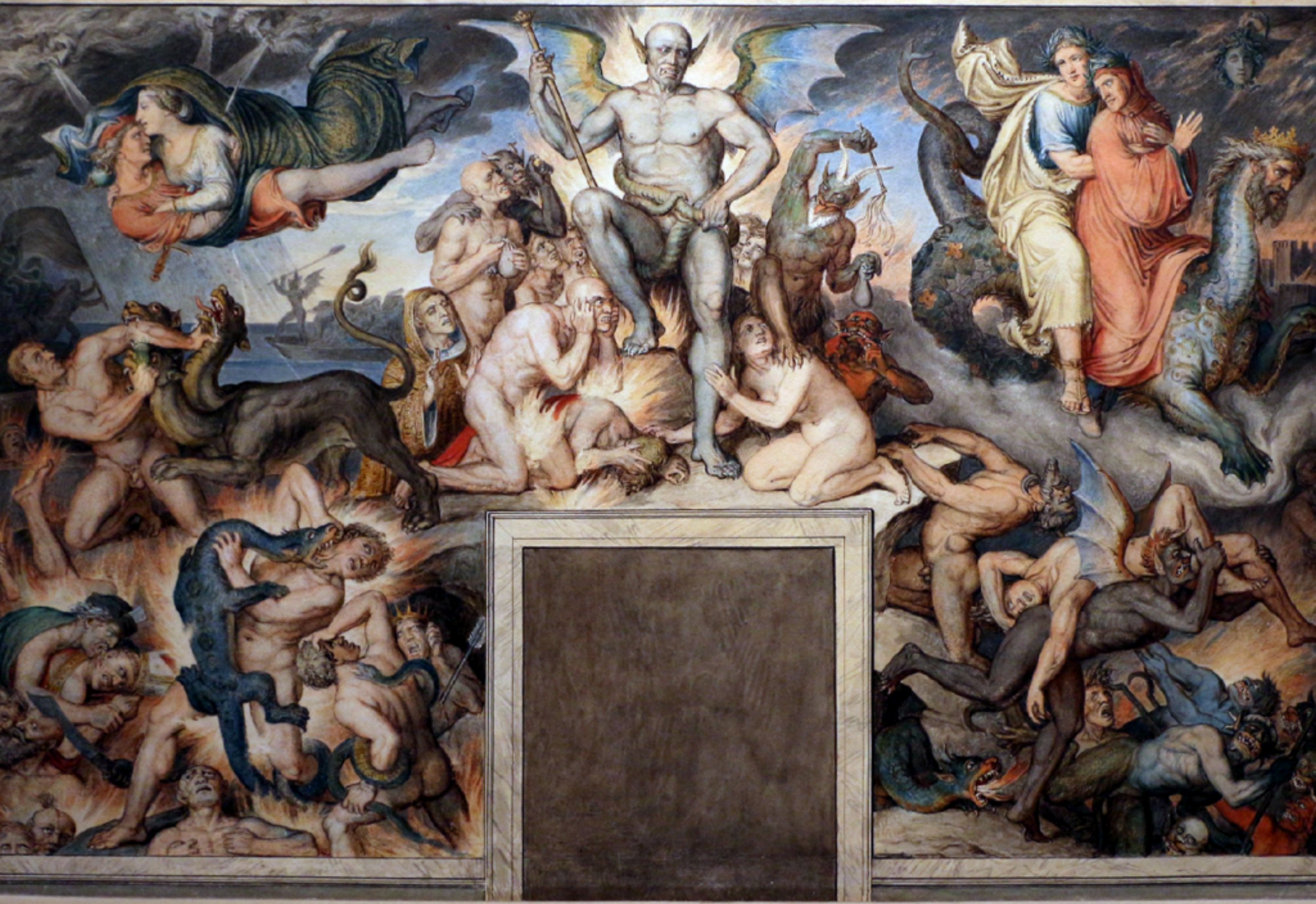
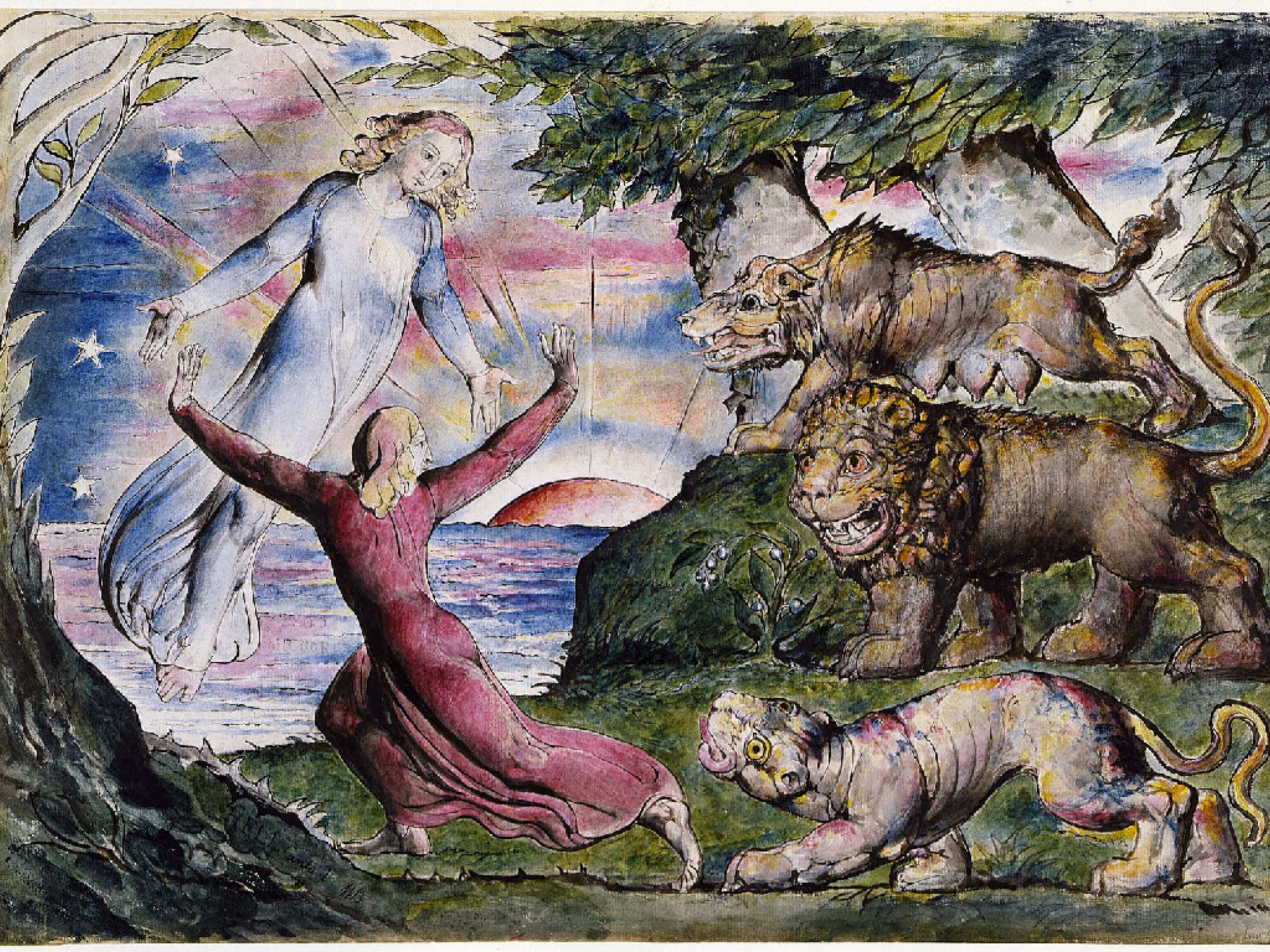
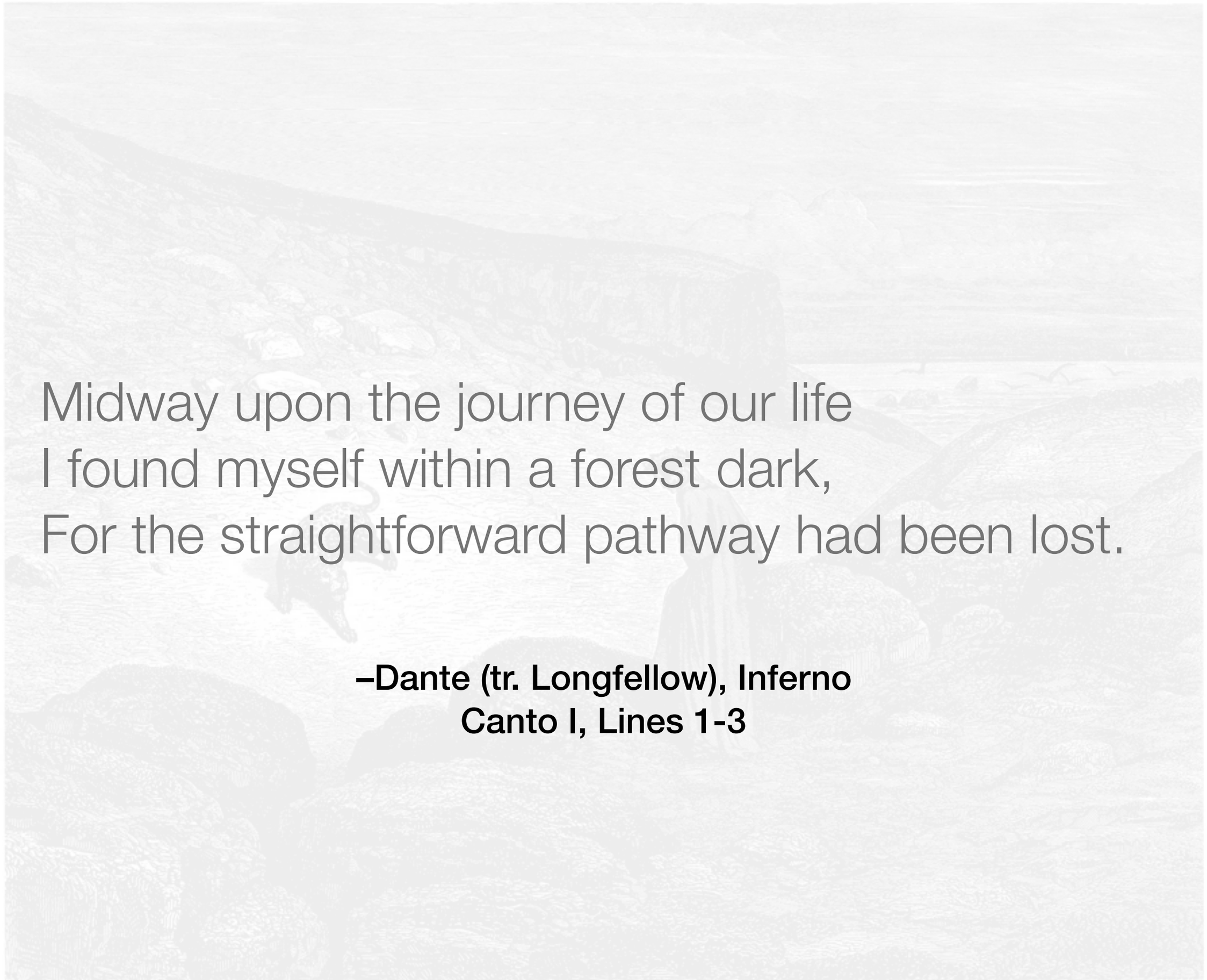
A Divine Data Comedy

Mike Harris







–Dante (tr. Longfellow), Inferno
Canto I, Lines 1-3

Canto

- Data Flow in a System
- The Language of Data Flow Manipulations
- Data Flow in Context

Canto

- **Data Flow in a System**
- The Language of Data Flow Manipulations
- Data Flow in Context

Example Data Set

```
create table stock_price (
    symbol varchar(5) not null
   ,price_date date not null
   ,price money not null
);
```

```
insert into stock_price
(symbol, price_date, price)
values
  ('ZVZZT', '2017-01-03', 10.02)
,('CBO'   , '2017-01-03', 18.99)
,('ZVZZT', '2017-01-04',  9.99)
,('CBO'   , '2017-01-04', 19.01)
;
```

Example Data Set

Symbol	Price Date	Price
ZVZZT	2017-01-03	10.02
CBO	2017-01-03	18.99
ZVZZT	2017-01-04	9.99
CBO	2017-01-04	19.01

SQL Statement

```
select symbol, price
   from stock_price
   where symbol = 'ZVZZT'
   order by price_date
;
```

