Introducció

ETSEIB/GIE

27 de novembre de 2023

1. Expressions

2. funcions predefinides
   - filter
   - map
   - reduce
   - any
   - all
   - Exemples

3. itertools
   - permutacions
   - combinacions
   - accumulate
   - product
   - tee

Expressions

funcions predefinides

itertools

En python tenim la possibilitat de crear llistes per comprensió. És a dir, en una expressió breu generem els valors de la llista. Per exemple,

Expressions

```
>>> [n for n in range(3,8)]
[3, 4, 5, 6, 7]
>>> [n*n for n in range(10)]
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
>>> [(i,j) for j in range(2) for i in range(3)]
[(0, 0), (1, 0), (2, 0), (0, 1), (1, 1), (2, 1)]
>>> [[j for j in range(2)] for i in range(3)]
[[0, 1], [0, 1], [0, 1]]
```

Expressions

funcions predefinides

itertools

En python tenim la possibilitat de crear llistes per comprensió. És a dir, en una expressió breu generem els valors de la llista. Per exemple,

Expressions

```
>>> [n for n in range(3,8)]
[3, 4, 5, 6, 7]
>>> [n*n for n in range(10)]
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
>>> [(i,j) for j in range(2) for i in range(3)]
[(0, 0), (1, 0), (2, 0), (0, 1), (1, 1), (2, 1)]
>>> [[j for j in range(2)] for i in range(3)]
[[0, 1], [0, 1], [0, 1]]
```

Expressions

funcions predefinides

itertools

En python tenim la possibilitat de crear llistes per comprensió. És a dir, en una expressió breu generem els valors de la llista. Per exemple,

Expressions

```
>>> [n for n in range(3,8)]
[3, 4, 5, 6, 7]
>>> [n*n for n in range(10)]
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
>>> [(i,j) for j in range(2) for i in range(3)]
[(0, 0), (1, 0), (2, 0), (0, 1), (1, 1), (2, 1)]
>>> [[j for j in range(2)] for i in range(3)]
[[0, 1], [0, 1], [0, 1]]
```

Expressions

funcions predefinides

itertools

En python tenim la possibilitat de crear llistes per comprensió. És a dir, en una expressió breu generem els valors de la llista. Per exemple,

Expressions

```
>>> [n for n in range(3,8)]
[3, 4, 5, 6, 7]
>>> [n*n for n in range(10)]
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
>>> [(i,j) for j in range(2) for i in range(3)]
[(0, 0), (1, 0), (2, 0), (0, 1), (1, 1), (2, 1)]
>>> [[j for j in range(2)] for i in range(3)]
[[0, 1], [0, 1], [0, 1]]
```

Expressions

funcions predefinides

itertools

En python tenim la possibilitat de crear llistes per comprensió. És a dir, en una expressió breu generem els valors de la llista. Per exemple,

Expressions

```
>>> [n for n in range(3,8)]
[3, 4, 5, 6, 7]
>>> [n*n for n in range(10)]
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
>>> [(i,j) for j in range(2) for i in range(3)]
[(0, 0), (1, 0), (2, 0), (0, 1), (1, 1), (2, 1)]
>>> [[j for j in range(2)] for i in range(3)]
[[0, 1], [0, 1], [0, 1]]
```

Expressions

funcions predefinides

itertools

En python tenim la possibilitat de crear llistes per comprensió. És a dir, en una expressió breu generem els valors de la llista. Per exemple,

Expressions

```
>>> [n for n in range(3,8)]
[3, 4, 5, 6, 7]
>>> [n*n for n in range(10)]
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
>>> [(i,j) for j in range(2) for i in range(3)]
[(0, 0), (1, 0), (2, 0), (0, 1), (1, 1), (2, 1)]
>>> [[j for j in range(2)] for i in range(3)]
[[0, 1], [0, 1], [0, 1]]
```

Expressions

funcions predefinides

itertools

També en podem crear generadors mitjançant les mateixes expressions.

Expressions

```
>>> (n for n in range(3,8))
<generator object <genexpr> at ...>
>>> for z in (n for n in range(3,8)):
...     print(z, end=', ')
...
3, 4, 5, 6, 7,
>>> dezeroau= (n for n in range(2))
>>> for n in dezeroau:
...     print(n, end=", ")
...
0, 1,
```

Expressions

funcions predefinides

itertools

També en podem crear generadors mitjançant les mateixes expressions.

Expressions

```
>>> (n for n in range(3,8))
<generator object <genexpr> at ...>
>>> for z in (n for n in range(3,8)):
...     print(z, end=', ')
...
3, 4, 5, 6, 7,
>>> dezeroau= (n for n in range(2))
>>> for n in dezeroau:
...     print(n, end=", ")
...
0, 1,
```

Expressions

funcions predefinides

itertools

També en podem crear generadors mitjançant les mateixes expressions.

Expressions

```
>>> (n for n in range(3,8))
<generator object <genexpr> at ...>
>>> for z in (n for n in range(3,8)):
...     print(z, end=', ')
...
3, 4, 5, 6, 7,
>>> dezeroau= (n for n in range(2))
>>> for n in dezeroau:
...     print(n, end=", ")
...
0, 1,
```

Expressions

funcions predefinides

itertools

També en podem crear generadors mitjançant les mateixes expressions.

Expressions

```
>>> (n for n in range(3,8))
<generator object <genexpr> at ...>
>>> for z in (n for n in range(3,8)):
...     print(z, end=', ')
...
3, 4, 5, 6, 7,
>>> dezeroau= (n for n in range(2))
>>> for n in dezeroau:
...     print(n, end=", ")
...
0, 1,
```

Expressions

funcions predefinides

itertools

També en podem crear generadors mitjançant les mateixes expressions.

Expressions

```
>>> (n for n in range(3,8))
<generator object <genexpr> at ...>
>>> for z in (n for n in range(3,8)):
...     print(z, end=', ')
...
3, 4, 5, 6, 7,
>>> dezeroau= (n for n in range(2))
>>> for n in dezeroau:
...     print(n, end=", ")
...
0, 1,
```

Expressions

funcions predefinides

itertools

També en podem crear generadors mitjançant les mateixes expressions.

Expressions

```
>>> (n for n in range(3,8))
<generator object <genexpr> at ...>
>>> for z in (n for n in range(3,8)):
...     print(z, end=', ')
...
3, 4, 5, 6, 7,
>>> dezeroau= (n for n in range(2))
>>> for n in dezeroau:
...     print(n, end=", ")
...
0, 1,
```

Expressions

funcions predefinides

itertools

Expressions

```
>>> for n in dezeroau:
...     print(n, end=", ")
...
>>>
>>> [(x for x in range(2))]
[<generator object <genexpr> at ...>]
```

Expressions

funcions predefinides

itertools

Expressions

```
>>> for n in dezeroau:
...     print(n, end=", ")
...
>>>
>>> [(x for x in range(2))]
[<generator object <genexpr> at ...>]
```

Expressions

funcions predefinides

itertools

Expressions

```
>>> for n in dezeroau:
...     print(n, end=", ")
...
>>>
>>> [(x for x in range(2))]
[<generator object <genexpr> at ...>]
```

Expressions

funcions predefinides

itertools

Expressions

```
>>> for n in dezeroau:
...     print(n, end=", ")
...
>>>
>>> [(x for x in range(2))]
[<generator object <genexpr> at ...>]
```

Expressions

funcions predefinides

itertools

Expressions

```
>>> for n in dezeroau:
...     print(n, end=", ")
...
>>>
>>> [(x for x in range(2))]
[<generator object <genexpr> at ...>]
```

Expressions

funcions predefinides

itertools

Expressions

```
>>> s="frase exemple amb unes quantes paraules"
>>> max(len(mot) for mot in s.split())
8
>>>  # Primer crea llista, després suma
>>> sum([x*x for x in range(10)])
285
>>> # suma sense usar memòria
>>> sum(x*x for x in range(10))
285
```

Expressions

funcions predefinides

itertools

Expressions

```
>>> s="frase exemple amb unes quantes paraules"
>>> max(len(mot) for mot in s.split())
8
>>>  # Primer crea llista, després suma
>>> sum([x*x for x in range(10)])
285
>>> # suma sense usar memòria
>>> sum(x*x for x in range(10))
285
```

Expressions

funcions predefinides

itertools

Expressions

```
>>> s="frase exemple amb unes quantes paraules"
>>> max(len(mot) for mot in s.split())
8
>>>  # Primer crea llista, després suma
>>> sum([x*x for x in range(10)])
285
>>> # suma sense usar memòria
>>> sum(x*x for x in range(10))
285
```

Expressions

funcions predefinides

itertools

Expressions

```
>>> s="frase exemple amb unes quantes paraules"
>>> max(len(mot) for mot in s.split())
8
>>>  # Primer crea llista, després suma
>>> sum([x*x for x in range(10)])
285
>>> # suma sense usar memòria
>>> sum(x*x for x in range(10))
285
```

Expressions

funcions predefinides

itertools

Expressions

```
>>> s="frase exemple amb unes quantes paraules"
>>> max(len(mot) for mot in s.split())
8
>>>  # Primer crea llista, després suma
>>> sum([x*x for x in range(10)])
285
>>> # suma sense usar memòria
>>> sum(x*x for x in range(10))
285
```

