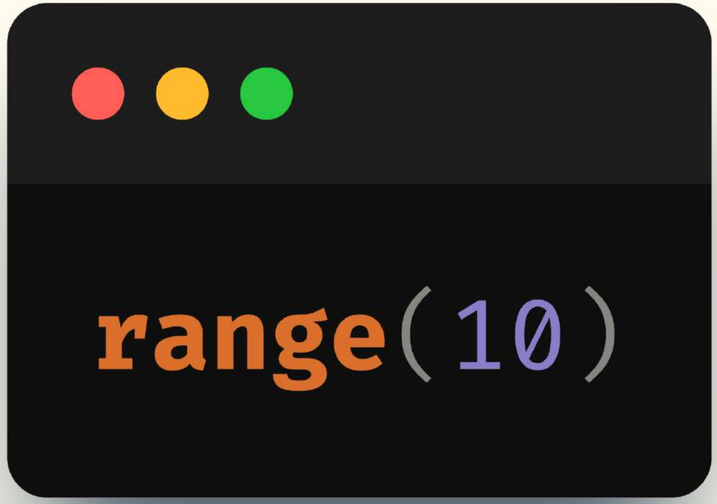
A tour of the module `itertools`

EuroPython 2025 🇨🇿

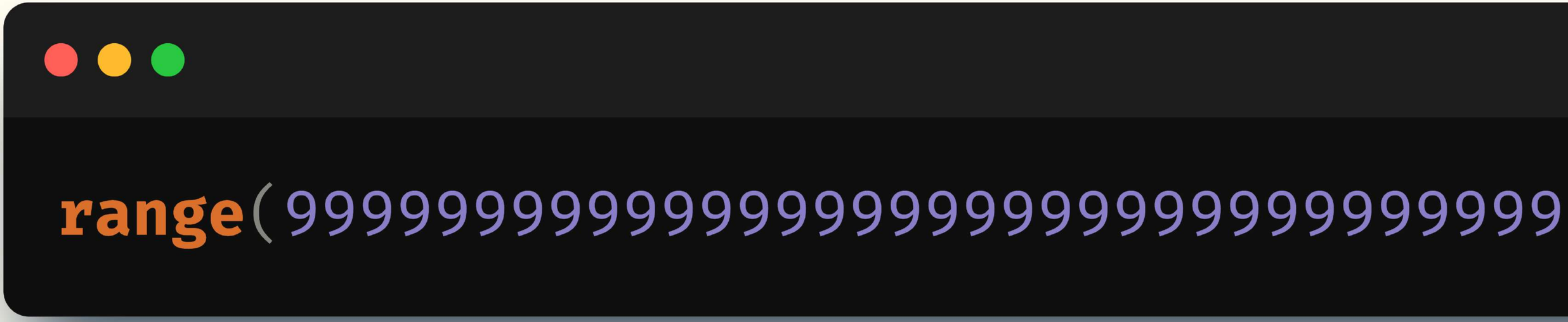
Rodrigo 🐍🚀

mathspp.com/drops

Why itertools?



vs



`range` *generates* elements on demand

`range` ***generates*** **elements on demand**

vs

Lists hold their elements in memory

Laziness is key

Lazy iterables are awesome

`itertools` ❤️ **iterables**

itertools menu

CONTENT WARNING

```
accumulate
batched
chain
combinations
combinations_with_replacement
compress
count
cycle
dropwhile
filterfalse
groupby
islice
pairwise
permutations
product
repeat
starmap
takewhile
tee
zip_longest
```

