First steps with Office Scripts: Sort an array of Range objects

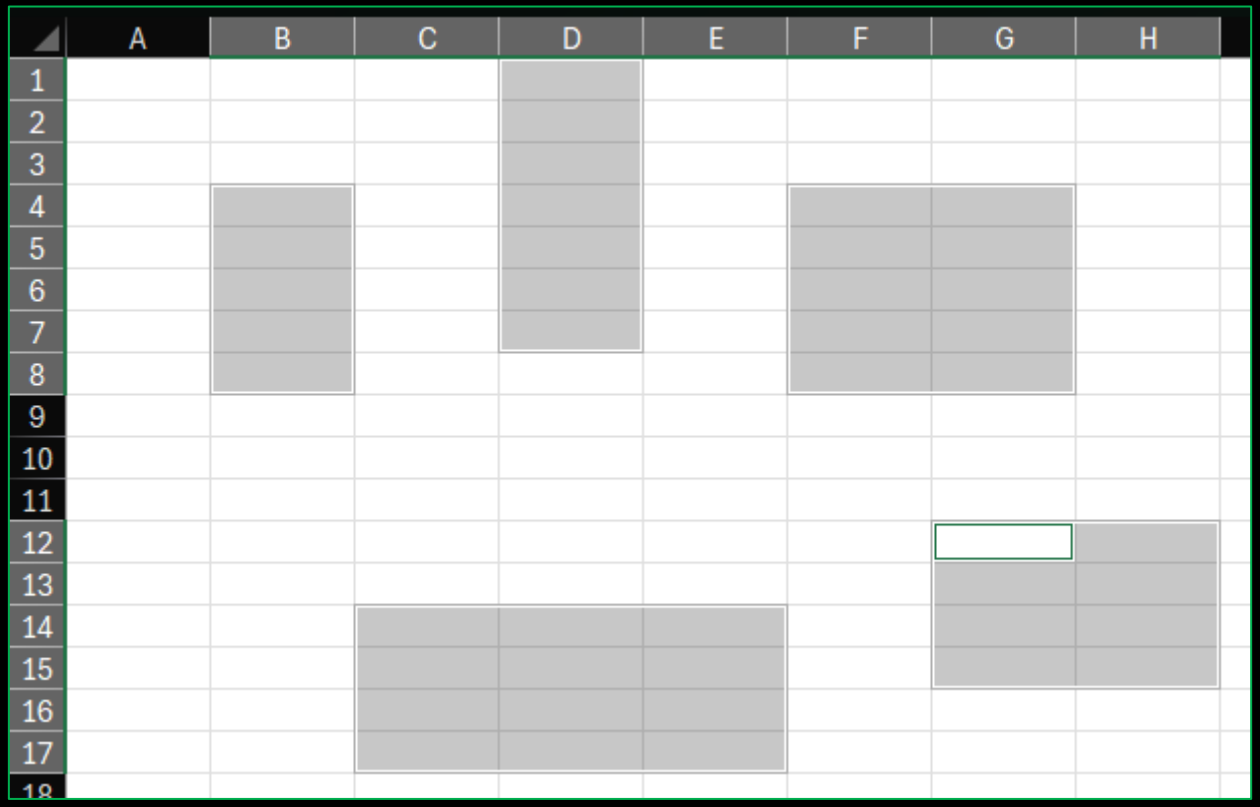
Owen Price
flexyourdata.com, @flexyourdata

What are Office Scripts?

"Office Scripts in Excel let you automate your day-to-day tasks. Use the Action Recorder to turn manual steps into reusable scripts. Edit those scripts or create new ones with the Code Editor. Let others in the workbook run these scripts with a single button. Then, share them with coworkers so everyone can improve their workflow."

https://learn.microsoft.com/en-us/office/dev/scripts/overview/excel?view=office-scripts?wt.mc_id=MVP_310565