Expressions
funcions predefinides
itertools

Expressions

```
>>> s="frase exemple amb unes quantes paraules"
>>> max(len(mot) for mot in s.split())
8
>>>  # Primer crea llista, després suma
>>> sum([x*x for x in range(10)])
285
>>> # suma sense usar memòria
>>> sum(x*x for x in range(10))
285
```

Expressions
funcions predefinides
itertools

Expressions

```
>>> s="frase exemple amb unes quantes paraules"
>>> max(len(mot) for mot in s.split())
8
>>>  # Primer crea llista, després suma
>>> sum([x*x for x in range(10)])
285
>>> # suma sense usar memòria
>>> sum(x*x for x in range(10))
285
```

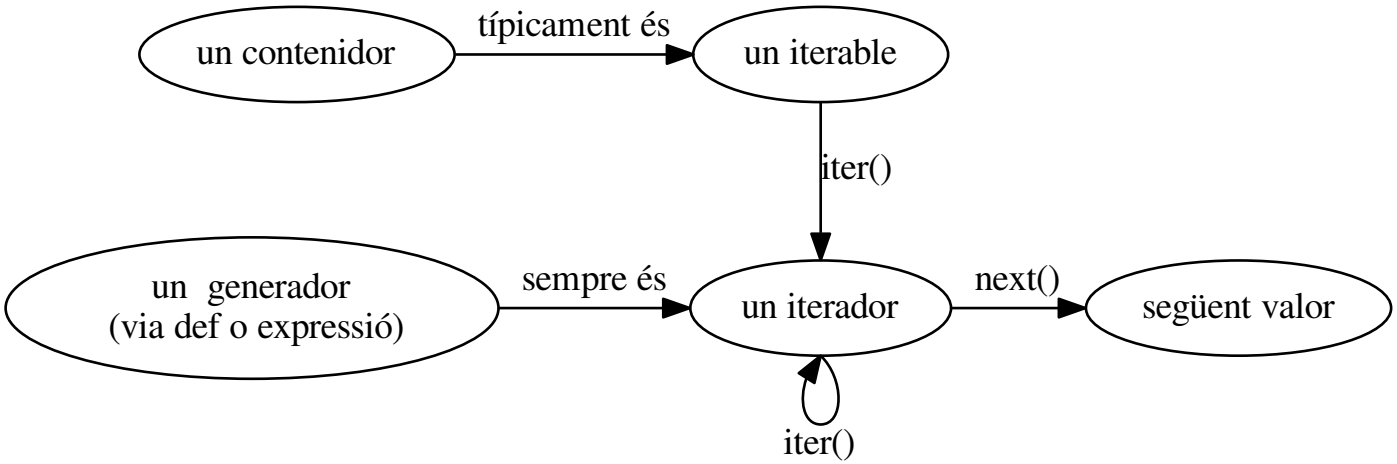
Expressions
funcions predefinides
itertools

Expressions

```
>>> s="frase exemple amb unes quantes paraules"
>>> max(len(mot) for mot in s.split())
8
>>>  # Primer crea llista, després suma
>>> sum([x*x for x in range(10)])
285
>>> # suma sense usar memòria
>>> sum(x*x for x in range(10))
285
```

Resum

- Un contenidor és un objecte que té l'operador `in`.
- Un iterador és un objecte que té el mètode `__next__()`
- Un iterable és un objecte que té el mètode `__iter__()`
- `for` accepta només iteradors que són a la vegada iterables.



Resum

- Si volem un dissenyar només un iterador:
  - → definició de funció generadora (`yield`), o
  - → definició d'expressió
- Si a més del comportament d'iterador (`__iter__()`, `__next__()`) volem afegir altres mètodes:
  - → definició de la classe (`class`)

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

`filter(propietat,iterable)` equival a

`(elem for elem in iterable if propietat(elem))`

o bé

```
def filter(propietat,iterable):
    for elem in iterable:
        if propietat(elem):
            yield elem
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Generar nombres parells.

filter

```
>>> def parell(n):
...     return (n % 2) == 0
...
>>> for i in filter(parell, range(10)):
...     print(i, end='-')
...
0-2-4-6-8-
>>> for i in filter(lambda x: (x % 2) == 0, range(10)
...     print(i, end='-')
...
0-2-4-6-8-
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Generar nombres parells.

filter

```
>>> def parell(n):
...     return (n % 2) == 0
...
>>> for i in filter(parell, range(10)):
...     print(i, end='-')
...
0-2-4-6-8-
>>> for i in filter(lambda x: (x % 2) == 0, range(10)
...     print(i, end='-')
...
0-2-4-6-8-
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Generar nombres parells.

filter

```
>>> def parell(n):
...     return (n % 2) == 0
...
>>> for i in filter(parell, range(10)):
...     print(i, end='-')
...
0-2-4-6-8-
>>> for i in filter(lambda x: (x % 2) == 0, range(10)):
...     print(i, end='-')
...
0-2-4-6-8-
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Generar nombres parells.

filter

```
>>> def parell(n):
...     return (n % 2) == 0
...
>>> for i in filter(parell, range(10)):
...     print(i, end='-')
...
0-2-4-6-8-
>>> for i in filter(lambda x: (x % 2) == 0, range(10)
...     print(i, end='-')
...
0-2-4-6-8-
```

Expressions
funcions predefinides
filter
map
reduce
any
all
Exemples
itertools

Generar nombres parells.

filter

```
>>> def parell(n):
...     return (n % 2) == 0
...
>>> for i in filter(parell, range(10)):
...     print(i, end='-')
...
0-2-4-6-8-
>>> for i in filter(lambda x: (x % 2) == 0, range(10)
...     print(i, end='-')
...
0-2-4-6-8-
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Construir una llista amb els divisors d'un nombre n excepte 1 i el pròpi nombre.

filter

```
>>> n=100
>>> for d in filter(lambda d: n % d== 0,range(2,n)):
...     print(d, end='-')
...
2-4-5-10-20-25-50-
>>> n=1
>>> for d in filter(lambda d: n % d== 0,range(2,n)):
...     print(d, end='-')
...
>>>
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Construir una llista amb els divisors d'un nombre n excepte 1 i el pròpi nombre.

filter

```
>>> n=100
>>> for d in filter(lambda d: n % d== 0,range(2,n)):
...     print(d, end='-')
...
2-4-5-10-20-25-50-
>>> n=1
>>> for d in filter(lambda d: n % d== 0,range(2,n)):
...     print(d, end='-')
...
>>>
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Construir una llista amb els divisors d'un nombre n excepte 1 i el pròpi nombre.

filter

```
>>> n=100
>>> for d in filter(lambda d: n % d== 0,range(2,n)):
...         print(d, end='-')
...
2-4-5-10-20-25-50-
>>> n=1
>>> for d in filter(lambda d: n % d== 0,range(2,n)):
...         print(d, end='-')
...
>>>
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Construir una llista amb els divisors d'un nombre n excepte 1 i el pròpi nombre.

filter

```
>>> n=100
>>> for d in filter(lambda d: n % d== 0,range(2,n)):
...       print(d, end='-')
...
2-4-5-10-20-25-50-
>>> n=1
>>> for d in filter(lambda d: n % d== 0,range(2,n)):
...       print(d, end='-')
...
>>>
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Construir una llista amb els divisors d'un nombre n excepte 1 i el pròpi nombre.

filter

```
>>> n=100
>>> for d in filter(lambda d: n % d== 0,range(2,n)):
...       print(d, end='-')
...
2-4-5-10-20-25-50-
>>> n=1
>>> for d in filter(lambda d: n % d== 0,range(2,n)):
...       print(d, end='-')
...
>>>
```

Expressions
funcions predefinides
filter
map
reduce
any
all
Exemples
itertools

Construir una llista amb els divisors d'un nombre n excepte 1 i el pròpi nombre.

filter

```
>>> n=100
>>> for d in filter(lambda d: n % d== 0,range(2,n)):
...       print(d, end='-')
...
2-4-5-10-20-25-50-
>>> n=1
>>> for d in filter(lambda d: n % d== 0,range(2,n)):
...       print(d, end='-')
...
>>>
```

Expressions
funcions predefinides
filter
map
reduce
any
all
Exemples
itertools

Construir una llista amb els divisors d'un nombre n excepte 1 i el pròpi nombre.

filter

```
>>> n=100
>>> for d in filter(lambda d: n % d== 0,range(2,n)):
...       print(d, end='-')
...
2-4-5-10-20-25-50-
>>> n=1
>>> for d in filter(lambda d: n % d== 0,range(2,n)):
...       print(d, end='-')
...
>>>
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

`map(funcio,iterable)` equival a
`(funcio(elem) for elem in iterable)`
o bé

```
def map(funcio, iterable):
    for elem in iterable:
        yield funcio(elem)
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Passar una llista de mesures en polzades britàniques a una llista amb centímetres.

map

```
>>> def a_cm(polzades):
...     return 2.54*polzades
...
>>> for m in map(a_cm, [1,2,3,4,5]):
...     print(m, end='-')
...
2.54-5.08-7.62-10.16-12.7-
>>> for m in map(lambda m: m*2.54, [1,2,3,4,5]):
...     print(m, end='-')
...
2.54-5.08-7.62-10.16-12.7-
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Passar una llista de mesures en polzades britàniques a una llista amb centímetres.