Filtering

- `compress`
- `dropwhile`
- `filterfalse`
- `takewhile`

Complementary

- `accumulate`
- `starmap`
- `zip_longest`

Reshaping

- `batched`
- `chain`
- `groupby`
- `islice`
- `pairwise`

Combinatorial

- `permutations`
- `product`
- `combinations`
- `combinations_with_replacement`

Infinite

- `count`
- `cycle`
- `repeat`

`tee`

Filtering

compress

dropwhile

filterfalse

takewhile

Complementary

accumulate

starmap

zip_longest

Reshaping

batched

chain

groupby

islice

pairwise

Combinatorial

permutations

product

combinations

combinations_with_replacement

Infinite

count

cycle

repeat

tee

Filtering

- compress
- dropwhile
- filterfalse
- takewhile

Complementary

- accumulate
- starmap
- zip_longest

Reshaping

- batched
- chain
- groupby
- islice
- pairwise

Combinatorial

- permutations
- product
- combinations
- combinations_with_replacement

Infinite

- count
- cycle
- repeat

tee

Filtering

- `compress`
- `dropwhile`
- `filterfalse`
- `takewhile`

Complementary

- `accumulate`
- `starmap`
- `zip_longest`

Reshaping

- `batched`
- `chain`
- `groupby`
- `islice`
- `pairwise`

Combinatorial

- `permutations`
- `product`
- `combinations`
- `combinations_with_replacement`

Infinite

- `count`
- `cycle`
- `repeat`

`tee`

Filtering

- compress
- dropwhile
- filterfalse
- takewhile

Complementary

- accumulate
- starmap
- zip_longest

Reshaping

- batched
- chain
- groupby
- islice
- pairwise

Combinatorial

- permutations
- product
- combinations
- combinations_with_replacement

Infinite

- count
- cycle
- repeat

tee

Filtering

- compress
- dropwhile
- filterfalse
- takewhile

Complementary

- accumulate
- starmap
- zip_longest

Reshaping

- batched
- chain
- groupby
- islice
- pairwise

Combinatorial

- permutations
- product
- combinations
- combinations_with_replacement

Infinite

- count
- cycle
- repeat

tee

Filtering

compress

dropwhile

filterfalse

takewhile

Complementary

accumulate

starmap

zip_longest

Reshaping

batched

chain

groupby

islice

pairwise

Combinatorial

permutations

product

combinations

combinations_with_replacement

Infinite

count

cycle

repeat

tee

combinations

```
flavours = ["chocolate", "vanilla", "strawberry"]
for scoops in combinations(flavours, 2):
    print(scoops)
```

```
flavours = ["chocolate", "vanilla", "strawberry"]
for scoops in combinations(flavours, 2):
    print(scoops)
```

```
('chocolate', 'vanilla')
('chocolate', 'strawberry')
('vanilla', 'strawberry')
```

permutations

```
flavours = ["chocolate", "vanilla", "strawberry"]
for scoops in permutations(flavours, 2):
    print(scoops)
```

```
flavours = ["chocolate", "vanilla", "strawberry"]
for scoops in permutations(flavours, 2):
    print(scoops)
```

```
('chocolate', 'vanilla')
('chocolate', 'strawberry')
('vanilla', 'chocolate')
('vanilla', 'strawberry')
('strawberry', 'chocolate')
('strawberry', 'vanilla')
```

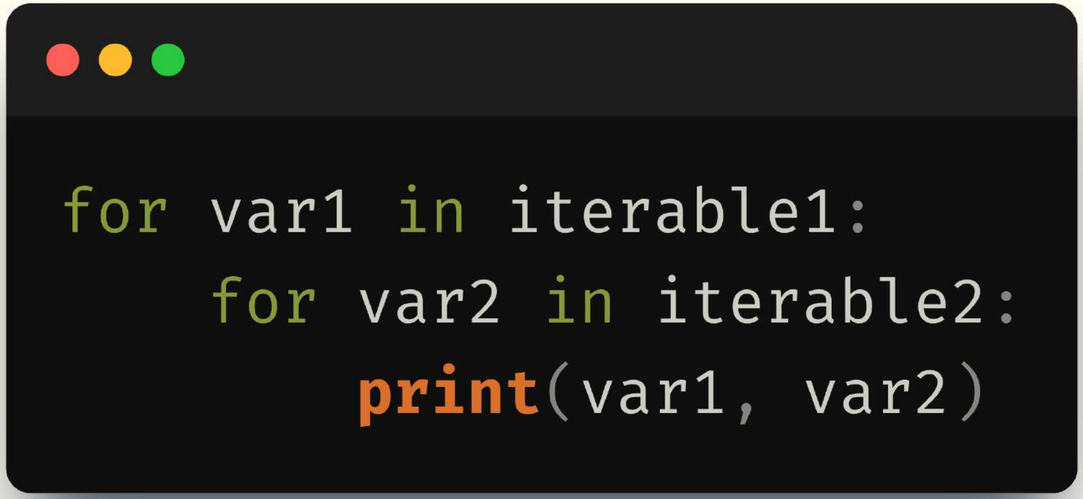
product

```
nr_scoops = [2, 3]
served_on = ["cup", "cone"]
```