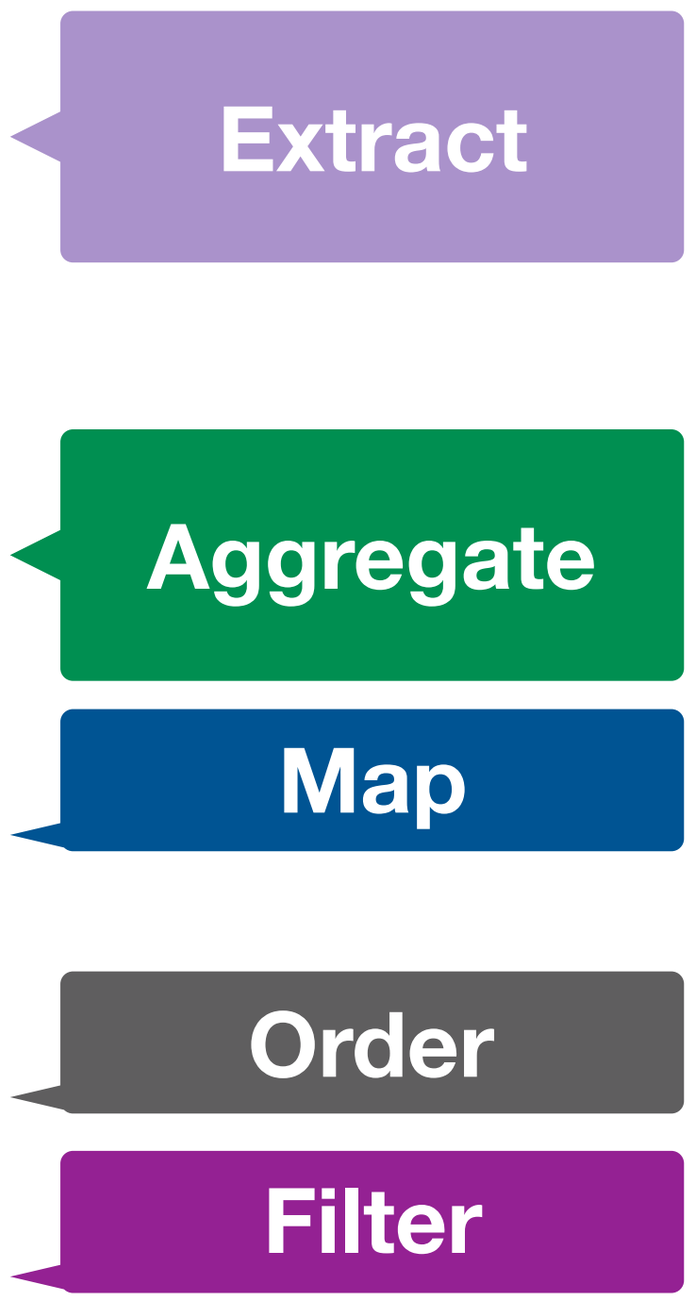
Symbol	Price Date	Price
ZVZZT	2017-01-03	10.02
ZVZZT	2017-01-04	9.99

Logical Order of a T-SQL Statement

1. FROM
2. ON
3. JOIN
4. WHERE
5. GROUP BY
6. WITH ROLLUP
7. HAVING
8. SELECT
9. DISTINCT
10. ORDER BY
11. TOP

Logical Order of a T-SQL Statement

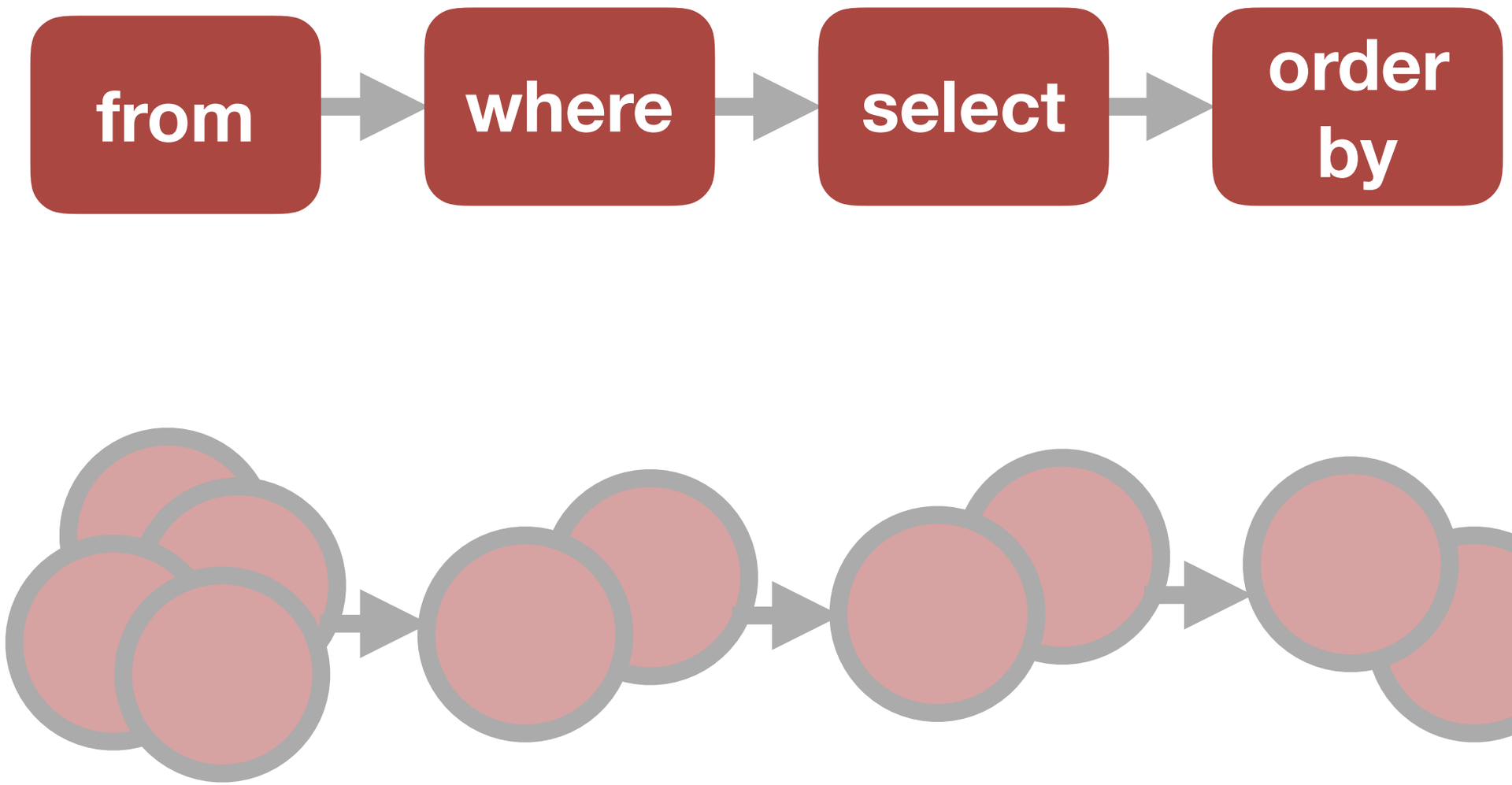
1. FROM
2. ON
3. JOIN
4. WHERE
5. GROUP BY
6. WITH ROLLUP
7. HAVING
8. SELECT
9. DISTINCT
10. ORDER BY
11. TOP



Logical Order of a T-SQL Statement

1. **FROM**
2. ON
3. JOIN
4. **WHERE**
5. GROUP BY
6. WITH ROLLUP
7. HAVING
8. **SELECT**
9. DISTINCT
10. **ORDER BY**
11. TOP

SQL Statement Data Flow



SQL Statement


```
select symbol, price
  from stock_price
 where symbol = 'ZVZZT'
 order by price_date
;
```

Symbol	Price Date	Price
ZVZZT	2017-01-03	10.02
ZVZZT	2017-01-04	9.99

Folded SQL

Folded SQL Statement

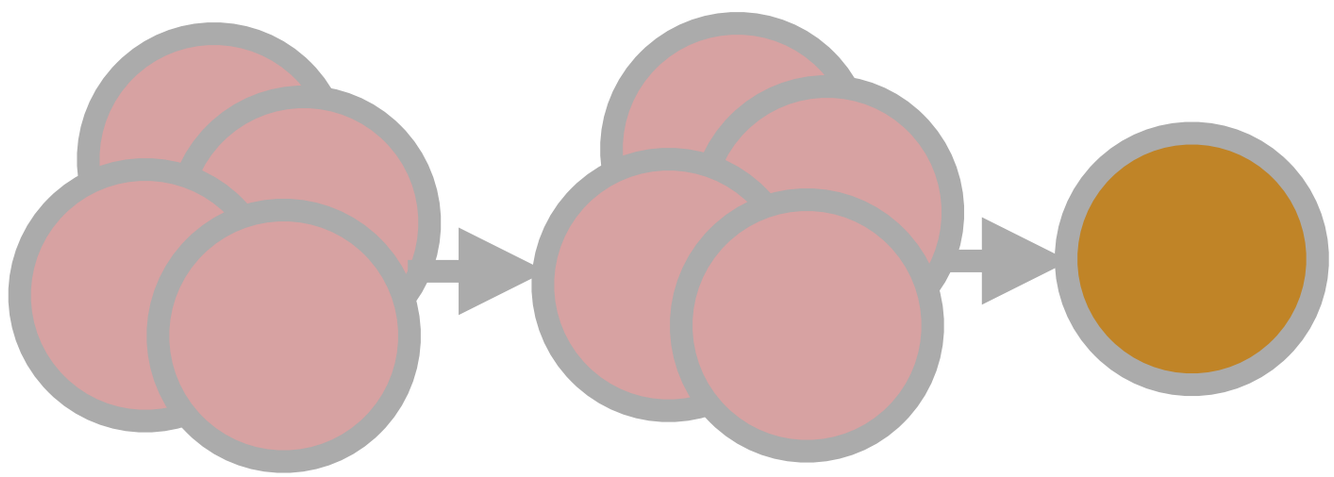
```
select max(price_date) as price_date
  from stock_price
;
```

Price Date
2017-01-04

Logical Order of a T-SQL Statement

1. **FROM**
2. ON
3. JOIN
4. WHERE
5. GROUP BY
6. WITH ROLLUP
7. HAVING
8. **SELECT**
9. DISTINCT
10. ORDER BY
11. TOP

Folded SQL Statement Data Flow



Folded SQL Statement

```
2  select max(price_date) as price_date
1    from stock_price
   ;
```

Price Date
2017-01-04

Group By Folded SQL

Group By Folded SQL Statement

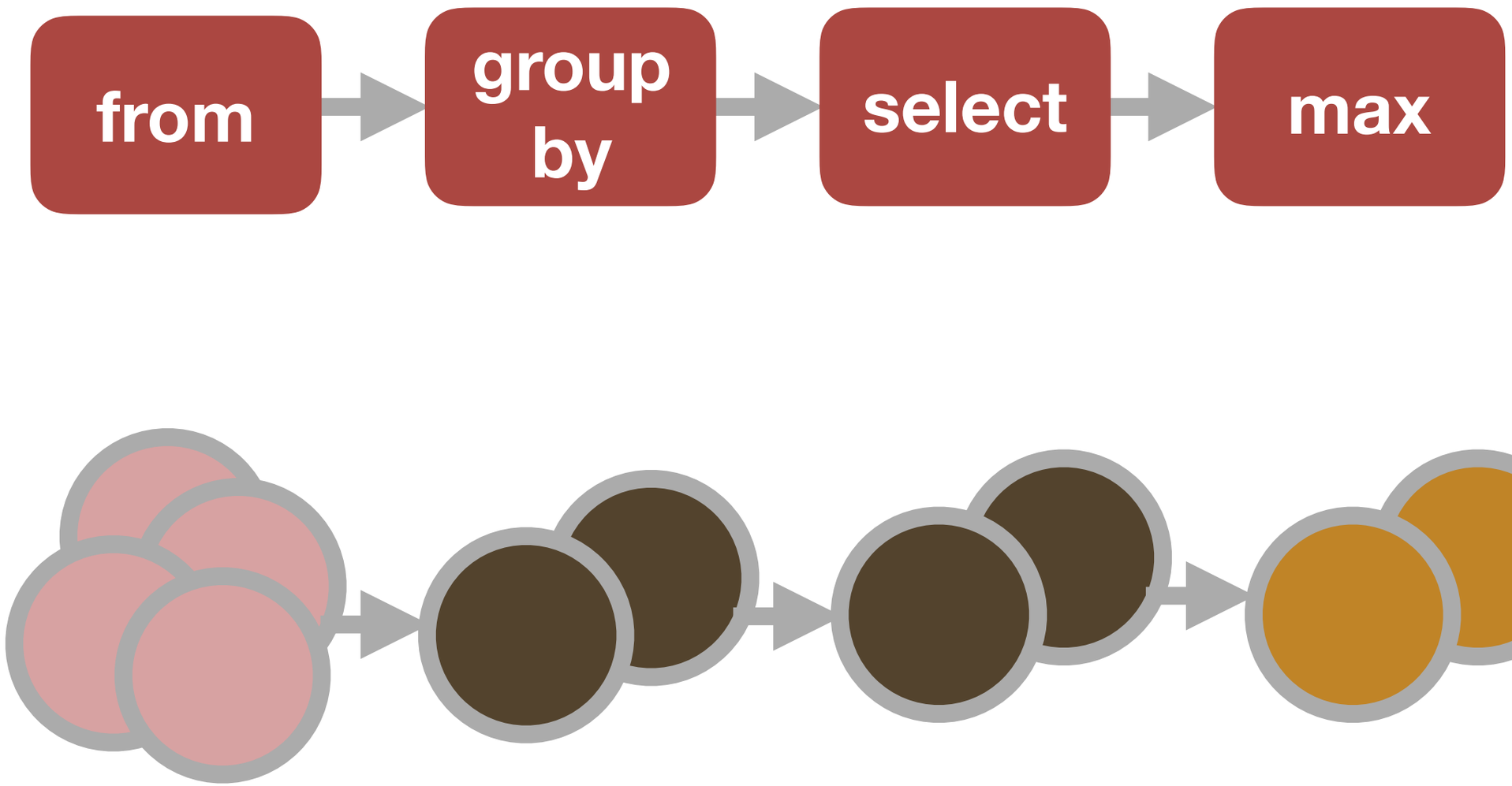
```
select
    symbol
   ,max(price_date) as price_date
   from stock_price
   group by symbol
;
```

Symbol	Price Date
CBO	2017-01-04
ZVZZT	2017-01-04

Logical Order of a T-SQL Statement

1. **FROM**
2. ON
3. JOIN
4. WHERE
5. **GROUP BY**
6. WITH ROLLUP
7. HAVING
8. **SELECT**
9. DISTINCT
10. ORDER BY
11. TOP

Group By Folded SQL Statement Data Flow



Group By Folded SQL Statement

```
3  select
       symbol
      ,max(price_date) as price_date
1  from stock_price
2  group by symbol
;
```

