Introducció:
for

iter() i **next()**

Generadors

range

in

Expressions

Expressions

Introducció

ETSEIB/GIE

8 de setembre de 2023

Introducció: **for**

iter() i **next()**

Generadors

range

in

Expressions

Expressions

1. Introducció: **for**
2. **iter()** i **next()**
3. Generadors
   - Exemples
4. **range**
5. **in**
6. Expressions
7. Expressions

Introducció: for
iter() i next()
Generadors
range
in
Expressions
Expressions

Introducció

- Sovint escrivim iteracions amb la composició de **for**. Avui coneixerem més a fons aquesta composició.
- Per exemple, **for** itera amb llistes:
- ```
  >>> for element in [1, 2, 3]:
  ...     print(element)
  ...
  1
  2
  3
  ```
  ```

Introducció: for
iter() i next()
Generadors
range
in
Expressions
Expressions

Introducció

- Sovint escrivim iteracions amb la composició de **for**. Avui coneixerem més a fons aquesta composició.
- Per exemple, **for** itera amb tuples:
- ```
  >>> for element in (1, 2, 3):
  ...     print(element)
  ...
  ```
  ```

Introducció

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

Introducció

- Sovint escrivim iteracions amb la composició de **for**. Avui coneixerem més a fons aquesta composició.
- Per exemple, **for** itera amb tuples:
- ```
  >>> for element in (1, 2, 3):
  ...     print(element)
  ...
  1
  2
  3
  ```
  ```

Introducció: for
iter() i next()
Generadors
range
in
Expressions
Expressions

Introducció

- Sovint escrivim iteracions amb la composició de **for**. Avui coneixerem més a fons aquesta composició.
- Per exemple, **for** itera amb diccionaris:
- ```
  >>> for clau in {'un': 1, 'dos': 2, 'tres': 3}:
  ...     print(clau)
  ...
  ```
  ```

Introducció: for
iter() i next()
Generadors
range
in
Expressions
Expressions

Introducció

- Sovint escrivim iteracions amb la composició de **for**. Avui coneixerem més a fons aquesta composició.
- Per exemple, **for** itera amb diccionaris:
- ```
  >>> for clau in {'un': 1, 'dos': 2, 'tres': 3}:
  ...     print(clau)
  ...
  dos
  tres
  un
  ```
  ```

Introducció: for
iter() i next()
Generadors
range
in
Expressions
Expressions

Introducció

- Sovint escrivim iteracions amb la composició de **for**. Avui coneixerem més a fons aquesta composició.
- Per exemple, **for** itera amb strings:
- ```
  >>> for caracter in '123':
  ...     print(caracter)
  ...
  ```
  ```

Introducció: for
iter() i next()
Generadors
range
in
Expressions
Expressions

Introducció

- Sovint escrivim iteracions amb la composició de **for**. Avui coneixerem més a fons aquesta composició.
- Per exemple, **for** itera amb strings:
- ```
  >>> for caracter in '123':
  ...     print(caracter)
  ...
  1
  2
  3
  ```
  ```

Introducció: **for**
iter() i next()
Generadors
range
in
Expressions
Expressions

Introducció

- Sovint escrivim iteracions amb la composició de **for**. Avui coneixerem més a fons aquesta composició.
- Per exemple, **for** itera amb fitxers:
- ```
  >>> for linia in open("arxiu"):
  ...     print(linia)
  ...
  ```
  ```

Introducció: for
iter() i next()
Generadors
range
in
Expressions
Expressions

Introducció

- En tots els exemples que acabem de veure, la iteració amb `for` es fa sobre un objecte *contenidor* com són les llistes, tuples, diccionaris, strings, etc.
- Els objectes *contenidor* tenen la particularitat de que se'ls pot aplicar l'operador **in**. (*Duck typing*)

```
>>> 'Maria' in ['Anna', 'Gloria', 'Maria', 'Gemma']
True
```

- També podem posar després d'**in** una funció com la funció predefinida `range`.

```
>>> for i in range(3):
...     print(i)
...
0
1
2
```

Introducció: for
iter() i next()
Generadors
range
in
Expressions
Expressions

Introducció

- En tots els exemples que acabem de veure, la iteració amb `for` es fa sobre un objecte *contenidor* com són les llistes, tuples, diccionaris, strings, etc.
- Els objectes *contenidor* tenen la particularitat de que se'ls pot aplicar l'operador **in**. (*Duck typing*)

```
>>> 'Maria' in ['Anna', 'Gloria', 'Maria', 'Gemma']
True
```

- També podem posar després d'**in** una funció com la funció predefinida `range`.

```
>>> for i in range(3):
...     print(i)
...
0
1
2
```

Introducció: for
iter() i next()
Generadors
range
in
Expressions
Expressions

Introducció

- En tots els exemples que acabem de veure, la iteració amb `for` es fa sobre un objecte *contenidor* com són les llistes, tuples, diccionaris, strings, etc.
- Els objectes *contenidor* tenen la particularitat de que se'ls pot aplicar l'operador **in**. (*Duck typing*)

```
>>> 'Maria' in ['Anna', 'Gloria', 'Maria', 'Gemma']
True
```

- També podem posar després d'**in** una funció com la funció predefinida `range`.

```
>>> for i in range(3):
...     print(i)
...
0
1
2
```

Introducció: for
iter() i next()
Generadors
range
in
Expressions
Expressions

Què passa si posem un `int` després de l'`in`?

```
>>> for d in 123456789:
...     print(d)
Traceback (most recent call last):
  File "/usr/lib/python3.9/doctest.py", line 1336,
 in __run
    exec(compile(example.source, filename, "single
",
  File ≪doctest noiterador.txt[0]>", line 1, in <
module>
    for d in 123456789:
TypeError: 'int' object is not iterable
```

