How APL made me a better Python developer

by Rodrigo Girão Serrão

Rodrigo 🐍🚀

🐦 @mathsppblog

Rodrigo 🐍🚀

🐦 @mathsppblog

Textualize

mathspp.com

mathspp.com/talks

“A language that doesn't affect the way you think about programming, is not worth knowing.”

— Alan J. Perlis

rodrigo@mathspp.com

What's APL?

APL = A Programming Language

rodrigo@mathspp.com

APL = A Programming Language

Terrible name, right?

rodrigo@mathspp.com

APL = A Programming Language

Terrible name, right?

Not really, it isn't a pun.

rodrigo@mathspp.com

Game of life

.apl

```
{↑1 ωv.∧3 4=+/,¯1 0 1∘.⊖¯1 0 1∘.⌽⊂ω}
```

rodrigo@mathspp.com

How APL made me a better Python developer

What learning APL taught me about Python

What learning APL taught me about Python

*mileage may vary

Stating the obvious can be good.

rodrigo@mathspp.com

Stating the obvious can be good.

Example: pigeonhole principle.

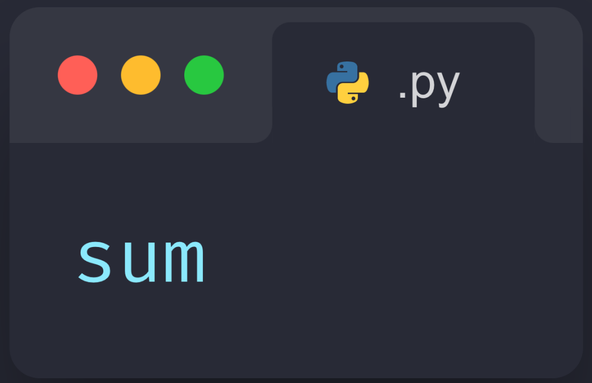