nr_scoops = [2, 3]
served_on = ["cup", "cone"]
for scoop_n, served_on in product(nr_scoops, served_on):
print(f"{scoop_n} scoops served on a {served_on}.")

```python
nr_scoops = [2, 3]
served_on = ["cup", "cone"]
for scoop_n, served_on in product(nr_scoops, served_on):
    print(f"{scoop_n} scoops served on a {served_on}.")
```

```
2 scoops served on a cup.
2 scoops served on a cone.
3 scoops served on a cup.
3 scoops served on a cone.
```


for var1 in iterable1:
for var2 in iterable2:
print(var1, var2)


for var1, var2 in product(iterable1, iterable2):
print(var1, var2)

Filtering

- compress
- dropwhile
- filterfalse
- takewhile

Complementary

- accumulate
- starmap
- zip_longest

Reshaping

- batched
- chain
- groupby
- islice
- pairwise

Combinatorial

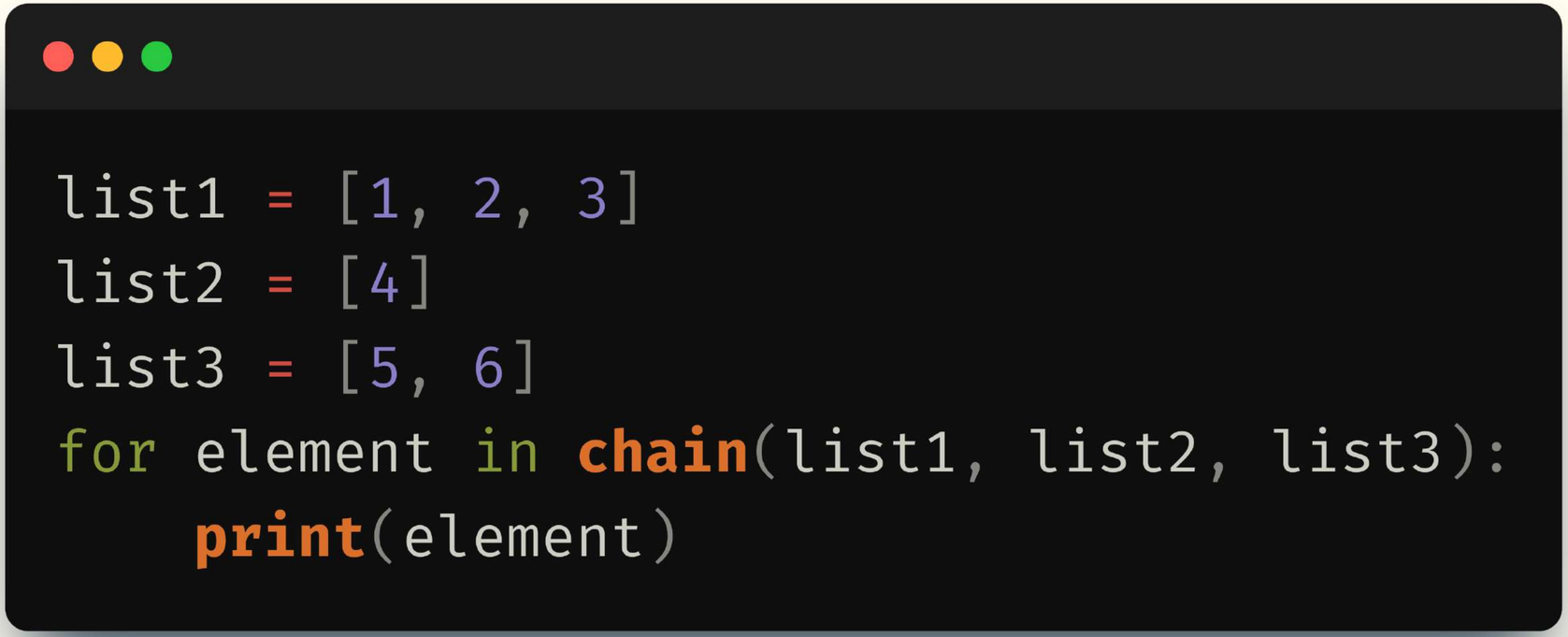
- permutations
- product
- combinations
- combinations_with_replacement