Symbol	Price Date
CBO	2017-01-04
ZVZZT	2017-01-04

Window Folded SQL

Window Folded SQL Statement

```
select distinct
    symbol
   ,max(price_date) over(
     partition by symbol) as price_date
   from stock_price
;
```

Symbol	Price Date
CBO	2017-01-04
ZVZZT	2017-01-04

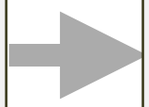
Logical Order of a T-SQL Statement

1. **FROM**
2. ON
3. JOIN
4. WHERE
5. GROUP BY
6. WITH ROLLUP
7. HAVING
8. **SELECT**
9. **DISTINCT**
10. ORDER BY
11. TOP

Window SQL Statement Data View

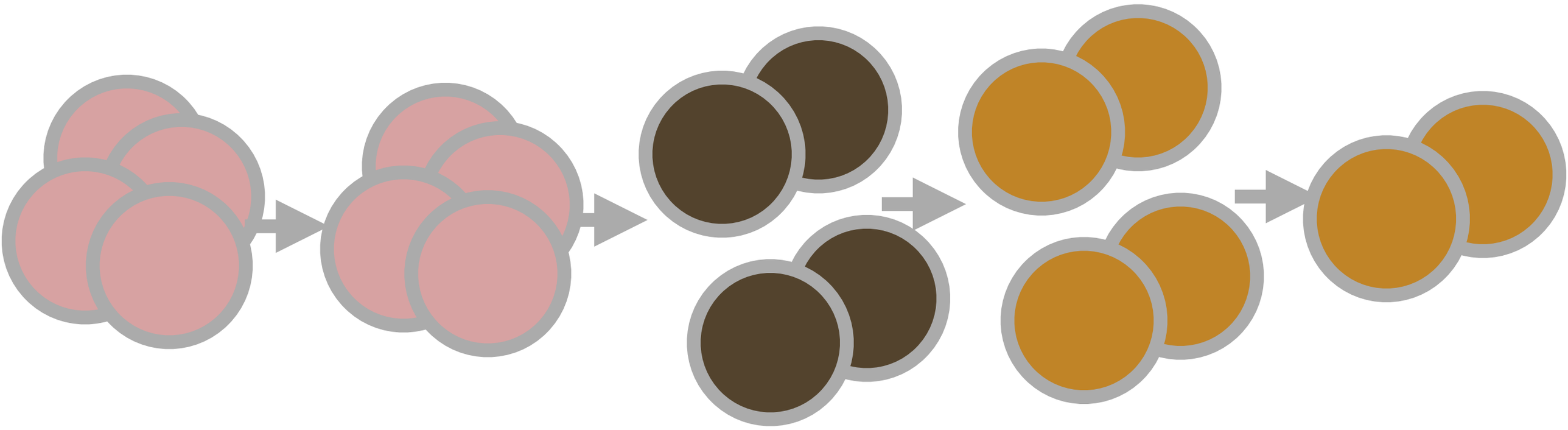
```
max(price_date) over(
 partition by symbol) as price_date
```

Symbol	Price_Date
CBO	2017-01-03
CBO	2017-01-04



Price_Date
2017-01-04
2017-01-04

Window SQL Statement Data Flow



Window Folded SQL Statement

```
  select distinct
2    symbol
    ,max(price_date) over(
      partition by symbol) as price_date
1  from stock_price
;
```

Symbol	Price Date
CBO	2017-01-04
ZVZZT	2017-01-04

Window SQL

Window SQL Statement

```
select
    symbol
   ,price
   ,lag(price, 1) over (
      partition by symbol
      order by price_date) as prev_price
   ,price - lag(price, 1) over (
      partition by symbol
      order by price_date) as price_change
   ,lead(price, 1) over (
     partition by symbol
     order by price_date) as next_price
   from stock_price
;
```

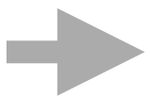
Window SQL Statement

Symbol	Price	prev_price	price_change	next_price
CBO	18.99	(null)	(null)	19.01
CBO	19.01	18.99	0.02	(null)
ZVZZT	10.02	(null)	(null)	9.99
ZVZZT	9.99	10.02	-0.03	(null)

Window SQL Statement Data View

```
lag(price, 1) over (
  partition by symbol
  order by price_date) as prev_price
```

Symbol	Price_Date	Price
CBO	2017-01-03	18.99
CBO	2017-01-04	19.01

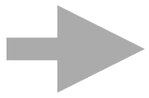


prev_price
(null)
18.99

Window SQL Statement Data View

```
lead(price, 1) over (
  partition by symbol
  order by price_date) as next_price
```

Symbol	Price_Date	Price
CBO	2017-01-03	18.99
CBO	2017-01-04	19.01

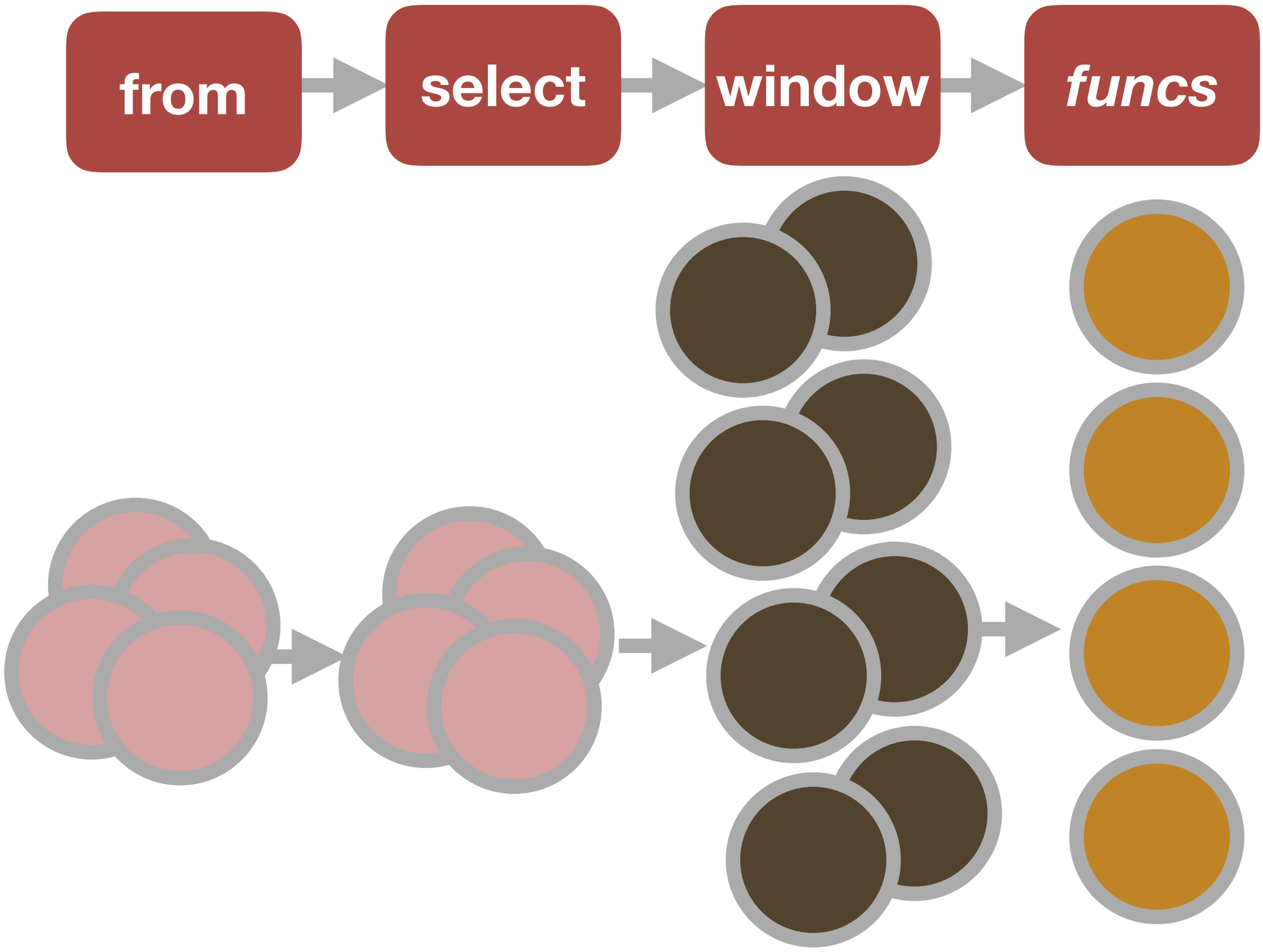


next_price
19.01
(null)

Logical Order of a T-SQL Statement

1. **FROM**
2. ON
3. JOIN
4. WHERE
5. GROUP BY
6. WITH ROLLUP
7. HAVING
8. **SELECT**
9. DISTINCT
10. ORDER BY
11. TOP

Window SQL Statement Data Flow



Window SQL Statement

```
select
    symbol
   ,price
   ,lag(price, 1) over (
      partition by symbol
      order by price_date) as prev_price
   ,price - lag(price, 1) over (
      partition by symbol
      order by price_date) as price_change
   ,lead(price, 1) over (
     partition by symbol
     order by price_date) as next_price
  from stock_price
;
```

2

1

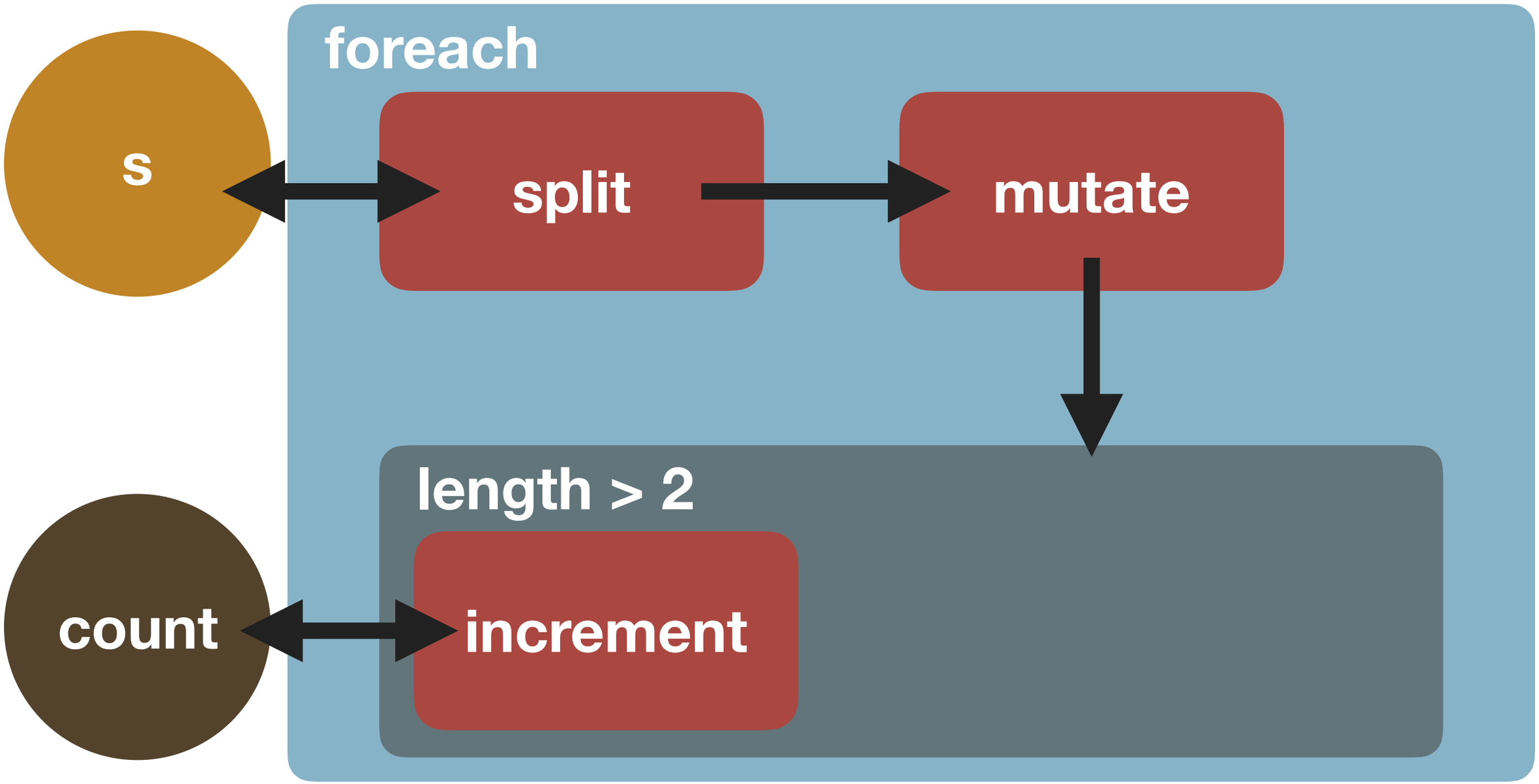
Canto

- Data Flow in a System
- **The Language of Data Flow Manipulations**
- Data Flow in Context

Imperative Example

```
var s = "Midway in our life's journey, I went astray";
var count = 0;
foreach(var word in s.Split(' '))
{
   var stripped =
     Regex.Replace(word, "('s)|\\W+", "");
   if (stripped.Length > 2)
     count++;
}
```