rodrigo@mathspp.com

The LOC that changed everything

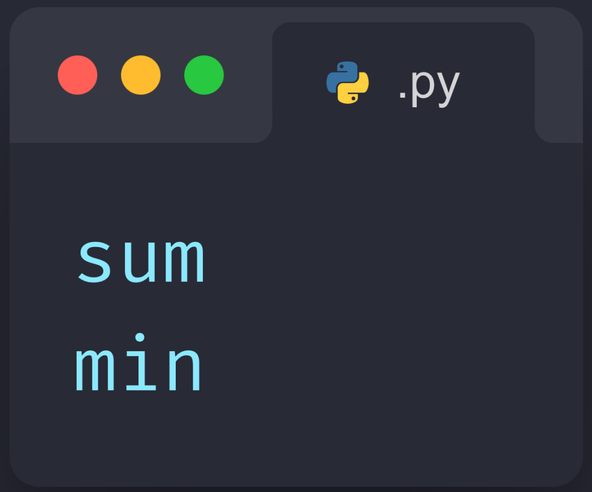
rodrigo@mathspp.com

********** are

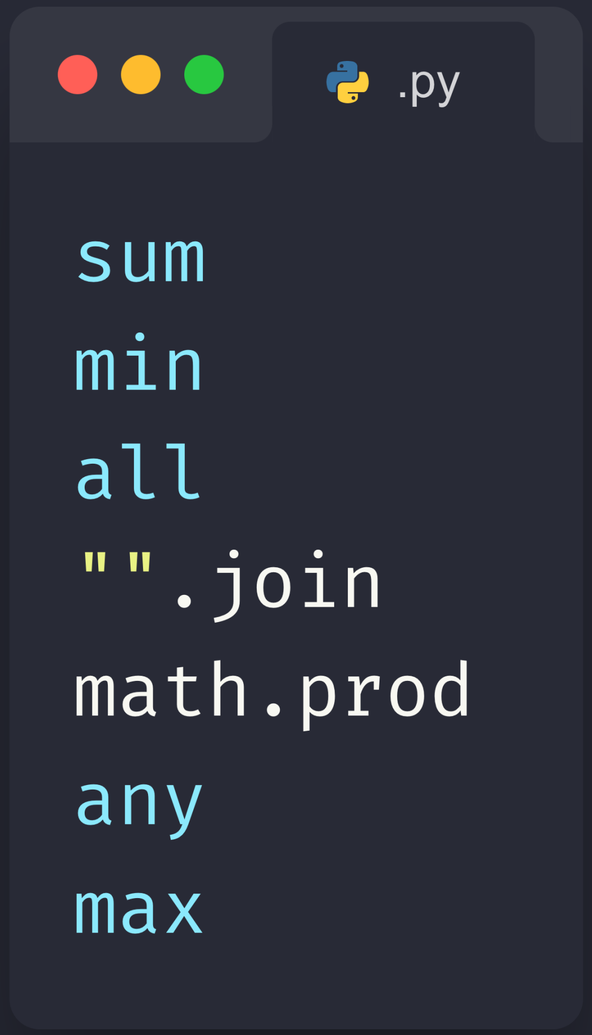
everywhere



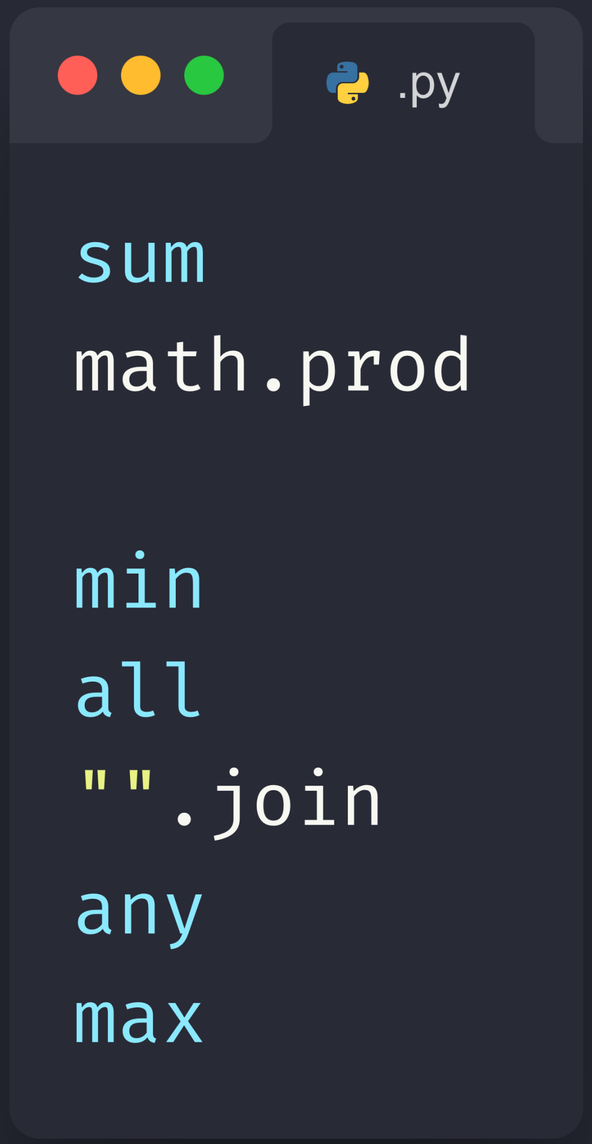
rodrigo@mathspp.com



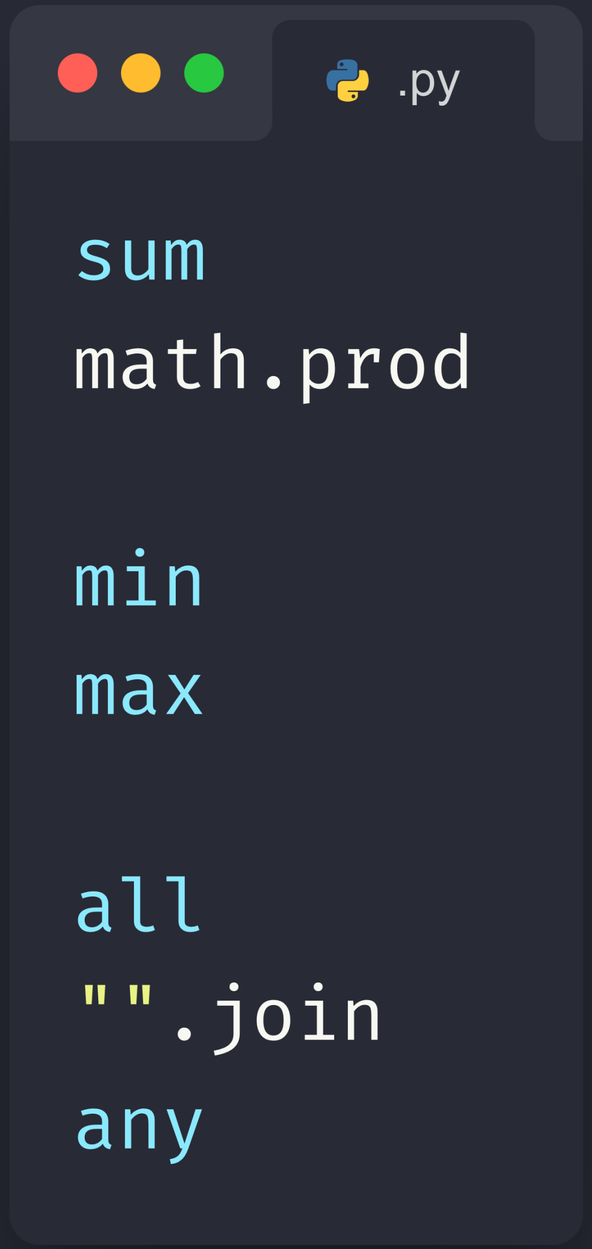
rodrigo@mathspp.com



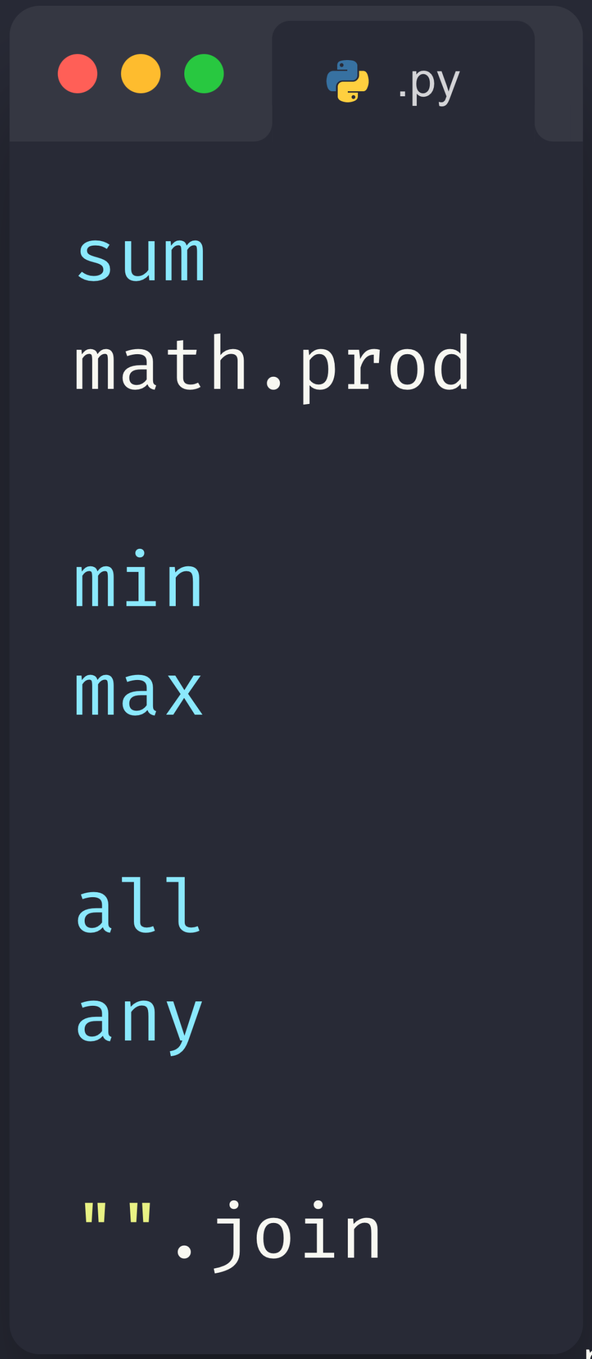
rodrigo@mathspp.com



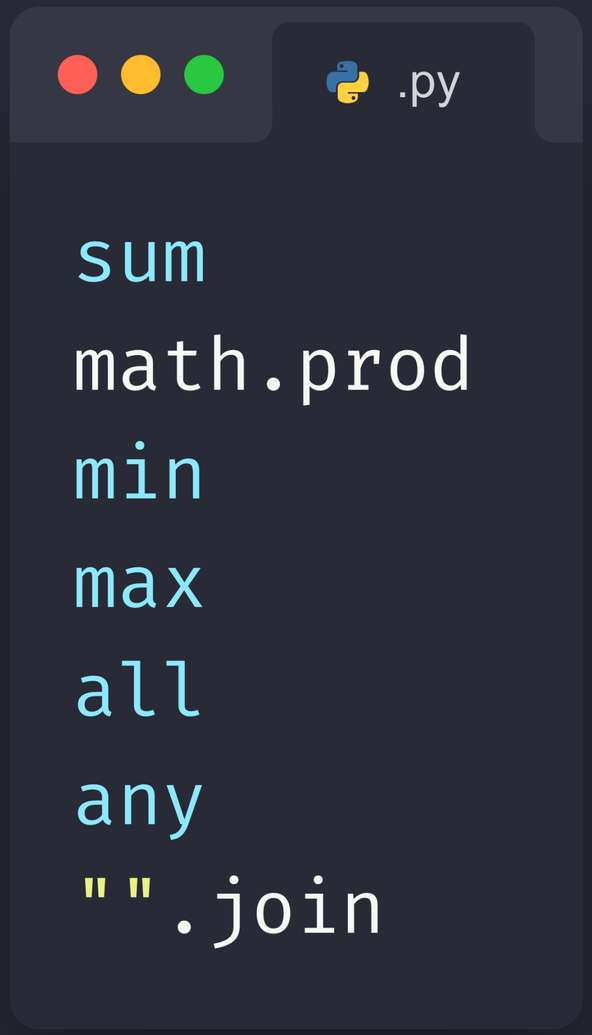
rodrigo@mathspp.com



rodrigo@mathspp.com



rodrigo@mathspp.com



rodrigo@mathspp.com

.py

```
sum
math.prod
min
max
all
any
"".join
```

.apl

```
+/
```

rodrigo@mathspp.com

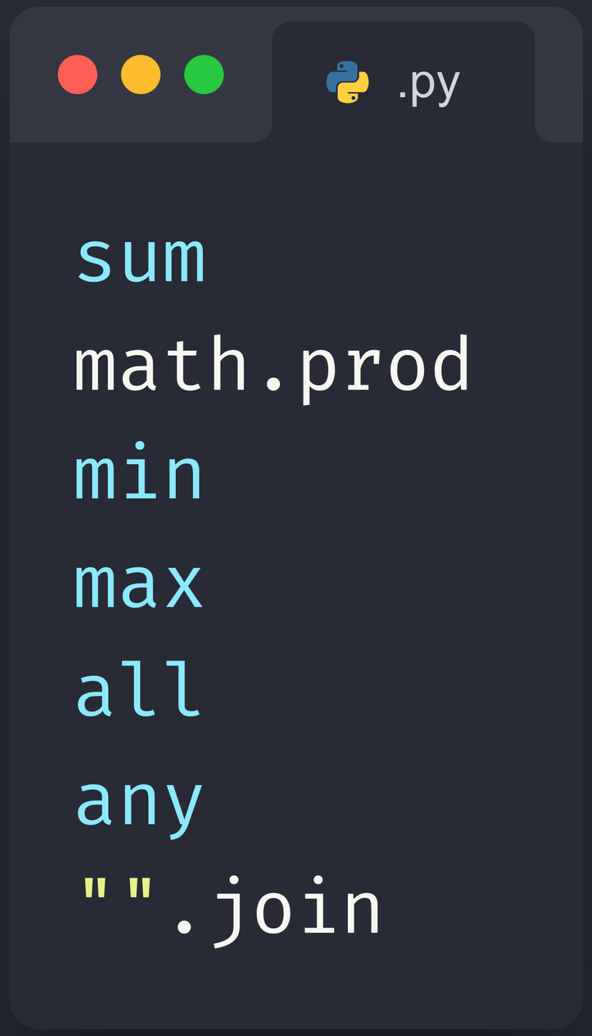
.py

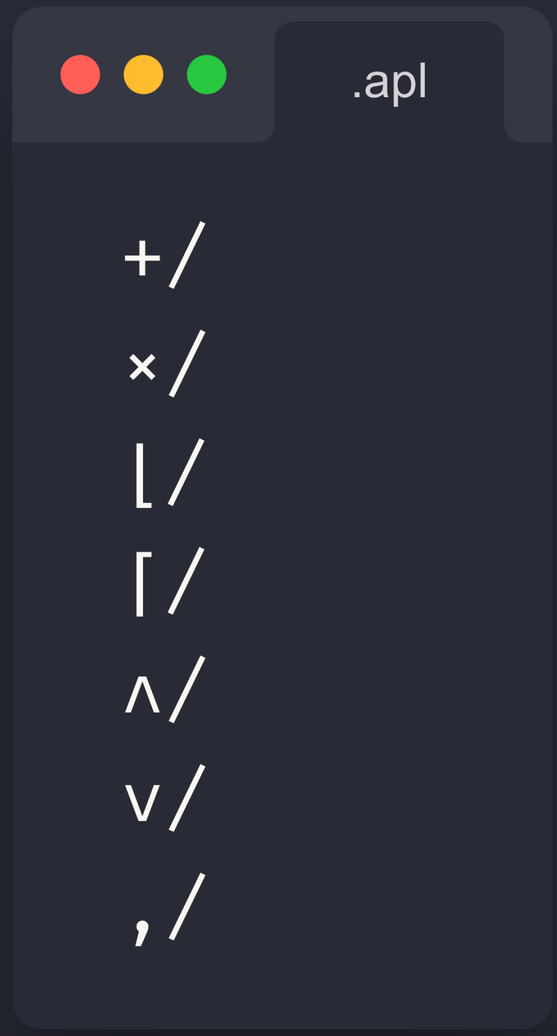
```
sum
math.prod
min
max
all
any
"".join
```

.apl

```
+/
×/
```

rodrigo@mathspp.com





rodrigo@mathspp.com

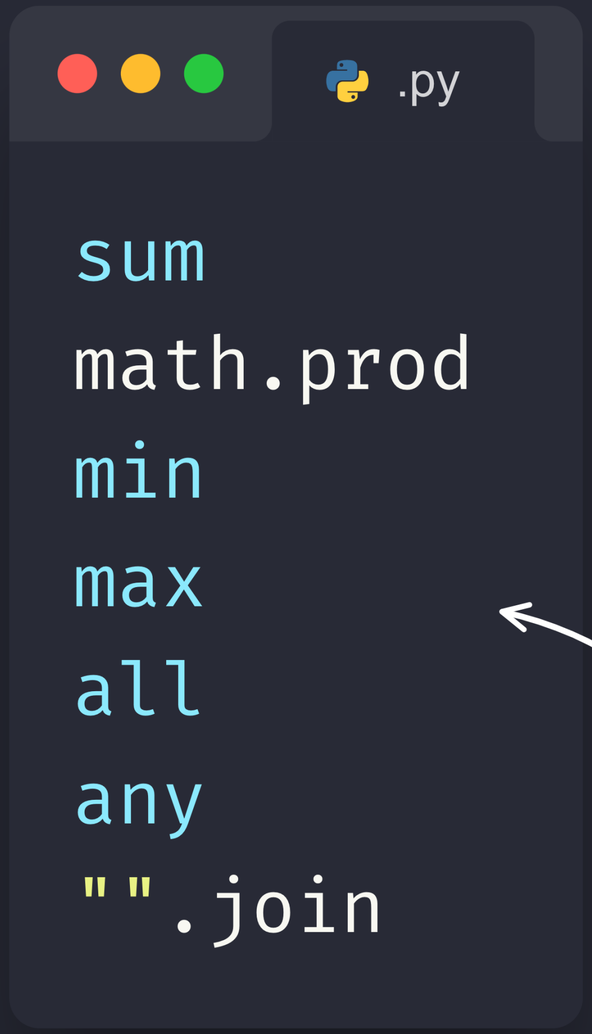
.py

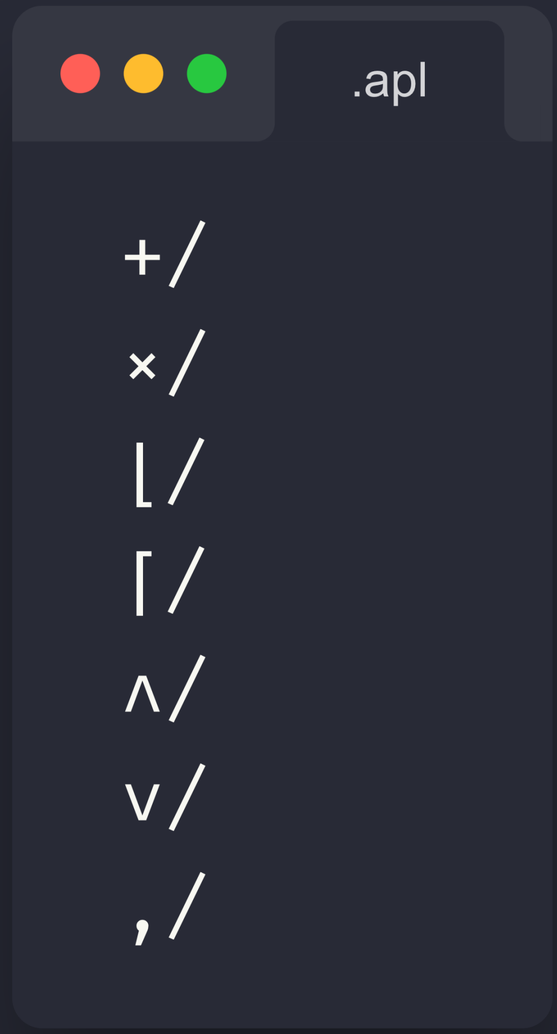
```
sum
math.prod
min
max
all
any
"".join
```

