

Résultats d'Expressions - Solution

Dans ces exercices vous devez devinez les résultats des expressions en simulant mentalement l'exécution des expressions.

1.1 Exercice : Résultats

Examinez les expressions suivantes. Quel est la valeur retournée par l'exécution des expressions suivantes ?

Exercice :

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

Solution.

```
[ 'second'
```

Exercice :

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

Solution.

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

Exercice :

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

Solution.

```
[ 5
```

Exercice :

```
[ 1 + 3 negated
```

Solution.

```
[ -2
```

Exercice :

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

Solution.

```
[ -2
```

Exercice :

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

Solution.

```
[ 32
```

Exercice :

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

Solution.

```
[ -32
```

Exercice :

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

Solution.

```
[ true
```

Exercice : Parenthèses superflues

Mettre plus de parenthèses que nécessaire est une bonne manière pour commencer. Cependant une telle pratique rend le code moins lisible. Réécrivez les expressions avec le minimum de parenthèses.

1.1 Exercice : Résultats

Exercice :

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

Solution.

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

Exercice :

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

Solution.

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

Exercice :

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

Solution.

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

Exercice :

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

Solution.

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

Exercice :

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

Solution.

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

Exercice :

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

Solution.

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

Exercice :

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

Solution.

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

Exercice :

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

Solution.

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

Exercice :

```
[ ((#(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
```

Exercice : Séquence d'Execution

Examinez chacune des expressions suivantes et décrivez la séquence des étapes de leur exécution (quel message est exécuté en premier).

Exercice :

```
[ Date today daysInMonth
```

Solution.

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

1.1 Exercice : Résultats

Exercice :

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

Solution.

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

Exercice :

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

Solution.

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

Exercice :

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

Solution.

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

Exercice :

```
[ 42 factorial decimalDigitLength
```

Solution.

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

Exercice :

```
[ (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:
```

Exercice :

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

Solution.

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

Exercice :

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

Solution.

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

Exercice :

```
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:
```

Exercice :

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

Solution.

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

Exercice :

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

Solution.

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

Exercice :

```
[ '^.*.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  
  asFileReference  
  contents  
  line  
  matches:
```