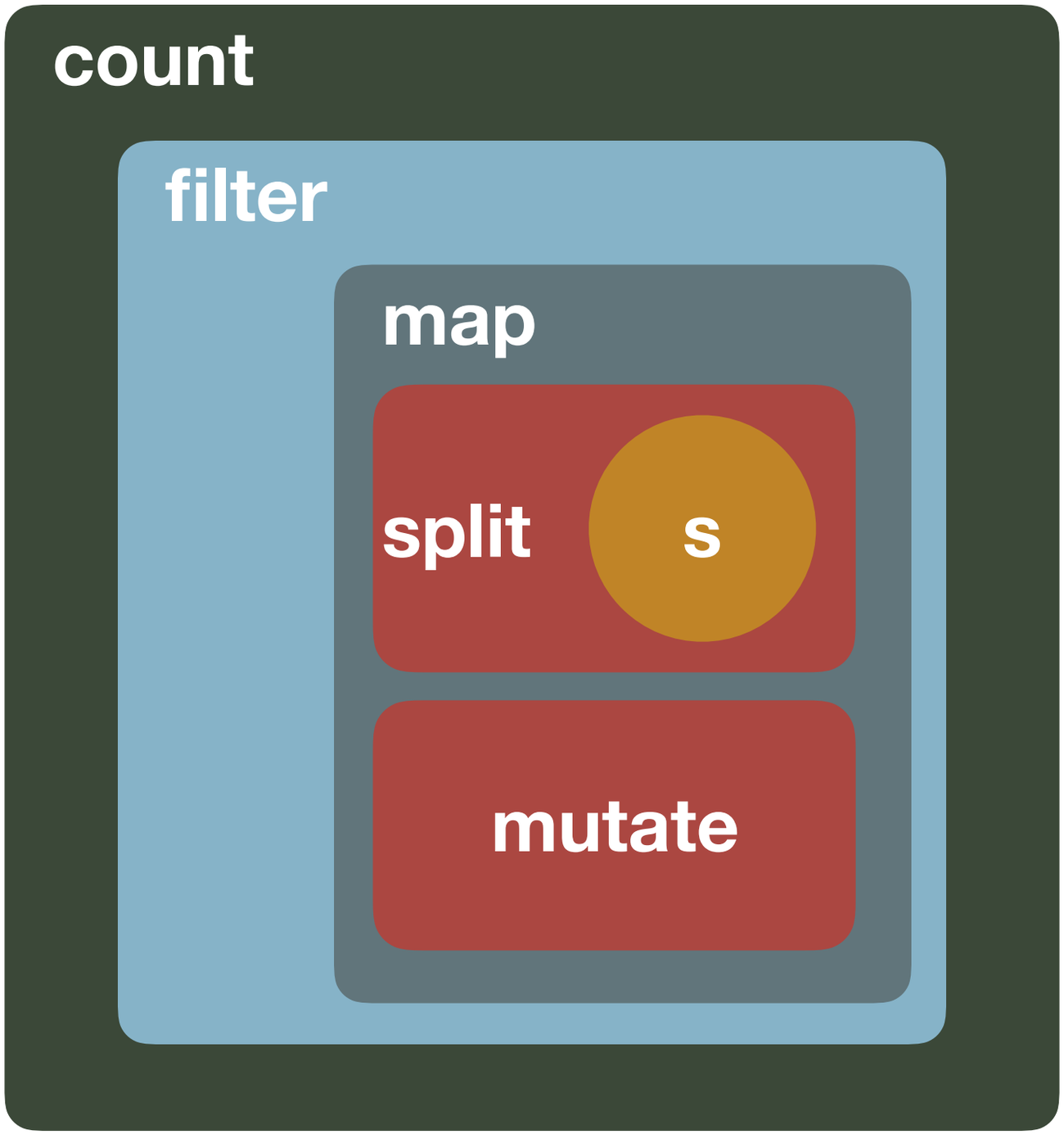
Imperative Data Flow



Nested Example

```
var s = "Midway in our life's journey, I went astray";
var count = Enumerable.Count(
  Enumerable.Where(
Enumerable.Select(
  s.Split(' '),
  word => Regex.Replace(
    word, "('s)|\\W+", "")),
stripped => stripped.Length > 2));
```

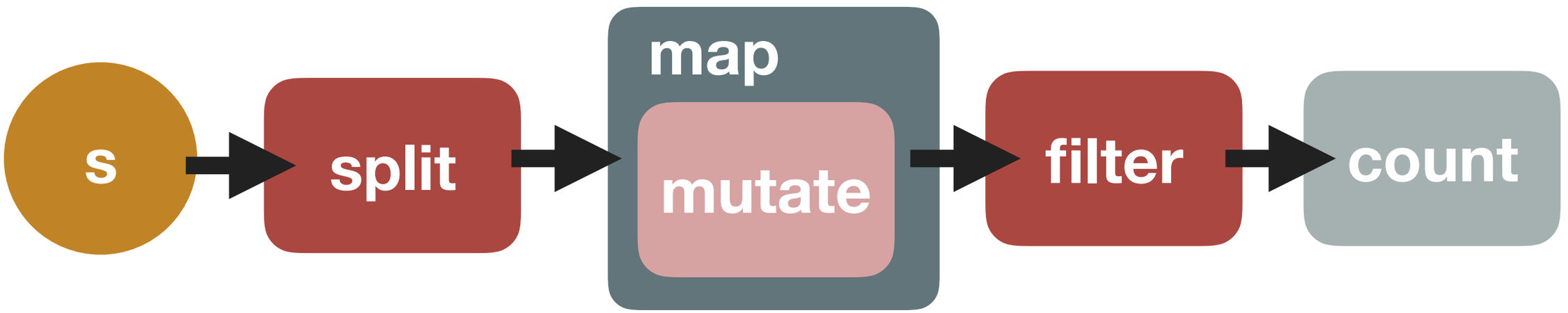
Nested Data Flow



Piped Example

```
var count = "Midway in our life's journey, I went astray”
  .Split(' ')
  .Select(word =>
    Regex.Replace(word, "('s)|\\W+", ""))
.Where(word => word.Length > 2)
.Count();
```

Piped Data Flow



The Language of Data Flow Manipulations

- map
- bind
- filter
- fold
- mutate
- group by
- order by

The Language of Data Flow Manipulations

- map: **Select**
- bind: **SelectMany**
- filter: **Where**
- fold: **Aggregate**
- mutate: **ForEach**
- group by: **GroupBy**
- order by: **OrderBy**

The Language of Data Flow Manipulations

- **map**
- bind
- filter
- fold
- mutate
- group by
- order by

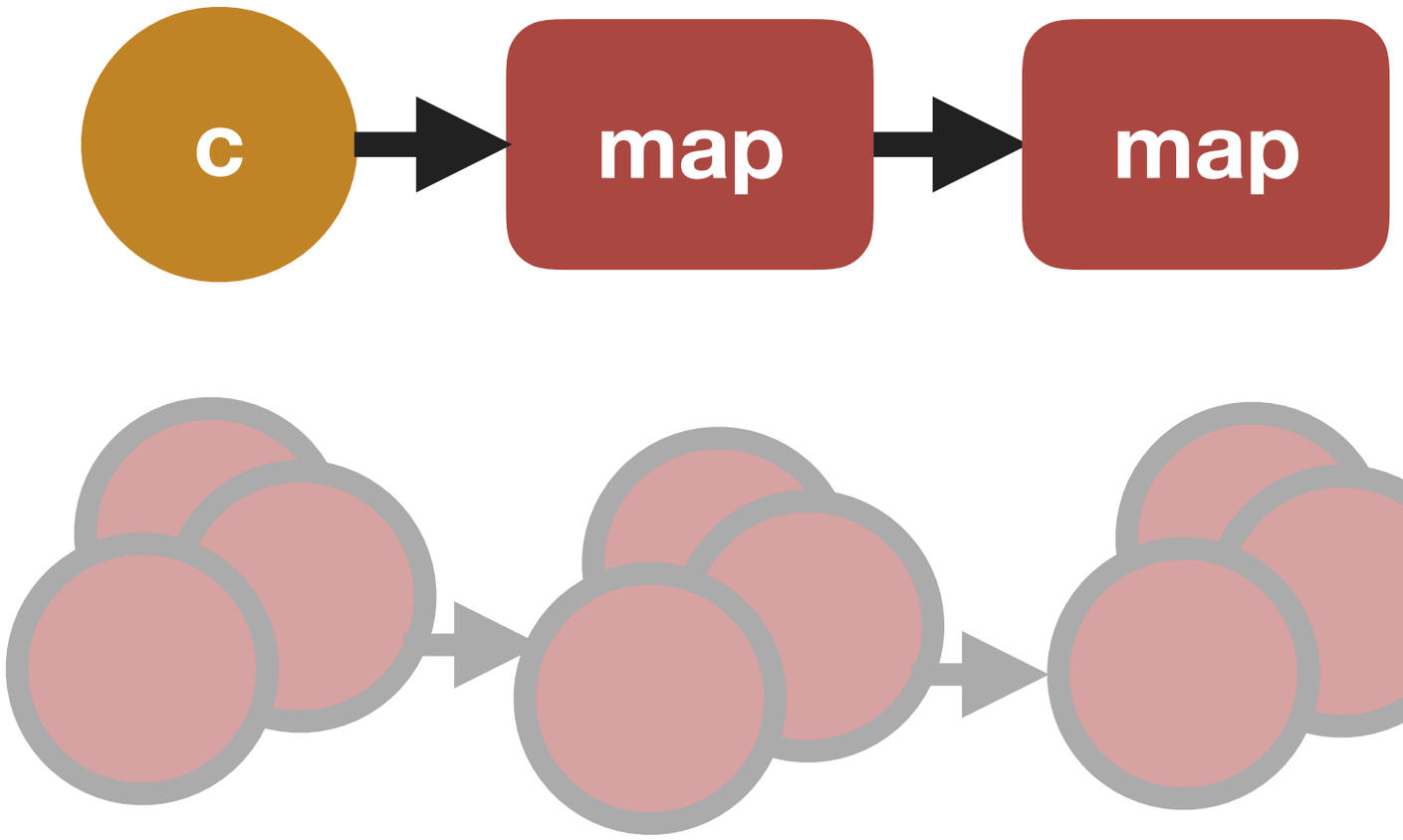
Map

Map Example

```
var numbers = new [] {3, 9, 10}
    .Select(x => x * 10)
    .Select(x => x + 3);
```

```
output>
new [] {33, 93, 103}
```

Map Data Flow



Select Source Code

```
private sealed class SelectArrayIterator<TSource, TResult> : Iterator<TResult>, IPartition<TResult>
{
  private readonly TSource[] _source;
  private readonly Func<TSource, TResult> _selector;

  public SelectArrayIterator(TSource[] source, Func<TSource, TResult> selector)
  {
    // ... assert we have something
    _source = source;
    _selector = selector;
  }

  public override bool MoveNext()
  {
    if (_state < 1 | _state == _source.Length + 1)
    {
      Dispose();
      return false;
    }

    int index = _state++ - 1;
    _current = _selector(_source[index]);
    return true;
  }
  // ... and more
```

https://github.com/dotnet/corefx/blob/7a673547563dc92f1f4cad802d7b57483cdf7500/src/System.Linq/src/System/Linq/Select.cs#L199-L226

Select Source Code - constructor

```
private sealed class SelectArrayIterator
  <TSource, TResult> : Iterator<TResult>,
                       IPartition<TResult>
{
  private readonly TSource[] _source;
  private readonly Func<TSource, TResult> _selector;

  public SelectArrayIterator(
    TSource[] source,
    Func<TSource, TResult> selector)
  {
    // ... assert we have something
    _source = source;
    _selector = selector;
  }
```

https://github.com/dotnet/corefx/blob/7a673547563dc92f1f4cad802d7b57483cdf7500/src/System.Linq/src/System/Linq/Select.cs#L199-L226

Select Source Code

```
public override bool MoveNext()
 {
    if (_state < 1 | _state == _source.Length + 1)
    {
       Dispose();
       return false;
    }

    int index = _state++ - 1;
    _current = _selector(_source[index]);
    return true;
}
```

https://github.com/dotnet/corefx/blob/7a673547563dc92f1f4cad802d7b57483cdf7500/src/System.Linq/src/System/Linq/Select.cs#L199-L226

Map Example

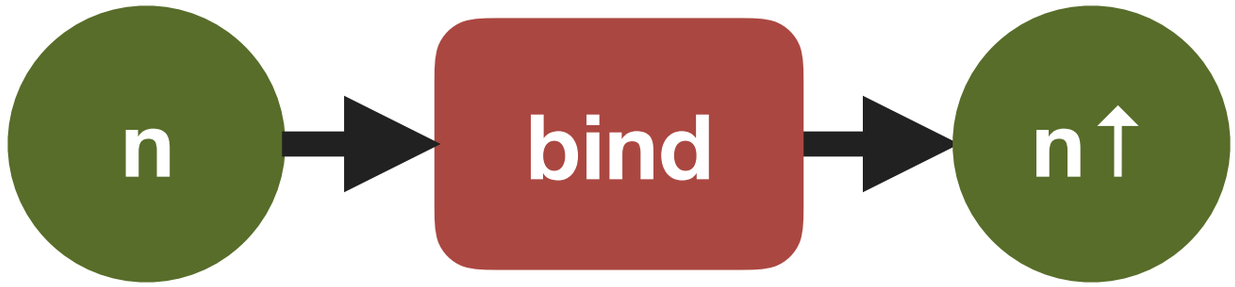
```
var numbers = new [] {3, 9, 10}
    .Select(x => x * 10)
    .Select(x => x + 3);
```

```
output>
new [] {33, 93, 103}
```

The Language of Data Flow Manipulations

- map
- **bind**
- filter
- fold
- mutate
- group by
- order by

Bind



Bind Example

```
var list = new [] {
    "Midway in our life's journey,",
    "I went astray"
  }
  .SelectMany(s => s.Split(' '))
  .Select(w => Regex.Replace(w, "('s)|\\W+", ""));
```