.apl

```
+/
×/
⌊/
⌈/
∧/
∨/
,/
```

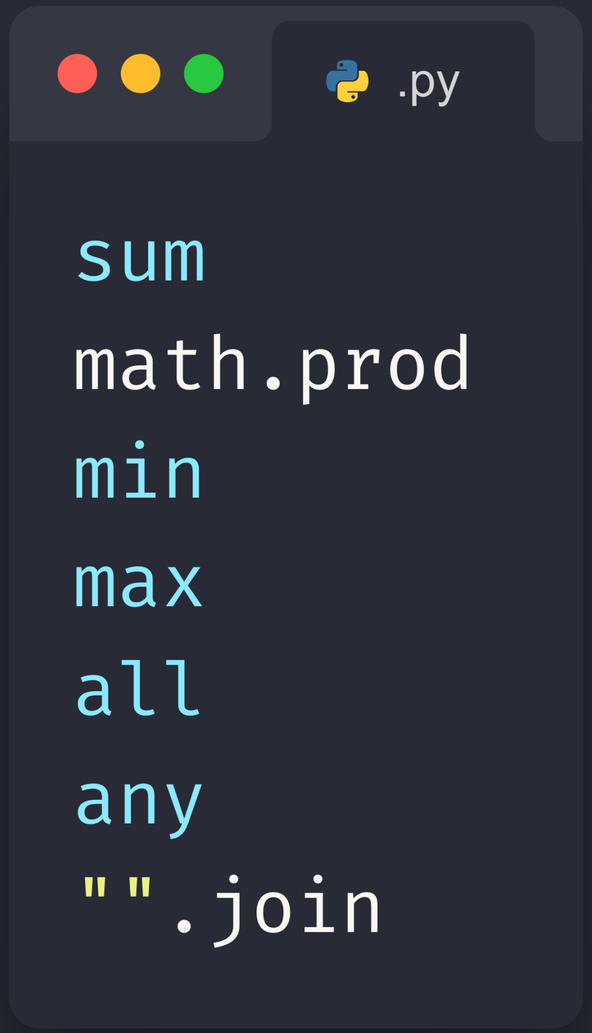
rodrigo@mathspp.com

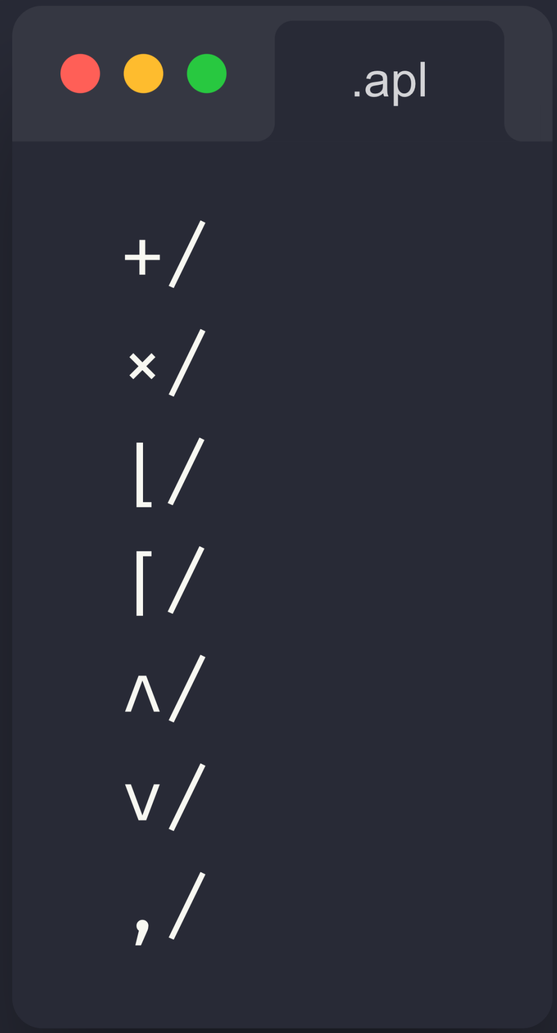




These are all the same thing!

rodrigo@mathspp.com





functools.reduce

rodrigo@mathspp.com

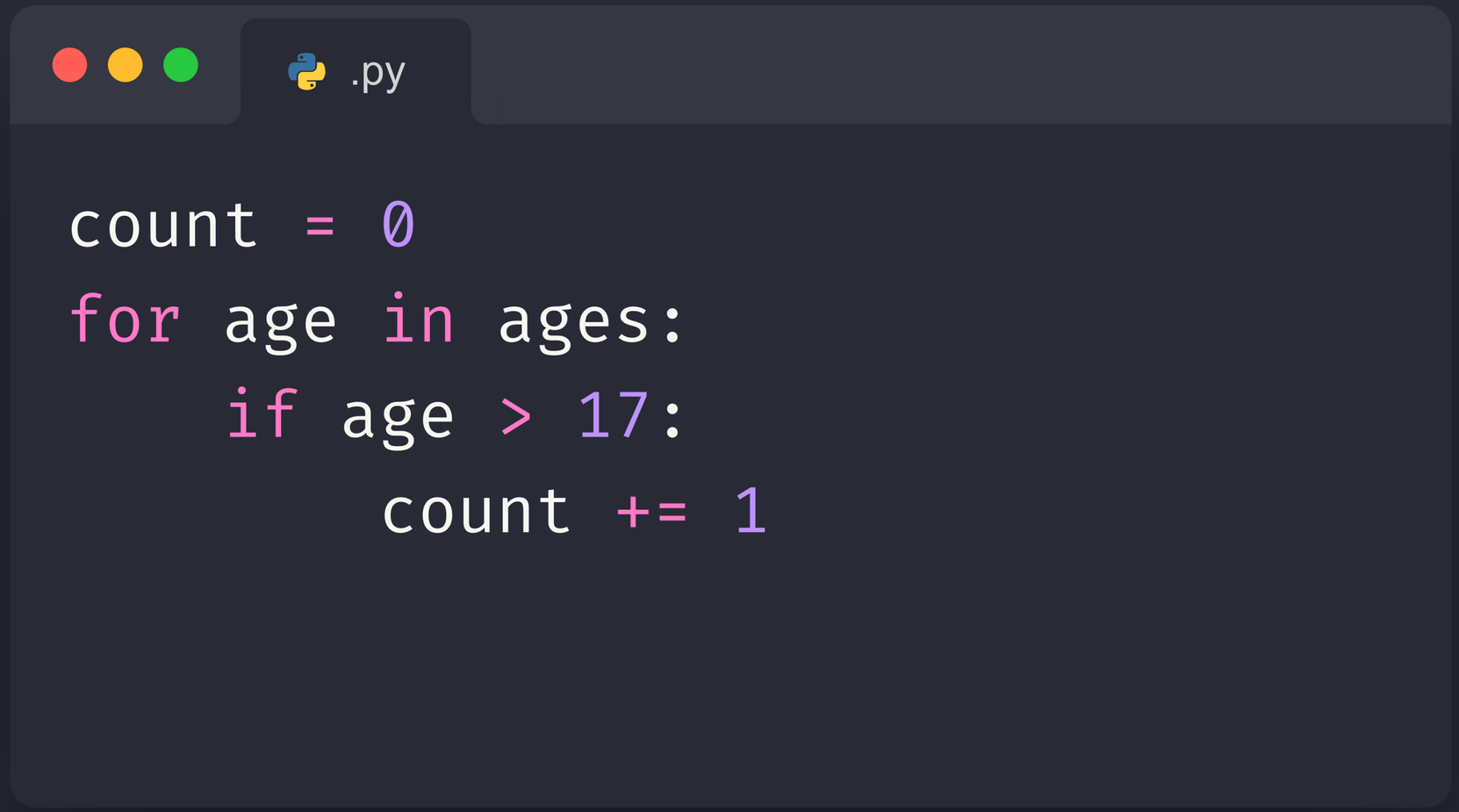
********** are

everywhere

Reductions are everywhere

Data-driven conditionals

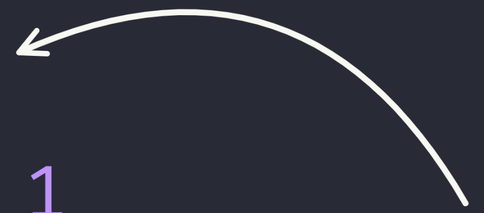
rodrigo@mathspp.com



rodrigo@mathspp.com

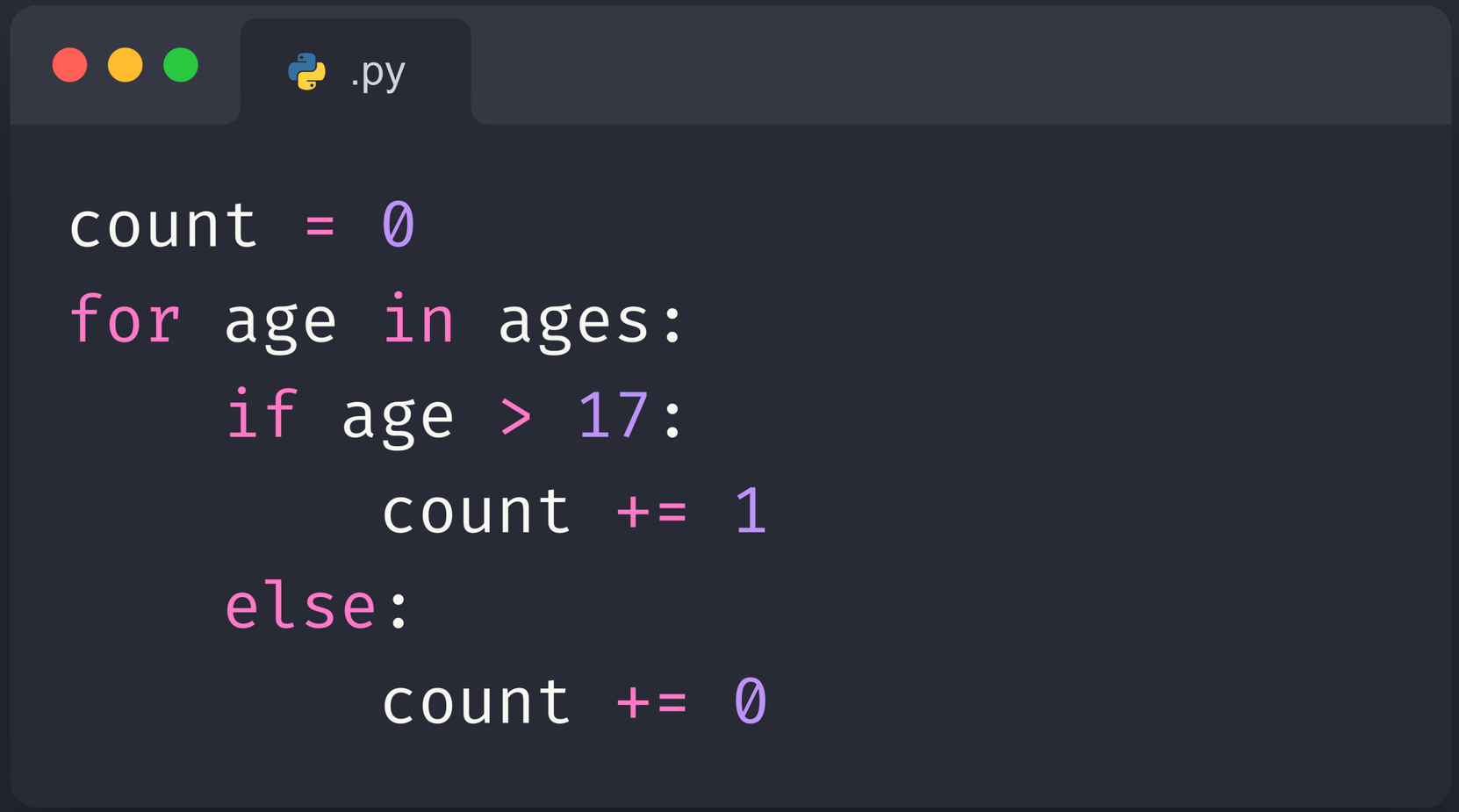
.py

```
count = 0
for age in ages:
    if age > 17:
        count += 1
```



