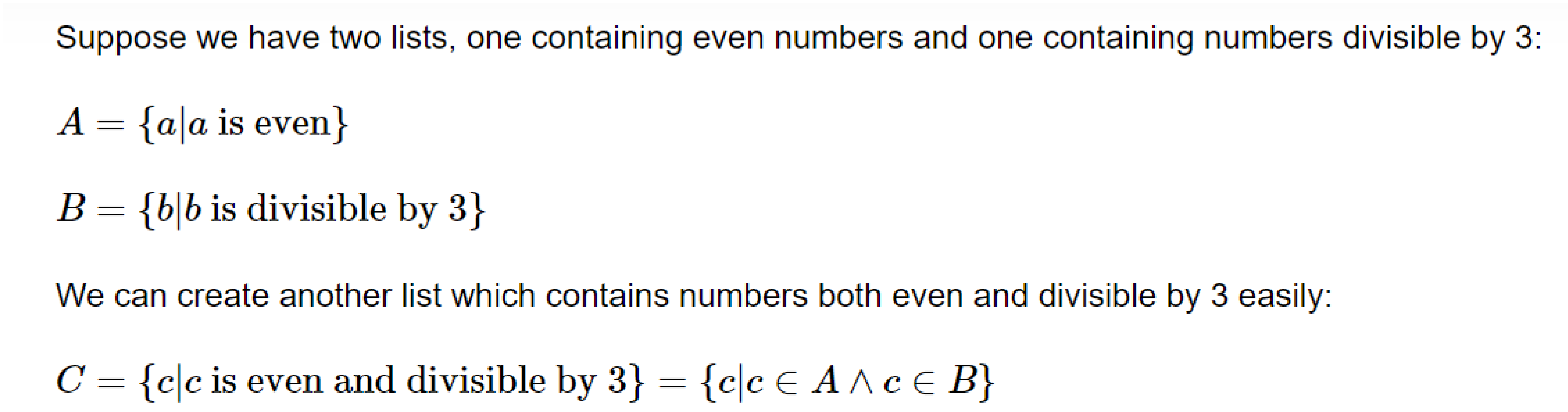
List comprehension: some examples...

List comprehension



Suppose we have two lists, one containing even numbers and one containing numbers divisible by 3:

$$A = \{a | a \text{ is even}\}$$

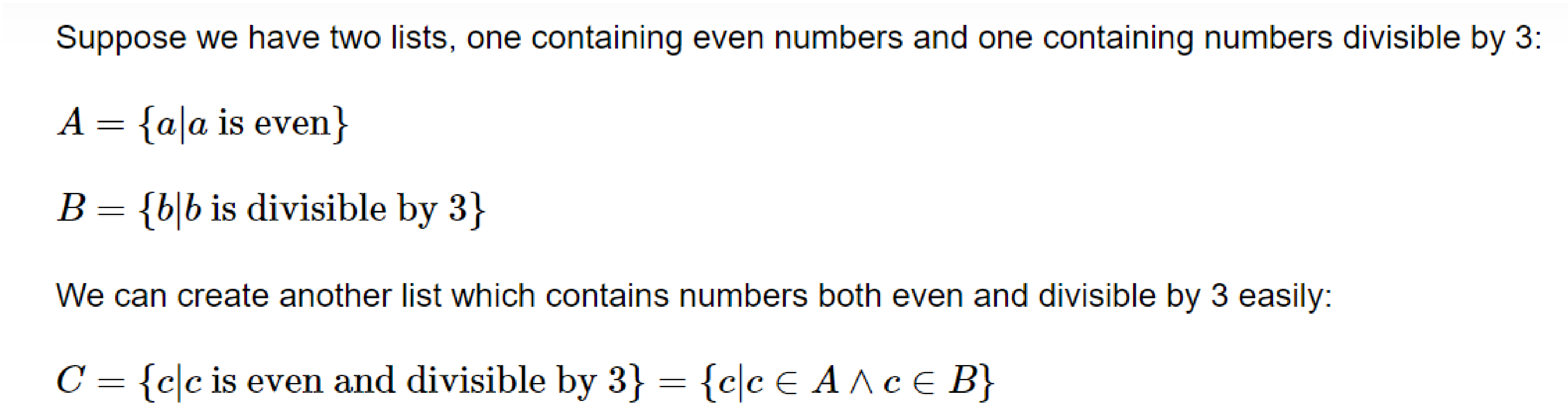
$$B = \{b | b \text{ is divisible by } 3\}$$