Suppose we have several selected ranges…



Owen Price
flexyourdata.com, @flexyourdata

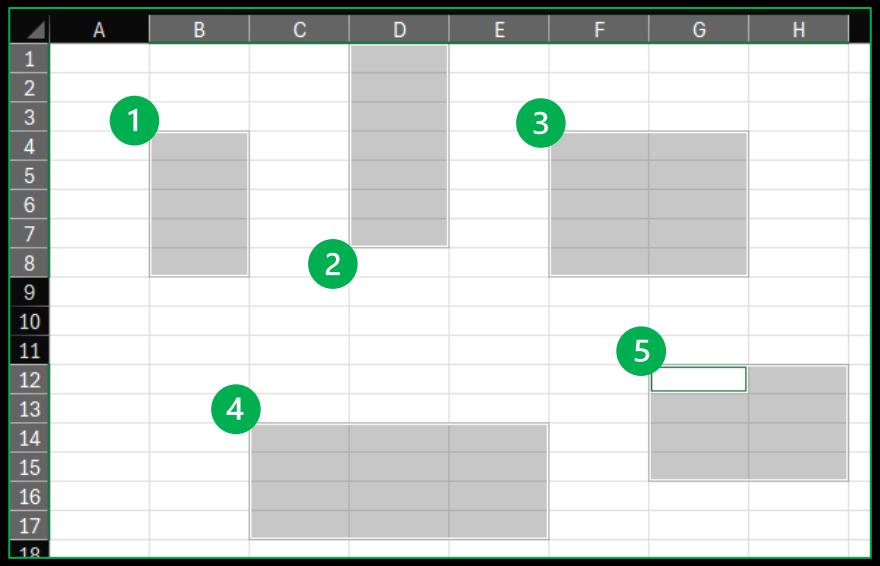
We can retrieve an array of ranges in what is known as a 'RangeAreas' object

```
function main(workbook: ExcelScript.Workbook) {

  const selectedAreas = workbook.getSelectedRanges()
  console.log(selectedAreas.getAreaCount()) // 5

}
```

Get the RangeAreas object



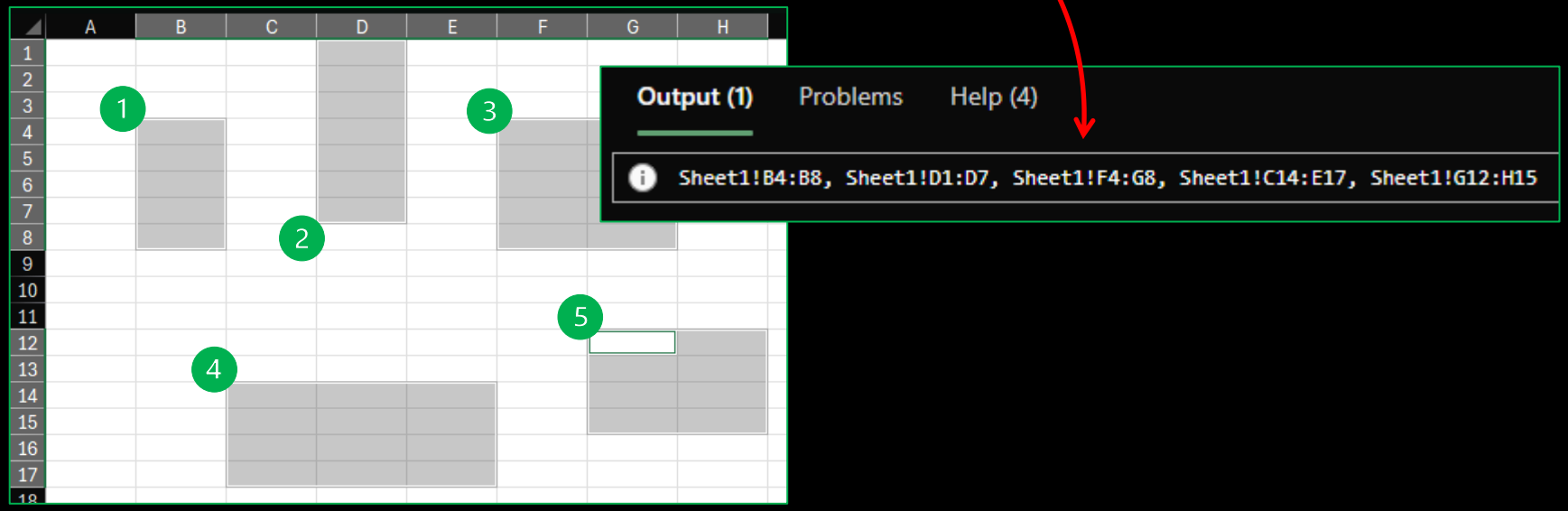
Owen Price
flexyourdata.com, @flexyourdata