Do we add? 🤔

rodrigo@mathspp.com



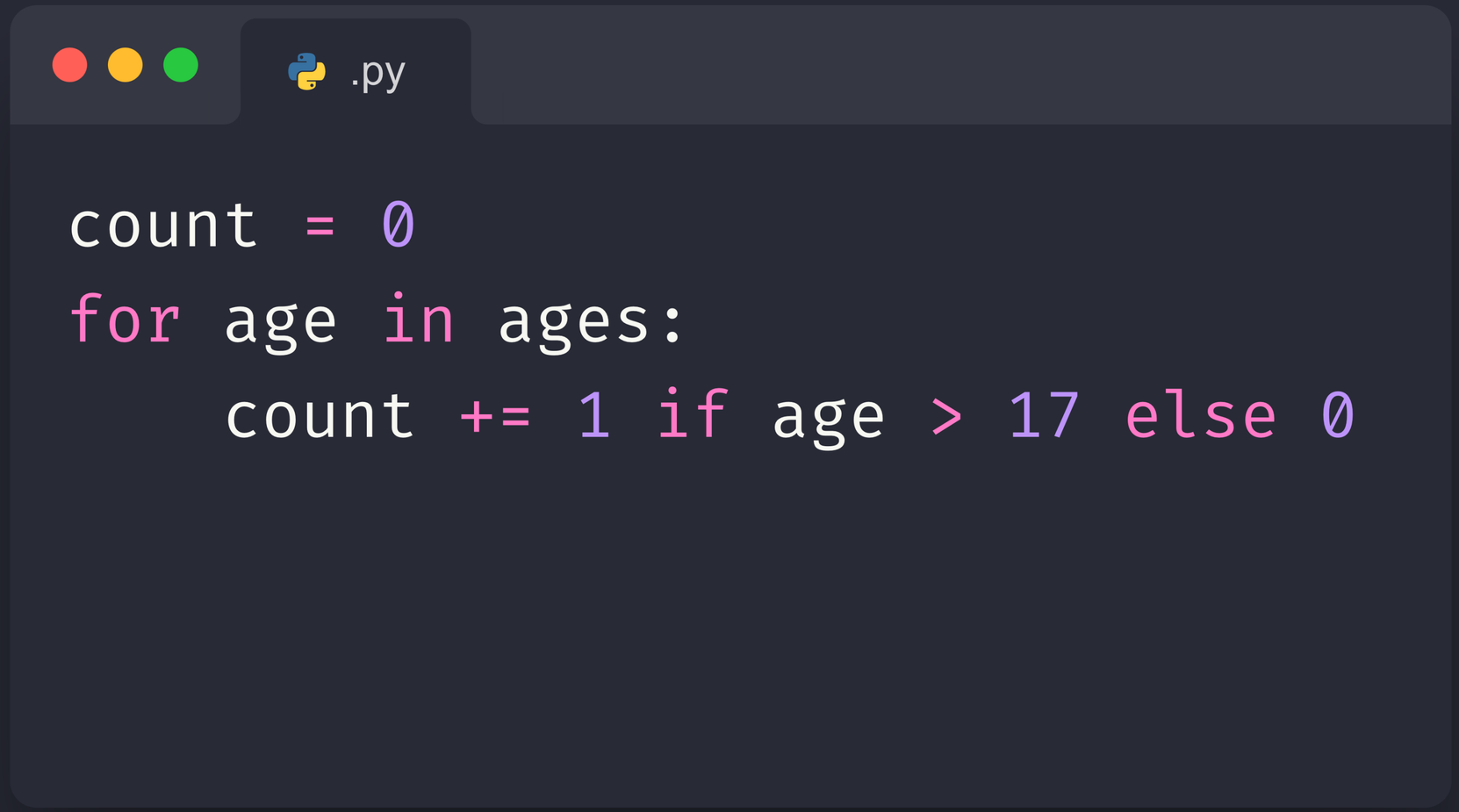
rodrigo@mathspp.com

.py

```python
count = 0
for age in ages:
    if age > 17:
        count += 1
    else:
        count += 0
```

“Useless” but correct

rodrigo@mathspp.com



rodrigo@mathspp.com

.py

```
count = 0
for age in ages:
    count += 1 if age > 17 else 0
```

What do we add? 🤔

rodrigo@mathspp.com

.py

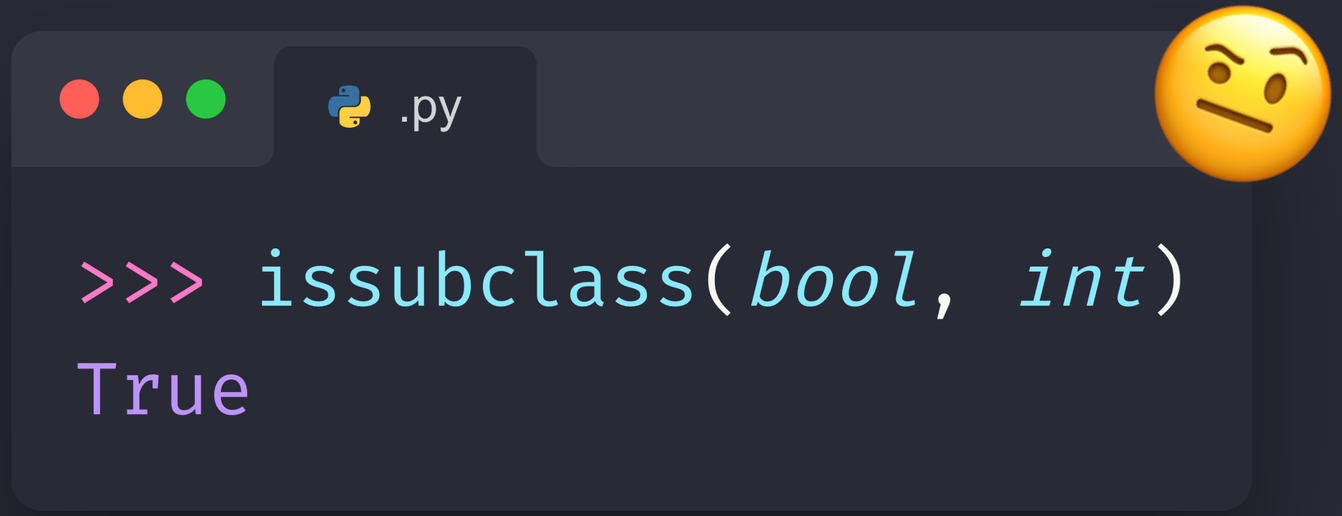
```
count = 0
for age in ages:
    count += 1 if age > 17 else 0
```