Introducció: for
iter() i next()
Generadors
range
in
Expressions
Expressions

Què passa si posem un `int` després de l'`in`?

```
>>> for d in 123456789:
...     print(d)
Traceback (most recent call last):
  File "/usr/lib/python3.9/doctest.py", line 1336,
 in __run
    exec(compile(example.source, filename, "single
",
  File ≪doctest noiterador.txt[0]>", line 1, in <
module>
    for d in 123456789:
TypeError: 'int' object is not iterable
```

Introducció: for
iter() i next()
Generadors
range
in
Expressions
Expressions

Què passa si posem un `int` després de l'`in`?

```
>>> for d in 123456789:
...     print(d)
Traceback (most recent call last):
  File "/usr/lib/python3.9/doctest.py", line 1336,
 in __run
    exec(compile(example.source, filename, "single
",
  File ≪doctest noiterador.txt[0]>", line 1, in <
module>
    for d in 123456789:
TypeError: 'int' object is not iterable
```

Introducció: **for**
iter() i next()
Generadors
range
in
Expressions
Expressions

Introducció

- Després de l'operador **in** del `for` s'espera trobar una expressió que doni com a resultat un objecte *iterable*.
- La idea d'*iterable* indica que hi ha la possibilitat d'iterar sobre l'objecte *iterable*. (Dit d'una altra manera, podem recòrrer l'objecte aplicant un `for`).
- En concret, si l'objecte `Iterable` és *iterable* es té que
  - La crida de la funció predefinida *iter* `iter(Iterable)` retorna un objecte *iterador* d'`Iterable`. A partir de l'objecte *iterador* realitzarem la iteració d'`Iterable`.
  - l'objecte té el mètode `__iter__()` o el mètode `__getitem__()`.

Introducció: for
iter() i next()
Generadors
range
in
Expressions
Expressions

Introducció

- Després de l'operador **in** del `for` s'espera trobar una expressió que doni com a resultat un objecte *iterable*.
- La idea d'*iterable* indica que hi ha la possibilitat d'iterar sobre l'objecte *iterable*. (Dit d'una altra manera, podem recòrrer l'objecte aplicant un `for`).
- En concret, si l'objecte `Iterable` és *iterable* es té que
  - La crida de la funció predefinida *iter* `iter(Iterable)` retorna un objecte *iterador* d'`Iterable`. A partir de l'objecte *iterador* realitzarem la iteració d'`Iterable`.
  - l'objecte té el mètode `__iter__()` o el mètode `__getitem__()`.

Introducció: for
iter() i next()
Generadors
range
in
Expressions
Expressions

Introducció

- Després de l'operador **in** del `for` s'espera trobar una expressió que doni com a resultat un objecte *iterable*.
- La idea d'*iterable* indica que hi ha la possibilitat d'iterar sobre l'objecte *iterable*. (Dit d'una altra manera, podem recòrrer l'objecte aplicant un `for`).
- En concret, si l'objecte `Iterable` és *iterable* es té que
  - La crida de la funció predefinida *iter* `iter(Iterable)` retorna un objecte *iterador* d'`Iterable`. A partir de l'objecte *iterador* realitzarem la iteració d'`Iterable`.
  - l'objecte té el mètode `__iter__()` o el mètode `__getitem__()`.

Introducció: for
iter() i next()
Generadors
range
in
Expressions
Expressions

Introducció

- Després de l'operador **in** del `for` s'espera trobar una expressió que doni com a resultat un objecte *iterable*.
- La idea d'*iterable* indica que hi ha la possibilitat d'iterar sobre l'objecte *iterable*. (Dit d'una altra manera, podem recòrrer l'objecte aplicant un `for`).
- En concret, si l'objecte `Iterable` és *iterable* es té que
  - La crida de la funció predefinida *iter* `iter(Iterable)` retorna un objecte *iterador* d'`Iterable`. A partir de l'objecte *iterador* realitzarem la iteració d'`Iterable`.
  - l'objecte té el mètode `__iter__()` o el mètode `__getitem__()`.

Introducció: for
iter() i next()
Generadors
range
in
Expressions
Expressions

Introducció

- Després de l'operador **in** del `for` s'espera trobar una expressió que doni com a resultat un objecte *iterable*.
- La idea d'*iterable* indica que hi ha la possibilitat d'iterar sobre l'objecte *iterable*. (Dit d'una altra manera, podem recòrrer l'objecte aplicant un `for`).
- En concret, si l'objecte `Iterable` és *iterable* es té que
  - La crida de la funció predefinida *iter* `iter(Iterable)` retorna un objecte *iterador* d'`Iterable`. A partir de l'objecte *iterador* realitzarem la iteració d'`Iterable`.
  - l'objecte té el mètode `__iter__()` o el mètode `__getitem__()`.

Introducció: for
iter() i next()
Generadors
range
in
Expressions
Expressions

Introducció

- Un *iterador* és el que fa possible que anem visitant cadascun dels elements de l'objecte a iterar.
- Un *iterador* és un objecte que pot ser argument de la funció predefinida `next()`. La funció `next()` és:
  - retorna el primer valor a tractar el primer cop que es crida.
  - retorna el següent valor a tractar cada cop que es crida.
  - indica que s'ha acabat llençant l'error `StopIteration` quan no queden més elements a tractar.
- És iterable: Com argument de la funció predefinida `iter()` retorna a sí mateix.
- Un cop hem usat l'iterador el seu estat no pot revertir i ja no ens serveix. Caduca!