And get an Array of the Ranges themselves

```
function main(workbook: ExcelScript.Workbook) {

  const selectedAreas = workbook.getSelectedRanges()
  const selectedRanges = selectedAreas.getAreas()
  console.log(selectedRanges.map(r => r.getAddress()).join(", "))
}
```

Get an array of Ranges

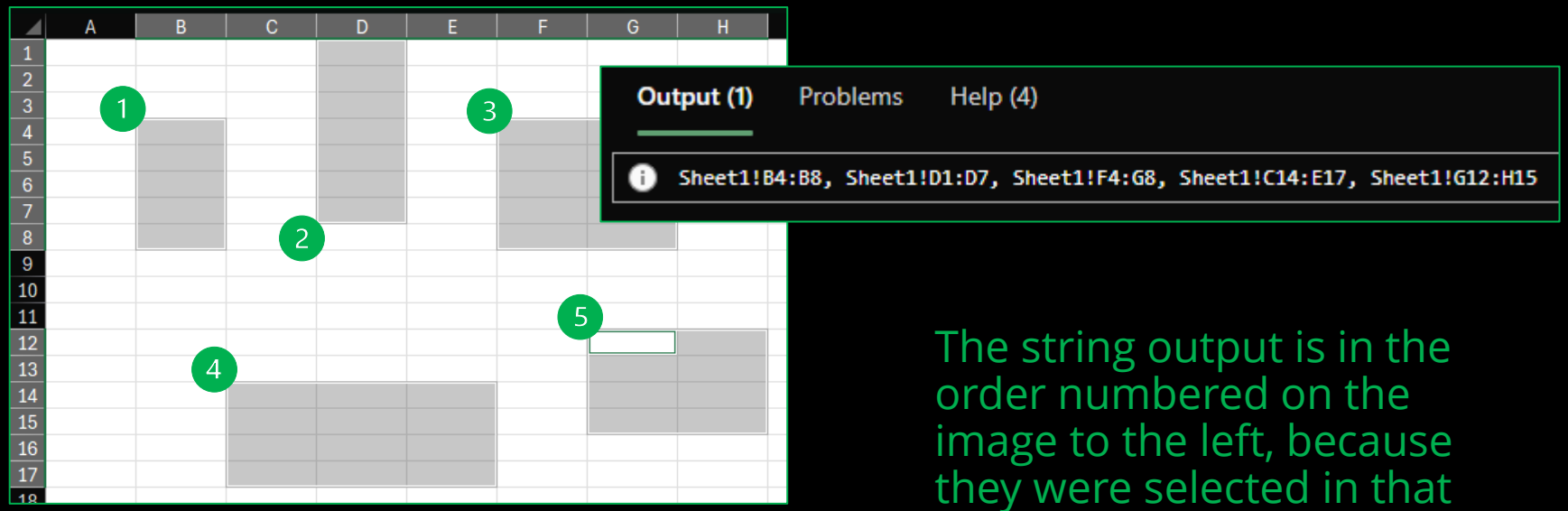


Owen Price
flexyourdata.com, @flexyourdata

The order of the Ranges in the array is dependent on the order in which they were selected

```
function main(workbook: ExcelScript.Workbook) {

  const selectedAreas = workbook.getSelectedRanges()
  const selectedRanges = selectedAreas.getAreas()
  console.log(selectedRanges.map(r => r.getAddress()).join(", "))
}
```