We can create another list which contains numbers both even and divisible by 3 easily:

$$C = \{c | c \text{ is even and divisible by } 3\} = \{c | c \in A \wedge c \in B\}$$

https://www.data-blogger.com/2017/11/16/python-list-comprehension/

List comprehension



Suppose we have two lists, one containing even numbers and one containing numbers divisible by 3:

$$A = \{a | a \text{ is even}\}$$

$$B = \{b | b \text{ is divisible by } 3\}$$

We can create another list which contains numbers both even and divisible by 3 easily:

$$C = \{c | c \text{ is even and divisible by } 3\} = \{c | c \in A \wedge c \in B\}$$

```
A = [a for a in range(20) if a % 2 == 0]
B = [b for b in range(20) if b % 3 == 0]
C = [c for c in range(20) if c in A and c in B]
```

https://www.data-blogger.com/2017/11/16/python-list-comprehension/

List comprehension

$$S = \{x^2 : x \text{ in } 1, ..., 10\}$$

(Read: S is the set of all x^2 such that x is a positive integer between 1 and 10 inclusive.)

$$M = \{x | x \text{ in } \; \text{S and } x \text{ odd}\}$$

(Read: M is the set of x such that x is in S and x is even.)

http://www.jessicayung.com/list-comprehensions-in-python/

List comprehension

$$S = \{x^2 : x \text{ in } 1, ..., 10\}$$

(Read: S is the set of all x^2 such that x is a positive integer between 1 and 10 inclusive.)

$$M = \{x | x \text{ in } \text{ S and } x \text{ odd}\}$$

(Read: M is the set of x such that x is in S and x is even.)

```
s = [x**2 for x in range(1,10)]
s
[1, 4, 9, 16, 25, 36, 49, 64, 81]

m = [x for x in s if x % 2 == 1]
m
[1, 9, 25, 49, 81]
```

http://www.jessicayung.com/list-comprehensions-in-python/

List comprehencion

L = [expression for variable in sequence [if condition]]

```
Something using a1,… an
   for a1 in … something to build a1 …
   for a2 in … something to build a2 …
   (...)
   for an in … something to build an …
```

This uses generator expressions…

Can use functions…

```
celsius = [22, 28, 33, 42, 52]

fahr = [e * 9/5 + 32 for e in celsius]




def c2f( v ) :
  return  v*9/5 + 32

fahr = [ c2f(e) for e in celsius]
```

http://zetcode.com/articles/pythonlistcomprehensions/

What the function do?

```
def within( l , lim1 , lim2):
    if lim1> lim2 :
       lim1,lim2 = lim2,lim1
    return [e for e in l if e >= lim1 and e<=lim2]

def greater( l , lim ):
    return [e for e in l if e > lim]

def even( l ):
    return [e for e in l if e%2==0]

def zip( l1 , l2 ) :
    if  len(l1)!=len(l2) :
        return []
    return [ (y,l2[x]) for x,y in enumerate(l1) ]

def zip( l1 , l2 ) :
    c = min(  len(l1) , len(l2) )
    return [ (l1[i],l2[i]) for i in range(0,c )]
```

http://zetcode.com/articles/pythonlistcomprehensions/

Simple examples

```
lst2=[]
for s in args:
    if ( len(s ) > 3 ) :
        s.upper()
        lst2.append(s)
```

```
args3 = [ s.upper()
for s in args
if len(s)>3 ]
```

```
Lst2=[]
for a in [1,2] :
    for b in nums :
     if b> 3 :
       elem = a,b
       lst2.append( elem )
```

```
[(a,b)
for a in [1,2]
for b in nums if
b>3 ]
```

```
Lst = []
For a in args :
   elem = a , len(a)
   lst.append( elem )
Dict( lst )

#other option

D =dict()
For a in args :
    dict[a]=len(a)
```

```
{ a: len(a) for a in args }
```

```
lst = [1, -3, 2]
lst2 = [] # init result
with empty list
for v in lst: # loop over
original list:
   v1 = v**2
   v2 = v+1
   v3 = v
   elem = v1, v2 , v3
   lst2.append(elem)
```

```
Lst2 = [
   (v**2, v+1,v)
   for v in lst ]
```