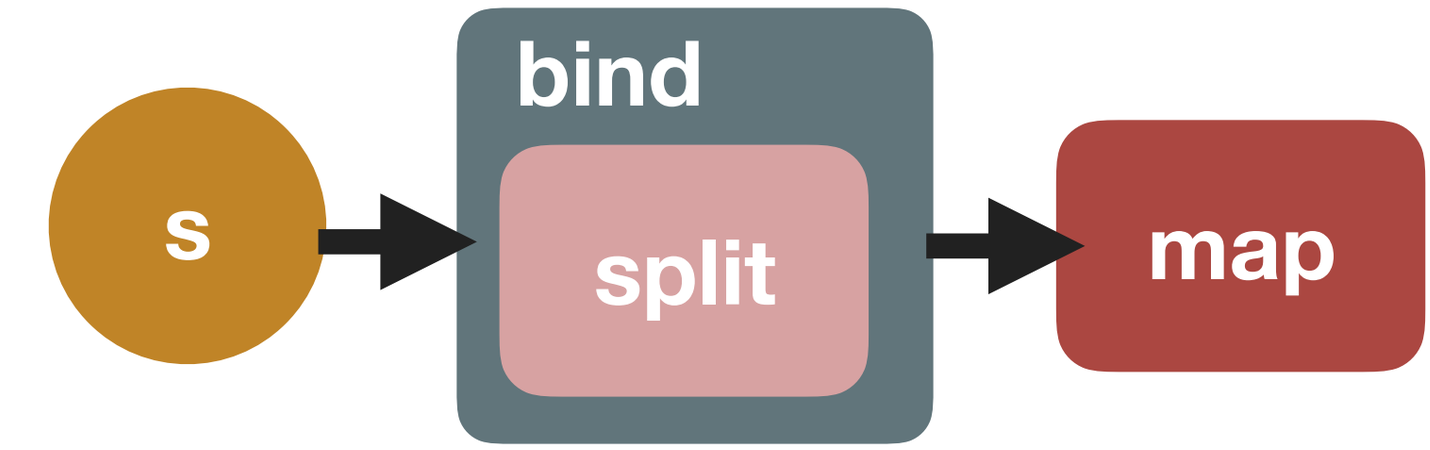
```
output>
new [] { "Midway", "in", “our",
         "life", "journey", “I",
         “went", "astray" }
```

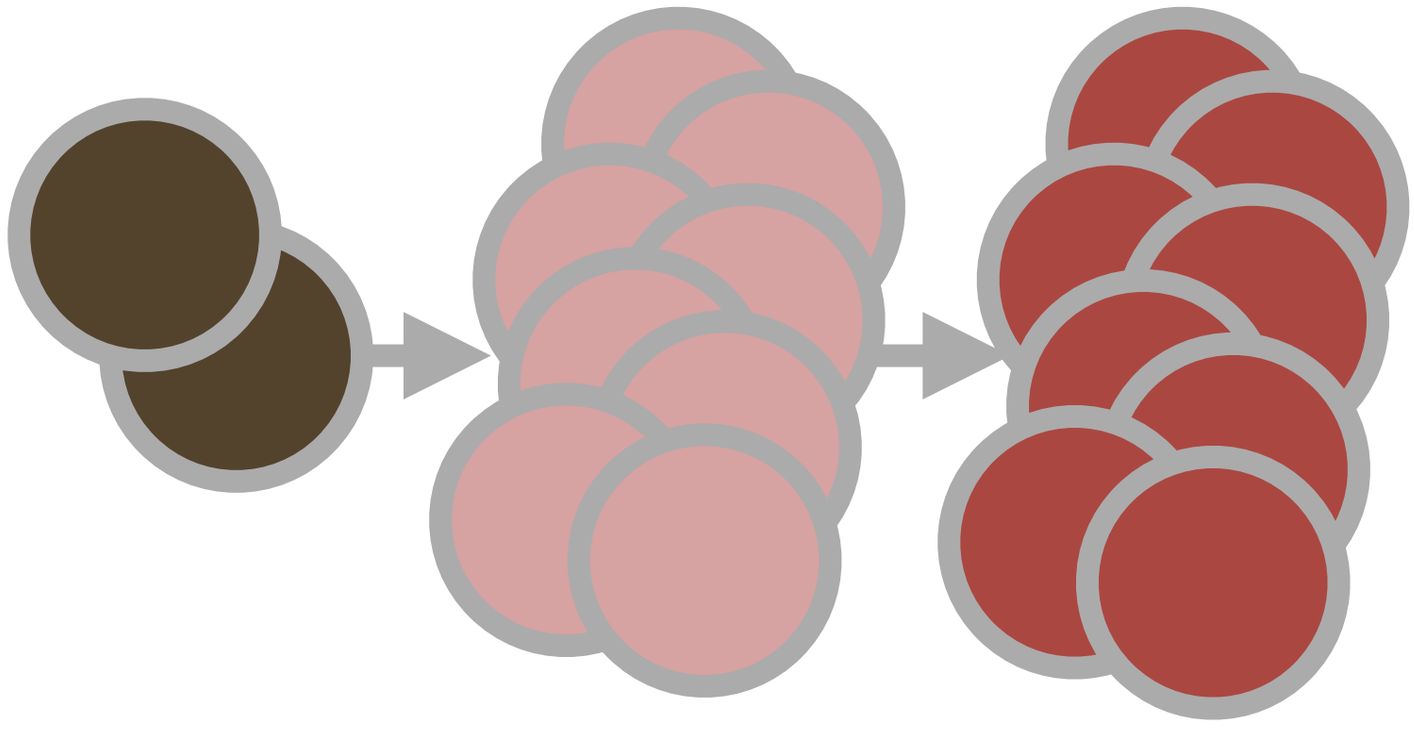
Bind Query Syntax Example

```
var list =
  from sentences in
    new [] {
      "Midway in our life's journey,",
      "I went astray"
    }
from words in sentences.Split(' ')
select Regex.Replace(words, "('s)|\\W+", "");
```

```
output>
new [] { "Midway", "in", “our",
         "life", "journey", “I",
         “went", "astray" }
```

Bind Query Syntax Data Flow





SelectMany Source Code - constructor

```
private sealed class SelectManySingleSelectorIterator
  <TSource, TResult> : Iterator<TResult>
                      ,IIListProvider<TResult>
{
  private readonly IEnumerable<TSource> _source;
  private readonly Func<TSource, IEnumerable<TResult>> _selector;
  private IEnumerator<TSource> _sourceEnumerator;
  private IEnumerator<TResult> _subEnumerator;

  internal SelectManySingleSelectorIterator(
    IEnumerable<TSource> source,
    Func<TSource, IEnumerable<TResult>> selector)
  {
      // ... assert we have something
      _source = source;
      _selector = selector;
  }
  // ... and more
```

https://github.com/dotnet/corefx/blob/7a673547563dc92f1f4cad802d7b57483cdf7500/src/System.Linq/src/System/Linq/SelectMany.cs#L127-L223

SelectMany Source Code

```
public override bool MoveNext()
{
    switch (_state)
    {
        case 1:
            // Retrieve the source enumerator.
            _sourceEnumerator = _source.GetEnumerator();
            _state = 2;
            goto case 2;
        case 2:
            // Take the next element from the source enumerator.
```

https://github.com/dotnet/corefx/blob/7a673547563dc92f1f4cad802d7b57483cdf7500/src/System.Linq/src/System/Linq/SelectMany.cs#L127-L223

SelectMany Source Code

```
case 2:
  // Take the next element from the source enumerator.
  if (!_sourceEnumerator.MoveNext())
  {
      break;
  }
  TSource element = _sourceEnumerator.Current;
  // Project it into a sub-collection and get its enumerator.
  _subEnumerator = _selector(element).GetEnumerator();
  _state = 3;
  goto case 3;
case 3:
  // Take the next element from the sub-collection and yield.
```

https://github.com/dotnet/corefx/blob/7a673547563dc92f1f4cad802d7b57483cdf7500/src/System.Linq/src/System/Linq/SelectMany.cs#L127-L223

SelectMany Source Code

```
case 3:
  // Take the next element from the sub-collection and yield.
  if (!_subEnumerator.MoveNext())
  {
    _subEnumerator.Dispose();
    _subEnumerator = null;
    _state = 2;
    goto case 2;
  }
  _current = _subEnumerator.Current;
  return true;
```

https://github.com/dotnet/corefx/blob/7a673547563dc92f1f4cad802d7b57483cdf7500/src/System.Linq/src/System/Linq/SelectMany.cs#L127-L223

Bind Example

```
var list = new [] {
    "Midway in our life's journey, I went astray"
  }
  .SelectMany(s => s.Split(' '))
  .Select(w => Regex.Replace(w, "('s)|\\W+", ""));
```

```
output>
new [] { "Midway", "in", “our",
         "life", "journey", “I",
         “went", "astray" }
```

The Language of Data Flow Manipulations

- map
- bind
- **filter**
- fold
- mutate
- group by
- order by

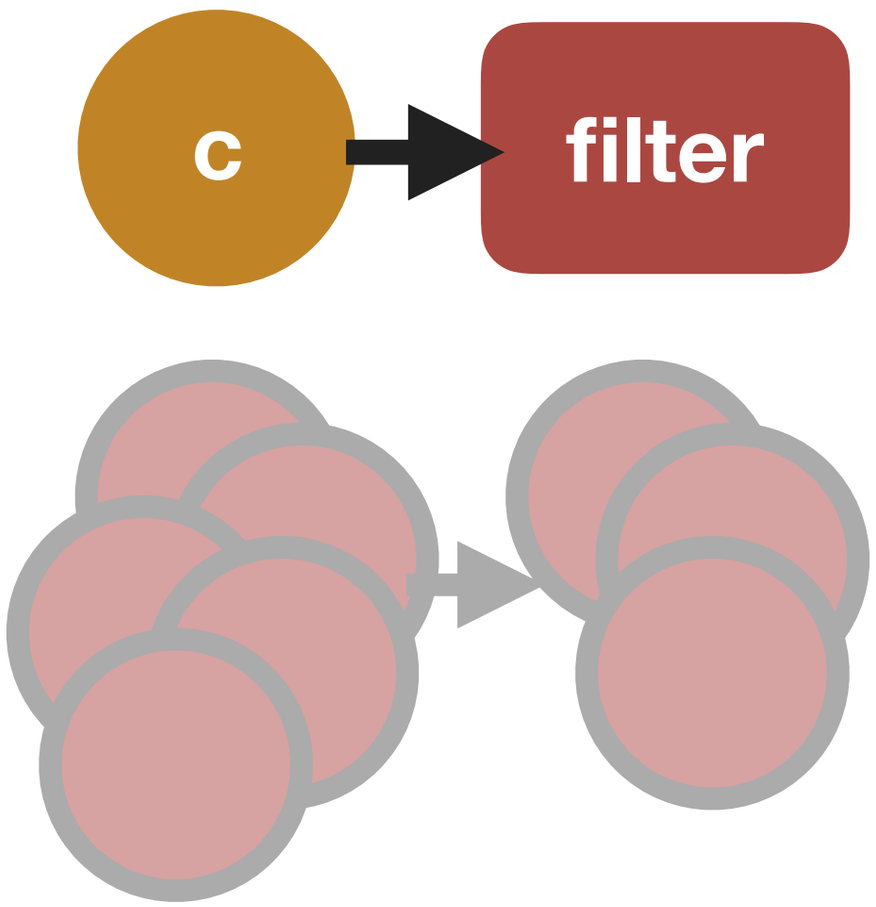
Filter

Filter Example

```
var numbers = new [] {3, 9, 10, 33, 100}
.Where(n => n % 2 != 0);
```

```
output>
new [] {3, 9, 33}
```

Filter Data Flow



Where Source Code - constructor

```
internal sealed class WhereArrayIterator<TSource>
  : Iterator<TSource>, IIListProvider<TSource>
{
    private readonly TSource[] _source;
    private readonly Func<TSource, bool> _predicate;
```

```
    public WhereArrayIterator(
      TSource[] source,
      Func<TSource, bool> predicate)
    {
        // ... assert we have something
        _source = source;
        _predicate = predicate;
    }
```

https://github.com/dotnet/corefx/blob/7a673547563dc92f1f4cad802d7b57483cdf7500/src/System.Linq/src/System/Linq/Where.cs#L198-L255

Where Source Code

```
public override bool MoveNext()
{
    int index = _state - 1;
    TSource[] source = _source;

    while (unchecked((uint)index < (uint)source.Length))
    {
        TSource item = source[index];
        index = _state++;
        if (_predicate(item))
        {
            _current = item;
            return true;
        }
    }

    Dispose();
    return false;
}
// ... and more
```

https://github.com/dotnet/corefx/blob/7a673547563dc92f1f4cad802d7b57483cdf7500/src/System.Linq/src/System/Linq/Where.cs#L198-L255