map

```
>>> def a_cm(polzades):
...     return 2.54*polzades
...
>>> for m in map(a_cm, [1,2,3,4,5]):
...     print(m, end='-')
...
2.54-5.08-7.62-10.16-12.7-
>>> for m in map(lambda m: m*2.54, [1,2,3,4,5]):
...     print(m, end='-')
...
2.54-5.08-7.62-10.16-12.7-
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Passar una llista de mesures en polzades britàniques a una llista amb centímetres.

map

```
>>> def a_cm(polzades):
...     return 2.54*polzades
...
>>> for m in map(a_cm, [1,2,3,4,5]):
...     print(m, end='-')
...
2.54-5.08-7.62-10.16-12.7-
>>> for m in map(lambda m: m*2.54, [1,2,3,4,5]):
...     print(m, end='-')
...
2.54-5.08-7.62-10.16-12.7-
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Passar una llista de mesures en polzades britàniques a una llista amb centímetres.

map

```
>>> def a_cm(polzades):
...     return 2.54*polzades
...
>>> for m in map(a_cm, [1,2,3,4,5]):
...     print(m, end='-')
...
2.54-5.08-7.62-10.16-12.7-
>>> for m in map(lambda m: m*2.54, [1,2,3,4,5]):
...     print(m, end='-')
...
2.54-5.08-7.62-10.16-12.7-
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Passar una llista de mesures en polzades britàniques a una llista amb centímetres.

map

```
>>> def a_cm(polzades):
...     return 2.54*polzades
...
>>> for m in map(a_cm, [1,2,3,4,5]):
...     print(m, end='-')
...
2.54-5.08-7.62-10.16-12.7-
>>> for m in map(lambda m: m*2.54, [1,2,3,4,5]):
...     print(m, end='-')
...
2.54-5.08-7.62-10.16-12.7-
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Trobar la longitud de la cadena més llarga que hagi a la llista *llista*.

map

```
>>> def longitud_maxima(llista):
...     return max(map(len, llista), default=0)
...
>>> longitud_maxima(['pera', 'albercoc', 'taronja', 'ma
duixa'])
8
>>> longitud_maxima([])
0
>>> longitud_maxima(['bar', 'cel', 'ona'])
3
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Trobar la longitud de la cadena més llarga que hagi a la llista *llista*.

map

```
>>> def longitud_maxima(llista):
...     return max(map(len, llista), default=0)
...
>>> longitud_maxima(['pera', 'albercoc', 'taronja', 'ma
duixa'])
8
>>> longitud_maxima([])
0
>>> longitud_maxima(['bar', 'cel', 'ona'])
3
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Trobar la longitud de la cadena més llarga que hagi a la llista *llista*.

map

```
>>> def longitud_maxima(llista):
...     return max(map(len, llista), default=0)
...
>>> longitud_maxima(['pera', 'albercoc', 'taronja', 'maduixa'])
8
>>> longitud_maxima([])
0
>>> longitud_maxima(['bar', 'cel', 'ona'])
3
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Trobar la longitud de la cadena més llarga que hagi a la llista *llista*.

map

```
>>> def longitud_maxima(llista):
...     return max(map(len, llista), default=0)
...
>>> longitud_maxima(['pera', 'albercoc', 'taronja', 'ma
duixa'])
8
>>> longitud_maxima([])
0
>>> longitud_maxima(['bar', 'cel', 'ona'])
3
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Trobar la longitud de la cadena més llarga que hagi a la llista *llista*.

map

```
>>> def longitud_maxima(llista):
...     return max(map(len, llista), default=0)
...
>>> longitud_maxima(['pera', 'albercoc', 'taronja', 'ma
duixa'])
8
>>> longitud_maxima([])
0
>>> longitud_maxima(['bar', 'cel', 'ona'])
3
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Trobar la longitud de la cadena més llarga que hagi a la llista *llista*.

map

```
>>> def longitud_maxima(llista):
...     return max(map(len, llista), default=0)
...
>>> longitud_maxima(['pera', 'albercoc', 'taronja', 'ma
duixa'])
8
>>> longitud_maxima([])
0
>>> longitud_maxima(['bar', 'cel', 'ona'])
3
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Pot haver més d'un iterador. La funció a passar caldrà que tingui tants arguments com iteradors.
Amb dos iteradors el comportament és:

```
def map(funció, iterable1, iterable2):
    it1 = iter(iterable1)
    it2 = iter(iterable2)
    while True:
        try:
            a = next(it1)
            b = next(it2)
            yield(funció(a,b))
        except StopIteration:
            break
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Pot haver més d'un iterador. La funció a passar caldrà que tingui tants arguments com iteradors.
Amb dos iteradors el comportament és:

```
def map(funció, iterable1, iterable2):
    for a, b in zip(iterable1, iterable2):
            yield(funció(a,b))
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Pot haver més d'un iterador. La funció a passar caldrà que tingui tants arguments com iteradors.
Amb *n* iteradors el comportament és:

```
def map(funció, *iterables):
    for valors in zip(*iterables):
        yield funció(*valors)
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Per exemple,

map

```
>>> for e in map(lambda a, b: a==b, [1,2,3], [3,2,1,0]):
...     print(e)
...
False
True
False
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Per exemple,

map

```
>>> for e in map(lambda a, b: a==b, [1,2,3], [3,2,1,0]):
...     print(e)
...
False
True
False
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Per exemple,

map

```
>>> for e in map(lambda a, b: a==b, [1,2,3], [3,2,1,0])
:
...     print(e)
...
False
True
False
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

reduce(funcio, iterable)

Si $iterable = [e_1, e_2, e_3, e_4]$,

$$\frac{[e_1, \quad e_2, \quad e_3, \quad e_4]}{funcio(e_1, e_2) \quad \frac{}{funcio(funcio(e_1, e_2), e3) \quad \frac{}{funcio(funcio(funcio(e_1, e_2), e_3), e_4)}}}$$

Si $iterable = [e_1]$ torna e_1

Si $reduce(funcio, [e_1], valorinicial)$ torna $funcio(valorinicial, e_1)$

reduce

Expressions
funcions predefinides
filter
map
reduce
any
all
Exemples
itertools

```
def reduce(funció, iterable):
    it = iter(iterable)
    compte = next(it)
    for elem in it:
        compte = funció(compte, elem)
    return compte
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

reduce (Exemples)

```
>>> import functools
>>> import operator
>>> functools.reduce(operator.add, [1,2,3,4])
10
>>> functools.reduce(operator.add, [])
Traceback (most recent call last):
  File ≪stdin>", line 1, in <module>
TypeError: reduce() of empty sequence with no initial v
alue
>>> functools.reduce(operator.add, [], 3)
3
>>> functools.reduce(operator.add, [1], 3)
4
>>> functools.reduce(operator.add, [1])
1
>>> sum([1,2,3,4])==functools.reduce(operator.add, [1,2
,3,4])
True
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

reduce (Exemples)

```
>>> import functools
>>> import operator
>>> functools.reduce(operator.add, [1,2,3,4])
10
>>> functools.reduce(operator.add, [])
Traceback (most recent call last):
  File ≪stdin>", line 1, in <module>
TypeError: reduce() of empty sequence with no initial v
alue
>>> functools.reduce(operator.add, [], 3)
3
>>> functools.reduce(operator.add, [1], 3)
4
>>> functools.reduce(operator.add, [1])
1
>>> sum([1,2,3,4])==functools.reduce(operator.add, [1,2
,3,4])
True
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

reduce (Exemples)

```
>>> import functools
>>> import operator
>>> functools.reduce(operator.add, [1,2,3,4])
10
>>> functools.reduce(operator.add, [])
Traceback (most recent call last):
  File «stdin>", line 1, in <module>
TypeError: reduce() of empty sequence with no initial v
alue
>>> functools.reduce(operator.add, [], 3)
3
>>> functools.reduce(operator.add, [1], 3)
4
>>> functools.reduce(operator.add, [1])
1
>>> sum([1,2,3,4])==functools.reduce(operator.add, [1,2
,3,4])
True
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

reduce (Exemples)

```
>>> import functools
>>> import operator
>>> functools.reduce(operator.add, [1,2,3,4])
10
>>> functools.reduce(operator.add, [])
Traceback (most recent call last):
  File ≪stdin>", line 1, in <module>
TypeError: reduce() of empty sequence with no initial v
alue
>>> functools.reduce(operator.add, [], 3)
3
>>> functools.reduce(operator.add, [1], 3)
4
>>> functools.reduce(operator.add, [1])
1
>>> sum([1,2,3,4])==functools.reduce(operator.add, [1,2
,3,4])
True
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

reduce (Exemples)

```
>>> import functools
>>> import operator
>>> functools.reduce(operator.add, [1,2,3,4])
10
>>> functools.reduce(operator.add, [])
Traceback (most recent call last):
  File «stdin>", line 1, in <module>
TypeError: reduce() of empty sequence with no initial v
alue
>>> functools.reduce(operator.add, [], 3)
3
>>> functools.reduce(operator.add, [1], 3)
4
>>> functools.reduce(operator.add, [1])
1
>>> sum([1,2,3,4])==functools.reduce(operator.add, [1,2
,3,4])
True
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

reduce (Exemples)

```
>>> import functools
>>> import operator
>>> functools.reduce(operator.add, [1,2,3,4])
10
>>> functools.reduce(operator.add, [])
Traceback (most recent call last):
  File ≪stdin>", line 1, in <module>
TypeError: reduce() of empty sequence with no initial v
alue
>>> functools.reduce(operator.add, [], 3)
3
>>> functools.reduce(operator.add, [1], 3)
4
>>> functools.reduce(operator.add, [1])
1
>>> sum([1,2,3,4])==functools.reduce(operator.add, [1,2
,3,4])
True
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

reduce (Exemples)

