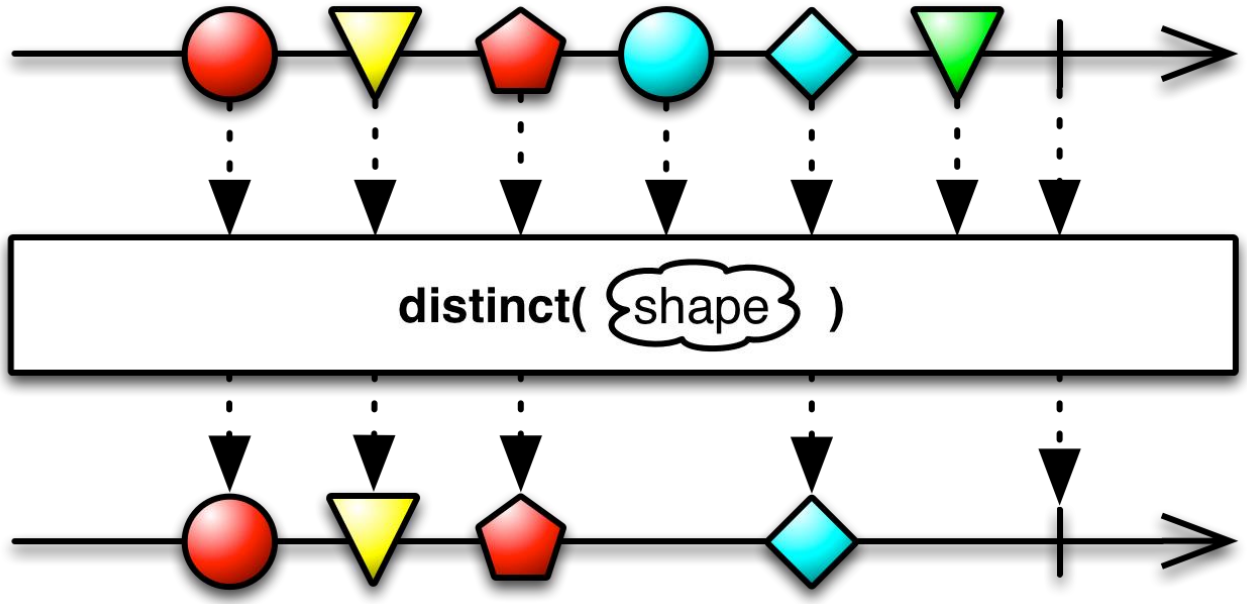
https://realpython.com/python-lambda/

# Other examples

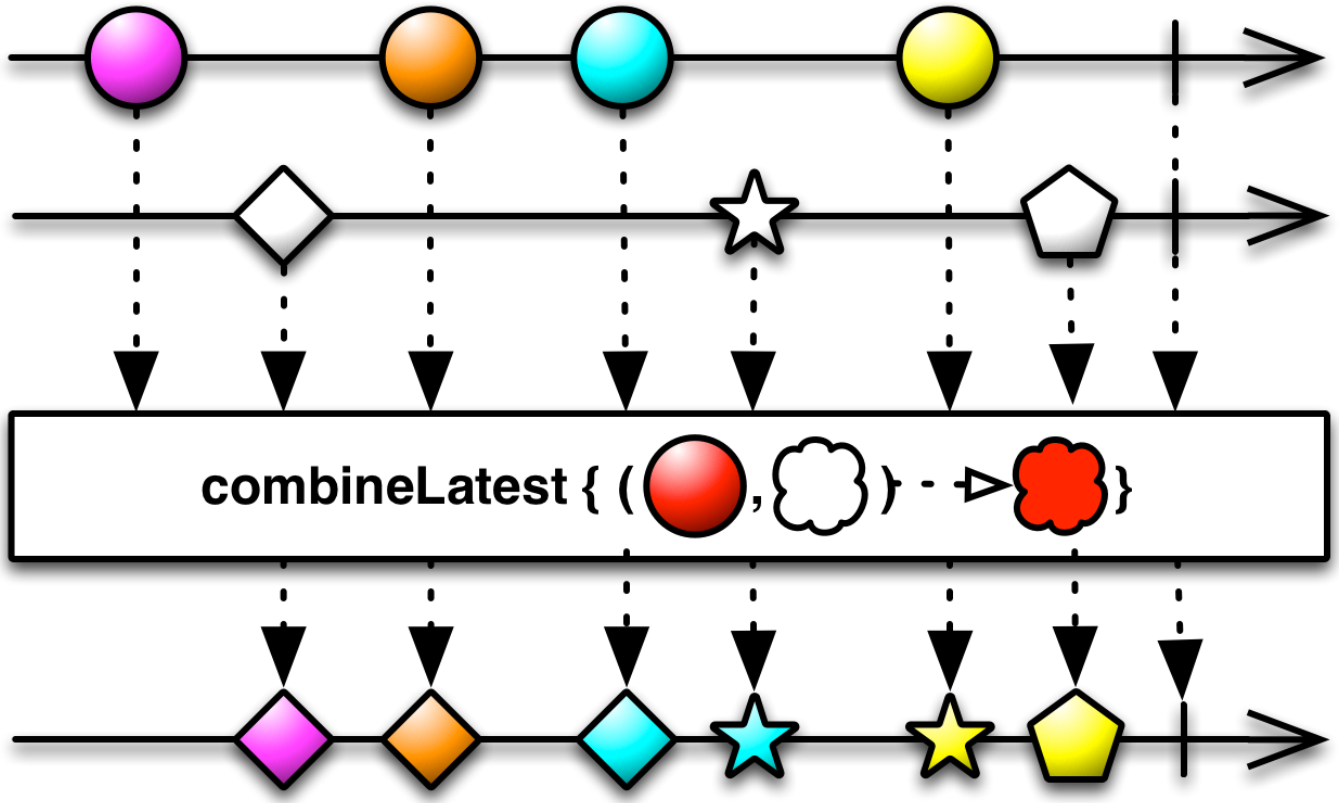
```
# Filter
numbers = [1, 2, 3, 4, 5]
filtered = list(filter(lambda x: x % 2 == 0, numbers))
print(filtered)

# List Comprehension
numbers = [1, 2, 3, 4, 5]
filtered = [number for number in numbers if number % 2
== 0]
print(filtered)
```