From expression to …

- set
  - Place list between { }
- Dict
  - Place list between dict( )
- Tupple
  - Place list between tupple()

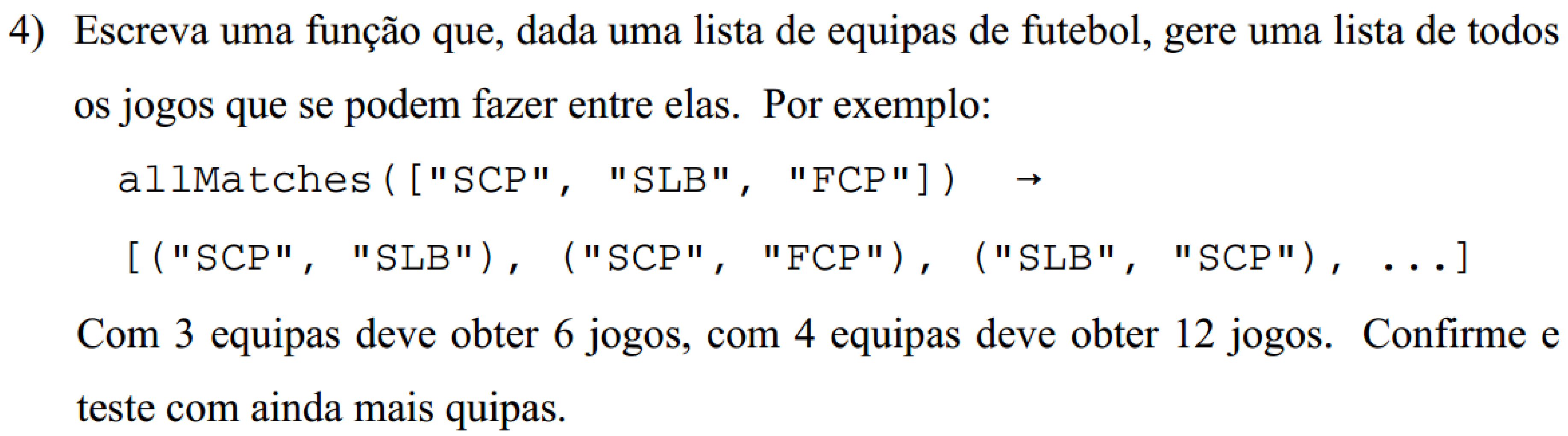
From Week05

4) Escreva uma função que, dada uma lista de equipas de futebol, gere uma lista de todos os jogos que se podem fazer entre elas. Por exemplo:

```
allMatches(["SCP", "SLB", "FCP"])  →
[("SCP", "SLB"), ("SCP", "FCP"), ("SLB", "SCP"), ...]
```

Com 3 equipas deve obter 6 jogos, com 4 equipas deve obter 12 jogos. Confirme e teste com ainda mais quipas.