```
>>> import functools
>>> import operator
>>> functools.reduce(operator.add, [1,2,3,4])
10
>>> functools.reduce(operator.add, [])
Traceback (most recent call last):
  File ≪stdin>", line 1, in <module>
TypeError: reduce() of empty sequence with no initial v
alue
>>> functools.reduce(operator.add, [], 3)
3
>>> functools.reduce(operator.add, [1], 3)
4
>>> functools.reduce(operator.add, [1])
1
>>> sum([1,2,3,4])==functools.reduce(operator.add, [1,2
,3,4])
True
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

reduce (Exemples)

```
>>> import functools
>>> import operator
>>> functools.reduce(operator.add, [1,2,3,4])
10
>>> functools.reduce(operator.add, [])
Traceback (most recent call last):
  File ≪stdin>", line 1, in <module>
TypeError: reduce() of empty sequence with no initial v
alue
>>> functools.reduce(operator.add, [], 3)
3
>>> functools.reduce(operator.add, [1], 3)
4
>>> functools.reduce(operator.add, [1])
1
>>> sum([1,2,3,4])==functools.reduce(operator.add, [1,2
,3,4])
True
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

reduce (Exemples)

```
>>> import functools
>>> import operator
>>> functools.reduce(operator.add, [1,2,3,4])
10
>>> functools.reduce(operator.add, [])
Traceback (most recent call last):
  File ≪stdin>", line 1, in <module>
TypeError: reduce() of empty sequence with no initial v
alue
>>> functools.reduce(operator.add, [], 3)
3
>>> functools.reduce(operator.add, [1], 3)
4
>>> functools.reduce(operator.add, [1])
1
>>> sum([1,2,3,4])==functools.reduce(operator.add, [1,2
,3,4])
True
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

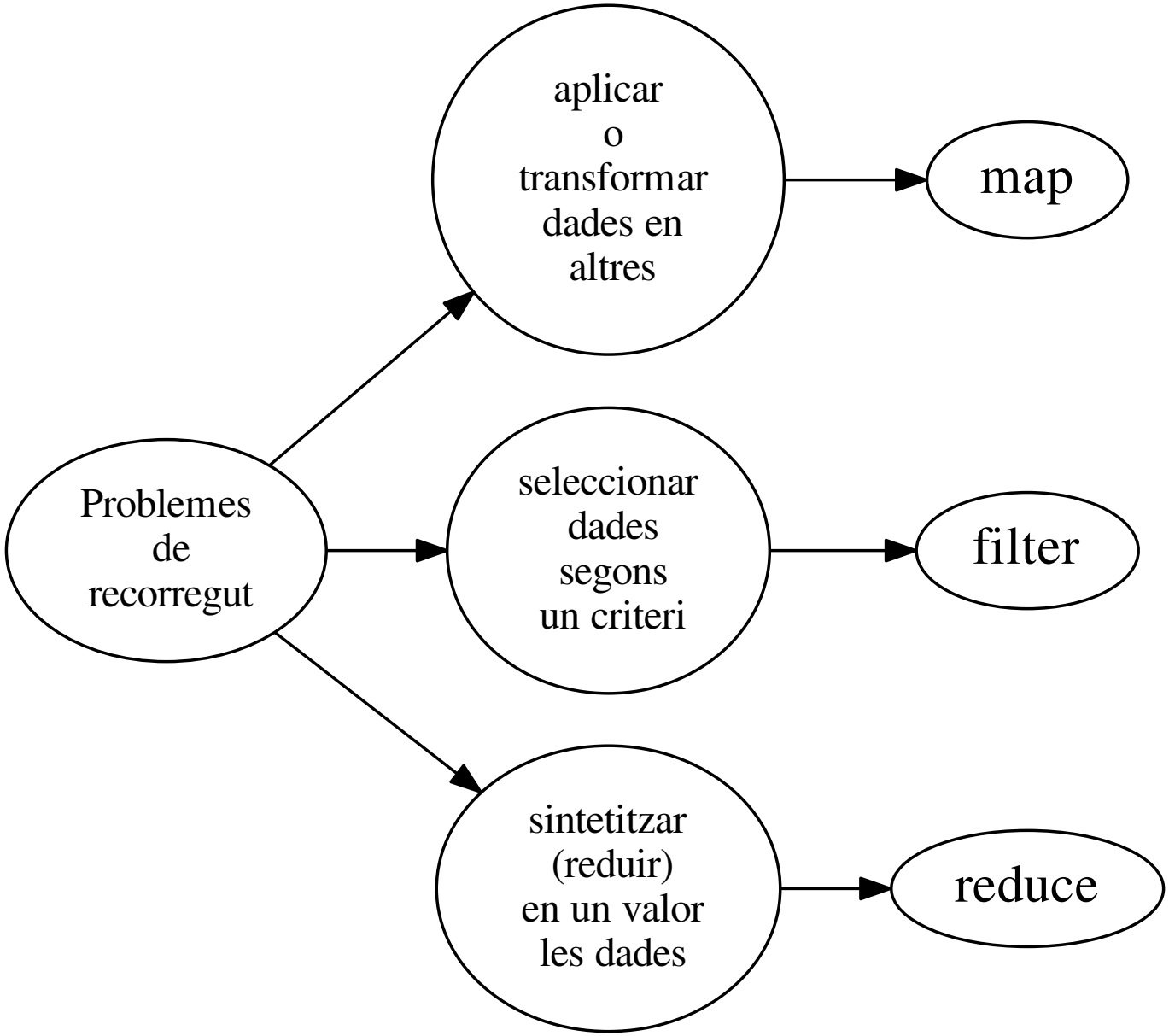
itertools

reduce (Exemples)

```
>>> import functools
>>> import operator
>>> functools.reduce(operator.add, [1,2,3,4])
10
>>> functools.reduce(operator.add, [])
Traceback (most recent call last):
  File ≪stdin>", line 1, in <module>
TypeError: reduce() of empty sequence with no initial v
alue
>>> functools.reduce(operator.add, [], 3)
3
>>> functools.reduce(operator.add, [1], 3)
4
>>> functools.reduce(operator.add, [1])
1
>>> sum([1,2,3,4])==functools.reduce(operator.add, [1,2
,3,4])
True
```

Resum

Expressions
funcions predefinides
filter
map
reduce
any
all
Exemples
itertools



Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

```
def any(iterable):
    for element in iterable:
        if element:
            return True
    return False
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

```
def algun_negatiu(l):
    """ Retorna True si la llista d'enters l
        conté algun element negatiu."""
```

any

```
>>> def algun_negatiu(l):
...     return any(map(lambda x: x < 0, l))
>>> algun_negatiu([1, 2, 3, 4])
False
>>> algun_negatiu([2, -4, 8, 5, 7])
True
>>> algun_negatiu([])
False
>>> any(e<0 for e in [1, 2, 3, -4, -5])
True
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

```
def algun_negatiu(l):
    """ Retorna True si la llista d'enters l
        conté algun element negatiu."""
```

any

```
>>> def algun_negatiu(l):
...     return any(map(lambda x: x < 0, l))
>>> algun_negatiu([1, 2, 3, 4])
False
>>> algun_negatiu([2, -4, 8, 5, 7])
True
>>> algun_negatiu([])
False
>>> any(e<0 for e in [1, 2, 3, -4, -5])
True
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

```
def algun_negatiu(l):
    """ Retorna True si la llista d'enters l
        conté algun element negatiu."""
```

any

```
>>> def algun_negatiu(l):
...     return any(map(lambda x: x < 0, l))
>>> algun_negatiu([1, 2, 3, 4])
False
>>> algun_negatiu([2, -4, 8, 5, 7])
True
>>> algun_negatiu([])
False
>>> any(e<0 for e in [1, 2, 3, -4, -5])
True
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

```
def algun_negatiu(l):
    """ Retorna True si la llista d'enters l
        conté algun element negatiu."""
```

any

```
>>> def algun_negatiu(l):
...     return any(map(lambda x: x < 0, l))
>>> algun_negatiu([1, 2, 3, 4])
False
>>> algun_negatiu([2, -4, 8, 5, 7])
True
>>> algun_negatiu([])
False
>>> any(e<0 for e in [1, 2, 3, -4, -5])
True
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

```
def algun_negatiu(l):
    """ Retorna True si la llista d'enters l
        conté algun element negatiu."""
```

any

```
>>> def algun_negatiu(l):
...     return any(map(lambda x: x < 0, l))
>>> algun_negatiu([1, 2, 3, 4])
False
>>> algun_negatiu([2, -4, 8, 5, 7])
True
>>> algun_negatiu([])
False
>>> any(e<0 for e in [1, 2, 3, -4, -5])
True
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

```
def algun_negatiu(l):
    """ Retorna True si la llista d'enters l
        conté algun element negatiu."""
```

any

```
>>> def algun_negatiu(l):
...     return any(map(lambda x: x < 0, l))
>>> algun_negatiu([1, 2, 3, 4])
False
>>> algun_negatiu([2, -4, 8, 5, 7])
True
>>> algun_negatiu([])
False
>>> any(e<0 for e in [1, 2, 3, -4, -5])
True
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