Filter Example

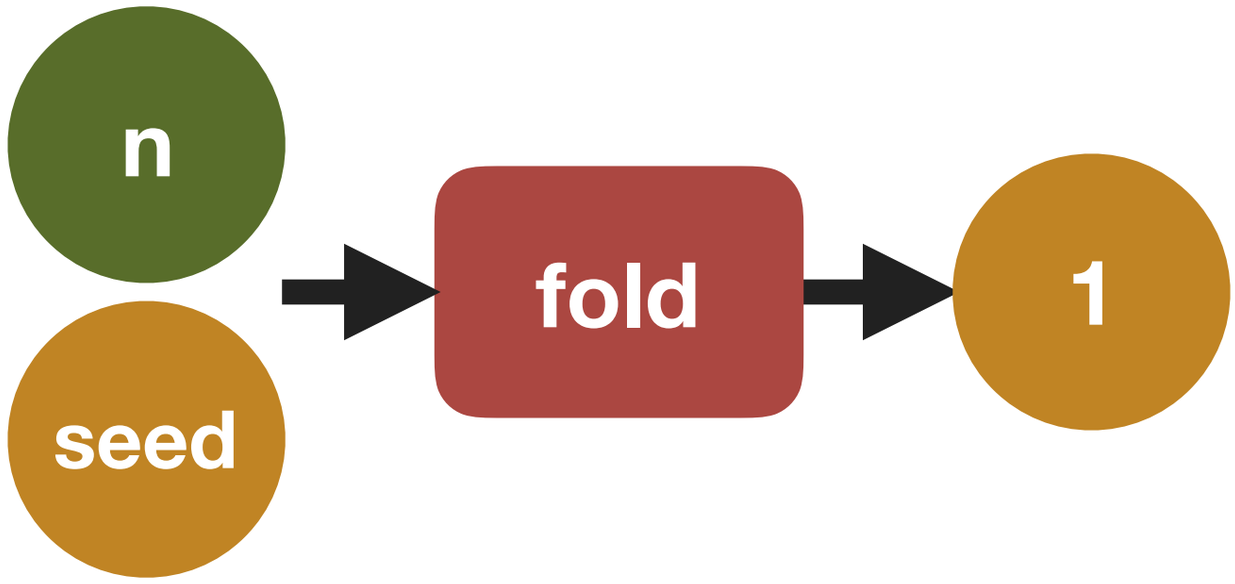
```
var numbers = new [] {3, 9, 10, 33, 100}
.Where(n => n % 2 != 0);
```

```
output>
new [] {3, 9, 33}
```

The Language of Data Flow Manipulations

- map
- bind
- filter
- **fold**
- mutate
- group by
- order by

Fold

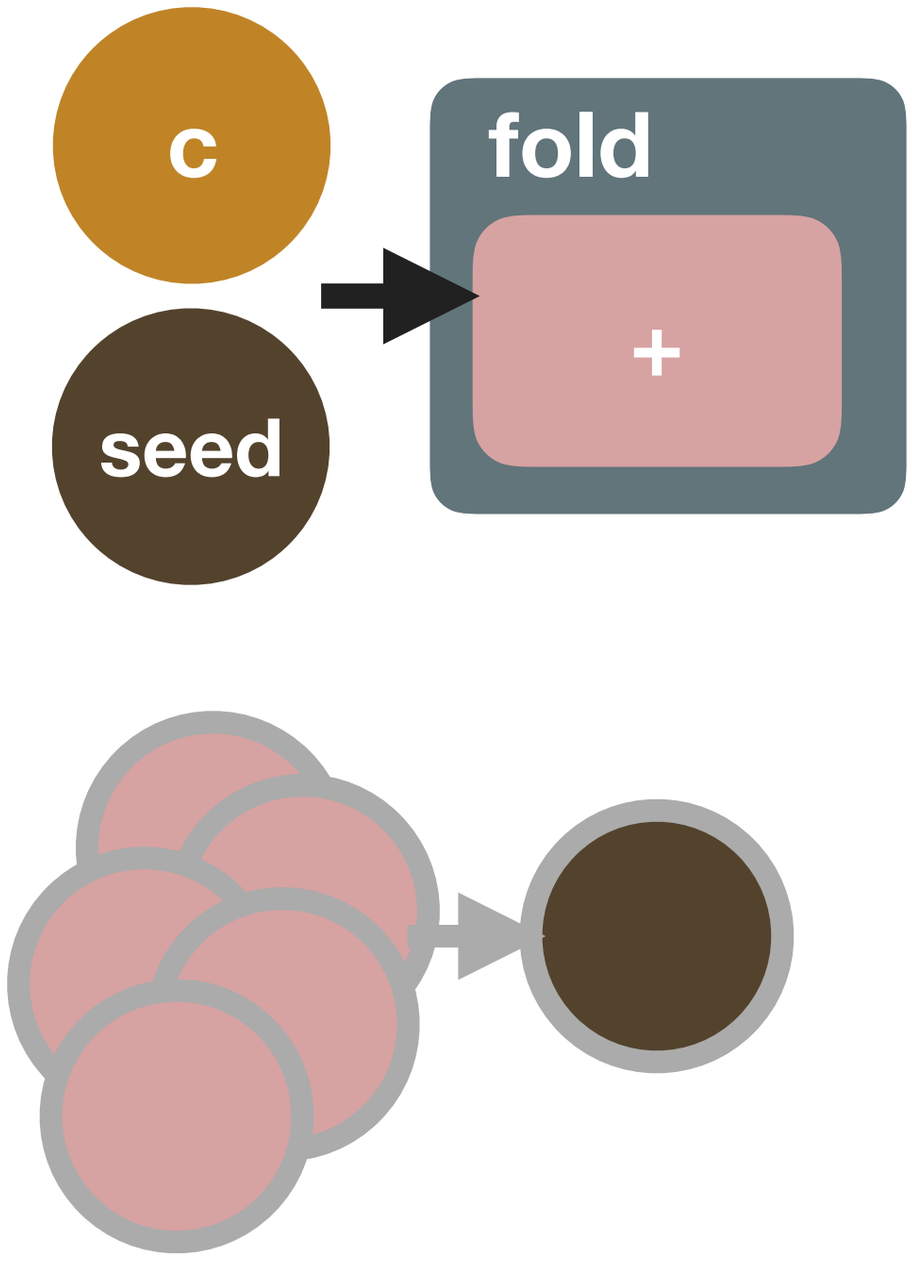


Fold Example

```
var sum = new [] {3, 9, 10, 33, 100}
.Aggregate(0, (m, n) => m + n);
```

```
output>
155
```

Fold Data Flow



Aggregate Source Code

```
public static TAccumulate Aggregate<TSource, TAccumulate>(
  this IEnumerable<TSource> source,
  TAccumulate seed,
  Func<TAccumulate, TSource, TAccumulate> func)
{
    // ... assert we have something
    TAccumulate result = seed;
    foreach (TSource element in source)
    {
        result = func(result, element);
    }

    return result;
}
```

https://github.com/dotnet/corefx/blob/7a673547563dc92f1f4cad802d7b57483cdf7500/src/System.Linq/src/System/Linq/Aggregate.cs#L40-L59

Fold Example

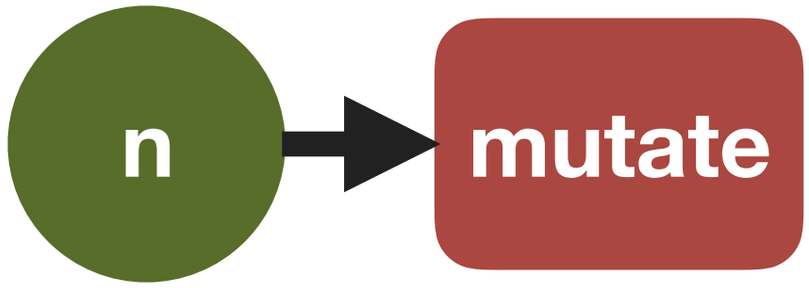
```
var sum = new [] {3, 9, 10, 33, 100}
.Aggregate(0, (m, n) => m + n);
```

```
output>
155
```

The Language of Data Flow Manipulations

- map
- bind
- filter
- fold
- **mutate**
- group by
- order by

Mutate



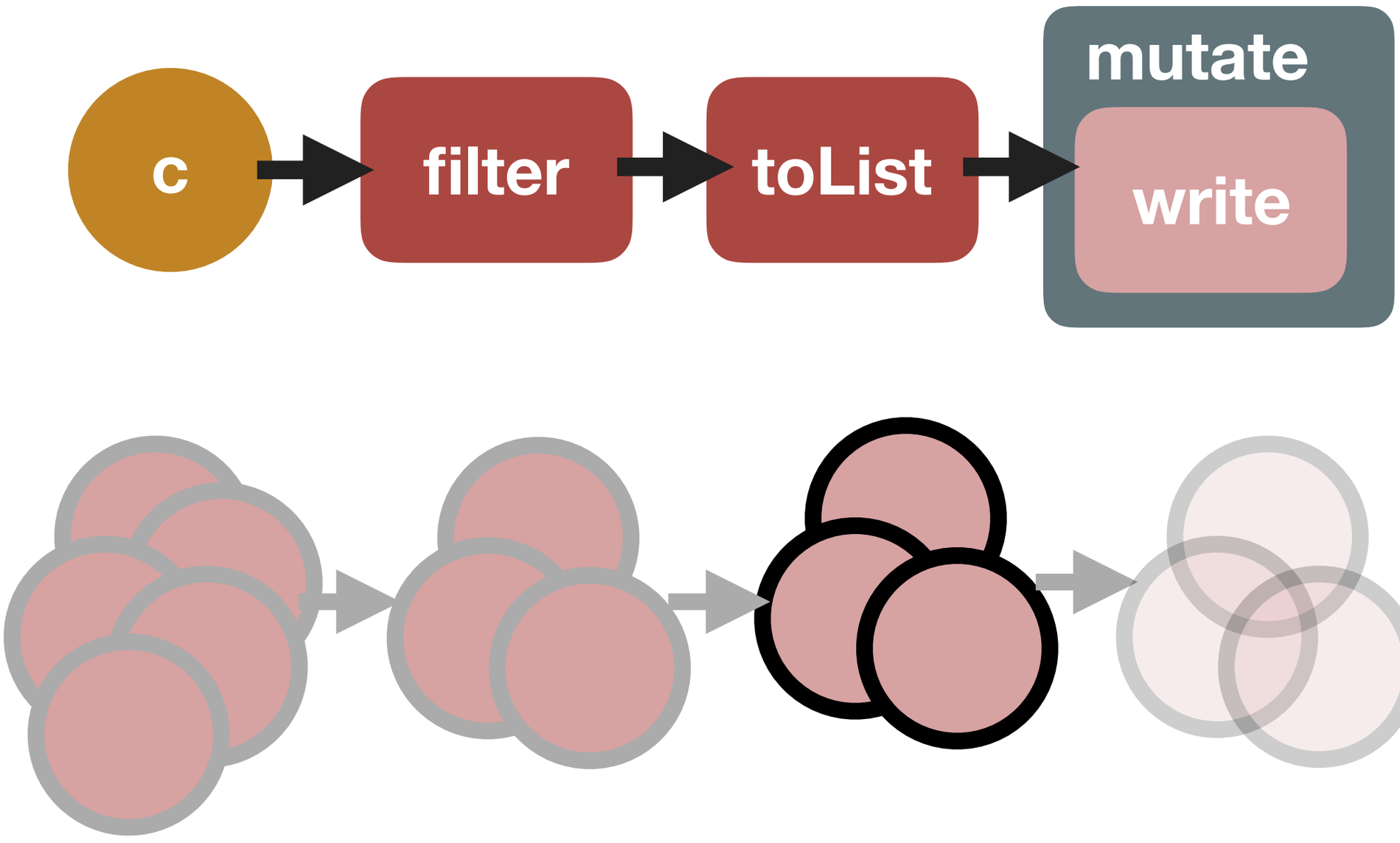
Mutate Example

```
var visited = new List<int>();
new [] {3, 9, 10, 33, 100}
   .Where(n => n % 3 == 0)
   .ToList()
   .ForEach(n => visited.Add(n));
```

```
output>
//void

var visited = new [] {3, 9, 33}
```

Mutate Data Flow



ForEach Source Code

```
public void ForEach(Action<T> action)
{
    // ... assert we have something
    int version = _version;
    for (int i = 0; i < _size; i++)
    {
        if (version !=  version)
        {
            break;
        }
        action(_items[i]);
    }
    // cannot alter list
    if (version != _version)
        throw new InvalidOperationException(
            SR.InvalidOperation_EnumFailedVersion);
}
```

https://github.com/dotnet/corefx/blob/bffef76f6af208e2042a2f27bc081ee908bb390b/src/System.Collections/src/System/Collections/Generic/List.cs#L598-L618

Mutate Example

```
var visited = new List<int>();
new [] {3, 9, 10, 33, 100}
   .Where(n => n % 3 == 0)
   .ToList()
   .ForEach(n => visited.Add(n));
```

```
output>
//void

var visited = new [] {3, 9, 33}
```