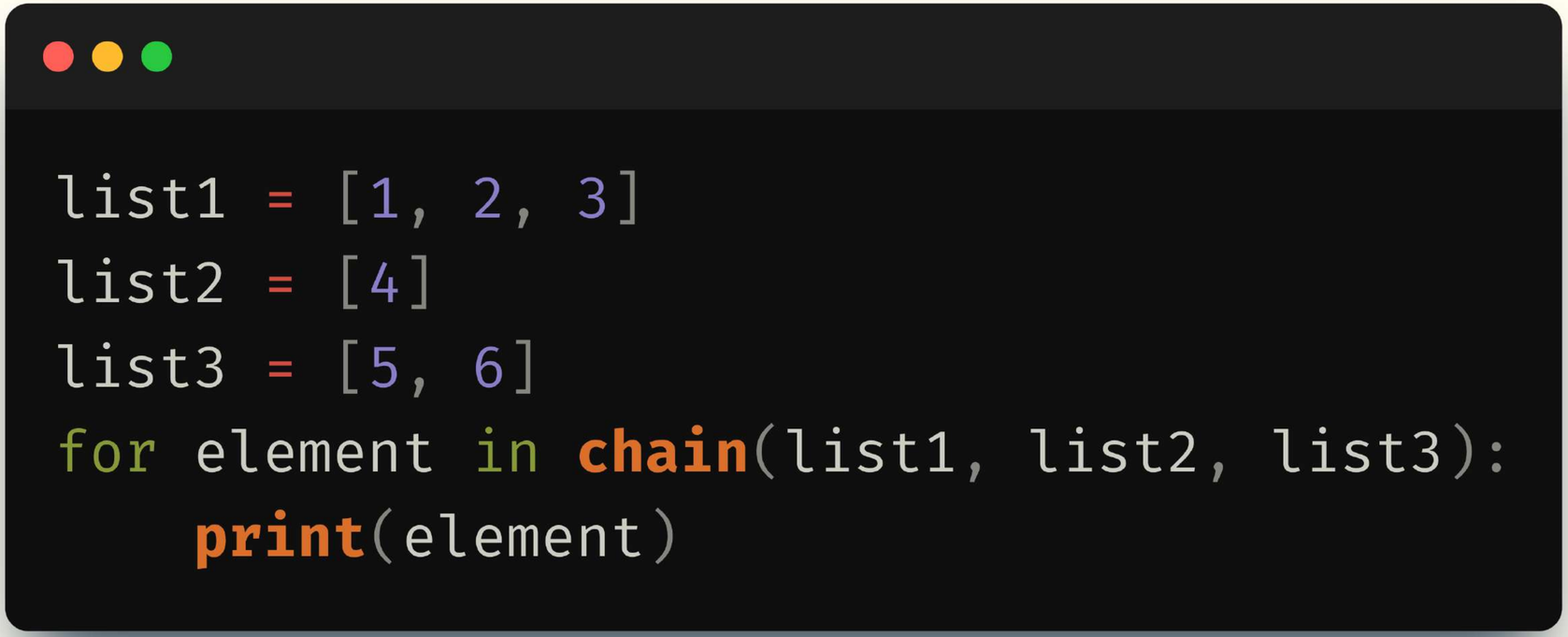
Infinite

- count
- cycle
- repeat

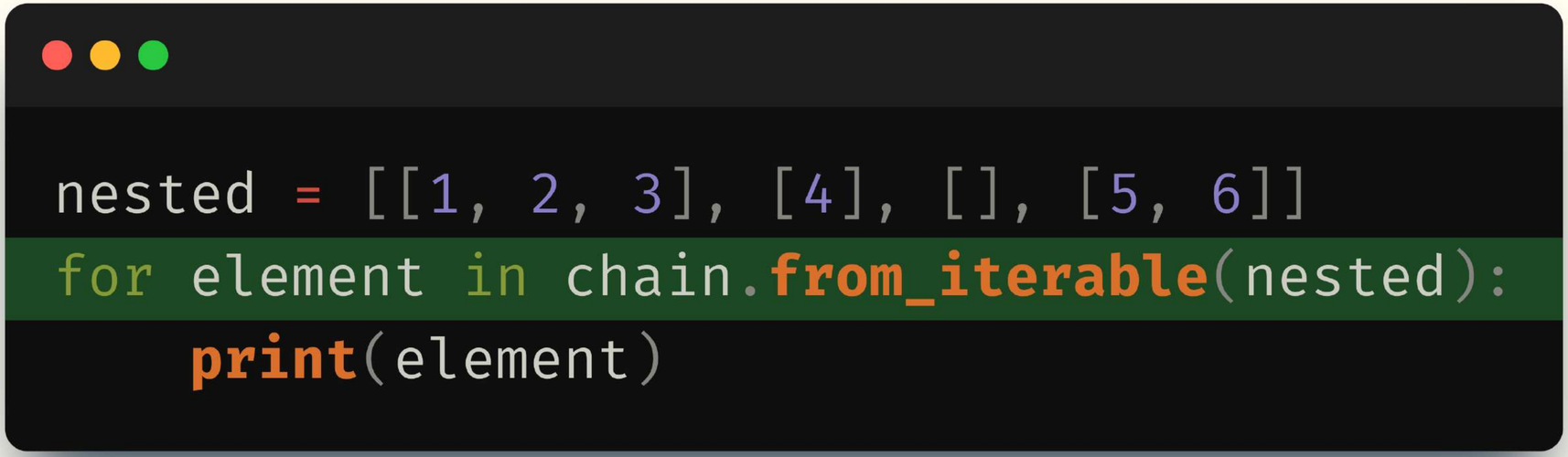
tee

chain


list1 = [1, 2, 3]
list2 = [4]
list3 = [5, 6]
for element in chain(list1, list2, list3):
print(element)


list1 = [1, 2, 3]
list2 = [4]
list3 = [5, 6]
for element in chain(list1, list2, list3):
print(element)

```
1
2
3
4
5
6
```


nested = [[1, 2, 3], [4], [], [5, 6]]
for element in chain.from_iterable(nested):
print(element)

```
1
2
3
4
5
6
```

islice


battleship_coords = product(range(10), range(10))
for square in battleship_coords[:10]: #?!
    print(square, end=" ")


battleship_coords = product(range(10), range(10))
for square in islice(battleship_coords, 10):
print(square, end=" ")

pairwise

```python
my_list = ["Hello", "from", "Prague"]
for before, after in zip(my_list[1:], my_list[:-1]):
    print(before, after)
```

```python
my_list = ["Hello", "from", "Prague"]
for before, after in zip(my_list[1:], my_list[:-1]):
    print(before, after)
```

```
Hello from
from Prague
```

```python
my_list = ["Hello", "from", "Prague"]
for before, after in pairwise(my_list):
    print(before, after)
```

```
Hello from
from Prague
```