```
def algun_negatiu(l):
    """ Retorna True si la llista d'enters l
        conté algun element negatiu."""
```

any

```
>>> def algun_negatiu(l):
...     return any(map(lambda x: x < 0, l))
>>> algun_negatiu([1, 2, 3, 4])
False
>>> algun_negatiu([2, -4, 8, 5, 7])
True
>>> algun_negatiu([])
False
>>> any(e<0 for e in [1, 2, 3, -4, -5])
True
```

all

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

```
def all(iterable):
    for element in iterable:
        if not element:
            return False
    return True
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

```
def cap_negatiu(l):
    """Retorna True si cap nombre de la llista l
       és negatiu i False altrament."""
```

all

```
>>> def cap_negatiu(l):
...     return all(map(lambda x: x >= 0, l))
>>> cap_negatiu([1, 2, 3, 4])
True
>>> cap_negatiu([2, -4, 8, 5, 7])
False
>>> cap_negatiu([])
True
>>> all(e>0 for e in [1, 2, 3, -4, -5])
False
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

```
def cap_negatiu(l):
    """Retorna True si cap nombre de la llista l
       és negatiu i False altrament."""
```

all

```
>>> def cap_negatiu(l):
...     return all(map(lambda x: x >= 0, l))
>>> cap_negatiu([1, 2, 3, 4])
True
>>> cap_negatiu([2, -4, 8, 5, 7])
False
>>> cap_negatiu([])
True
>>> all(e>0 for e in [1, 2, 3, -4, -5])
False
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

```
def cap_negatiu(l):
    """Retorna True si cap nombre de la llista l
       és negatiu i False altrament."""
```

all

```
>>> def cap_negatiu(l):
...     return all(map(lambda x: x >= 0, l))
>>> cap_negatiu([1, 2, 3, 4])
True
>>> cap_negatiu([2, -4, 8, 5, 7])
False
>>> cap_negatiu([])
True
>>> all(e>0 for e in [1, 2, 3, -4, -5])
False
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

```
def cap_negatiu(l):
    """Retorna True si cap nombre de la llista l
       és negatiu i False altrament."""
```

all

```
>>> def cap_negatiu(l):
...     return all(map(lambda x: x >= 0, l))
>>> cap_negatiu([1, 2, 3, 4])
True
>>> cap_negatiu([2, -4, 8, 5, 7])
False
>>> cap_negatiu([])
True
>>> all(e>0 for e in [1, 2, 3, -4, -5])
False
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

```
def cap_negatiu(l):
    """Retorna True si cap nombre de la llista l
       és negatiu i False altrament."""
```

all

```
>>> def cap_negatiu(l):
...     return all(map(lambda x: x >= 0, l))
>>> cap_negatiu([1, 2, 3, 4])
True
>>> cap_negatiu([2, -4, 8, 5, 7])
False
>>> cap_negatiu([])
True
>>> all(e>0 for e in [1, 2, 3, -4, -5])
False
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

```
def cap_negatiu(l):
    """Retorna True si cap nombre de la llista l
       és negatiu i False altrament."""
```

all

```
>>> def cap_negatiu(l):
...     return all(map(lambda x: x >= 0, l))
>>> cap_negatiu([1, 2, 3, 4])
True
>>> cap_negatiu([2, -4, 8, 5, 7])
False
>>> cap_negatiu([])
True
>>> all(e>0 for e in [1, 2, 3, -4, -5])
False
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

```
def cap_negatiu(l):
    """Retorna True si cap nombre de la llista l
       és negatiu i False altrament."""
```

all

```
>>> def cap_negatiu(l):
...     return all(map(lambda x: x >= 0, l))
>>> cap_negatiu([1, 2, 3, 4])
True
>>> cap_negatiu([2, -4, 8, 5, 7])
False
>>> cap_negatiu([])
True
>>> all(e>0 for e in [1, 2, 3, -4, -5])
False
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Resum

Problemes de cerca:

- $\exists x \in S, \quad x < 0 \rightarrow$ `any(propietat, S)`
- $\forall x \in S, \quad x < 0 \rightarrow$ `all(propietat, S)`

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Dissenyar la funció esPrimer(x) que torna True si el nombre x és primer, i False en cas contrari.

esPrimer amb all i map

```
>>> import math
>>> def esPrimer(x):
...     return all(map(lambda d: x % d!=0,
...                    range(2,int(math.sqrt(x))+1)))
...
>>> esPrimer(9)
False
>>> esPrimer(5)
True
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Dissenyar la funció esPrimer(x) que torna True si el nombre x és primer, i False en cas contrari.

esPrimer amb all i map

```
>>> import math
>>> def esPrimer(x):
...     return all(map(lambda d: x % d!=0,
...                    range(2,int(math.sqrt(x))+1)))
...
>>> esPrimer(9)
False
>>> esPrimer(5)
True
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Dissenyar la funció esPrimer(x) que torna True si el nombre x és primer, i False en cas contrari.

esPrimer amb all i map

```
>>> import math
>>> def esPrimer(x):
...     return all(map(lambda d: x % d!=0,
...                    range(2,int(math.sqrt(x))+1)))
...
>>> esPrimer(9)
False
>>> esPrimer(5)
True
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Dissenyar la funció esPrimer(x) que torna True si el nombre x és primer, i False en cas contrari.

esPrimer amb all i map

```
>>> import math
>>> def esPrimer(x):
...     return all(map(lambda d: x % d!=0,
...                    range(2,int(math.sqrt(x))+1)))
...
>>> esPrimer(9)
False
>>> esPrimer(5)
True
```

Expressions
funcions predefinides
filter
map
reduce
any
all
Exemples
itertools

Dissenyar la funció esPrimer(x) que torna True si el nombre x és primer, i False en cas contrari.

esPrimer amb all i map

```
>>> import math
>>> def esPrimer(x):
...     return all(map(lambda d: x % d!=0,
...                    range(2,int(math.sqrt(x))+1)))
...
>>> esPrimer(9)
False
>>> esPrimer(5)
True
```

Expressions
funcions predefinides
filter
map
reduce
any
all
Exemples
itertools

Dissenyar la funció esPrimer(x) que torna True si el nombre x és primer, i False en cas contrari.

esPrimer amb all i map

```
>>> import math
>>> def esPrimer(x):
...     return all(map(lambda d: x % d!=0,
...                    range(2,int(math.sqrt(x))+1)))
...
>>> esPrimer(9)
False
>>> esPrimer(5)
True
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Escriviu una expressió que torni la llista de tots els nombres primers inferiors a 130

llista dels n primers primers

```
>>> import math
>>> list(filter(lambda x: all(map(lambda d: x % d!=0,
...                               range(2,int(math.sqrt
(x))+1))),
...                           range(1,130)))
[1, 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43,
 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103,
107, 109, 113, 127]
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Escriviu una expressió que torni la llista de tots els nombres primers inferiors a 130

llista dels n primers primers

```
>>> import math
>>> list(filter(lambda x: all(map(lambda d: x % d!=0,
...                               range(2,int(math.sqrt
(x))+1))),
...                           range(1,130)))
[1, 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43,
 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103,
107, 109, 113, 127]
```

Expressions
funcions predefinides
filter
map
reduce
any
all
Exemples
itertools

Escriviu una expressió que torni la llista de tots els nombres primers inferiors a 130

llista dels n primers primers

```
>>> import math
>>> list(filter(lambda x: all(map(lambda d: x % d!=0,
...                               range(2,int(math.sqrt
(x))+1))),
...                           range(1,130)))
[1, 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43,
 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103,
107, 109, 113, 127]
```

Expressions
funcions predefinides
filter
map
reduce
any
all
Exemples
itertools

Escriviu una expressió que torni la llista de tots els nombres primers inferiors a 130

llista dels n primers primers

```
>>> import math
>>> list(filter(lambda x: all(map(lambda d: x % d!=0,
...                               range(2,int(math.sqrt
(x))+1))),
...                                range(1,130)))
[1, 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43,
 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103,
107, 109, 113, 127]
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Escriviu una expressió amb reduce que torni el màxim d'un iterable

Exemples

```
>>> import functools
>>> lst= [2,30,4,5,66, 1]
>>> functools.reduce(lambda x,y: x if x>y else y, lst)
66
>>> functools.reduce(lambda x,y: x if x>y else y, lst)
== max(lst)
True
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Escriviu una expressió amb reduce que torni el màxim d'un iterable

Exemples

```
>>> import functools
>>> lst= [2,30,4,5,66, 1]
>>> functools.reduce(lambda x,y: x if x>y else y, lst)
66
>>> functools.reduce(lambda x,y: x if x>y else y, lst)
== max(lst)
True
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Escriviu una expressió amb reduce que torni el màxim d'un iterable

Exemples

```
>>> import functools
>>> lst= [2,30,4,5,66, 1]
>>> functools.reduce(lambda x,y: x if x>y else y, lst)
66
>>> functools.reduce(lambda x,y: x if x>y else y, lst)
== max(lst)
True
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Escriviu una expressió amb reduce que torni el màxim d'un iterable

Exemples

```
>>> import functools
>>> lst= [2,30,4,5,66, 1]
>>> functools.reduce(lambda x,y: x if x>y else y, lst)
66
>>> functools.reduce(lambda x,y: x if x>y else y, lst)
== max(lst)
True
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Escriviu una expressió amb reduce que torni el màxim d'un iterable

Exemples

```
>>> import functools
>>> lst= [2,30,4,5,66, 1]
>>> functools.reduce(lambda x,y: x if x>y else y, lst)
66
>>> functools.reduce(lambda x,y: x if x>y else y, lst)
== max(lst)
True
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Escriviu una expressió amb reduce que torni el màxim d'un iterable

Exemples

```
>>> import functools
>>> lst= [2,30,4,5,66, 1]
>>> functools.reduce(lambda x,y: x if x>y else y, lst)
66
>>> functools.reduce(lambda x,y: x if x>y else y, lst)
== max(lst)
True
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Escriviu la funció factorial(n) que torna el factorial de n sense usar cap for.

Exemples

```
>>> import functools
>>> def factorial(n):
...     return  functools.reduce(lambda x,y: x*y,
...                              range(1,n+1))
...
>>> factorial(3)
6
>>> factorial(25)
15511210043330985984000000
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Escriviu la funció factorial(n) que torna el factorial de n sense usar cap for.