From Week05



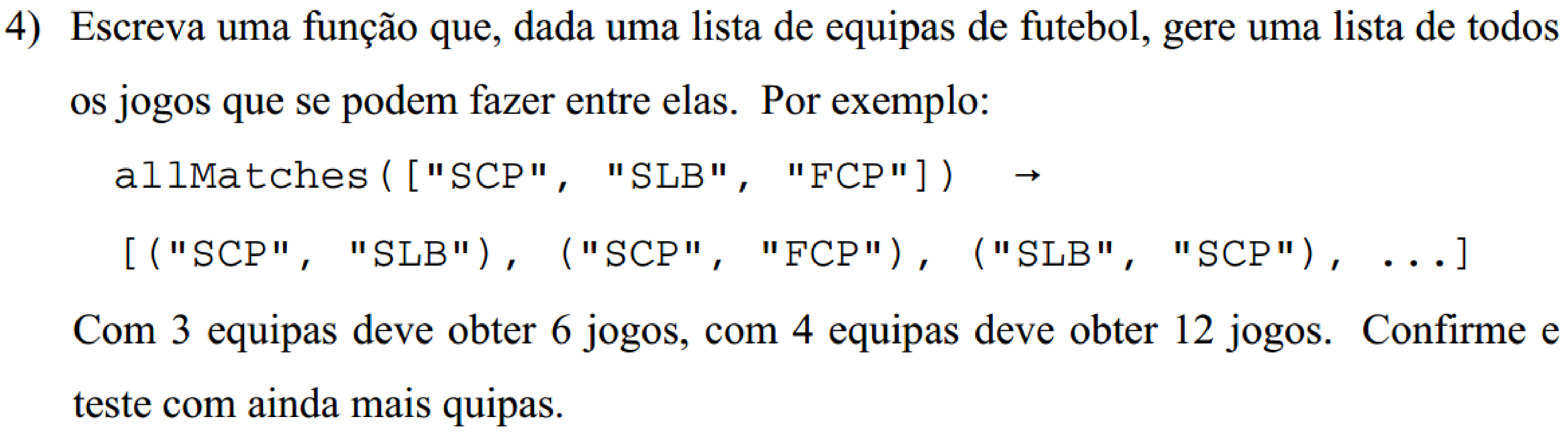
4) Escreva uma função que, dada uma lista de equipas de futebol, gere uma lista de todos os jogos que se podem fazer entre elas. Por exemplo:

```
allMatches(["SCP", "SLB", "FCP"])  →
[("SCP", "SLB"), ("SCP", "FCP"), ("SLB", "SCP"), ...]
```

Com 3 equipas deve obter 6 jogos, com 4 equipas deve obter 12 jogos. Confirme e teste com ainda mais quipas.

```
equipas = ["SCP", "SLB", "FCP"]
l = [(e1,e2) for e1 in equipas for e2 in equipas]
```

From Week05



4) Escreva uma função que, dada uma lista de equipas de futebol, gere uma lista de todos os jogos que se podem fazer entre elas. Por exemplo:

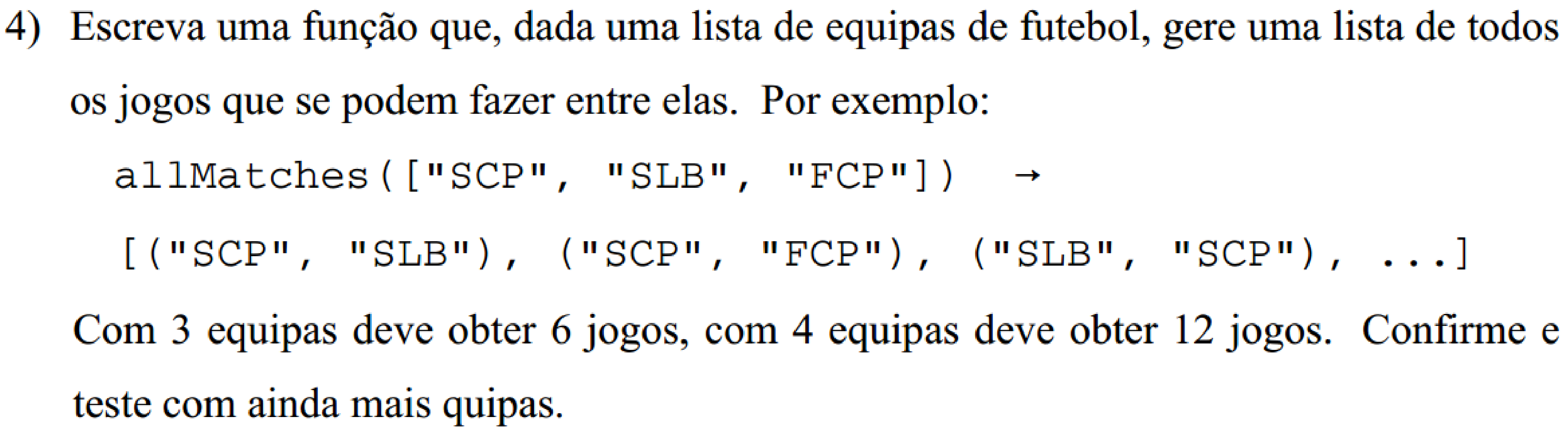
```
allMatches(["SCP", "SLB", "FCP"])  →
[("SCP", "SLB"), ("SCP", "FCP"), ("SLB", "SCP"), ...]
```