Filtering

- `compress`
- `dropwhile`
- `filterfalse`
- `takewhile`

Complementary

- `accumulate`
- `starmap`
- `zip_longest`

Reshaping

- `batched`
- `chain`
- `groupby`
- `islice`
- `pairwise`

Combinatorial

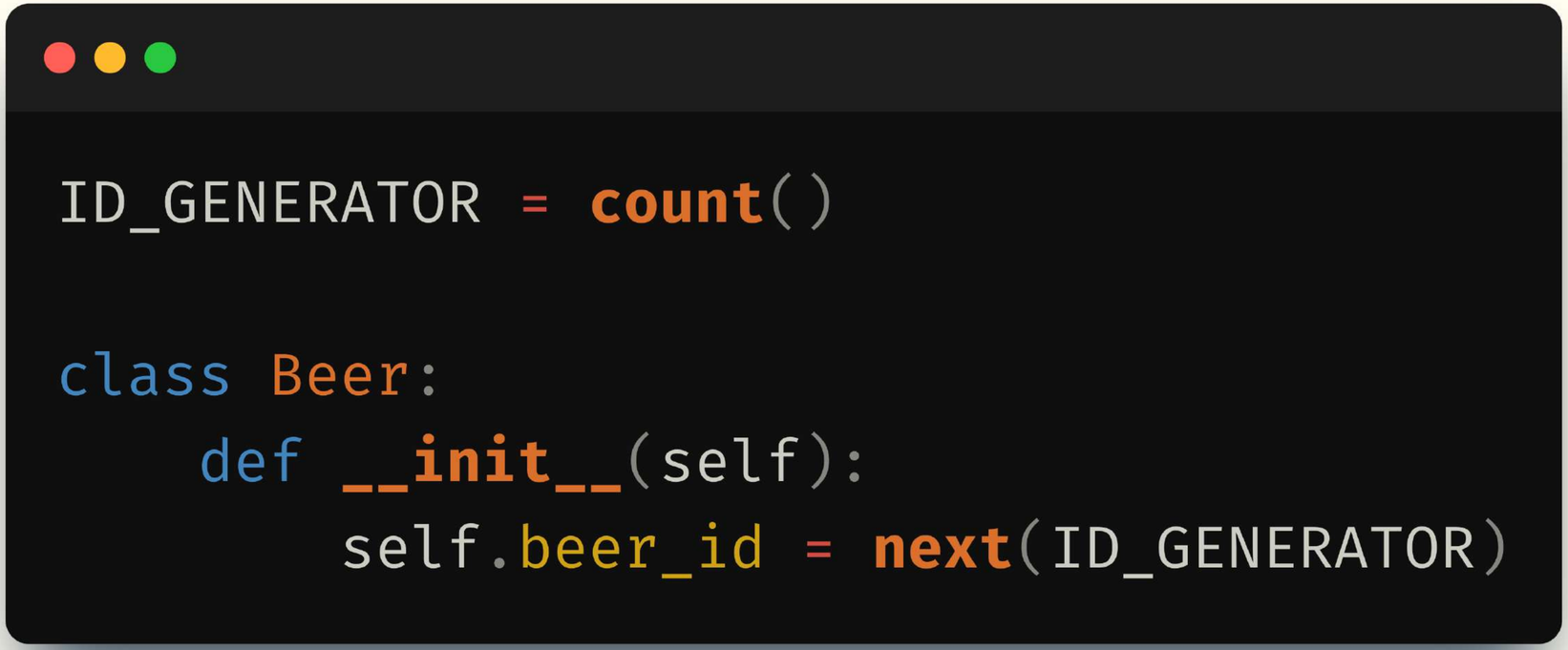
- `permutations`
- `product`
- `combinations`
- `combinations_with_replacement`

Infinite

- `count`
- `cycle`
- `repeat`

`tee`

count


ID_GENERATOR = count()
class Beer:
def __init__(self):
self.beer_id = next(ID_GENERATOR)

```python
ID_GENERATOR = count()

class Beer:
    def __init__(self):
        self.beer_id = next(ID_GENERATOR)
```

```python
b1, b2, b3 = Beer(), Beer(), Beer()
print(b1.beer_id)  # 0
print(b2.beer_id)  # 1
print(b3.beer_id)  # 2
```

cycle

```python
chckn_words = ["Chicken", "chicken", "Egg", "chicken", "chkn"]

for word in cycle(chckn_words):
    print(word, end=" ")
```

```
chckn_words = ["Chicken", "chicken", "Egg", "chicken", "chkn"]

for word in islice(cycle(chckn_words), 8):
    print(word, end=" ")
```

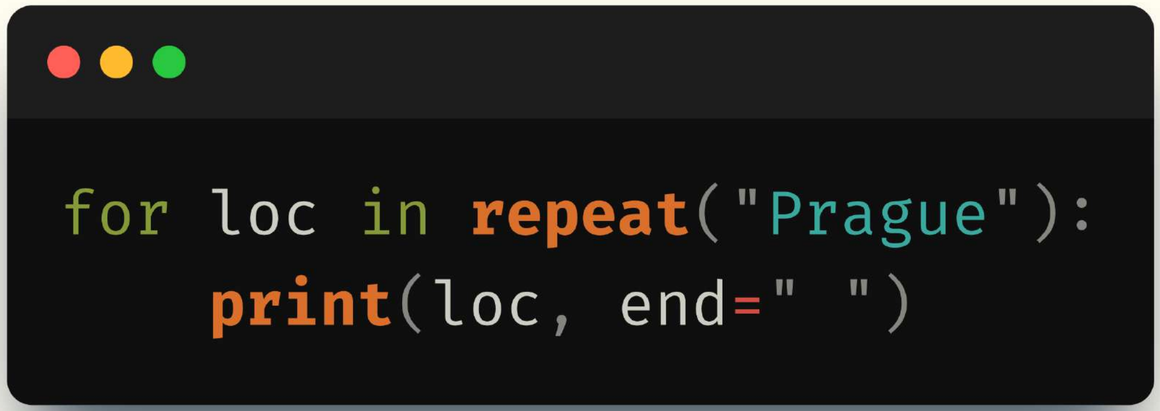
```
Chicken chicken Egg chicken chkn Chicken chicken Egg
```

```python
chckn_words = ["Chicken", "chicken", "Egg", "chicken", "chkn"]

print(" ".join(islice(cycle(chckn_words), 8)))
```

```
Chicken chicken Egg chicken chkn Chicken chicken Egg
```