What do we add? 🤔

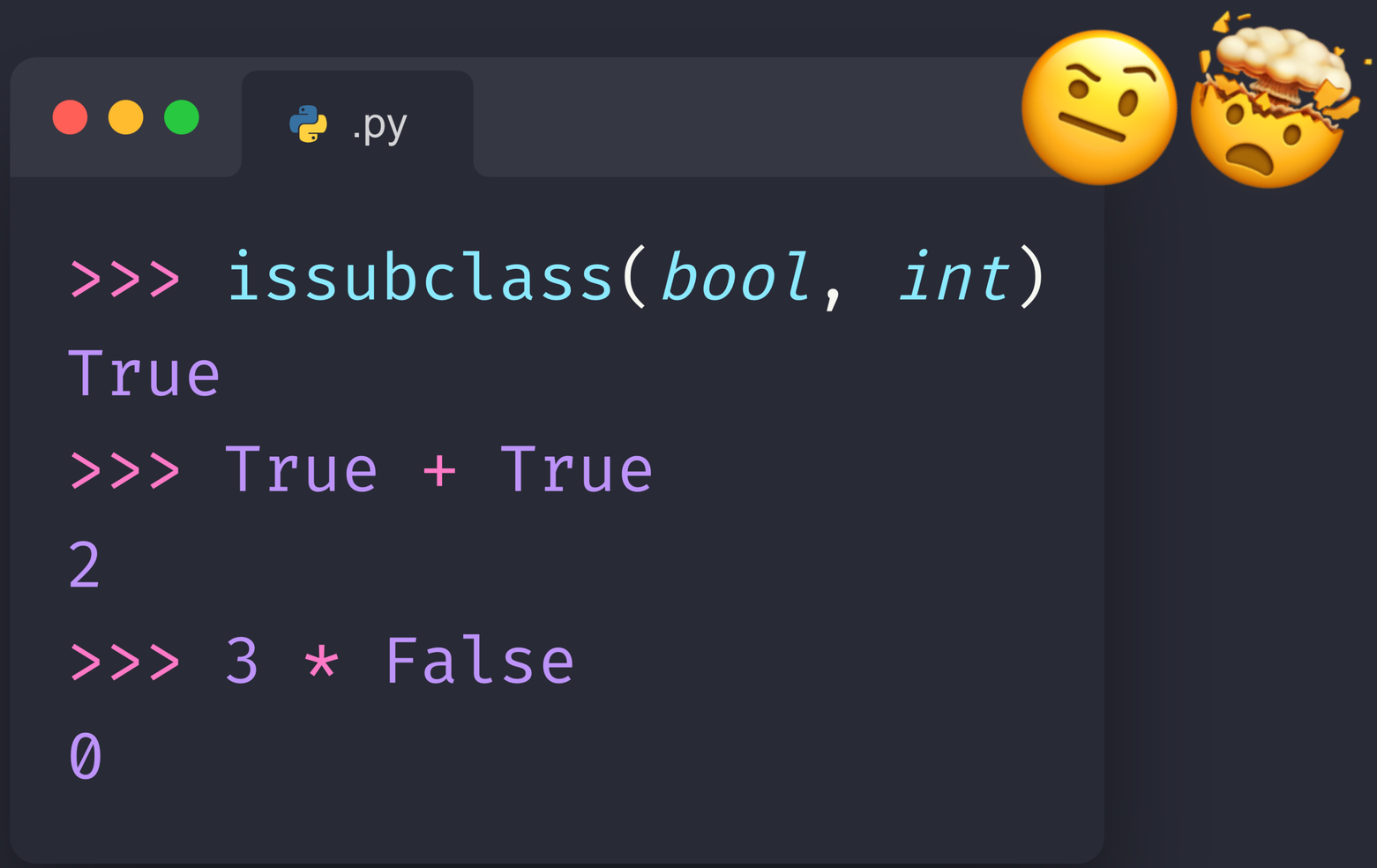
Data-driven conditionals

rodrigo@mathspp.com

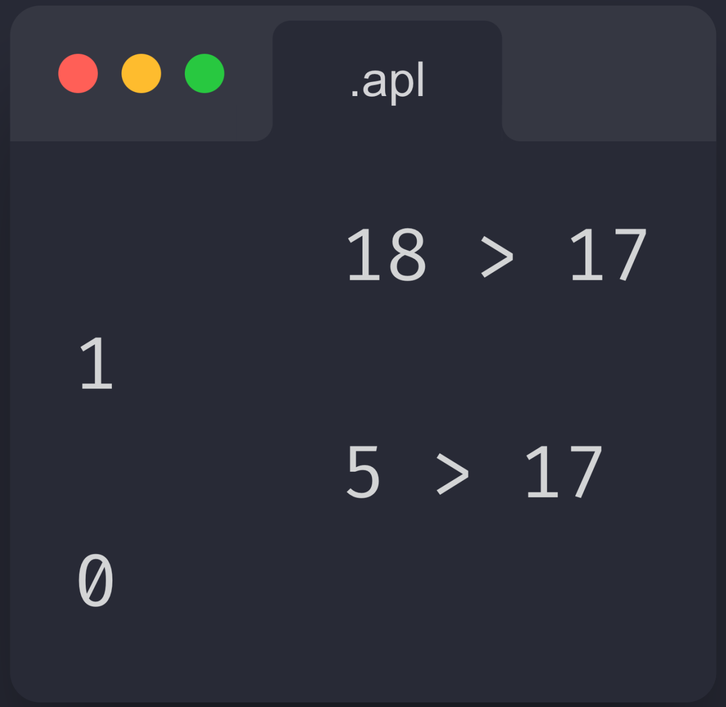
Booleans and 0/1



rodrigo@mathspp.com

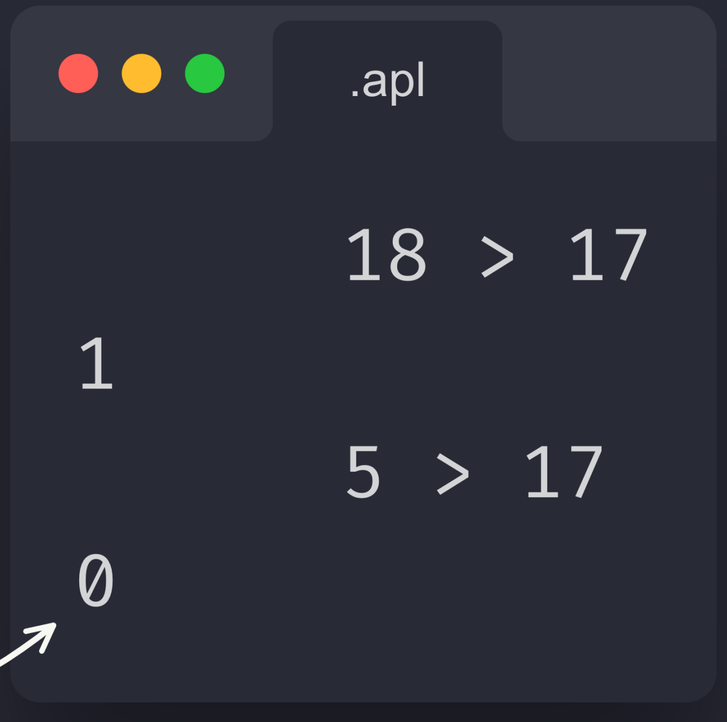


rodrigo@mathspp.com

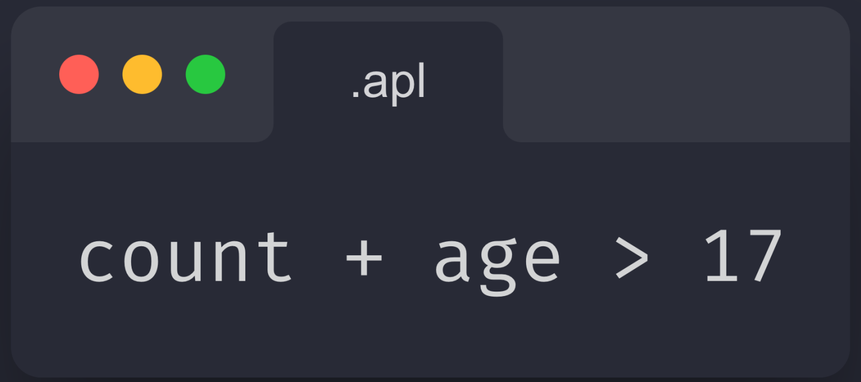


rodrigo@mathspp.com

APL doesn't have Boolean values



rodrigo@mathspp.com



rodrigo@mathspp.com

.apl

```
count + age > 17
```

.py

```python
count = 0
for age in ages:
    count += 1 if age > 17 else 0
```

rodrigo@mathspp.com

.apl

```
count + age > 17
```

.py

```
count = 0
for age in ages:
    count += age > 17
```

rodrigo@mathspp.com

.apl

```
count + age > 17
```

.py

```
count = 0
for age in ages:
    count += age > 17
```

Using the Boolean directly

rodrigo@mathspp.com

.apl

```
count + age > 17
```

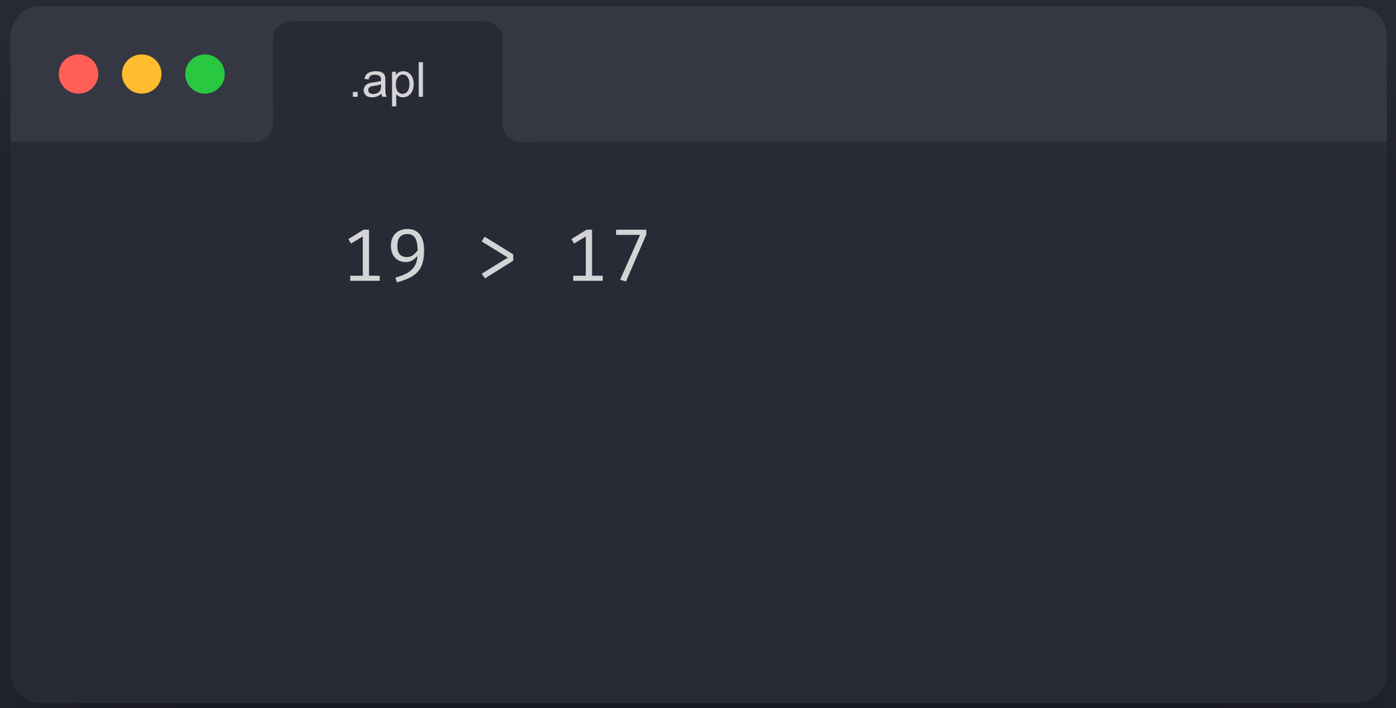
.py

```
count = 0
for age in ages:
    count += age > 17
```

