

# Expressions Results - Solution

In this exercise, we ask you to guess the results of the expressions by mentally simulating the execution of expressions.

## 1.1 Exercise: Results

Examine the following expressions. What is the value returned by the execution of the following expression?

Exercise:

```
[ | anArray |  
  anArray := #('first' 'second' 'third' 'fourth').  
  anArray at: 2
```

**Solution.**

```
[ 'second'
```

Exercise:

```
[ #(2 3 -10 3) collect: [ :each | each * each]
```

**Solution.**

```
[ #(4 9 100 9)
```

Exercise:

```
[ 6 + 4 / 2
```

**Solution.**

```
[ 5
```

Exercise:

```
[ 1 + 3 negated
```

**Solution.**

```
[ -2
```

Exercise:

```
[ 1 + (3 negated)
```

**Solution.**

```
[ -2
```

Exercise:

```
[ 2 raisedTo: 3 + 2
```

**Solution.**

```
[ 32
```

Exercise:

```
[ 2 negated raisedTo: 3 + 2
```

**Solution.**

```
[ -32
```

Exercise:

```
[ #(a b c d e f) includesAll: #(f d b)
```

**Solution.**

```
[ true
```

### Exercise: unneeded parentheses

Putting more parentheses than necessary is a good way to get started. Such practice however leads to less readable expressions. Rewrite the following expressions using the least number of parentheses.

## 1.1 Exercise: Results

Exercise:

```
[ ((3 + 4) + (2 * 2) + (2 * 3))
```

**Solution.**

```
[ 3 + 4 + (2 * 2) + (2 * 3)
```

Exercise:

```
[ x between: (pt1 x) and: (pt2 y)
```

**Solution.**

```
[ x between: pt1 x and: pt2 y
```

Exercise:

```
[ (x isZero)
  ifTrue: [...].
  (x includes: y)
  ifTrue: [...]
```

**Solution.**

```
[ x isZero
  ifTrue: [...].
  (x includes: y)
  ifTrue: [...]
```

Exercise:

```
[ (Integer primesUpTo: 64) sum
```

**Solution.**

```
[ (Integer primesUpTo: 64) sum
```

Exercise:

```
[ (OrderedCollection new)
  add: 56;
  add: 33;
  yourself
```

**Solution.**

```
[ OrderedCollection new
  add: 56;
  add: 33;
  yourself
```

Exercise:

```
[ ('http://www.pharo.org' asUrl) retrieveContents
```

**Solution.**

```
[ 'http://www.pharo.org' asUrl retrieveContents
```

Exercise:

```
[ (('2014-07-01' asDate) - '2013/2/1' asDate) days
```

**Solution.**

```
[ ('2014-07-01' asDate - '2013/2/1' asDate) days
```

Exercise:

```
[ (((ZnEasy getPng: 'http://pharo.org/web/files/pharo.png')
  asMorph) openInWindow)
```

**Solution.**

```
[ (ZnEasy getPng: 'http://pharo.org/web/files/pharo.png')
  asMorph openInWindow
```

Exercise:

```
[ ((#(a b c d e f) asSet) intersection: #(f d b) asSet))
```

**Solution.**

```
[ #(a b c d e f) asSet intersection: #(f d b) asSet
```

### Exercise: Execution sequence

Examine each of the following expression and write down the sequence of steps of their execution (which message is executed first and so on).

Exercise:

```
[ Date today daysInMonth
```

**Solution.**

```
[ Date today daysInMonth
  today
  daysInMonth
```

## 1.1 Exercise: Results

Exercise:

```
[ 5@5 extent: 6.2 truncated @ 7
```

**Solution.**

```
[ 5@5 extent: 6.2 truncated @ 7
  6.2 truncated
  5@5
  6@7
  extent:
```

Exercise:

```
[ Transcript show: (45 + 9) printString
```

**Solution.**

```
[ Transcript show: (45 + 9) printString
  (45 + 9)
  printString
  show:
```

Exercise:

```
[ ('2014-07-01' asDate - '2013/2/1' asDate) days
```

**Solution.**

```
[ ('2014-07-01' asDate - '2013/2/1' asDate) days
  asDate
  asDate
  -
  days
```

Exercise:

```
[ 42 factorial decimalDigitLength
```

**Solution.**

```
[ 42 factorial decimalDigitLength
  factorial
  decimalDigitLength
```

Exercise:

```
[ (ZnServer startDefaultOn: 8080)
  onRequestRespond: [ :request | ZnResponse ok: (ZnEntity
    with: DateAndTime now printString) ]
```

**Solution.**

```

(ZnServer startDefaultOn: 8080)
  onRequestRespond: [ :request | ZnResponse ok: (ZnEntity
with: DateAndTime now printString) ]
startDefaultOn:
onRequestRespond:
now
printString
with:
ok:

```

Exercise:

```
(1914 to: 1945) count: [ :each | Year isLeapYear: each ].
```

**Solution.**

```

(1914 to: 1945) count: [ :each | Year isLeapYear: each ].
to:
count:
isLeapYear:

```

Exercise:

```
$/ join: ($- split: '1969-07-20') reverse
```

**Solution.**

```

$/ join: ($- split: '1969-07-20') reverse
split:
reverse
join:

```

Exercise:

```

DateAndTime fromUnixTime:
  ((ByteArray readHexFrom: 'CAFEBABE4422334400FF')
copyFrom: 5 to: 8) asInteger

```

**Solution.**

```

DateAndTime fromUnixTime:
  ((ByteArray readHexFrom: 'CAFEBABE4422334400FF')
copyFrom: 5 to: 8) asInteger
readHexFrom:
copyFrom:to:
asInteger
fromUnixTime:

```

Exercise:

```
[ (String new: 32) collect: [ :each | 'abcdef' atRandom ]
```

**Solution.**

```
[ (String new: 32) collect: [ :each | 'abcdef' atRandom ]  
  new:  
  collect:  
  atRandom
```

Exercise:

```
[ 'http://www.pharo.org' asUrl saveContentsToFile: 'page.html'
```

**Solution.**

```
[ 'http://www.pharo.org' asUrl saveContentsToFile: 'page.html'  
  asUrl  
  saveContentsToFile:
```

Exercise:

```
[ '^.*.jpg' asRegex in: [ :regex |  
  '/tmp/foo.txt' asFileReference contents lines  
  select: [ :line | regex matches: line ] ]
```

**Solution.**

```
[ '^.*.jpg' asRegex in: [ :regex |  
  '/tmp/foo.txt' asFileReference contents lines  
  select: [ :line | regex matches: line ] ]  
  asRegex  
  in:  
  asFileReference  
  contents  
  line  
  select:  
  matches:
```