repeat


for loc in repeat("Prague"):
print(loc, end=" ")

```python
for loc in repeat("Prague"):
    print(loc, end=" ")
```

```
Prague Prague Prague Prague Prague Prague Prague Prague P
 Prague Prague Prague Prague Prague Prague Prague Prague
e Prague Prague Prague Prague Prague Prague Prague Prague
ue Prague Prague Prague Prague Prague Prague Prague Pragu
gue Prague Prague Prague Prague Prague Prague Prague Prag
ague Prague Prague Prague Prague Prague Prague Prague Pra
rague Prague Prague Prague Prague Prague Prague Prague Pr
Prague Prague Prague Prague Prague Prague Prague Prague P
 Prague Prague Prague Prague Prague Prague Prague Prague
e Prague Prague Prague Prague Prague Prague Prague Prague
ue Prague Prague Prague Prague Prague Prague Prague Pragu
gue Prague Prague Prague Prague Prague Prague Prague Prag
ague Prague Prague Prague Prague Prague Prague Prague Pra
rague Prague Prague Prague Prague Prague Prague Prague Pr
Prague Prague Prague Prague Prague Prague Prague Prague P
 Prague Prague Prague Prague Prague Prague Prague Prague
e Prague Prague Prague Prague Prague Prague Prague Prague
ue Prague Prague Prague Prague Prague Prague Prague Pragu
gue Prague Prague Prague Prague Prague Prague Prague Prag
ague Prague Prague Prague Prague Prague Prague Prague Pra
rague Prague Prague Prague Prague Prague Prague Prague Pr
Prague Prague Prague Prague Prague Prague Prague Prague P
 Prague Prague Prague Prague Prague Prague Prague Prague
e Prague Prague Prague Prague Prague Prague Prague Prague
ue Prague Prague Prague Prague Prague Prague Prague Pragu
gue Prague Prague Prague Prague Prague Prague Prague Prag
ague Prague Prague Prague Prague Prague Prague Prague Pra
rague Prague Prague Prague Prague Prague Prague Prague Pr
```


for loc in repeat("Prague", 3):
print(loc, end=" ")

```python
for loc in repeat("Prague", 3):
    print(loc, end=" ")
```

```
Prague Prague Prague
```

```python
for year, loc in zip(range(2023, 2026), repeat("Prague")):
        print(f"EP {year} location: {loc}")
```

```python
for year, loc in zip(range(2023, 2026), repeat("Prague")):
        print(f"EP {year} location: {loc}")
```

```
EP 2023 location: Prague
EP 2024 location: Prague
EP 2025 location: Prague
```

```python
locations = chain(
    ["online", "Dublin"],
    repeat("Prague", 3),
)

for year, loc in zip(count(2021), locations):
    print(f"EP {year} location: {loc}")
```


locations = chain(
["online", "Dublin"],
repeat("Prague", 3),
)
for year, loc in zip(count(2021), locations):
print(f"EP {year} location: {loc}")

```python
locations = chain(
    ["online", "Dublin"],
    repeat("Prague", 3),
)

for year, loc in zip(count(2021), locations):
    print(f"EP {year} location: {loc}")
```

```
EP 2021 location: online
EP 2022 location: Dublin
EP 2023 location: Prague
EP 2024 location: Prague
EP 2025 location: Prague
```