The Language of Data Flow Manipulations

- map
- bind
- filter
- fold
- mutate
- **group by**
- order by

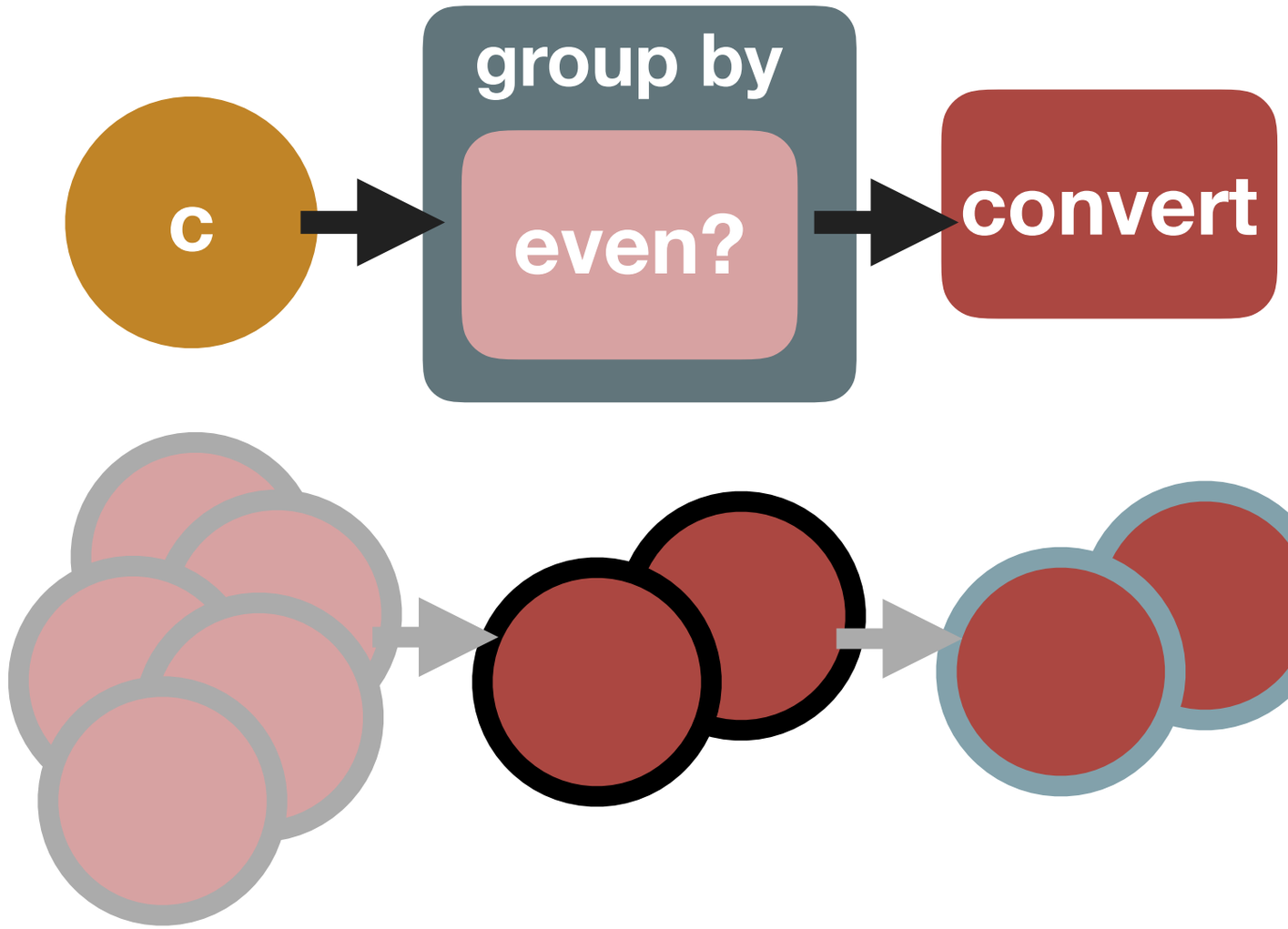
Group By

Group By Example

```
var groups = new [] {3, 9, 10, 33, 100}
  .GroupBy(n => n % 2 == 0)
  .ToDictionary(g => g.Key, g => g.ToList());
```

```
output>
new Dictionary<bool, List<int>>()
{
  {true, new List<int>() {10, 100}},
  {false, new List<int>() {3, 9, 33}}
}
```

Group By Data Flow



GroupBy Source Code

```
internal static Lookup<TKey, TElement> Create(
  IEnumerable<TElement> source,
  Func<TElement, TKey> keySelector,
  IEqualityComparer<TKey> comparer)
{
    // ... assert we have something
    Lookup<TKey, TElement> lookup =
      new Lookup<TKey, TElement>(comparer);
    foreach (TElement item in source)
    {
        lookup.GetGrouping(
          keySelector(item),create: true).Add(item);
    }

    return lookup;
}
```

https://github.com/dotnet/corefx/blob/7a673547563dc92f1f4cad802d7b57483cdf7500/src/System.Linq/src/System/Linq/Lookup.cs#L88-L100

Group By Example

```
var groups = new [] {3, 9, 10, 33, 100}
  .GroupBy(n => n % 2 == 0)
  .ToDictionary(g => g.Key, g => g.ToList());
```

```
output>
new Dictionary<bool, List<int>>()
{
  {true, new List<int>() {10, 100}},
  {false, new List<int>() {3, 9, 33}}
}
```

The Language of Data Flow Manipulations

- map
- bind
- filter
- fold
- mutate
- group by
- **order by**

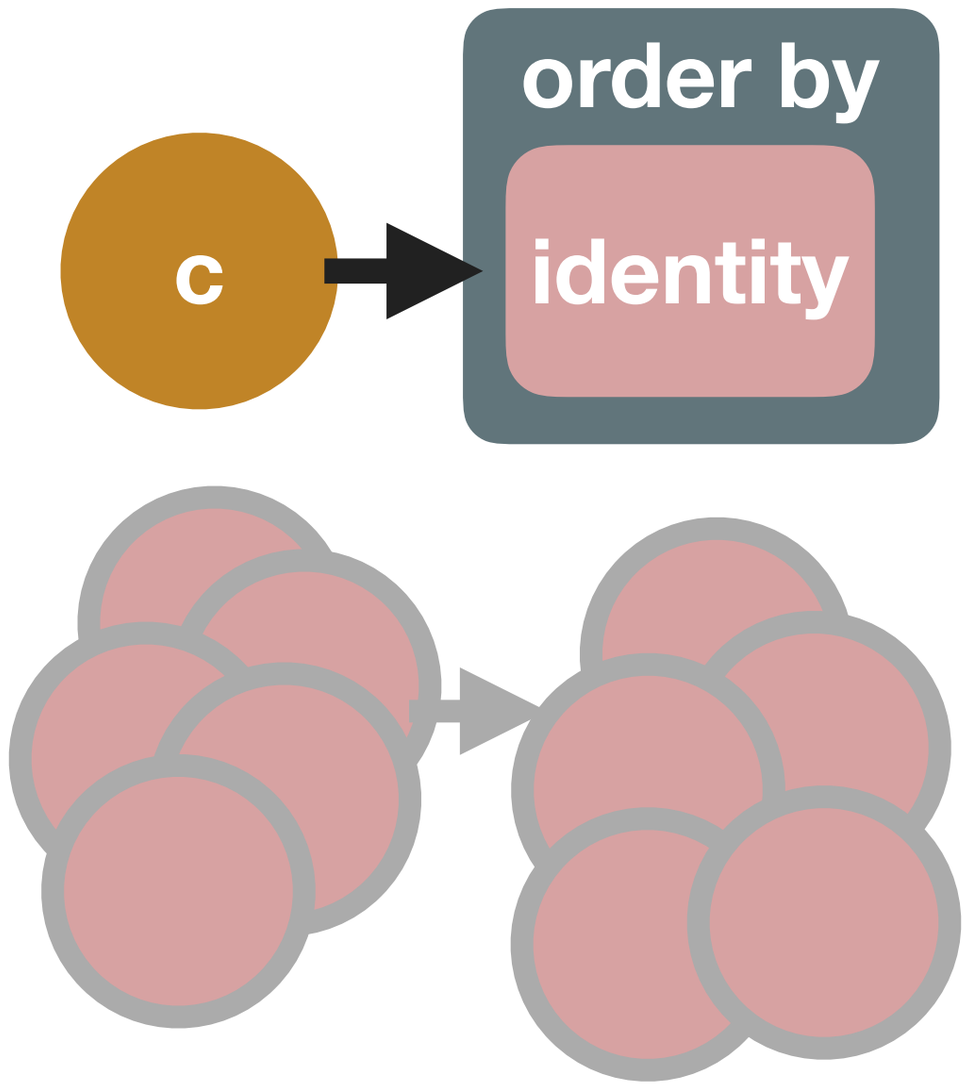
Order By

Order By Example

```
var unordered = new [] {33, 9, 100, 3, 10}
.OrderBy(n => n);
```

```
output>
new [] {3, 9, 10, 33, 100}
```

Order By Data Flow



OrderBy Source Code

```
internal int[] Sort(TElement[] elements, int count)
{
    int[] map = ComputeMap(elements, count);
    QuickSort(map, 0, count - 1);
    return map;
}
```

https://github.com/dotnet/corefx/blob/f17f1e847aeab830de77f8a46656339d7b0f1b43/src/System.Linq/src/System/Linq/OrderedEnumerable.cs#L508-L513

Order By Example

```
var unordered = new [] {33, 9, 100, 3, 10}
.OrderBy(n => n);
```

```
output>
new [] {3, 9, 10, 33, 100}
```

Canto

- Data Flow in a System
- The Language of Data Flow Manipulations
- **Data Flow in Context**

You shall leave everything you love most dearly:
this is the arrow that the bow of exile
shoots first. You are to know the bitter taste
of others' bread, how salt it is, and know
how hard a path it is for one who goes
descending and ascending others' stairs.

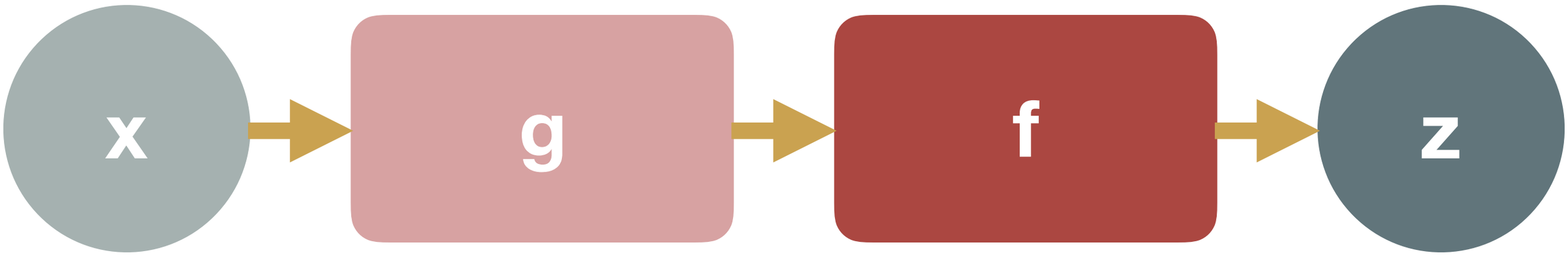
–Dante (tr. Barolini), Paradiso
Canto XVII, Lines 55-60

Domain and Codomain

Domain and Codomain

Symbols	Natural Language
$x \in X$	**X** is domain of **f**
$y \in Y$	**Y** is codomain of **f**

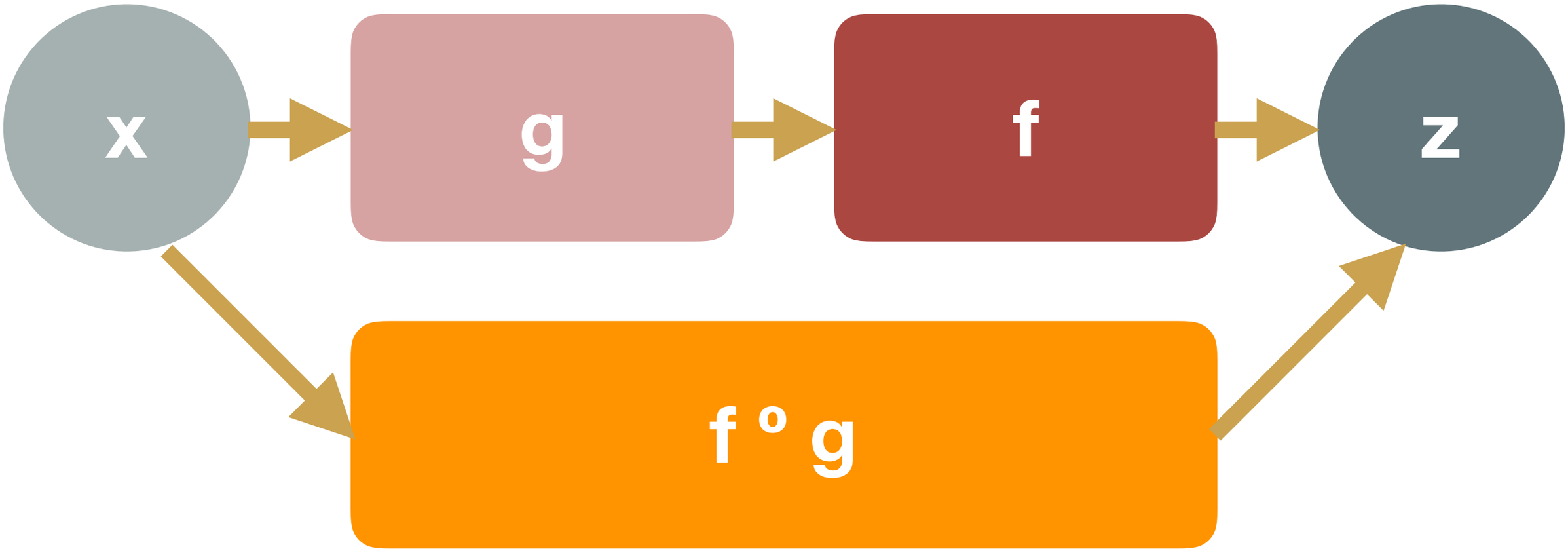
Functional Composition




```
Y g (X x)
{
  return y;
}
```

```
Z f (Y y)
{
  return z;
}
```

Functional Composition

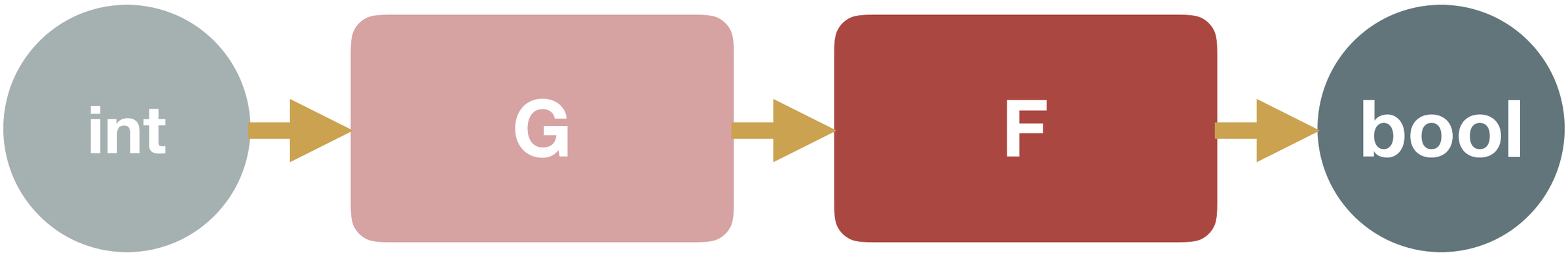


Example of Composed Functions

```
string G (int x)
    => x.ToString();

bool F (string y)
    => int.TryParse(y, out int _);
```