Introducció: for
iter() i next()
Generadors
range
in
Expressions
Expressions

Introducció

- Un *iterador* és el que fa possible que anem visitant cadascun dels elements de l'objecte a iterar.
- Un *iterador* és un objecte que pot ser argument de la funció predefinida `next()`. La funció `next()` és:
  - retorna el primer valor a tractar el primer cop que es crida.
  - retorna el següent valor a tractar cada cop que es crida.
  - indica que s'ha acabat llençant l'error `StopIteration` quan no queden més elements a tractar.
- És iterable: Com argument de la funció predefinida `iter()` retorna a sí mateix.
- Un cop hem usat l'iterador el seu estat no pot revertir i ja no ens serveix. Caduca!

Introducció: for
iter() i next()
Generadors
range
in
Expressions
Expressions

Introducció

- Un *iterador* és el que fa possible que anem visitant cadascun dels elements de l'objecte a iterar.
- Un *iterador* és un objecte que pot ser argument de la funció predefinida `next()`. La funció `next()` és:
  - retorna el primer valor a tractar el primer cop que es crida.
  - retorna el següent valor a tractar cada cop que es crida.
  - indica que s'ha acabat llençant l'error `StopIteration` quan no queden més elements a tractar.
- És iterable: Com argument de la funció predefinida `iter()` retorna a sí mateix.
- Un cop hem usat l'iterador el seu estat no pot revertir i ja no ens serveix. Caduca!

Introducció: for
iter() i next()
Generadors
range
in
Expressions
Expressions

Introducció

- Un *iterador* és el que fa possible que anem visitant cadascun dels elements de l'objecte a iterar.
- Un *iterador* és un objecte que pot ser argument de la funció predefinida `next()`. La funció `next()` és:
  - retorna el primer valor a tractar el primer cop que es crida.
  - retorna el següent valor a tractar cada cop que es crida.
  - indica que s'ha acabat llençant l'error `StopIteration` quan no queden més elements a tractar.
- És iterable: Com argument de la funció predefinida `iter()` retorna a sí mateix.
- Un cop hem usat l'iterador el seu estat no pot revertir i ja no ens serveix. Caduca!

Introducció: for
iter() i next()
Generadors
range
in
Expressions
Expressions

Introducció

- Un *iterador* és el que fa possible que anem visitant cadascun dels elements de l'objecte a iterar.
- Un *iterador* és un objecte que pot ser argument de la funció predefinida `next()`. La funció `next()` és:
  - retorna el primer valor a tractar el primer cop que es crida.
  - retorna el següent valor a tractar cada cop que es crida.
  - indica que s'ha acabat llençant l'error `StopIteration` quan no queden més elements a tractar.
- És iterable: Com argument de la funció predefinida `iter()` retorna a sí mateix.
- Un cop hem usat l'iterador el seu estat no pot revertir i ja no ens serveix. Caduca!

Introducció: for
iter() i next()
Generadors
range
in
Expressions
Expressions

Introducció

- Un *iterador* és el que fa possible que anem visitant cadascun dels elements de l'objecte a iterar.
- Un *iterador* és un objecte que pot ser argument de la funció predefinida `next()`. La funció `next()` és:
  - retorna el primer valor a tractar el primer cop que es crida.
  - retorna el següent valor a tractar cada cop que es crida.
  - indica que s'ha acabat llençant l'error `StopIteration` quan no queden més elements a tractar.
- És iterable: Com argument de la funció predefinida `iter()` retorna a sí mateix.
- Un cop hem usat l'iterador el seu estat no pot revertir i ja no ens serveix. Caduca!

Introducció:
for
iter() i next()
Generadors
range
in
Expressions
Expressions

Introducció

- Un *iterador* és el que fa possible que anem visitant cadascun dels elements de l'objecte a iterar.
- Un *iterador* és un objecte que pot ser argument de la funció predefinida `next()`. La funció `next()` és:
  - retorna el primer valor a tractar el primer cop que es crida.
  - retorna el següent valor a tractar cada cop que es crida.
  - indica que s'ha acabat llençant l'error `StopIteration` quan no queden més elements a tractar.
- És iterable: Com argument de la funció predefinida `iter()` retorna a sí mateix.
- Un cop hem usat l'iterador el seu estat no pot revertir i ja no ens serveix. Caduca!

Introducció: for
iter() i next()
Generadors
range
in
Expressions
Expressions

Introducció

- Un *iterador* és el que fa possible que anem visitant cadascun dels elements de l'objecte a iterar.
- Un *iterador* és un objecte que pot ser argument de la funció predefinida `next()`. La funció `next()` és:
  - retorna el primer valor a tractar el primer cop que es crida.
  - retorna el següent valor a tractar cada cop que es crida.
  - indica que s'ha acabat llençant l'error `StopIteration` quan no queden més elements a tractar.
- És iterable: Com argument de la funció predefinida `iter()` retorna a sí mateix.
- Un cop hem usat l'iterador el seu estat no pot revertir i ja no ens serveix. Caduca!

