The paradox of `itertools.tee`

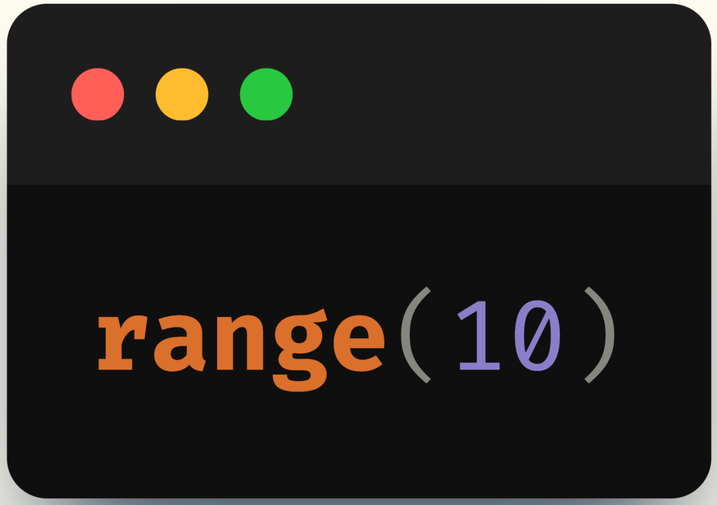
Python Users Berlin 🇩🇪

Feb 2026

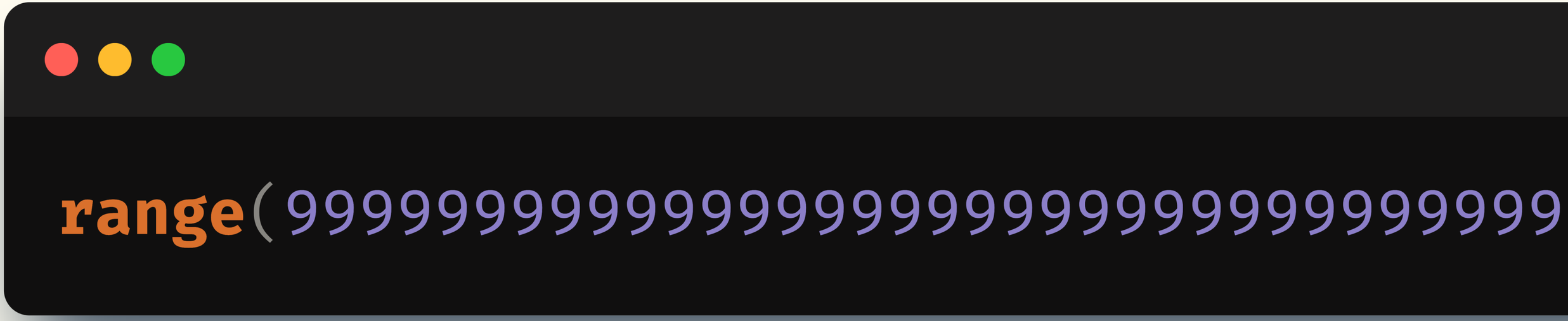
Rodrigo 🐍🚀

mathspp.com/insider

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
```

accumulate

batched

chain

combinations

combinations_with_replacement

compress

count

cycle

dropwhile

filterfalse

groupby

islice

pairwise

permutations

product

repeat

starmap

takewhile

tee

zip_longest

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

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

```
Hello from
from Lisbon
```

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

```
Hello from
from Lisbon
```

```
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

Iterables vs iterators

Iterable: something you can iterate over

Iterable: something you can iterate over

vs

Iterator: low-level object that knows how to iterate

List

List

vs

- `list_iterator`
- `list_reverseiterator`

Why `tee`?

Iterator laziness comes at a cost

Iterator laziness comes at a cost

Iterators are less flexible

Iterator laziness comes at a cost

Iterators are less flexible

Iterators can only be traversed *once*

Live ~~bugs~~demo

(Find the live demo notebook here: https://github.com/mathspp/talks)

- tee is low-level

- tee is low-level
- allows you to defy the iterator protocol

- tee is low-level
- allows you to defy the iterator protocol
- but it's all smoke and mirrors

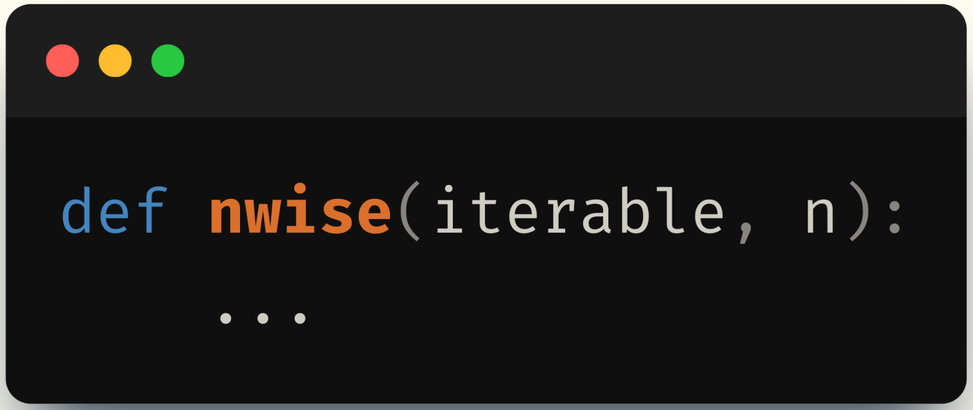
Understanding the inner workings of Python

Understanding the inner workings of Python

Freedom and flexibility

Homework

Implement `nwise`,
a generalisation of `pairwise`



Become the smartest Python dev in the room 🐍🚀