Example of Composed Functions



Example of Composed Functions

```
new [] { 33 }
.Select(G)
.Select(F)
.First();
```

```
output>
true
```

Functions in Context

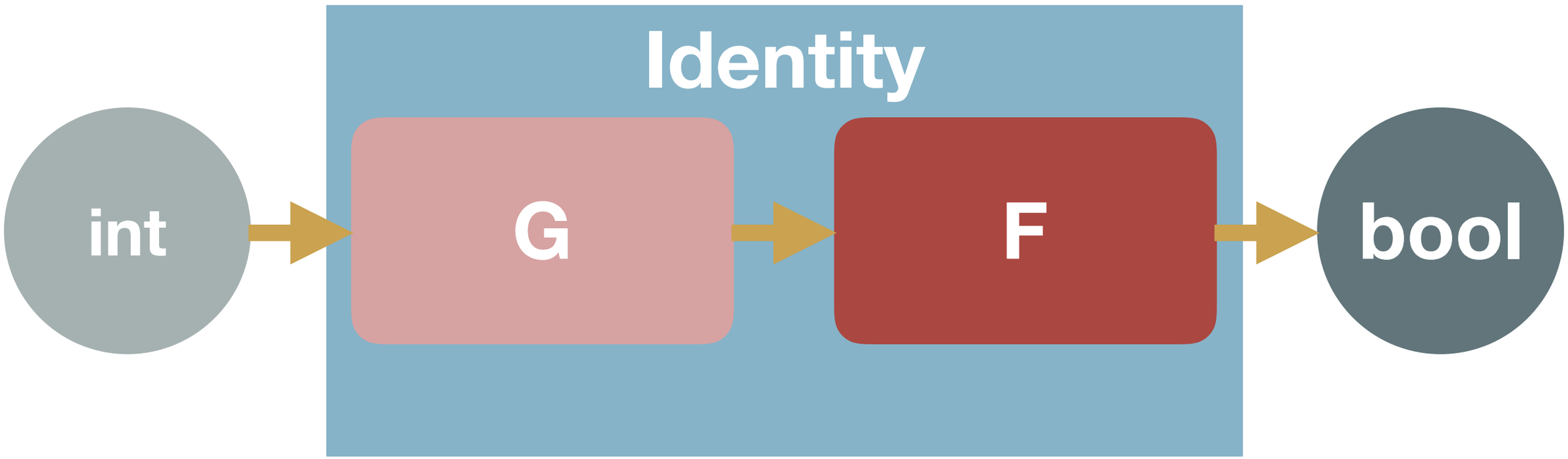
Example of Composed Functions

```
class Identity
{
   public static T [] Of<T>(T x)
      => new [] { x };
}

Identity.Of(33)
   .Select(G)
   .Select(F)
   .First();
```

```
output>
true
```

Example of Composed Functions



Example of Composed Functions

```
Identity.Of(33)
        .Select(G)
        .Select(F)
        .First();
```

Lift

Context

Remove

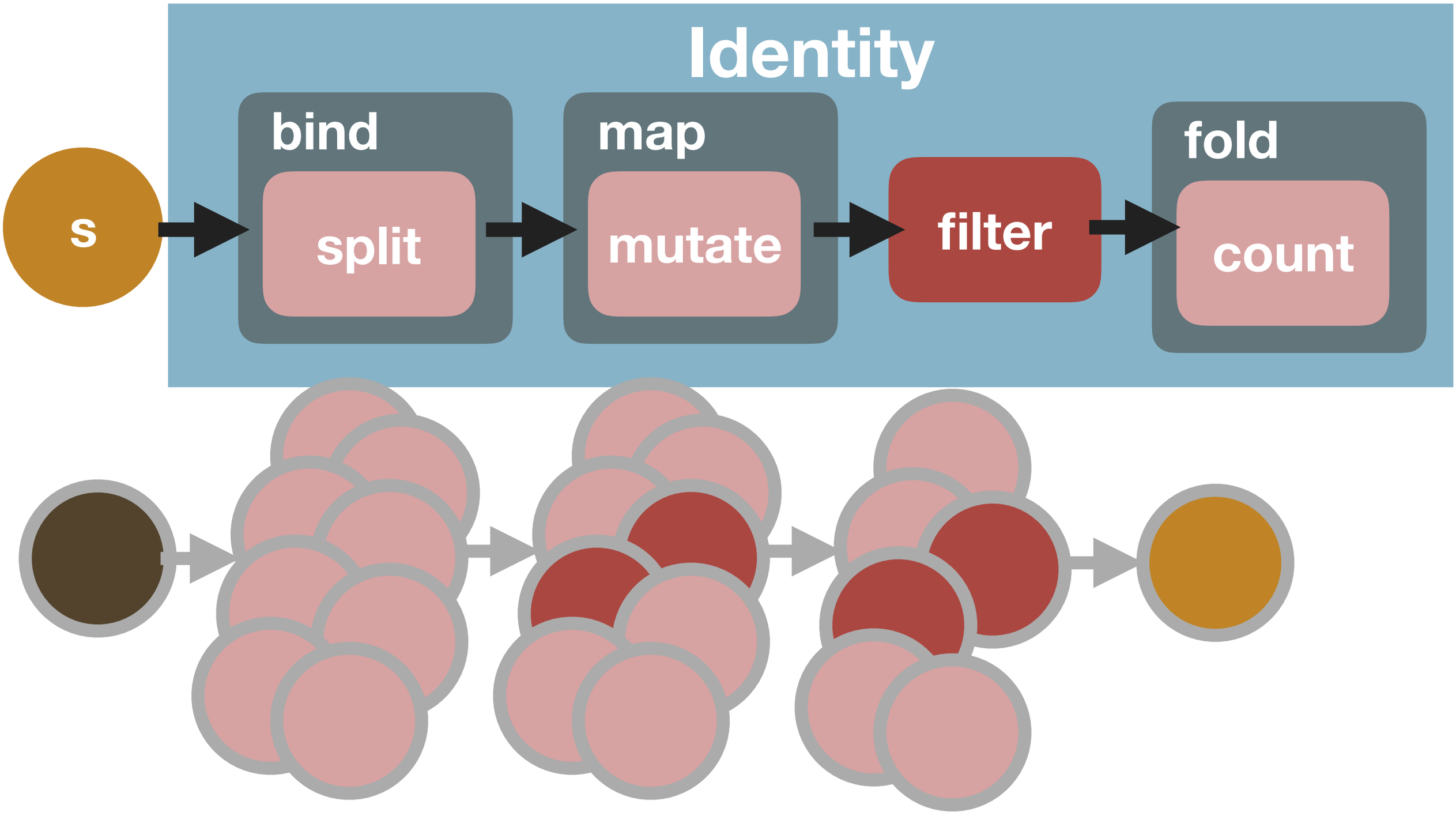
```
output>
true
```

Functions in Context Example

```
var count = Identity.Of(
    "Midway in our life's journey, I went astray")
  .SelectMany(s => s.Split(' '))
  .Select(word =>
    Regex.Replace(word, "('s)|\\W+", ""))
.Where(word => word.Length > 2)
.Aggregate(0, (number, _) => number + 1);
```

```
output>
6
```

Functions in Context Data Flow



Functions in Context Example

```
var count = Identity.Of(
    "Midway in our life's journey, I went astray")
    .SelectMany(s => s.Split(' '))
    .Select(word =>
      Regex.Replace(word, "('s)|\\W+", ""))
    .Where(word => word.Length > 2)
    .Aggregate(0, (number, _) => number + 1);
```

Lift

Context

Remove

```
output>
6
```

String in Context Example

```
var count =
  “Midway in our life's journey, I went astray”
  .Split(' ')
  .Select(word =>
    Regex.Replace(word, "('s)|\\W+", ""))
.Where(word => word.Length > 2)
.Count();
```

```
output>
6
```

String in Context Example

Lift

Context

```
var count =
    "Midway in our life's journey, I went astray"
        .Split(' ')
        .Select(word =>
          Regex.Replace(word, "('s)|\\W+", ""))
        .Where(word => word.Length > 2)
        .Count();
```

Remove

```
output>
6
```

QVI COELVM CECINIT MEDIVMQVE IMVMQVE TRIBVNAL LVSTRAVITQVE ANIMO CVNCTA POETA SVO DOCTVS ADEST DANTES SVA QVEM FLORENTIA SAEPE
SENSIT CONSILIIS AC PIETATE PATREM NIL POTVIT TANTO MORS SAEVA NOCERE POETAE QVEM VIVVM VIRTVS CARMEN IMAGO FACIT

Thank you!

Mike Harris

@MikeMKH
http://comp-phil.blogspot.com/

https://www.slideshare.net/secret/ySVIRZU0HF3E8L

Next Steps

- StrangeLoop (conference)
  https://www.thestrangeloop.com/

- Brian Lonsdorf - Professor Frisby Introduces Composable Functional JavaScript (video series)
  https://egghead.io/courses/professor-frisby-introduces-composable-functional-javascript

- Aditya Bhargava - Functors, Applicatives, And Monads In Pictures (blog post)
  http://adit.io/posts/2013-04-17-functors,_applicatives,_and_monads_in_pictures.html

- Scott Wlaschin - F# for Fun and Profit (blog series)
  http://fsharpforfunandprofit.com/series/thinking-functionally.html

- Dixin Yan - LINQ and Functional Programming via C# (blog series)
  https://weblogs.asp.net/dixin/linq-via-csharp

Code from C# Source Code

- Select, https://github.com/dotnet/corefx/blob/7a673547563dc92f1f4cad802d7b57483cdf7500/src/System.Linq/src/System/Linq/Select.cs#L199-L226
- SelectMany, https://github.com/dotnet/corefx/blob/7a673547563dc92f1f4cad802d7b57483cdf7500/src/System.Linq/src/System/Linq/SelectMany.cs#L127-L223
- Where, https://github.com/dotnet/corefx/blob/7a673547563dc92f1f4cad802d7b57483cdf7500/src/System.Linq/src/System/Linq/Where.cs#L198-L255
- Aggregate, https://github.com/dotnet/corefx/blob/7a673547563dc92f1f4cad802d7b57483cdf7500/src/System.Linq/src/System/Linq/Aggregate.cs#L40-L59
- ForEach, https://github.com/dotnet/corefx/blob/bffef76f6af208e2042a2f27bc081ee908bb390b/src/System.Collections/src/System/Collections/Generic/List.cs#L598-L618
- GroupBy, https://github.com/dotnet/corefx/blob/7a673547563dc92f1f4cad802d7b57483cdf7500/src/System.Linq/src/System/Linq/Lookup.cs#L88-L100
- OrderBy, https://github.com/dotnet/corefx/blob/f17f1e847aeab830de77f8a46656339d7b0f1b43/src/System.Linq/src/System/Linq/OrderedEnumerable.cs#L508-L513

Code

- SQL Code, http://sqlfiddle.com/#!6/50442/6
- C# Code, https://gist.github.com/MikeMKH/cabd17faa9290d6f15592c5df8eded9f
- PowerShell Code, https://gist.github.com/MikeMKH/97b38013b1871f98ba1ed843efcca6f9

Biography

- Barolini, Teodolinda. Commento Baroliniano, Digital Dante. New York, NY: Columbia University Libraries, 2017. https://digitaldante.columbia.edu/dante/divine-comedy/
- Ben-Gan, Itzik. Microsoft SQL Server 2012 high-performance T-SQL using Window functions: Microsoft Press, 2012.
- Byham, Rick. Microsoft Docs. https://docs.microsoft.com/en-us/sql/t-sql/queries/select-transact-sql
- Cook, William R. and Herzman, Ronald B. Dante's Divine Comedy. The Great Courses. http://www.thegreatcourses.com/courses/dante-s-divine-comedy.html
- Equity Regulatory Alert #2016 - 3 Guidance On Test Stock Usage https://www.nasdaqtrader.com/MicroNews.aspx?id=ERA2016-3
- Petricek, Tomas. Beyond the Monad fashion (I.): Writing idioms in LINQ- http://tomasp.net/blog/idioms-in-linq.aspx/
- Securities Industry Automation Corporation. New York, 10 September 2015. https://www.nyse.com/publicdocs/ctaplan/notifications/trader-update/CTS_CQS%20%20NEW%20DEDICATED%20TEST%20SYMBOL_09102015.pdf
- Wickham, Hadley, and Garrett Grolemund. R for data science : import, tidy, transform, visualize, and model data. Sebastopol, CA: O'Reilly Media, 2016. http://r4ds.had.co.nz/

Images

- By Domenico di Michelino - Jastrow, Self-photographed, Public Domain, https://commons.wikimedia.org/w/index.php?curid=970608
- By Sailko - Own work, CC BY 3.0, https://commons.wikimedia.org/w/index.php?curid=49841035
- By Lua - Sotheby's, London, 17 December 2015, lot 22, Public Domain, https://commons.wikimedia.org/w/index.php?curid=32782393
- By Sailko - Own work, CC BY 3.0, https://commons.wikimedia.org/w/index.php?curid=56266161
- By William Blake - http://www.blakearchive.org/exist/blake/archive/work.xq?workid=but812&java=no, Public Domain, https://commons.wikimedia.org/w/index.php?curid=27118518
- By Salvatore Postiglione - http://www.hampel-auctions.com/en/92-325/onlinecatalog-detail-n229.html, Public Domain, https://commons.wikimedia.org/w/index.php?curid=26540023
- Di Giovanni Stradano - Opera propria, 2007-10-25, Pubblico dominio, https://commons.wikimedia.org/w/index.php?curid=2981981