Introducció:
for

iter() i next()

Generadors

range

in

Expressions

Expressions

```
>>> s = 'abc'
>>> it = iter(s)
>>> it
<iterator object at 0x00A1DB50>
>>> it is iter(it)
True
>>> next(it)
'a'
>>> next(it)
'b'
>>> next(it)
'c'
>>> next(it)

Traceback (most recent call last):
  File "<stdin>", line 1, in ?
    next(it)
StopIteration
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

```
>>> s = 'abc'
>>> it = iter(s)
>>> it
<iterator object at 0x00A1DB50>
>>> it is iter(it)
True
>>> next(it)
'a'
>>> next(it)
'b'
>>> next(it)
'c'
>>> next(it)

Traceback (most recent call last):
  File "<stdin>", line 1, in ?
    next(it)
StopIteration
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

```
>>> s = 'abc'
>>> it = iter(s)
>>> it
<iterator object at 0x00A1DB50>
>>> it is iter(it)
True
>>> next(it)
'a'
>>> next(it)
'b'
>>> next(it)
'c'
>>> next(it)

Traceback (most recent call last):
  File "<stdin>", line 1, in ?
    next(it)
StopIteration
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

```
>>> s = 'abc'
>>> it = iter(s)
>>> it
<iterator object at 0x00A1DB50>
>>> it is iter(it)
True
>>> next(it)
'a'
>>> next(it)
'b'
>>> next(it)
'c'
>>> next(it)

Traceback (most recent call last):
  File "<stdin>", line 1, in ?
    next(it)
StopIteration
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

```
>>> s = 'abc'
>>> it = iter(s)
>>> it
<iterator object at 0x00A1DB50>
>>> it is iter(it)
True
>>> next(it)
'a'
>>> next(it)
'b'
>>> next(it)
'c'
>>> next(it)

Traceback (most recent call last):
  File "<stdin>", line 1, in ?
    next(it)
StopIteration
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

```
>>> s = 'abc'
>>> it = iter(s)
>>> it
<iterator object at 0x00A1DB50>
>>> it is iter(it)
True
>>> next(it)
'a'
>>> next(it)
'b'
>>> next(it)
'c'
>>> next(it)

Traceback (most recent call last):
  File "<stdin>", line 1, in ?
    next(it)
StopIteration
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

```
>>> s = 'abc'
>>> it = iter(s)
>>> it
<iterator object at 0x00A1DB50>
>>> it is iter(it)
True
>>> next(it)
'a'
>>> next(it)
'b'
>>> next(it)
'c'
>>> next(it)

Traceback (most recent call last):
  File "<stdin>", line 1, in ?
    next(it)
StopIteration
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

```
>>> s = 'abc'
>>> it = iter(s)
>>> it
<iterator object at 0x00A1DB50>
>>> it is iter(it)
True
>>> next(it)
'a'
>>> next(it)
'b'
>>> next(it)
'c'
>>> next(it)

Traceback (most recent call last):
  File "<stdin>", line 1, in ?
    next(it)
StopIteration
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

```
>>> s = 'abc'
>>> it = iter(s)
>>> it
<iterator object at 0x00A1DB50>
>>> it is iter(it)
True
>>> next(it)
'a'
>>> next(it)
'b'
>>> next(it)
'c'
>>> next(it)

Traceback (most recent call last):
  File "<stdin>", line 1, in ?
    next(it)
StopIteration
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

```
>>> s = 'abc'
>>> it = iter(s)
>>> it
<iterator object at 0x00A1DB50>
>>> it is iter(it)
True
>>> next(it)
'a'
>>> next(it)
'b'
>>> next(it)
'c'
>>> next(it)

Traceback (most recent call last):
  File "<stdin>", line 1, in ?
    next(it)
StopIteration
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

```
>>> s = 'abc'
>>> it = iter(s)
>>> it
<iterator object at 0x00A1DB50>
>>> it is iter(it)
True
>>> next(it)
'a'
>>> next(it)
'b'
>>> next(it)
'c'
>>> next(it)

Traceback (most recent call last):
  File "<stdin>", line 1, in ?
    next(it)
StopIteration
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

```
>>> s = 'abc'
>>> it = iter(s)
>>> it
<iterator object at 0x00A1DB50>
>>> it is iter(it)
True
>>> next(it)
'a'
>>> next(it)
'b'
>>> next(it)
'c'
>>> next(it)

Traceback (most recent call last):
  File "<stdin>", line 1, in ?
    next(it)
StopIteration
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

```
>>> s = 'abc'
>>> it = iter(s)
>>> it
<iterator object at 0x00A1DB50>
>>> it is iter(it)
True
>>> next(it)
'a'
>>> next(it)
'b'
>>> next(it)
'c'
>>> next(it)

Traceback (most recent call last):
  File "<stdin>", line 1, in ?
    next(it)
StopIteration
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

```
for element in iterable:
    tractar(element)
```

és equivalent a:

```
iterador = iter(iterable)
while True:
    try:
        element = next(iterador)
        tractar(element)
    except (StopIteration, IndexError):
        break
```

Introducció:
for
iter() i next()
Generadors
Exemples
range
in
Expressions
Expressions

Generadors: yield

- Un generador o funció generadora és una funció especial que retorna un objecte iterador anomenat també generador.
- La funció generadora usa la instrucció `yield`. Sintàcticament, distinguirem una funció generadora de la resta per la presència del mot `yield`.
- Cada cop que invoquem `next` a un generador, executa el codi de la funció generadora fins que troba un `yield`. El valor de l'expressió que hi ha al yield és el valor que retornarà `next`.
- El següent `next` farà que la funció s'executi des de la instrucció següent al darrer `yield` executat fins el següent yield. Si no n'hi ha més `yield` a trobar, `next` llançarà un `StopIteration`
- La funció generadora no té cap `return` per retornar valor.

Introducció: for
iter() i next()
Generadors
Exemples
range
in
Expressions
Expressions

Generadors: yield

- Un generador o funció generadora és una funció especial que retorna un objecte iterador anomenat també generador.
- La funció generadora usa la instrucció `yield`. Sintàcticament, distinguirem una funció generadora de la resta per la presència del mot `yield`.
- Cada cop que invoquem `next` a un generador, executa el codi de la funció generadora fins que troba un `yield`. El valor de l'expressió que hi ha al yield és el valor que retornarà `next`.
- El següent `next` farà que la funció s'executi des de la instrucció següent al darrer `yield` executat fins el següent yield. Si no n'hi ha més `yield` a trobar, `next` llançarà un `StopIteration`
- La funció generadora no té cap `return` per retornar valor.

Introducció: for
iter() i next()
Generadors
Exemples
range
in
Expressions
Expressions

Generadors: yield

- Un generador o funció generadora és una funció especial que retorna un objecte iterador anomenat també generador.
- La funció generadora usa la instrucció `yield`. Sintàcticament, distinguirem una funció generadora de la resta per la presència del mot `yield`.
- Cada cop que invoquem `next` a un generador, executa el codi de la funció generadora fins que troba un `yield`. El valor de l'expressió que hi ha al yield és el valor que retornarà `next`.
- El següent `next` farà que la funció s'executi des de la instrucció següent al darrer `yield` executat fins el següent yield. Si no n'hi ha més `yield` a trobar, `next` llançarà un `StopIteration`
- La funció generadora no té cap `return` per retornar valor.

Introducció: for

iter() i next()

Generadors

Exemples

range

in

Expressions

Expressions

- Un generador o funció generadora és una funció especial que retorna un objecte iterador anomenat també generador.
- La funció generadora usa la instrucció `yield`. Sintàcticament, distinguirem una funció generadora de la resta per la presència del mot `yield`.
- Cada cop que invoquem `next` a un generador, executa el codi de la funció generadora fins que troba un `yield`. El valor de l'expressió que hi ha al yield és el valor que retornarà `next`.
- El següent `next` farà que la funció s'executi des de la instrucció següent al darrer `yield` executat fins el següent yield. Si no n'hi ha més `yield` a trobar, `next` llançarà un `StopIteration`
- La funció generadora no té cap `return` per retornar valor.

Introducció: for

iter() i next()

Generadors

Exemples

range

in

Expressions

Expressions

- Un generador o funció generadora és una funció especial que retorna un objecte iterador anomenat també generador.
- La funció generadora usa la instrucció `yield`. Sintàcticament, distinguirem una funció generadora de la resta per la presència del mot `yield`.
- Cada cop que invoquem `next` a un generador, executa el codi de la funció generadora fins que troba un `yield`. El valor de l'expressió que hi ha al yield és el valor que retornarà `next`.
- El següent `next` farà que la funció s'executi des de la instrucció següent al darrer `yield` executat fins el següent yield. Si no n'hi ha més `yield` a trobar, `next` llançarà un `StopIteration`
- La funció generadora no té cap `return` per retornar valor.

Introducció: for

iter() i next()

Generadors

Exemples

range

in

Expressions

Expressions

Per exemple, un comptador que vagi des de `minim` fins un `maxim`:

Fitxer `comptador2.py`:

```
def Comptador(minim, maxim):
    actual = minim
    while actual <= maxim:
        yield actual
        actual = actual + 1
```

Generadors: yield

Introducció: for
iter() i next()
Generadors
Exemples
range
in
Expressions
Expressions

```
>>> from comptador2 import Comptador
>>> for i in Comptador(3,8):
...     print(i)
...
3
4
5
6
7
8
```

Generadors: yield

Introducció: for
iter() i next()
Generadors
Exemples
range
in
Expressions
Expressions

```
>>> from comptador2 import Comptador
>>> for i in Comptador(3,8):
...     print(i)
...
3
4
5
6
7
8
```

Introducció: for

iter() i next()

Generadors

Exemples

range

in

Expressions

Expressions

```
>>> from comptador2 import Comptador
>>> for i in Comptador(3,8):
...     print(i)
...
3
4
5
6
7
8
```

Introducció: for

iter() i next()

Generadors

Exemples

range

in

Expressions

Expressions

```
>>> from comptador2 import Comptador
>>> for i in Comptador(3,8):
...         print(i)
...
3
4
5
6
7
8
```

Generadors: yield

Introducció: for
iter() i next()
Generadors
Exemples
range
in
Expressions
Expressions

`yield` no té res a veure amb `return`

```
>>> def Comptador(minim, maxim):
...     actual = minim
...     while actual <= maxim:
...         print('actual:', actual)
...         yield actual
...         actual = actual + 1
...         print('següent:', actual)
...
>>> g = Comptador(0,2)
>>> next(g)
actual: 0
0
>>> next(g)
següent: 1
actual: 1
1
>>> next(g)
següent: 2
actual: 2
2
>>> next(g)
següent: 3
Traceback (most recent call last):
  File ≪stdin>", line 1, in <module>
StopIteration
```

Generadors: yield

Introducció: for
iter() i next()
Generadors
Exemples
range
in
Expressions
Expressions

`yield` no té res a veure amb `return`

```
>>> def Comptador(minim, maxim):
...     actual = minim
...     while actual <= maxim:
...         print('actual:', actual)
...         yield actual
...         actual = actual + 1
...         print('següent:', actual)
...
>>> g = Comptador(0,2)
>>> next(g)
actual: 0
0
>>> next(g)
següent: 1
actual: 1
1
>>> next(g)
següent: 2
actual: 2
2
>>> next(g)
següent: 3
Traceback (most recent call last):
  File ≪stdin>", line 1, in <module>
StopIteration
```

Generadors: yield

Introducció: for
iter() i next()
Generadors
Exemples
range
in
Expressions
Expressions

yield no té res a veure amb `return`