# The END

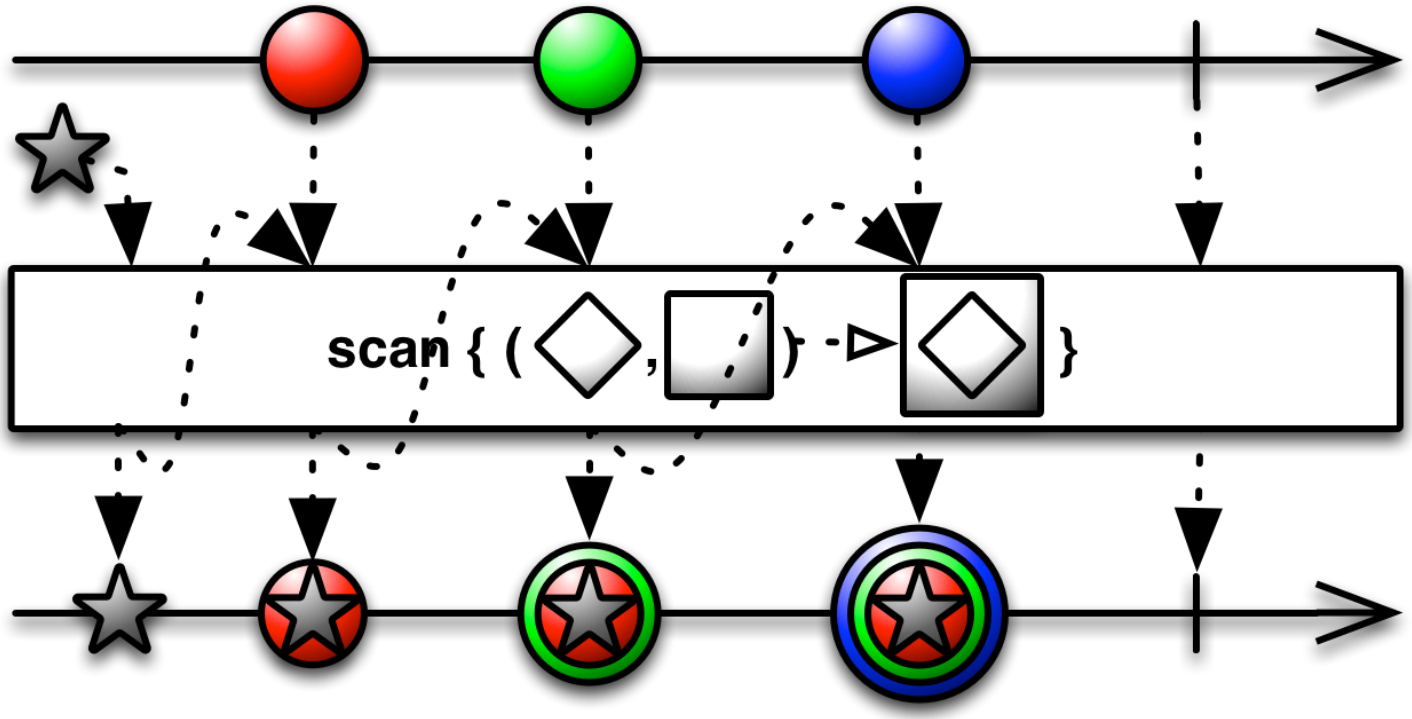
# challenges




https://github.com/Froussios/Intro-To-RxJava/blob/master/Part%202%20-%20Sequence%20Basics/2.%20Reducing%20a%20sequence.md




https://pursuit.purescript.org/packages/purescript-rx-observable/1.1.3/docs/RxJS.Observable


scan { ( , ) -> }