COMPREHENSIONS

PYTHON

THE BASICS

”Comprehensions are constructs that allow sequences to be built from other sequences”

LIST COMPREHENSIONS

”List comprehensions creates a **list** in memory that is destroyed after use unless stored”

You might remember this syntax from mathematics lessons at school?

```
q = { x2 : x in {0,3,6,9}}
```

This is the Python syntax for the same expression.

```
q = [ x**2 for x in [0,3,6,9]]
```

LIST COMPREHENSIONS

A list comprehension consists of the following parts

- Input sequence
- A variable representing members of the input sequence
- An optional Predicate expression
- An Output Expression producing elements of the output list from members of the Input Sequence that satisfy the predicate.

ANOTHER EXAMPLE

This is our original list with content

```
myList = [1, '4', 9, 'a', 0, 4]
```

We now wants to grab every integer from the list and raised them by 2.

This is a List Comprehension to solve our problem:

```
newList = [x**2 for x in myList if isinstance(x, int)]
```

PYTHON

READABILITY?

Informator
A SOPRANO COMPANY

```
newList = [x**2 for x in myList if isinstance(x, int)]
```

We could write the above comprehension like this:

```
newList = [
    x**2
    for x in myList
    if isinstance(x, int)
]
```

Follow your Python programming guideline regarding syntaxes!

BEFORE WE CONTINUE

We don’t need comprehensions!

By combining map, filter, reduce and lambda functions we can do the same thing.

Compare this:

```
a = map(lambda x: x**2, filter(lambda x: x%2 == 1,numbers))
```

With this:

```
a = [x**2 for x in numbers if x%2 == 1]
```

Calling functions is ”expensive” in Python compared to comprehensions.

PYTHON

NESTING

A comprehension could be nested within another comprehension.

Like this:

```
[[ 1 if item_idx == row_idx else 0 for item_idx in range(0, 3) ]
for row_idx in range(0, 3) ]
```

PYTHON

EXERCISE

Informator
A SOFRANO COMPANY

What is the output of the nested comprehension below?

Try to figure out **without** running the code.

```
[[ 1 if item_idx == row_idx else 0 for item_idx in range(0, 3) ]
for row_idx in range(0, 3) ]
```

DICT COMPREHENSIONS

Dictionaries is just another iterable in Python so the principles are the same.

```
myDict = {'a':10, 'b': 34, 'A': 7, 'Z':3}

myDict2 = {
     k.lower() : myDict.get(k.lower(),0) + myDict.get(k.upper(),0)
    for k in myDict
}
```

```
Result: {'a': 17, 'z': 3, 'b': 34}
```

SET COMPREHENSIONS


```
names = [ 'RASMUS', 'lena', 'jonas', 'RalF', 'A', 'RaSmUs' ]

names2 = {
      name[0].upper() + name[1:].lower()
      for name in names
      if len(name) > 1
}
```

A set is an unordered collection with no duplicate elements. Basic uses include membership testing and eliminating duplicate entries. Set objects also support mathematical operations like union, intersection, difference, and symmetric difference.

PYTHON

EXERCISE

Informator
A SOFRANO COMPANY

Grab all strings from the list below and then merge them to one string.

Use a list comprehension to grab the strings from the list.

Merge the strings to a single string using any technique you know about.

```
myList = ['e',4j,'l',8,'u',4,'r',' ','u','o','y',88,5]
```

THE RESULT


```
[[ 1 if item_idx == row_idx else 0 for item_idx in range(0, 3) ]
for row_idx in range(0, 3) ]
```

The nested above comprehension gives the following result:

```
[ [ 1, 0, 0 ],
  [ 0, 1, 0 ],
  [ 0, 0, 1 ] ]
```