Filtering

- compress
- dropwhile
- filterfalse
- takewhile

Complementary

- accumulate
- starmap
- zip_longest

Reshaping

- batched
- chain
- groupby
- islice
- pairwise

Combinatorial

- permutations
- product
- combinations
- combinations_with_replacement

Infinite

- count
- cycle
- repeat

tee

filterfalse
(vs filter)

```
people = [("Harry", 17), ("Anne", 21), ("George", 5)]

def is_adult(person): return person[1] >= 18
```

```python
people = [("Harry", 17), ("Anne", 21), ("George", 5)]

def is_adult(person): return person[1] >= 18
```

```python
for name, _ in filter(is_adult, people):
    print(name, end=" ")
```

```python
people = [("Harry", 17), ("Anne", 21), ("George", 5)]

def is_adult(person): return person[1] >= 18
```

```python
for name, _ in filter(is_adult, people):
    print(name, end=" ")
```

```
Anne
```

```python
people = [("Harry", 17), ("Anne", 21), ("George", 5)]

def is_adult(person): return person[1] >= 18
```

```python
for name, _ in filter(is_adult, people):
    print(name, end=" ")
```

```
Anne
```

```python
for name, _ in filterfalse(is_adult, people):
    print(name, end=" ")
```

```python
people = [("Harry", 17), ("Anne", 21), ("George", 5)]

def is_adult(person): return person[1] >= 18
```

```python
for name, _ in filter(is_adult, people):
    print(name, end=" ")
```

```
Anne
```

```python
for name, _ in filterfalse(is_adult, people):
    print(name, end=" ")
```

```
Harry George
```

Filtering

- compress
- dropwhile
- filterfalse
- takewhile

Complementary

- accumulate
- starmap
- zip_longest

Reshaping

- batched
- chain
- groupby
- islice
- pairwise

Combinatorial

- permutations
- product
- combinations
- combinations_with_replacement

Infinite

- count
- cycle
- repeat

tee

zip_longest (vs zip)


names = ["Harry", "Anne", "George"]
ages = [17, 21]

```
names = ["Harry", "Anne", "George"]
ages = [17, 21]
```

```
for name, age in zip(names, ages)
    print(name, age)
```

```
names = ["Harry", "Anne", "George"]
ages = [17, 21]
```

```
for name, age in zip(names, ages)
    print(name, age)
```

```
Harry 17
Anne 21
```

```
names = ["Harry", "Anne", "George"]
ages = [17, 21]
```

```
for name, age in zip(names, ages)
    print(name, age)
```

```
Harry 17
Anne 21
```

```
for name, age in zip_longest(names, ages):
    print(name, age)
```

```
names = ["Harry", "Anne", "George"]
ages = [17, 21]
```

```
for name, age in zip(names, ages)
    print(name, age)
```

```
Harry 17
Anne 21
```

```
for name, age in zip_longest(names, ages):
    print(name, age)
```

```
Harry 17
Anne 21
George None
```

```
for name, age in zip_longest(names, ages, fillvalue=99):
    print(name, age)
```

```
Harry 17
Anne 21
George 99
```

Filtering

- `compress`
- `dropwhile`
- `filterfalse`
- `takewhile`

Complementary

- `accumulate`
- `starmap`
- `zip_longest`

Reshaping

- `batched`
- `chain`
- `groupby`
- `islice`
- `pairwise`

Combinatorial

- `permutations`
- `product`
- `combinations`
- `combinations_with_replacement`

Infinite

- `count`
- `cycle`
- `repeat`

tee

Maybe next year ;)

Filtering

compress

dropwhile

filterfalse

takewhile

Complementary

accumulate

starmap

zip_longest

Reshaping

batched

chain

groupby

islice

pairwise

Combinatorial

permutations

product

combinations

combinations_with_replacement

Infinite

count

cycle

repeat

tee

Filtering

- `compress`
- `dropwhile`
- `filterfalse`
- `takewhile`

Complementary

- `accumulate`
- `starmap`
- `zip_longest`

Reshaping

- `batched`
- `chain`
- `groupby`
- `islice`
- `pairwise`

Combinatorial

- `permutations`
- `product`
- `combinations`
- `combinations_with_replacement`

Infinite

- `count`
- `cycle`
- `repeat`

`tee`

How to write good code?

How to write good code?

Go write bad code!

Rodrigo 🐍🚀

mathspp.com/drops