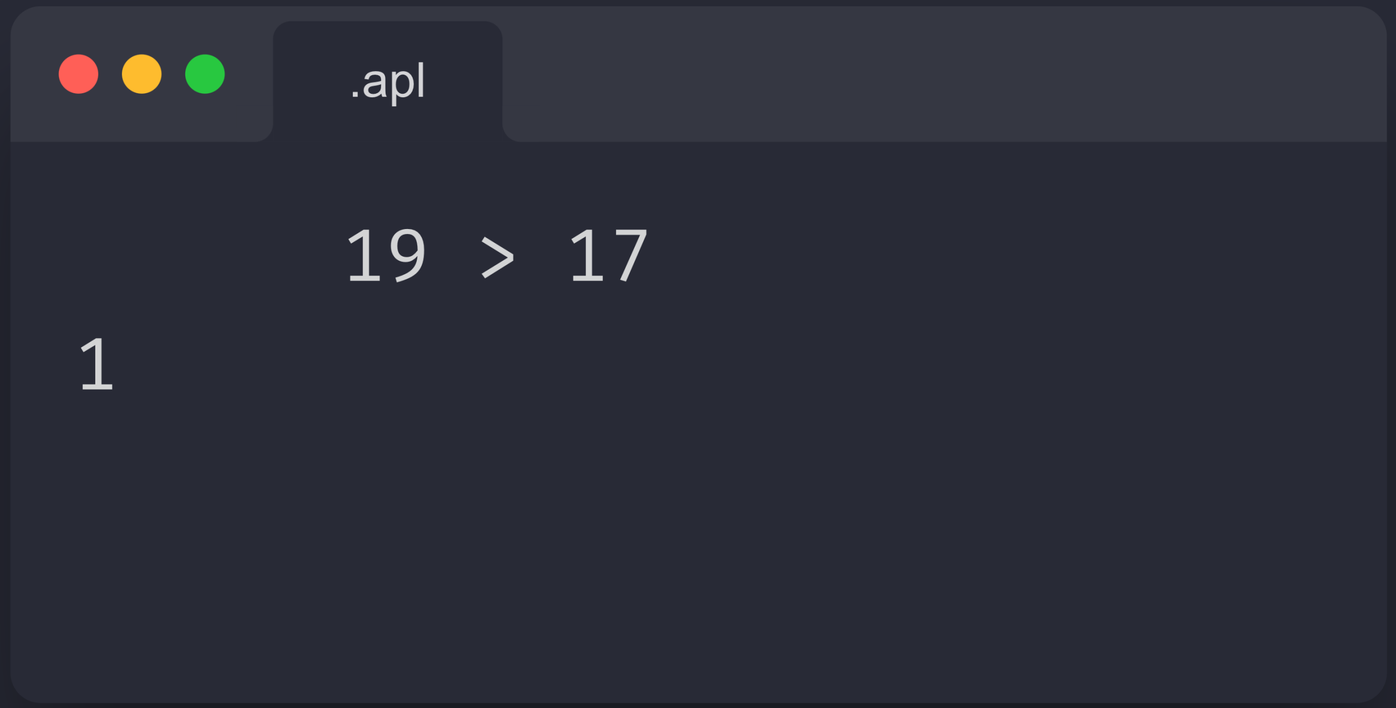
Scalar functions & list comprehensions

Using the Boolean directly

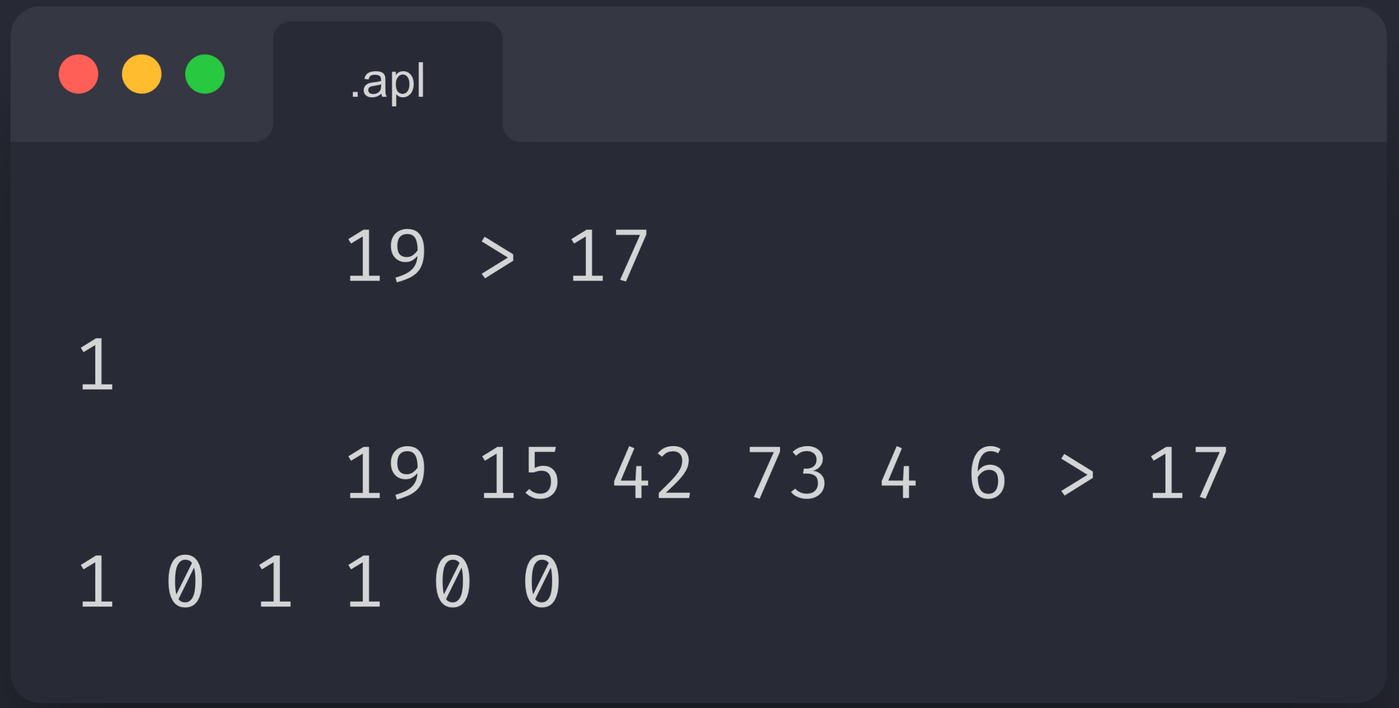
rodrigo@mathspp.com



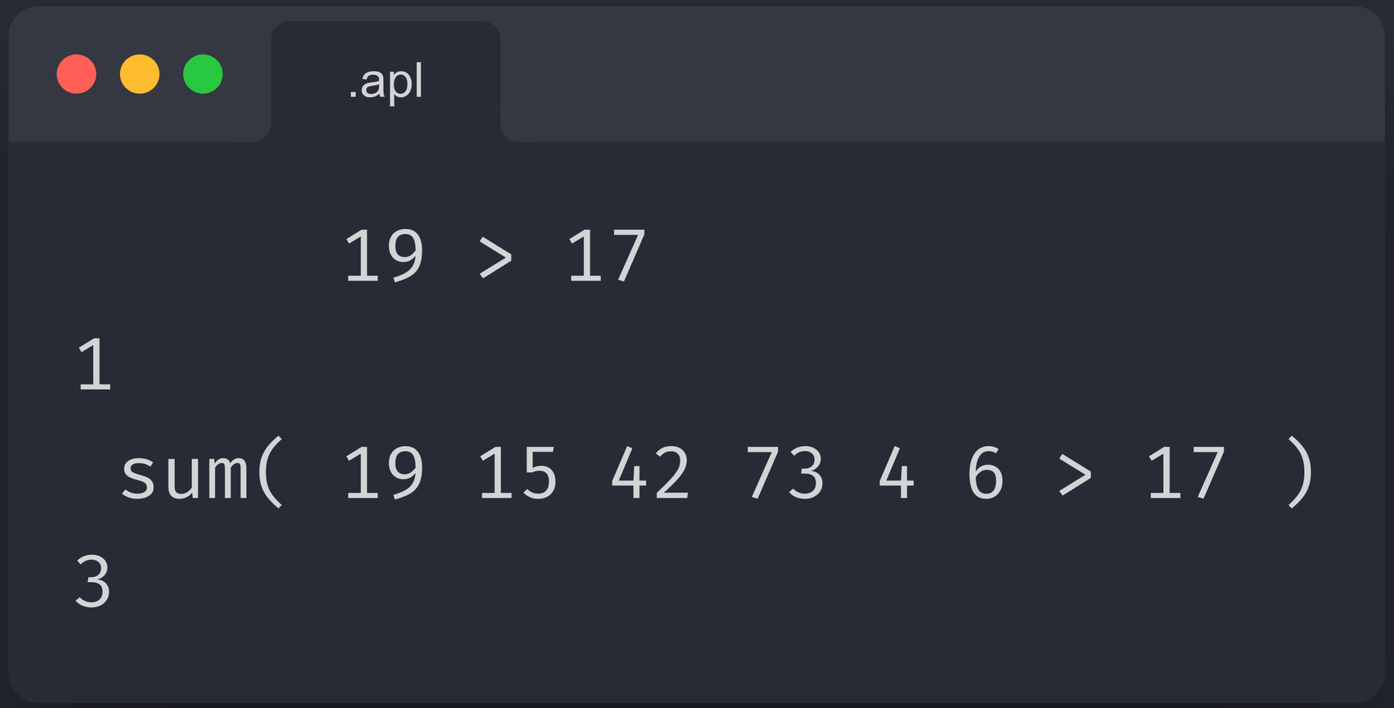
rodrigo@mathspp.com



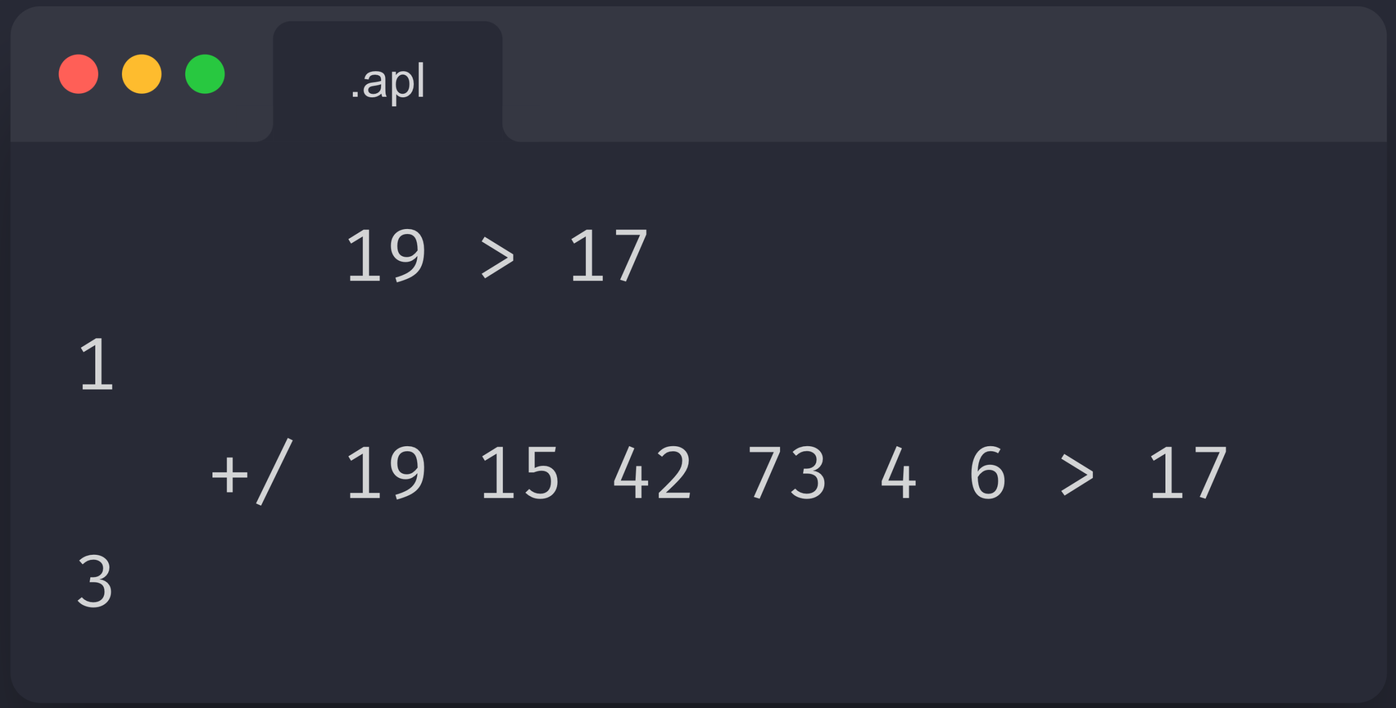
rodrigo@mathspp.com



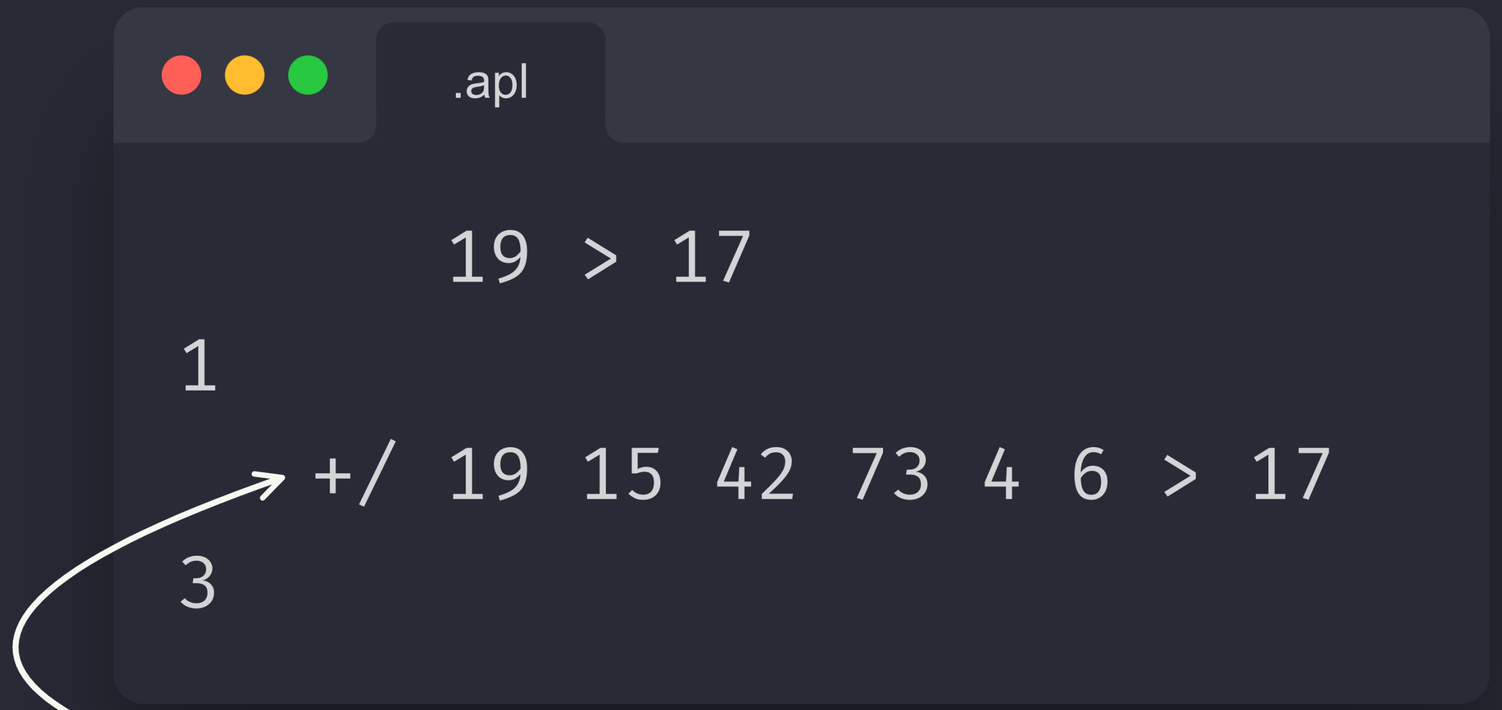
rodrigo@mathspp.com



rodrigo@mathspp.com

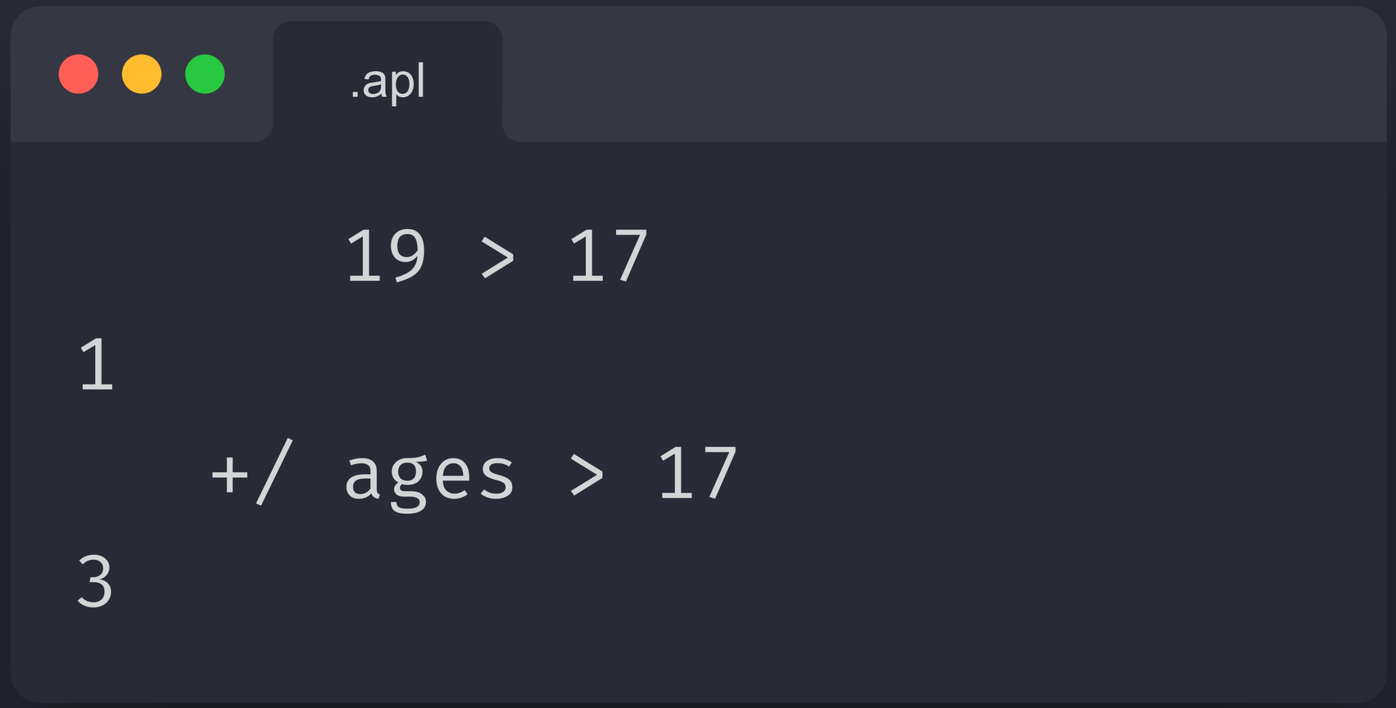


rodrigo@mathspp.com



APL way of
counting things