Com 3 equipas deve obter 6 jogos, com 4 equipas deve obter 12 jogos. Confirme e teste com ainda mais quipas.

```
equipas = ["SCP", "SLB", "FCP"]
l = [(e1,e2) for e1 in equipas for e2 in equipas
      if e1!=e2]
```

Order matters…. We can refer e1
because it was already defined…

From Week05



4) Escreva uma função que, dada uma lista de equipas de futebol, gere uma lista de todos os jogos que se podem fazer entre elas. Por exemplo:

```
allMatches(["SCP", "SLB", "FCP"])  →
[("SCP", "SLB"), ("SCP", "FCP"), ("SLB", "SCP"), ...]
```

Com 3 equipas deve obter 6 jogos, com 4 equipas deve obter 12 jogos. Confirme e teste com ainda mais quipas.

```
equipas = ["SCP", "SLB", "FCP"]
l = [(e1,e2) for e1 in equipas for e2 in equipas
      if e1!=e2]
```

Order matters…. We can refer e1
because it was already defined…

From Week05

4) Escreva uma função que, dada uma lista de equipas de futebol, gere uma lista de todos os jogos que se podem fazer entre elas. Por exemplo:

```
allMatches(["SCP", "SLB", "FCP"])  →
[("SCP", "SLB"), ("SCP", "FCP"), ("SLB", "SCP"), ...]
```

Com 3 equipas deve obter 6 jogos, com 4 equipas deve obter 12 jogos. Confirme e teste com ainda mais quipas.

```
def allMatches(equipas) :
   return [(e1,e2) for e1 in equipas for e2 in equipas
      if e1!=e2]
```

Order matters.... We can refer e1
because it was already defined...

From Week05

```
def allMatches(equipas) :
  l=[]
  for e1 in equipas :
   for e2 in  equipas :
     if e1!=e2 :
       l.append( (e1,e2) )
```

```
def allMatches(equipas) :
   return [(e1,e2) for e1
in equipas for e2 in
equipas
      if e1!=e2]
```

set comprehension in primes

```
# Construct a list of integers which are not prime (which are
the product of two integers)

no_primes = {a * multiplier for multiplier in range(2, 100) for
a in range(2, 100)}

# Since 1 is not a prime number we have to add it to this list
no_primes.add(1)
# Now construct a list of primes out of this list
primes = {p for p in range(1, 100) if p not in no_primes}

# Show the result
print(primes)
```

https://www.data-blogger.com/2017/11/16/python-list-comprehension/

O que faz?

```
l1=[a*b for a in range(1,11) for b in [2] ]
```

O que faz?

```
l1=[a*b for a in range(1,11) for b in [2] ]

l1=[ (a,b,a*b) for a in range(1,11) for b in [2] ]
```

O que faz?

```
l1=[a*b for a in range(1,11) for b in [2] ]

l1=[ (a,b,a*b) for a in range(1,11) for b in [2] ]

For a,b,c  in l1 :
   print(a,’x’,b,’=‘,c)
```

The END

```
# Generate a list of candidates
L = [n for n in range(2, 40)]

# Remove all multiples of the first value
L = [n for n in L if n == L[0] or n % L[0] > 0]

# Remove all multiples of the second value
L = [n for n in L if n == L[1] or n % L[1] > 0]

# Remove all multiples of the third value
L = [n for n in L if n == L[2] or n % L[2] > 0]
```

https://jakevdp.github.io/WhirlwindTourOfPython/12-generators.html

```
[p for p in range(2,N) if 0 not in [p%d for d in
range(2,p)]]

[p for p in range(2,N) if 0 not in [p%d for d in
range(2,p/2+1)]]

[p for p in range(3,N,2) if 0 not in [p%d for d in
range(2,p)]]
```

http://code.activestate.com/recipes/162479-generating-a-list-of-prime-numbers-in-one-statemen/

```
def gen_primes(N):
    """Generate primes up to N"""
    primes = set()
    for n in range(2, N):
        if all(n % p > 0 for p in primes):
            primes.add(n)
            yield n

print(*gen_primes(100))
```

https://jakevdp.github.io/WhirlwindTourOfPython/12-generators.html