Exemples

```
>>> import functools
>>> def factorial(n):
...     return  functools.reduce(lambda x,y: x*y,
...                              range(1,n+1))
...
>>> factorial(3)
6
>>> factorial(25)
15511210043330985984000000
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Escriviu la funció factorial(n) que torna el factorial de n sense usar cap for.

Exemples

```
>>> import functools
>>> def factorial(n):
...     return  functools.reduce(lambda x,y: x*y,
...                              range(1,n+1))
...
>>> factorial(3)
6
>>> factorial(25)
15511210043330985984000000
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Escriviu la funció factorial(n) que torna el factorial de n sense usar cap for.

Exemples

```
>>> import functools
>>> def factorial(n):
...     return  functools.reduce(lambda x,y: x*y,
...                              range(1,n+1))
...
>>> factorial(3)
6
>>> factorial(25)
15511210043330985984000000
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Escriviu la funció factorial(n) que torna el factorial de n sense usar cap for.

Exemples

```
>>> import functools
>>> def factorial(n):
...     return  functools.reduce(lambda x,y: x*y,
...                              range(1,n+1))
...
>>> factorial(3)
6
>>> factorial(25)
15511210043330985984000000
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Escriviu la funció factorial(n) que torna el factorial de n sense usar cap for.

Exemples

```
>>> import functools
>>> def factorial(n):
...     return  functools.reduce(lambda x,y: x*y,
...                                 range(1,n+1))
...
>>> factorial(3)
6
>>> factorial(25)
15511210043330985984000000
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Funcions predefinides:

```
>>> for e in zip(range(5),['a','b','c','d','e'],
...                       ['A','B','C','D','E']):
...     print(e)
...
(0, 'a', 'A') (1, 'b', 'B') (2, 'c', 'C') (3, 'd', 'D')
(4, 'e', 'E')
```

Expressions

funcions predefinides

filter

map

reduce

any

all

Exemples

itertools

Funcions predefinides:

```
>>> for e in zip(range(5),['a','b','c','d','e'],
...                       ['A','B','C','D','E']):
...     print(e)
...
(0, 'a', 'A') (1, 'b', 'B') (2, 'c', 'C') (3, 'd', 'D')
(4, 'e', 'E')
```

Funcions predefinides:

```
>>> for e in zip(range(5),['a','b','c','d','e'],
...                       ['A','B','C','D','E']):
...     print(e)
...
(0, 'a', 'A') (1, 'b', 'B') (2, 'c', 'C') (3, 'd', 'D')
(4, 'e', 'E')
```

Funcions predefinides:

```
>>> for e in zip(range(5),['a','b','c','d','e'],
...                       ['A','B','C','D','E']):
...     print(e)
...
(0, 'a', 'A') (1, 'b', 'B') (2, 'c', 'C') (3, 'd', 'D')
(4, 'e', 'E')
```

Funcions predefinides:

```
>>> for e in zip(range(5),['a','b','c','d','e'],
...                       ['A','B','C','D','E']):
...     print(e)
...
(0, 'a', 'A') (1, 'b', 'B') (2, 'c', 'C') (3, 'd', 'D')
(4, 'e', 'E')
```

Itertools: permutacions entre elements

Expressions
funcions predefinides
itertools
permutacions
combinacions
accumulate
product
tee

```
>>> for l in itertools.permutations("ABC"):
...     print(l)
...
('A', 'B', 'C')
('A', 'C', 'B')
('B', 'A', 'C')
('B', 'C', 'A')
('C', 'A', 'B')
('C', 'B', 'A')
```

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

Itertools: permutacions entre elements

```
>>> for l in itertools.permutations("ABC"):
...     print(l)
...
('A', 'B', 'C')
('A', 'C', 'B')
('B', 'A', 'C')
('B', 'C', 'A')
('C', 'A', 'B')
('C', 'B', 'A')
```

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

Itertools: permutacions entre elements

```
>>> for l in itertools.permutations("ABC"):
...     print(l)
...
('A', 'B', 'C')
('A', 'C', 'B')
('B', 'A', 'C')
('B', 'C', 'A')
('C', 'A', 'B')
('C', 'B', 'A')
```

Itertools: permutacions entre elements

Expressions
funcions predefinides
itertools
permutacions
combinacions
accumulate
product
tee

```
>>> for l in itertools.permutations("ABC"):
...     print(l)
...
('A', 'B', 'C')
('A', 'C', 'B')
('B', 'A', 'C')
('B', 'C', 'A')
('C', 'A', 'B')
('C', 'B', 'A')
```

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

Itertools: Combinacions entre elements

```
>>> lst = [1, 2, 3]
>>> [list(x) for x in itertools.combinations(lst, 1)]
  [[1], [2], [3]]
>>> [list(x) for x in itertools.combinations(lst, 2)]
[[1, 2], [1, 3], [2, 3]]
>>> [list(x) for x in itertools.combinations(lst, 3)]
[[1, 2, 3]]
```

Expressions
funcions predefinides
itertools
permutacions
combinacions
accumulate
product
tee

Itertools: Combinacions entre elements

```
>>> lst = [1, 2, 3]
>>> [list(x) for x in itertools.combinations(lst, 1)]
  [[1], [2], [3]]
>>> [list(x) for x in itertools.combinations(lst, 2)]
[[1, 2], [1, 3], [2, 3]]
>>> [list(x) for x in itertools.combinations(lst, 3)]
[[1, 2, 3]]
```

Expressions
funcions predefinides
itertools
permutacions
combinacions
accumulate
product
tee

Itertools: Combinacions entre elements

```
>>> lst = [1, 2, 3]
>>> [list(x) for x in itertools.combinations(lst, 1)]
  [[1], [2], [3]]
>>> [list(x) for x in itertools.combinations(lst, 2)]
[[1, 2], [1, 3], [2, 3]]
>>> [list(x) for x in itertools.combinations(lst, 3)]
[[1, 2, 3]]
```

Expressions
funcions predefinides
itertools
permutacions
combinacions
accumulate
product
tee

Itertools: Combinacions entre elements

```
>>> lst = [1, 2, 3]
>>> [list(x) for x in itertools.combinations(lst, 1)]
  [[1], [2], [3]]
>>> [list(x) for x in itertools.combinations(lst, 2)]
[[1, 2], [1, 3], [2, 3]]
>>> [list(x) for x in itertools.combinations(lst, 3)]
[[1, 2, 3]]
```

Expressions
funcions predefinides
itertools
permutacions
combinacions
accumulate
product
tee

Itertools: Combinacions entre elements

```
>>> lst = [1, 2, 3]
>>> [list(x) for x in itertools.combinations(lst, 1)]
  [[1], [2], [3]]
>>> [list(x) for x in itertools.combinations(lst, 2)]
[[1, 2], [1, 3], [2, 3]]
>>> [list(x) for x in itertools.combinations(lst, 3)]
 [[1, 2, 3]]
```

Expressions
funcions predefinides
itertools
permutacions
combinacions
accumulate
product
tee

Itertools: Combinacions entre elements

```
>>> lst = [1, 2, 3]
>>> [list(x) for x in itertools.combinations(lst, 1)]
  [[1], [2], [3]]
>>> [list(x) for x in itertools.combinations(lst, 2)]
[[1, 2], [1, 3], [2, 3]]
>>> [list(x) for x in itertools.combinations(lst, 3)]
 [[1, 2, 3]]
```

Expressions
funcions predefinides
itertools
permutacions
combinacions
accumulate
product
tee

Itertools: Combinacions entre elements

```
>>> lst = [1, 2, 3]
>>> [list(x) for x in itertools.combinations(lst, 1)]
  [[1], [2], [3]]
>>> [list(x) for x in itertools.combinations(lst, 2)]
[[1, 2], [1, 3], [2, 3]]
>>> [list(x) for x in itertools.combinations(lst, 3)]
[[1, 2, 3]]
```

Expressions
funcions predefinides
itertools
permutacions
combinacions
accumulate
product
tee

Itertools: Combinacions entre elements

```
>>> lst = [1, 2, 3]
>>> [list(x) for x in itertools.combinations(lst, 1)]
  [[1], [2], [3]]
>>> [list(x) for x in itertools.combinations(lst, 2)]
[[1, 2], [1, 3], [2, 3]]
>>> [list(x) for x in itertools.combinations(lst, 3)]
[[1, 2, 3]]
```

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

itertools accumulate