rodrigo@mathspp.com



rodrigo@mathspp.com

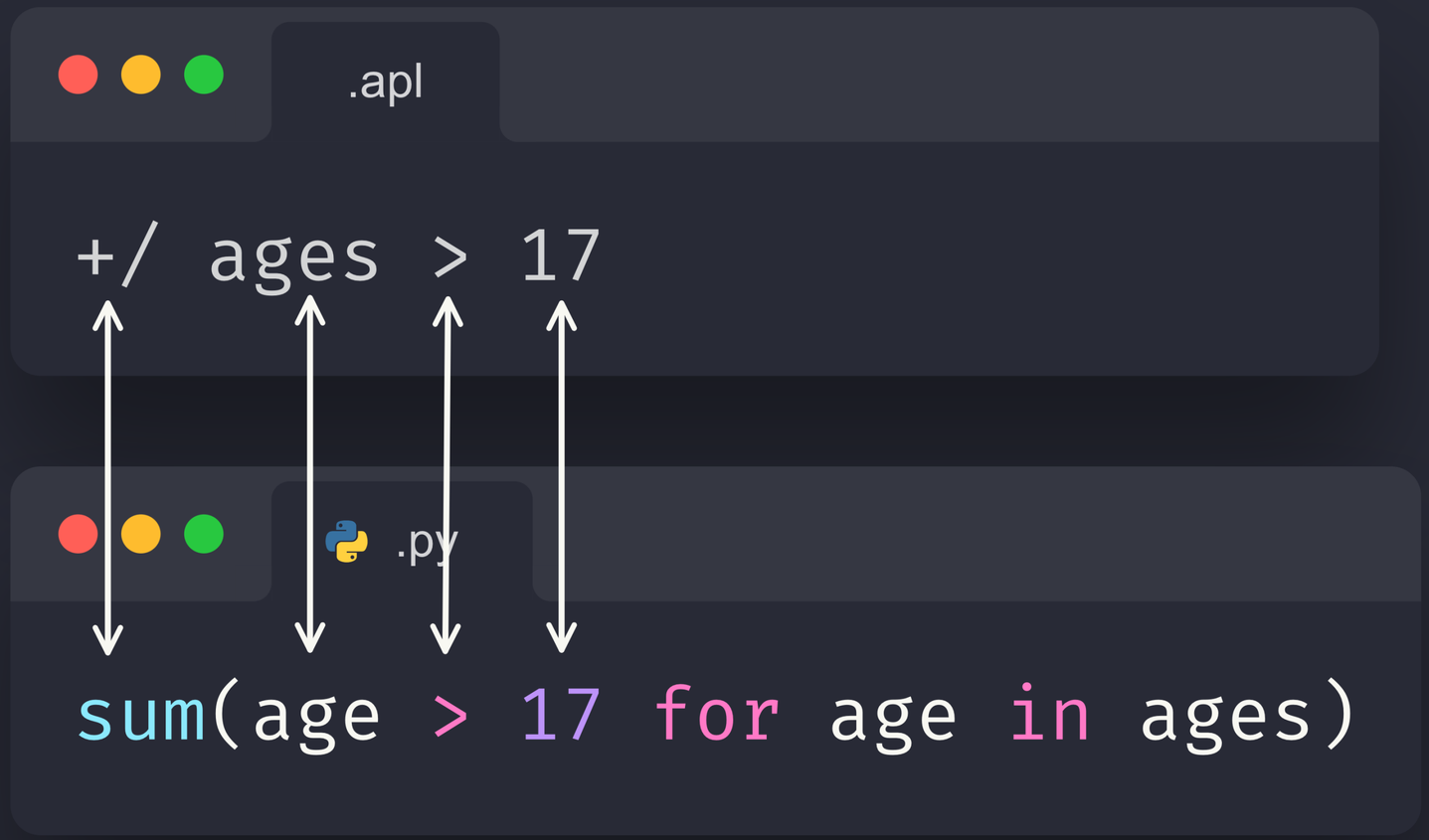
.apl

```
+/ ages > 17
```

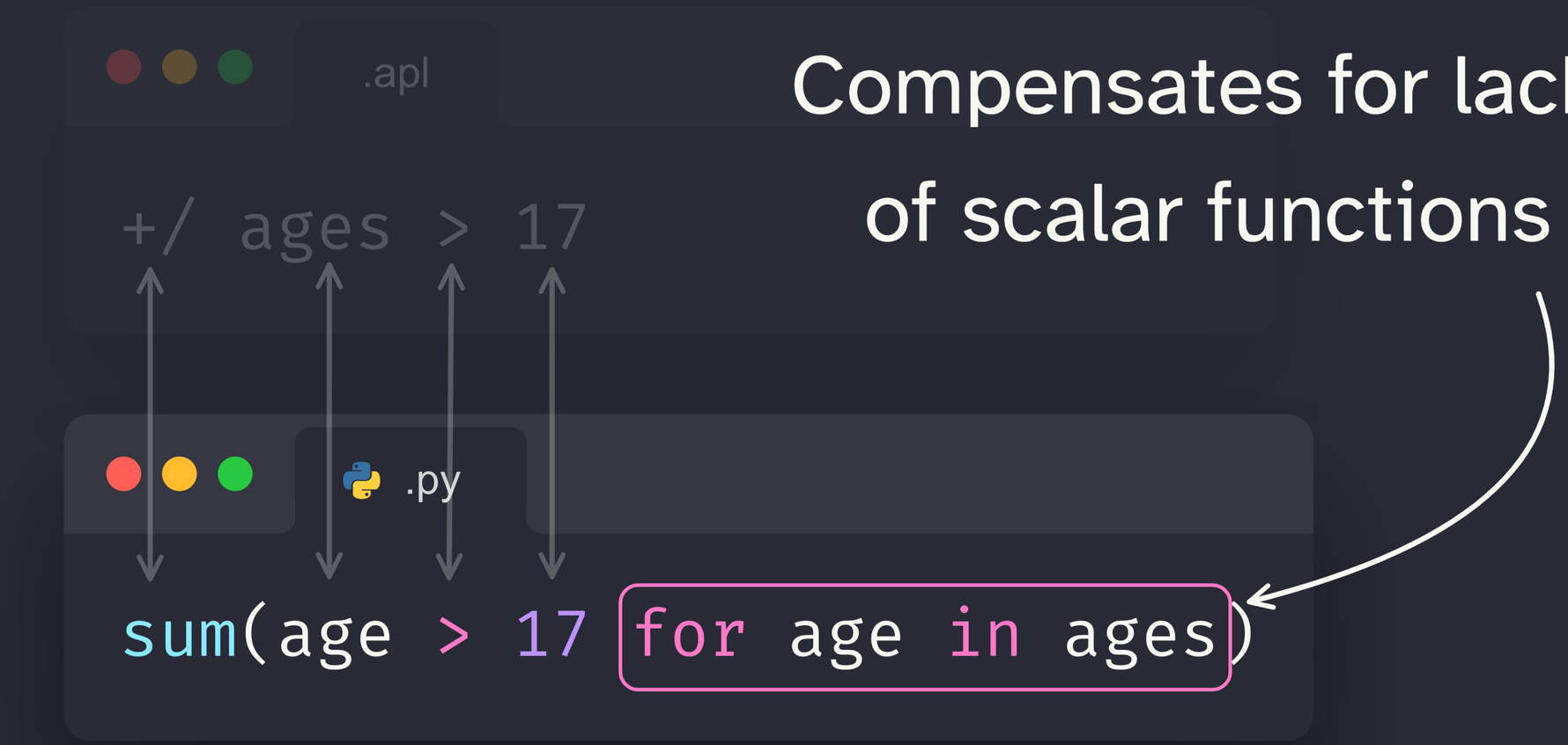
.py

```
sum(age > 17 for age in ages)
```

rodrigo@mathspp.com

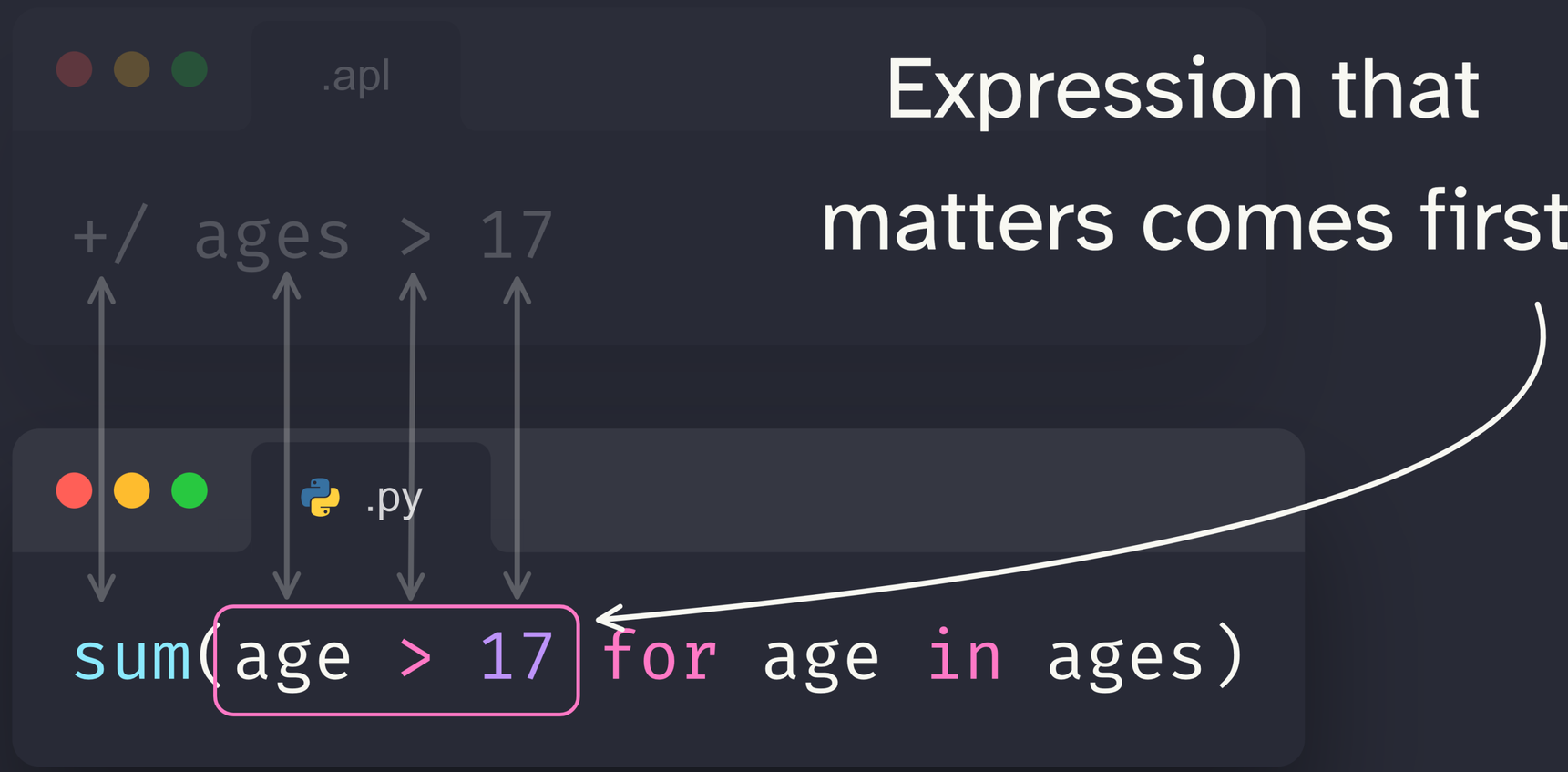


rodrigo@mathspp.com



Compensates for lack of scalar functions

rodrigo@mathspp.com



Expression that matters comes first

rodrigo@mathspp.com

Expression that matters comes first

.py