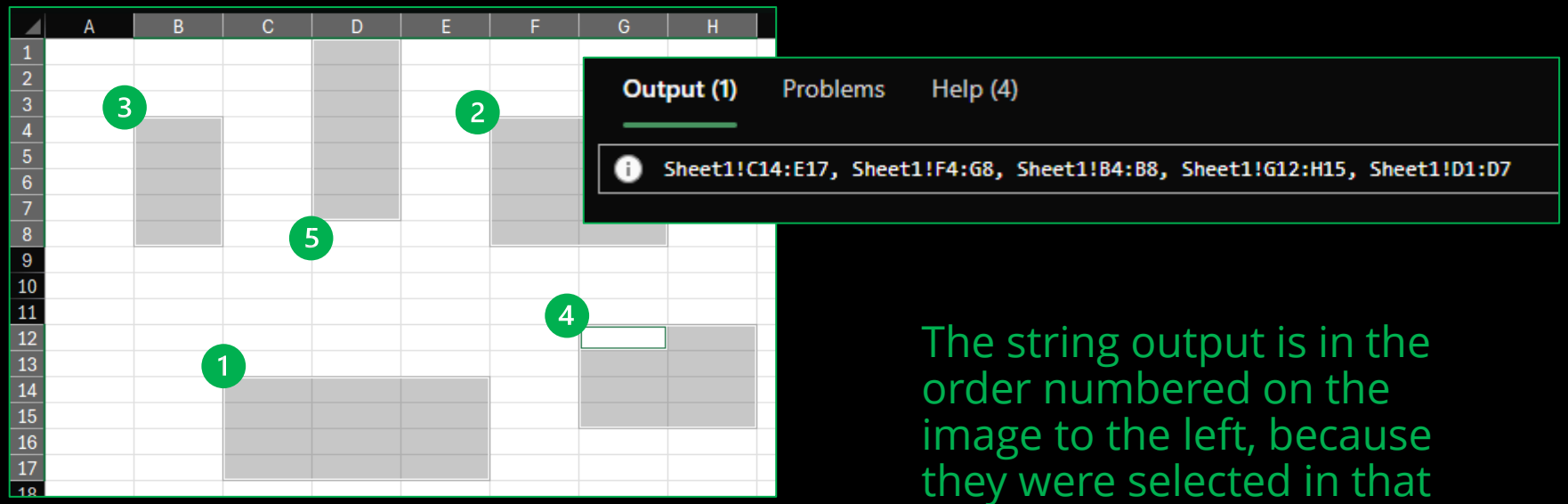
The string output is in the order numbered on the image to the left, because they were selected in that order

Owen Price
flexyourdata.com, @flexyourdata

If they are selected in a different order, the array reflects that order

```
function main(workbook: ExcelScript.Workbook) {

  const selectedAreas = workbook.getSelectedRanges()
  const selectedRanges = selectedAreas.getAreas()
  console.log(selectedRanges.map(r => r.getAddress()).join(", "))
}
```



The string output is in the order numbered on the image to the left, because they were selected in that order

Owen Price
flexyourdata.com, @flexyourdata

But what if I want to process them in row-major order? Array.sort seems like a good option

```
function main(workbook: ExcelScript.Workbook) {

  const selectedAreas = workbook.getSelectedRanges()
  const selectedRanges = selectedAreas.getAreas()

  const sortedRanges =
  selectedRanges.sort()

}
```

```
sort(compareFn?: (a: ExcelScript.Range, b:
ExcelScript.Range) => number): ExcelScript.Range[]
```

Function used to determine the order of the elements. It is expected to return a negative value if first argument is less than second argument, zero if they're equal and a positive
value otherwise. If omitted, the elements are sorted in ascending, ASCII character order.

```ts
[11,2,22,1].sort((a, b) => a - b)
```

Sorts an array.

Row-major order means “row-by-row from the top, and within each row, column-by-column from the left”

This optional ‘compareFn’ works the same way as the Comparer functions in Power Query

Unfortunately, we can't access the Office Scripts API inside Array.sort! ☹️

```
function main(workbook: ExcelScript.Workbook) {

  const selectedAreas = workbook.getSelectedRanges()
  const selectedRanges = selectedAreas.getAreas()

  const sortedRanges = selectedRanges.sort((a, b) => {

    const aRowIndex = a.getRowIndex()
    const bRowIndex = b.getRowIndex()

    if (aRowIndex !== bRowIndex) {
      return aRowIndex - bRowIndex
    }
    return a.getColumnIndex() - b.getColumnIndex()
  })

}
```

These 'getRow|ColumnIndex()' methods return the 0-indexed index of the top-left cell in the range.