```
>>> import itertools
>>> import operator
>>> list(itertools.accumulate([1,2,3,4], operator.add))
[1, 3, 6, 10]
>>> list(itertools.accumulate([1,2,3,4], operator.mul))
[1, 2, 6, 24]
>>> import functools
>>> functools.reduce(operator.add, [1,2,3,4])
10
>>> functools.reduce(operator.mul, [1,2,3,4])
24
```

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

itertools accumulate

```
>>> import itertools
>>> import operator
>>> list(itertools.accumulate([1,2,3,4], operator.add))
[1, 3, 6, 10]
>>> list(itertools.accumulate([1,2,3,4], operator.mul))
[1, 2, 6, 24]
>>> import functools
>>> functools.reduce(operator.add, [1,2,3,4])
10
>>> functools.reduce(operator.mul, [1,2,3,4])
24
```

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

itertools accumulate

```
>>> import itertools
>>> import operator
>>> list(itertools.accumulate([1,2,3,4], operator.add))
[1, 3, 6, 10]
>>> list(itertools.accumulate([1,2,3,4], operator.mul))
[1, 2, 6, 24]
>>> import functools
>>> functools.reduce(operator.add, [1,2,3,4])
10
>>> functools.reduce(operator.mul, [1,2,3,4])
24
```

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

itertools accumulate

```
>>> import itertools
>>> import operator
>>> list(itertools.accumulate([1,2,3,4], operator.add))
[1, 3, 6, 10]
>>> list(itertools.accumulate([1,2,3,4], operator.mul))
[1, 2, 6, 24]
>>> import functools
>>> functools.reduce(operator.add, [1,2,3,4])
10
>>> functools.reduce(operator.mul, [1,2,3,4])
24
```

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

itertools accumulate

```
>>> import itertools
>>> import operator
>>> list(itertools.accumulate([1,2,3,4], operator.add))
[1, 3, 6, 10]
>>> list(itertools.accumulate([1,2,3,4], operator.mul))
[1, 2, 6, 24]
>>> import functools
>>> functools.reduce(operator.add, [1,2,3,4])
10
>>> functools.reduce(operator.mul, [1,2,3,4])
24
```

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

itertools accumulate

```
>>> import itertools
>>> import operator
>>> list(itertools.accumulate([1,2,3,4], operator.add))
[1, 3, 6, 10]
>>> list(itertools.accumulate([1,2,3,4], operator.mul))
[1, 2, 6, 24]
>>> import functools
>>> functools.reduce(operator.add, [1,2,3,4])
10
>>> functools.reduce(operator.mul, [1,2,3,4])
24
```

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

itertools accumulate

```
>>> import itertools
>>> import operator
>>> list(itertools.accumulate([1,2,3,4], operator.add))
[1, 3, 6, 10]
>>> list(itertools.accumulate([1,2,3,4], operator.mul))
[1, 2, 6, 24]
>>> import functools
>>> functools.reduce(operator.add, [1,2,3,4])
10
>>> functools.reduce(operator.mul, [1,2,3,4])
24
```

Expressions
funcions predefinides
itertools
permutacions
combinacions
accumulate
product
tee

itertools accumulate

```
>>> import itertools
>>> import operator
>>> list(itertools.accumulate([1,2,3,4], operator.add))
[1, 3, 6, 10]
>>> list(itertools.accumulate([1,2,3,4], operator.mul))
[1, 2, 6, 24]
>>> import functools
>>> functools.reduce(operator.add, [1,2,3,4])
10
>>> functools.reduce(operator.mul, [1,2,3,4])
24
```

Expressions
funcions predefinides
itertools
permutacions
combinacions
accumulate
product
tee

itertools accumulate

```
>>> import itertools
>>> import operator
>>> list(itertools.accumulate([1,2,3,4], operator.add))
[1, 3, 6, 10]
>>> list(itertools.accumulate([1,2,3,4], operator.mul))
[1, 2, 6, 24]
>>> import functools
>>> functools.reduce(operator.add, [1,2,3,4])
10
>>> functools.reduce(operator.mul, [1,2,3,4])
24
```

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

Exemple de producte cartesià d'una llista amb sí mateixa

```
>>> import itertools
>>> it=itertools.product([1,2,3],repeat=2)
>>> for i,n in enumerate(it):
...     if i % 3 ==0: print()
...     print(n,end='')
...
<BLANKLINE>
(1, 1)(1, 2)(1, 3)
(2, 1)(2, 2)(2, 3)
(3, 1)(3, 2)(3, 3)
>>>
```

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

Exemple de producte cartesià d'una llista amb sí mateixa

```
>>> import itertools
>>> it=itertools.product([1,2,3],repeat=2)
>>> for i,n in enumerate(it):
...     if i % 3 ==0: print()
...     print(n,end='')
...
<BLANKLINE>
(1, 1)(1, 2)(1, 3)
(2, 1)(2, 2)(2, 3)
(3, 1)(3, 2)(3, 3)
>>>
```

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

Exemple de producte cartesià d'una llista amb sí mateixa

```
>>> import itertools
>>> it=itertools.product([1,2,3],repeat=2)
>>> for i,n in enumerate(it):
...     if i % 3 ==0: print()
...     print(n,end='')
...
<BLANKLINE>
(1, 1)(1, 2)(1, 3)
(2, 1)(2, 2)(2, 3)
(3, 1)(3, 2)(3, 3)
>>>
```

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

Exemple de producte cartesià d'una llista amb sí mateixa

```
>>> import itertools
>>> it=itertools.product([1,2,3],repeat=2)
>>> for i,n in enumerate(it):
...     if i % 3 ==0: print()
...     print(n,end='')
...
<BLANKLINE>
(1, 1)(1, 2)(1, 3)
(2, 1)(2, 2)(2, 3)
(3, 1)(3, 2)(3, 3)
>>>
```

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

Exemple de producte cartesià d'una llista amb sí mateixa

```
>>> import itertools
>>> it=itertools.product([1,2,3],repeat=2)
>>> for i,n in enumerate(it):
...     if i % 3 ==0: print()
...     print(n,end='')
...
<BLANKLINE>
(1, 1)(1, 2)(1, 3)
(2, 1)(2, 2)(2, 3)
(3, 1)(3, 2)(3, 3)
>>>
```

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

Exemple de producte cartesià d'una llista amb sí mateixa

```
>>> import itertools
>>> it=itertools.product([1,2,3],repeat=2)
>>> for i,n in enumerate(it):
...     if i % 3 ==0: print()
...     print(n,end='')
...
<BLANKLINE>
(1, 1)(1, 2)(1, 3)
(2, 1)(2, 2)(2, 3)
(3, 1)(3, 2)(3, 3)
>>>
```

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

Sense el `itertools.product` obtindrem un resultat semblant fent ...

product [1,2,3]*[1,2,3]

```
>>> lst = [1,2,3]
>>> for i in lst:
...     for j in lst:
...         print('({},{})'.format(i,j), end='')
...     print()
...
(1,1)(1,2)(1,3)
(2,1)(2,2)(2,3)
(3,1)(3,2)(3,3)
```

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

Sense el `itertools.product` obtindrem un resultat semblant fent ...

product [1,2,3]*[1,2,3]

```
>>> lst = [1,2,3]
>>> for i in lst:
...     for j in lst:
...         print('({},{})'.format(i,j), end='')
...     print()
...
(1,1)(1,2)(1,3)
(2,1)(2,2)(2,3)
(3,1)(3,2)(3,3)
```

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

Sense el `itertools.product` obtindrem un resultat semblant fent ...

product [1,2,3]*[1,2,3]

```
>>> lst = [1,2,3]
>>> for i in lst:
...     for j in lst:
...         print('({},{})'.format(i,j), end='')
...     print()
...
(1,1)(1,2)(1,3)
(2,1)(2,2)(2,3)
(3,1)(3,2)(3,3)
```

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

Sense el `itertools.product` obtindrem un resultat semblant fent ...

product [1,2,3]*[1,2,3]

```
>>> lst = [1,2,3]
>>> for i in lst:
...     for j in lst:
...         print('({},{})'.format(i,j), end='')
...     print()
...
(1,1)(1,2)(1,3)
(2,1)(2,2)(2,3)
(3,1)(3,2)(3,3)
```

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

... amb iteradors

```
>>> it = iter([1,2,3])
>>> for i in it:
...     for j in it:
...         print('({},{})'.format(i,j), end='')
...     print()
...
(1,2)(1,3)
>>>
```

L'iterador s'ha esgotat.

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

... amb iteradors

```
>>> it = iter([1,2,3])
>>> for i in it:
...     for j in it:
...         print('({},{})'.format(i,j), end='')
...     print()
...
(1,2)(1,3)
>>>
```

L'iterador s'ha esgotat.

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

... amb iteradors

```
>>> it = iter([1,2,3])
>>> for i in it:
...     for j in it:
...         print('({},{})'.format(i,j), end='')
...     print()
...
(1,2)(1,3)
>>>
```

L'iterador s'ha esgotat.

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

... amb iteradors

```
>>> it = iter([1,2,3])
>>> for i in it:
...     for j in it:
...         print('({},{})'.format(i,j), end='')
...     print()
...
(1,2)(1,3)
>>>
```

L'iterador s'ha esgotat.

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

... amb iteradors