```
>>> def Comptador(minim, maxim):
...     actual = minim
...     while actual <= maxim:
...         print('actual:', actual)
...         yield actual
...         actual = actual + 1
...         print('següent:', actual)
...
>>> g = Comptador(0,2)
>>> next(g)
actual: 0
0
>>> next(g)
següent: 1
actual: 1
1
>>> next(g)
següent: 2
actual: 2
2
>>> next(g)
següent: 3
Traceback (most recent call last):
  File ≪stdin>", line 1, in <module>
StopIteration
```

Generadors: yield

Introducció: for
iter() i next()
Generadors
Exemples
range
in
Expressions
Expressions

yield no té res a veure amb `return`

```
>>> def Comptador(minim, maxim):
...     actual = minim
...     while actual <= maxim:
...         print('actual:', actual)
...         yield actual
...         actual = actual + 1
...         print('següent:', actual)
...
>>> g = Comptador(0,2)
>>> next(g)
actual: 0
0
>>> next(g)
següent: 1
actual: 1
1
>>> next(g)
següent: 2
actual: 2
2
>>> next(g)
següent: 3
Traceback (most recent call last):
  File ≪stdin>", line 1, in <module>
StopIteration
```

Generadors: yield

Introducció: for
iter() i next()
Generadors
Exemples
range
in
Expressions
Expressions

yield no té res a veure amb `return`

```
>>> def Comptador(minim, maxim):
...     actual = minim
...     while actual <= maxim:
...         print('actual:', actual)
...         yield actual
...         actual = actual + 1
...         print('següent:', actual)
...
>>> g = Comptador(0,2)
>>> next(g)
actual: 0
0
>>> next(g)
següent: 1
actual: 1
1
>>> next(g)
següent: 2
actual: 2
2
>>> next(g)
següent: 3
Traceback (most recent call last):
  File ≪stdin>", line 1, in <module>
StopIteration
```

Generadors: yield

Introducció: for
iter() i next()
Generadors
Exemples
range
in
Expressions
Expressions

yield no té res a veure amb return

```
>>> def Comptador(minim, maxim):
...     actual = minim
...     while actual <= maxim:
...         print('actual:', actual)
...         yield actual
...         actual = actual + 1
...         print('següent:', actual)
...
>>> g = Comptador(0,2)
>>> next(g)
actual: 0
0
>>> next(g)
següent: 1
actual: 1
1
>>> next(g)
següent: 2
actual: 2
2
>>> next(g)
següent: 3
Traceback (most recent call last):
  File ≪stdin>", line 1, in <module>
StopIteration
```

Generadors: yield

Introducció: for
iter() i next()
Generadors
Exemples
range
in
Expressions
Expressions

yield no té res a veure amb `return`

```
>>> def Comptador(minim, maxim):
...     actual = minim
...     while actual <= maxim:
...         print('actual:', actual)
...         yield actual
...         actual = actual + 1
...         print('següent:', actual)
...
>>> g = Comptador(0,2)
>>> next(g)
actual: 0
0
>>> next(g)
següent: 1
actual: 1
1
>>> next(g)
següent: 2
actual: 2
2
>>> next(g)
següent: 3
Traceback (most recent call last):
  File ≪stdin>", line 1, in <module>
StopIteration
```

Generadors: yield

Introducció: for
iter() i next()
Generadors
Exemples
range
in
Expressions
Expressions

yield no té res a veure amb `return`

```
>>> def Comptador(minim, maxim):
...     actual = minim
...     while actual <= maxim:
...         print('actual:', actual)
...         yield actual
...         actual = actual + 1
...         print('següent:', actual)
...
>>> g = Comptador(0,2)
>>> next(g)
actual: 0
0
>>> next(g)
següent: 1
actual: 1
1
>>> next(g)
següent: 2
actual: 2
2
>>> next(g)
següent: 3
Traceback (most recent call last):
  File ≪stdin>", line 1, in <module>
StopIteration
```

Introducció: for

iter() i next()

Generadors

Exemples

range

in

Expressions

Expressions

Generació dies setmana:

```
def setmana():
    yield 'dilluns'
    yield 'dimarts'
    yield 'dimecres'
    yield 'dijous'
    yield 'divendres'
    yield 'dissabte'
    yield 'diumenge'
```

Introducció:
for
iter() i next()
Generadors
Exemples
range
in
Expressions
Expressions

Dies de la setmana

```
>>> for d in setmana():
...     print(d)
...
dilluns
dimarts
dimecres
dijous
divendres
dissabte
diumenge
```

Introducció: for

iter() i next()

Generadors

Exemples

range

in

Expressions

Expressions

Generació sèrie Fibonacci:

```
def fibonacci():
    a = 0
    yield a
    b = 1
    yield b
    while True:
        a, b = b, a + b
        yield b
```

Introducció: for

iter() i next()

Generadors

Exemples

range

in

Expressions

Expressions

Si teniu ganes de veure els límits del vostre ordinador:

```
>>> list(fibonacci())
```

Una forma d'enumerar els 10 primers seria:

Fibonacci

```
>>> from fibonacci import fibonacci
>>> for i, fib in enumerate(fibonacci()):
...     print(fib, end=' ')
...     if i==9:
...         break
...
0 1 1 2 3 5 8 13 21 34
```

Introducció: for

iter() i next()

Generadors

Exemples

range

in

Expressions

Expressions

Si teniu ganes de veure els límits del vostre ordinador:

```
>>> list(fibonacci())
```

Una forma d'enumerar els 10 primers seria:

Fibonacci

```
>>> from fibonacci import fibonacci
>>> for i, fib in enumerate(fibonacci()):
...     print(fib, end=' ')
...     if i==9:
...         break
...
0 1 1 2 3 5 8 13 21 34
```

Introducció: for

iter() i next()

Generadors

Exemples

range

in

Expressions

Expressions

Si teniu ganes de veure els límits del vostre ordinador:

```
>>> list(fibonacci())
```

Una forma d'enumerar els 10 primers seria:

Fibonacci

```
>>> from fibonacci import fibonacci
>>> for i, fib in enumerate(fibonacci()):
...     print(fib, end=' ')
...     if i==9:
...         break
...
0 1 1 2 3 5 8 13 21 34
```

Introducció:
for

iter() i next()

Generadors

Exemples

range

in

Expressions

Expressions

Si teniu ganes de veure els límits del vostre ordinador:

```
>>> list(fibonacci())
```

Una forma d'enumerar els 10 primers seria:

Fibonacci

```
>>> from fibonacci import fibonacci
>>> for i, fib in enumerate(fibonacci()):
...     print(fib, end=' ')
...     if i==9:
...         break
...
0 1 1 2 3 5 8 13 21 34
```

Introducció:
for

iter() i next()

Generadors

Exemples

range

in

Expressions

Expressions

Fibonacci

```
>>> def elsPrimers(g, n):
...         for i in range(n):
...             yield next(g)
...
>>> for f in elsPrimers(fibonacci(), 10):
...     print(f, end='-')
...
0-1-1-2-3-5-8-13-21-34-
```

Introducció:
for

iter() i next()

Generadors

Exemples

range

in

Expressions

Expressions

Fibonacci

```
>>> def elsPrimers(g, n):
...         for i in range(n):
...             yield next(g)
...
>>> for f in elsPrimers(fibonacci(), 10):
...     print(f, end='-')
...
0-1-1-2-3-5-8-13-21-34-
```

Introducció:
for

iter() i next()

Generadors

Exemples

range

in

Expressions

Expressions

Fibonacci

```
>>> def elsPrimers(g, n):
...         for i in range(n):
...             yield next(g)
...
>>> for f in elsPrimers(fibonacci(), 10):
...     print(f, end='-')
...
0-1-1-2-3-5-8-13-21-34-
```

Introducció: for

iter() i next()

Generadors

Exemples

range

in

Expressions

Expressions

Fibonacci

```
>>> def elsPrimers(g, n):
...         for i in range(n):
...                 yield next(g)
...
>>> for f in elsPrimers(fibonacci(), 10):
...     print(f, end='-')
...
0-1-1-2-3-5-8-13-21-34-
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

La funció `range` retorna un objecte de tipus `range` que funciona com iterable:

```
>>> it = iter(range(2))
>>> next(it)
0
>>> next(it)
1
>>> next(it)
Traceback (most recent call last):
  File «stdin>", line 1, in <module>
StopIteration
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

La funció `range` retorna un objecte de tipus `range` que funciona com iterable:

```
>>> it = iter(range(2))
>>> next(it)
0
>>> next(it)
1
>>> next(it)
Traceback (most recent call last):
  File «stdin>", line 1, in <module>
StopIteration
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

La funció `range` retorna un objecte de tipus `range` que funciona com iterable:

```
>>> it = iter(range(2))
>>> next(it)
0
>>> next(it)
1
>>> next(it)
Traceback (most recent call last):
  File «stdin>", line 1, in <module>
StopIteration
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

La funció `range` retorna un objecte de tipus `range` que funciona com iterable:

```
>>> it = iter(range(2))
>>> next(it)
0
>>> next(it)
1
>>> next(it)
Traceback (most recent call last):
  File «stdin>", line 1, in <module>
StopIteration
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

La funció `range` retorna un objecte de tipus `range` que funciona com iterable:

```
>>> it = iter(range(2))
>>> next(it)
0
>>> next(it)
1
>>> next(it)
Traceback (most recent call last):
  File «stdin>", line 1, in <module>
StopIteration
```

Introducció:
for
iter() i next()
Generadors
range
in
Expressions
Expressions

La funció `range` retorna un objecte de tipus `range` que funciona com iterable:

```
>>> it = iter(range(2))
>>> next(it)
0
>>> next(it)
1
>>> next(it)
Traceback (most recent call last):
  File ≪stdin>", line 1, in <module>
StopIteration
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

funciona com a contenidor:

```
>>> 3 in range(0,10,2)
False
>>> 6 in range(0,10,2)
True
```

funciona com una llista (sense llesques, ni concatenació):

```
>>> range(0,10,2)[3]
6
>>> range(0,10,2)[9]
Traceback (most recent call last):
  File «stdin>", line 1, in <module>
IndexError: range object index out of range
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

funciona com a contenidor:

```
>>> 3 in range(0,10,2)
False
>>> 6 in range(0,10,2)
True
```

funciona com una llista (sense llesques, ni concatenació):

```
>>> range(0,10,2)[3]
6
>>> range(0,10,2)[9]
Traceback (most recent call last):
  File «stdin>", line 1, in <module>
IndexError: range object index out of range
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

funciona com a contenidor:

```
>>> 3 in range(0,10,2)
False
>>> 6 in range(0,10,2)
True
```

funciona com una llista (sense llesques, ni concatenació):

```
>>> range(0,10,2)[3]
6
>>> range(0,10,2)[9]
Traceback (most recent call last):
  File «stdin>", line 1, in <module>
IndexError: range object index out of range
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

funciona com a contenidor:

```
>>> 3 in range(0,10,2)
False
>>> 6 in range(0,10,2)
True
```

funciona com una llista (sense llesques, ni concatenació):

```
>>> range(0,10,2)[3]
6
>>> range(0,10,2)[9]
Traceback (most recent call last):
  File «stdin>", line 1, in <module>
IndexError: range object index out of range
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

funciona com a contenidor:

```
>>> 3 in range(0,10,2)
False
>>> 6 in range(0,10,2)
True
```

funciona com una llista (sense llesques, ni concatenació):

```
>>> range(0,10,2)[3]
6
>>> range(0,10,2)[9]
Traceback (most recent call last):
  File «stdin>", line 1, in <module>
IndexError: range object index out of range
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

funciona com a contenidor:

```
>>> 3 in range(0,10,2)
False
>>> 6 in range(0,10,2)
True
```

funciona com una llista (sense llesques, ni concatenació):

```
>>> range(0,10,2)[3]
6
>>> range(0,10,2)[9]
Traceback (most recent call last):
  File «stdin>", line 1, in <module>
IndexError: range object index out of range
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

funciona com a contenidor:

```
>>> 3 in range(0,10,2)
False
>>> 6 in range(0,10,2)
True
```

funciona com una llista (sense llesques, ni concatenació):

```
>>> range(0,10,2)[3]
6
>>> range(0,10,2)[9]
Traceback (most recent call last):
  File «stdin>", line 1, in <module>
IndexError: range object index out of range
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

L'operador `in` crida al mètode `__contains__` si aquest existeix. Si no crida a `__iter__`. Si aquest no existeix, llavors crida a `__getitem__`.
Com que els iteradors son iterables, tenim:

```
>>> it = iter([1,2,3])
>>> 1 in it
True
>>> 1 in it
False
```

Fals doncs `it` ha avançat després de trobar el primer 1. A més, en aquest segon cop `it` s'ha esgotat al no trobar cap 1 més endavant.

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

L'operador `in` crida al mètode `__contains__` si aquest existeix. Si no crida a `__iter__`. Si aquest no existeix, llavors crida a `__getitem__`.
Com que els iteradors son iterables, tenim:

```
>>> it = iter([1,2,3])
>>> 1 in it
True
>>> 1 in it
False
```

Fals doncs `it` ha avançat després de trobar el primer 1. A més, en aquest segon cop `it` s'ha esgotat al no trobar cap 1 més endavant.

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

L'operador `in` crida al mètode `__contains__` si aquest existeix. Si no crida a `__iter__`. Si aquest no existeix, llavors crida a `__getitem__`.
Com que els iteradors son iterables, tenim:

```
>>> it = iter([1,2,3])
>>> 1 in it
True
>>> 1 in it
False
```

Fals doncs `it` ha avançat després de trobar el primer 1. A més, en aquest segon cop `it` s'ha esgotat al no trobar cap 1 més endavant.

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

L'operador `in` crida al mètode `__contains__` si aquest existeix. Si no crida a `__iter__`. Si aquest no existeix, llavors crida a `__getitem__`.
Com que els iteradors son iterables, tenim:

```
>>> it = iter([1,2,3])
>>> 1 in it
True
>>> 1 in it
False
```

Fals doncs `it` ha avançat després de trobar el primer 1. A més, en aquest segon cop `it` s'ha esgotat al no trobar cap 1 més endavant.

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

L'operador `in` crida al mètode `__contains__` si aquest existeix. Si no crida a `__iter__`. Si aquest no existeix, llavors crida a `__getitem__`.
Com que els iteradors son iterables, tenim:

```
>>> it = iter([1,2,3])
>>> 1 in it
True
>>> 1 in it
False
```

Fals doncs `it` ha avançat després de trobar el primer 1. A més, en aquest segon cop `it` s'ha esgotat al no trobar cap 1 més endavant.

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

Com que està esgotat, no troba res:

```
>>> 3 in it
False
```

Un altre exemple:

```
>>> it = iter([1,2,3])
>>> 3 in it
True
>>> 2 in it
False
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

Com que està esgotat, no troba res:

```
>>> 3 in it
False
```

Un altre exemple:

```
>>> it = iter([1,2,3])
>>> 3 in it
True
>>> 2 in it
False
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

Com que està esgotat, no troba res:

```
>>> 3 in it
False
```

Un altre exemple:

```
>>> it = iter([1,2,3])
>>> 3 in it
True
>>> 2 in it
False
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

Com que està esgotat, no troba res:

```
>>> 3 in it
False
```

Un altre exemple:

```
>>> it = iter([1,2,3])
>>> 3 in it
True
>>> 2 in it
False
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

Com que està esgotat, no troba res:

```
>>> 3 in it
False
```

Un altre exemple:

```
>>> it = iter([1,2,3])
>>> 3 in it
True
>>> 2 in it
False
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

Com que està esgotat, no troba res:

```
>>> 3 in it
False
```

Un altre exemple:

```
>>> it = iter([1,2,3])
>>> 3 in it
True
>>> 2 in it
False
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

Com que està esgotat, no troba res:

```
>>> 3 in it
False
```

Un altre exemple:

```
>>> it = iter([1,2,3])
>>> 3 in it
True
>>> 2 in it
False
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

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]]
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

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]]
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

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]]
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

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]]
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

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]]
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

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]]
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

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,
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

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,
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

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,
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

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,
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

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,
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

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,
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

Expressions

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

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

Expressions

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

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

Expressions

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

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

Expressions

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

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

Expressions

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

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

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
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

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
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

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
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

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
```

Introducció: for

iter() i next()

Generadors

range

in

Expressions

Expressions

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
```

Introducció: for
iter() i next()
Generadors
range
in
Expressions
Expressions

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
```

Introducció: for
iter() i next()
Generadors
range
in
Expressions
Expressions

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
```

Introducció: for
iter() i next()
Generadors
range
in
Expressions
Expressions

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
```