Output (1) Problems (4) Help (4)

```
See line 8, column 25: Usage of Office Scripts API in Array.sort is not allowed.
See line 9, column 25: Usage of Office Scripts API in Array.sort is not allowed.
See line 14, column 14: Usage of Office Scripts API in Array.sort is not allowed.
See line 14, column 35: Usage of Office Scripts API in Array.sort is not allowed.
```

i.e. "If two ranges aren't on the same row, then return the difference between the row indices of their top-left cells. Otherwise, they are on the same row, so return the difference between the column indices of their top-left cells." The sign of the return value tells the sort method how they should be sorted relative to each other.

But we *can* retrieve the row and column indices *before* calling sort😀

```
function main(workbook: ExcelScript.Workbook) {

  const selectedAreas = workbook.getSelectedRanges()
  const selectedRanges = selectedAreas.getAreas()

  const rangeDetails: [ExcelScript.Range, number, number][] = selectedRanges.map(r =>
    [r, r.getRowIndex(), r.getColumnIndex()]
  )

  let sortedRangeDetails = rangeDetails.sort((a, b) => {
    if (a[1] !== b[1]) {
      return a[1] - b[1]
    }
    return a[2] - b[2]
  })

}
```

Use Array.map to create a new array of

[Range, number, number]

containing the Range and it's starting row and column indices

Then use Array.sort on the array containing the row and column indices. This circumvents the need to call the Office Scripts API within Array.sort

Owen Price
flexyourdata.com, @flexyourdata

The result is that even though the ranges are not selected in that order, they are returned in row-major order

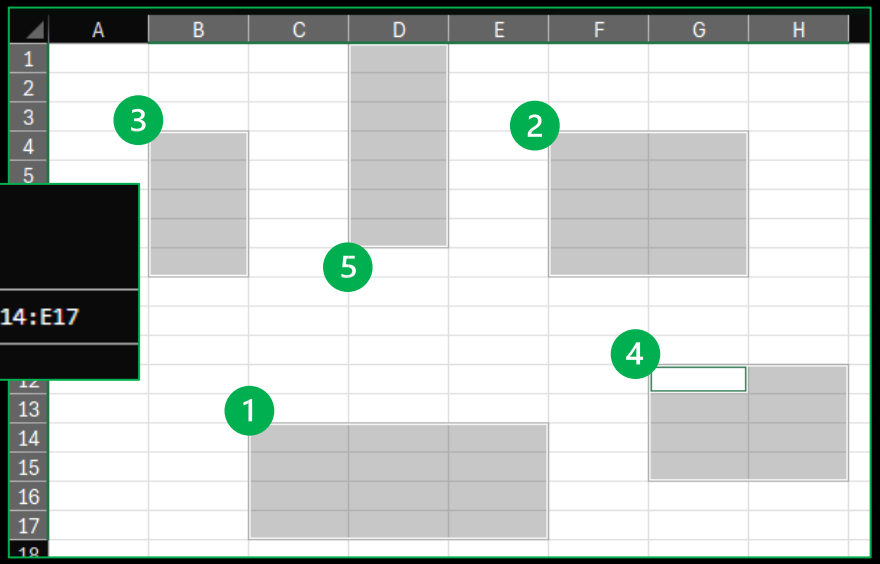
```
...
  let sortedRangeDetails = rangeDetails.sort((a, b) => {
    if (a[1] !== b[1]) {
      return a[1] - b[1]
    }
    return a[2] - b[2]
  })
  console.log(sortedRangeDetails.map(r => r[0].getAddress()).join(", "))

}
```

Element zero of each rangeDetails object contains the Range object itself, so that's how we access the A1-style address

Output (1) Problems Help (4)

```
Sheet1!D1:D7, Sheet1!B4:B8, Sheet1!F4:G8, Sheet1!G12:H15, Sheet1!C14:E17
```



Owen Price
flexyourdata.com, @flexyourdata

Takeaways:

1. The workbook.getSelectedAreas() method returns a RangeAreas object containing the currently selected ranges
2. We use the getAreas() method on a RangeAreas object to get an array of Range objects
3. They are returned in the order they were selected in the UI
4. We can't use methods of Range objects within the comparer function of the Array.sort method
5. A workaround is to use Array.map to retrieve the properties of the ranges (like the row and column index of the top-left cell) and store them with the range in a new Array and then sort that array instead
6. Both Array.map and Array.sort use a callback function (AKA a lambda) to control their output