```
>>> it = iter([1,2,3])
>>> for i in it:
...     for j in it:
...         print('({},{})'.format(i,j), end='')
...     print()
...
(1,2)(1,3)
>>>
```

L'iterador s'ha esgotat.

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

tee

```
>>> import itertools
>>> iteradors = itertools.tee(iter([1,2,3]),2)
>>> print(iteradors)
(<itertools._tee object at ...>, <itertools._tee object at
..>)
>>> for i in iteradors[0]:
...     for j in iteradors[1]:
...         print('({},{})'.format(i,j), end='')
...     print()
...
(1,1)(1,2)(1,3)
<BLANKLINE>
<BLANKLINE>
>>>
```

L'iterador `iteradors[1]` s'ha esgotat en la primera línia. Per això les altres dues línies estan buides.

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

tee

```
>>> import itertools
>>> iteradors = itertools.tee(iter([1,2,3]),2)
>>> print(iteradors)
(<itertools._tee object at ...>, <itertools._tee object at
..>)
>>> for i in iteradors[0]:
...     for j in iteradors[1]:
...         print('({},{})'.format(i,j), end='')
...     print()
...
(1,1)(1,2)(1,3)
<BLANKLINE>
<BLANKLINE>
>>>
```

L'iterador `iteradors[1]` s'ha esgotat en la primera línia. Per això les altres dues línies estan buides.

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

tee

```
>>> import itertools
>>> iteradors = itertools.tee(iter([1,2,3]),2)
>>> print(iteradors)
(<itertools._tee object at ...>, <itertools._tee object at
..>)
>>> for i in iteradors[0]:
...     for j in iteradors[1]:
...         print('({},{})'.format(i,j), end='')
...     print()
...
(1,1)(1,2)(1,3)
<BLANKLINE>
<BLANKLINE>
>>>
```

L'iterador `iteradors[1]` s'ha esgotat en la primera línia. Per això les altres dues línies estan buides.

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

tee

```
>>> import itertools
>>> iteradors = itertools.tee(iter([1,2,3]),2)
>>> print(iteradors)
(<itertools._tee object at ...>, <itertools._tee object at
..>)
>>> for i in iteradors[0]:
...     for j in iteradors[1]:
...         print('({},{})'.format(i,j), end='')
...     print()
...
(1,1)(1,2)(1,3)
<BLANKLINE>
<BLANKLINE>
>>>
```

L'iterador `iteradors[1]` s'ha esgotat en la primera línia. Per això les altres dues línies estan buides.

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

tee

```
>>> import itertools
>>> iteradors = itertools.tee(iter([1,2,3]),2)
>>> print(iteradors)
(<itertools._tee object at ...>, <itertools._tee object at
..>)
>>> for i in iteradors[0]:
...     for j in iteradors[1]:
...         print('({},{})'.format(i,j), end='')
...     print()
...
(1,1)(1,2)(1,3)
<BLANKLINE>
<BLANKLINE>
>>>
```

L'iterador `iteradors[1]` s'ha esgotat en la primera línia. Per això les altres dues línies estan buides.

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

tee

```
>>> import itertools
>>> iteradors = itertools.tee(iter([1,2,3]),2)
>>> print(iteradors)
(<itertools._tee object at ...>, <itertools._tee object at
..>)
>>> for i in iteradors[0]:
...     for j in iteradors[1]:
...         print('({},{})'.format(i,j), end='')
...     print()
...
(1,1)(1,2)(1,3)
<BLANKLINE>
<BLANKLINE>
>>>
```

L'iterador `iteradors[1]` s'ha esgotat en la primera línia. Per això les altres dues línies estan buides.

Expressions
funcions predefinides
itertools
permutacions
combinacions
accumulate
product
tee

tee

```
>>> import itertools
>>> iteradors = itertools.tee(iter([1,2,3]),2)
>>> print(iteradors)
(<itertools._tee object at ...>, <itertools._tee object at
..>)
>>> for i in iteradors[0]:
...     for j in iteradors[1]:
...         print('({},{})'.format(i,j), end='')
...     print()
...
(1,1)(1,2)(1,3)
<BLANKLINE>
<BLANKLINE>
>>>
```

L'iterador `iteradors[1]` s'ha esgotat en la primera línia. Per això les altres dues línies estan buides.

Posant-hi remei ...

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

```
>>> it = iter([1,2,3])
>>> iteradors = itertools.tee(it,4)
>>> print(iteradors)
(<itertools._tee object at ...>, <itertools._tee object at
..>, <itertools._tee object at ...>, <itertools._tee objec
at ...>)
>>> its = iter(iteradors)
>>> for i in next(its):
...     for j in next(its):
...         print('({},{})'.format(i,j), end='')
...     print()
...
(1,1)(1,2)(1,3)
(2,1)(2,2)(2,3)
(3,1)(3,2)(3,3)
>>>
```

On iterem sobre un tuple d'iteradors.

Posant-hi remei ...

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

```
>>> it = iter([1,2,3])
>>> iteradors = itertools.tee(it,4)
>>> print(iteradors)
(<itertools._tee object at ...>, <itertools._tee object at
..>, <itertools._tee object at ...>, <itertools._tee objec
at ...>)
>>> its = iter(iteradors)
>>> for i in next(its):
...     for j in next(its):
...         print('({},{})'.format(i,j), end='')
...     print()
...
(1,1)(1,2)(1,3)
(2,1)(2,2)(2,3)
(3,1)(3,2)(3,3)
>>>
```

On iterem sobre un tuple d'iteradors.

Posant-hi remei ...

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

```
>>> it = iter([1,2,3])
>>> iteradors = itertools.tee(it,4)
>>> print(iteradors)
(<itertools._tee object at ...>, <itertools._tee object at
..>, <itertools._tee object at ...>, <itertools._tee objec
at ...>)
>>> its = iter(iteradors)
>>> for i in next(its):
...     for j in next(its):
...         print('({},{})'.format(i,j), end='')
...     print()
...
(1,1)(1,2)(1,3)
(2,1)(2,2)(2,3)
(3,1)(3,2)(3,3)
>>>
```

On iterem sobre un tuple d'iteradors.

Posant-hi remei ...

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

```
>>> it = iter([1,2,3])
>>> iteradors = itertools.tee(it,4)
>>> print(iteradors)
(<itertools._tee object at ...>, <itertools._tee object at
..>, <itertools._tee object at ...>, <itertools._tee objec
at ...>)
>>> its = iter(iteradors)
>>> for i in next(its):
...     for j in next(its):
...         print('({},{})'.format(i,j), end='')
...     print()
...
(1,1)(1,2)(1,3)
(2,1)(2,2)(2,3)
(3,1)(3,2)(3,3)
>>>
```

On iterem sobre un tuple d'iteradors.

Posant-hi remei ...

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

```
>>> it = iter([1,2,3])
>>> iteradors = itertools.tee(it,4)
>>> print(iteradors)
(<itertools._tee object at ...>, <itertools._tee object at
..>, <itertools._tee object at ...>, <itertools._tee objec
at ...>)
>>> its = iter(iteradors)
>>> for i in next(its):
...     for j in next(its):
...         print('({},{})'.format(i,j), end='')
...     print()
...
(1,1)(1,2)(1,3)
(2,1)(2,2)(2,3)
(3,1)(3,2)(3,3)
>>>
```

On iterem sobre un tuple d'iteradors.

Posant-hi remei ...

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

```
>>> it = iter([1,2,3])
>>> iteradors = itertools.tee(it,4)
>>> print(iteradors)
(<itertools._tee object at ...>, <itertools._tee object at
..>, <itertools._tee object at ...>, <itertools._tee objec
at ...>)
>>> its = iter(iteradors)
>>> for i in next(its):
...     for j in next(its):
...         print('({},{})'.format(i,j), end='')
...     print()
...
(1,1)(1,2)(1,3)
(2,1)(2,2)(2,3)
(3,1)(3,2)(3,3)
>>>
```

On iterem sobre un tuple d'iteradors.

Posant-hi remei ...

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

```
>>> it = iter([1,2,3])
>>> iteradors = itertools.tee(it,4)
>>> print(iteradors)
(<itertools._tee object at ...>, <itertools._tee object at
..>, <itertools._tee object at ...>, <itertools._tee objec
at ...>)
>>> its = iter(iteradors)
>>> for i in next(its):
...     for j in next(its):
...         print('({},{})'.format(i,j), end='')
...     print()
...
(1,1)(1,2)(1,3)
(2,1)(2,2)(2,3)
(3,1)(3,2)(3,3)
>>>
```

On iterem sobre un tuple d'iteradors.

Expressions

funcions predefinides

itertools

permutacions

combinacions

accumulate

product

tee

Posant-hi remei ...

```
>>> it = iter([1,2,3])
>>> iteradors = itertools.tee(it,4)
>>> print(iteradors)
(<itertools._tee object at ...>, <itertools._tee object at
..>, <itertools._tee object at ...>, <itertools._tee objec
at ...>)
>>> its = iter(iteradors)
>>> for i in next(its):
...     for j in next(its):
...         print('({},{})'.format(i,j), end='')
...     print()
...
(1,1)(1,2)(1,3)
(2,1)(2,2)(2,3)
(3,1)(3,2)(3,3)
>>>
```

On iterem sobre un tuple d'iteradors.