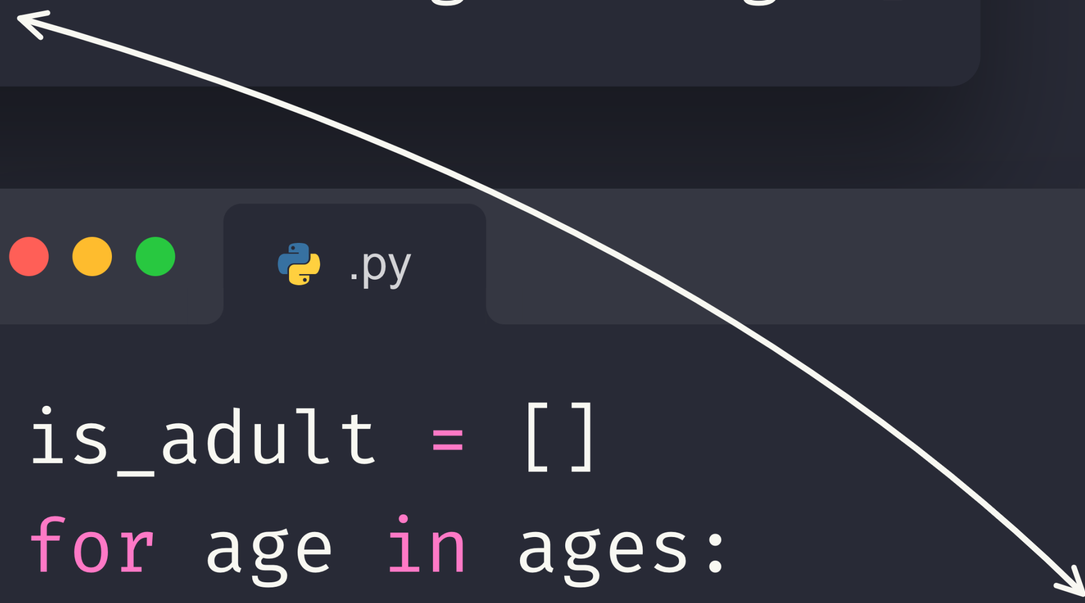
```
is_adult = [age > 17 for age in ages]
```

rodrigo@mathspp.com

Expression that matters comes first

.py

```
is_adult = [age > 17 for age in ages]
```



.py

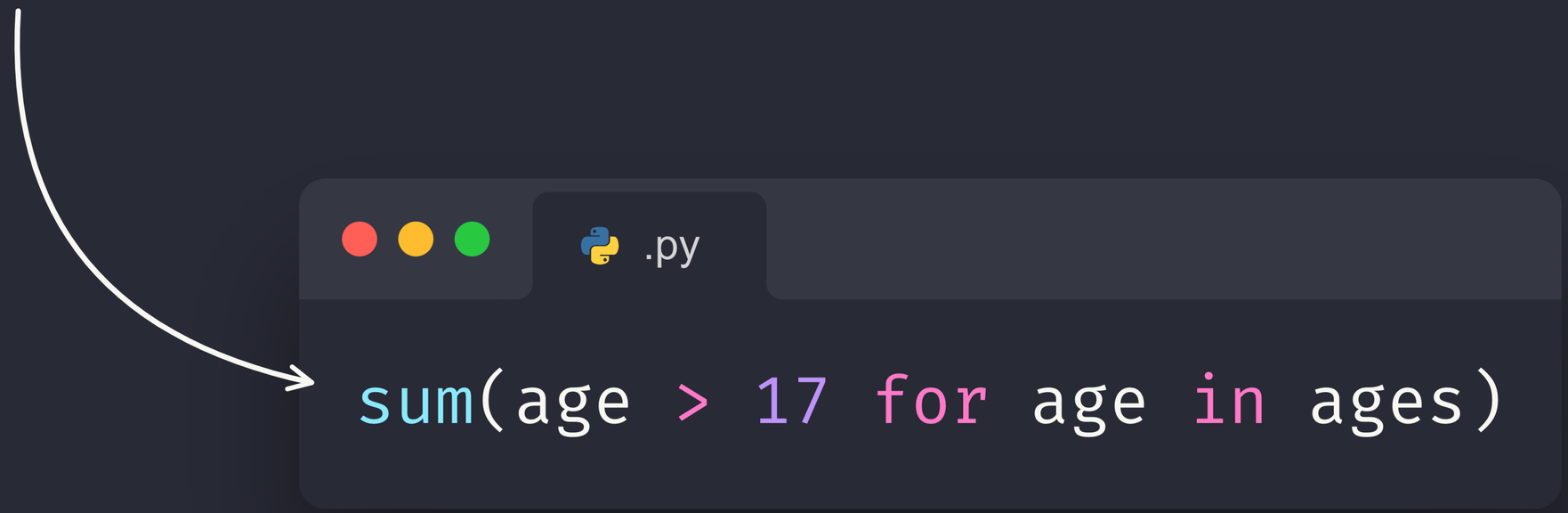
```
is_adult = []
for age in ages:
    is_adult.append(age > 17)
```

rodrigo@mathspp.com

rodrigo@mathspp.com

Recap

Reductions are everywhere



rodrigo@mathspp.com

Reductions are everywhere

Booleans, 0/1, and data-driven conditionals

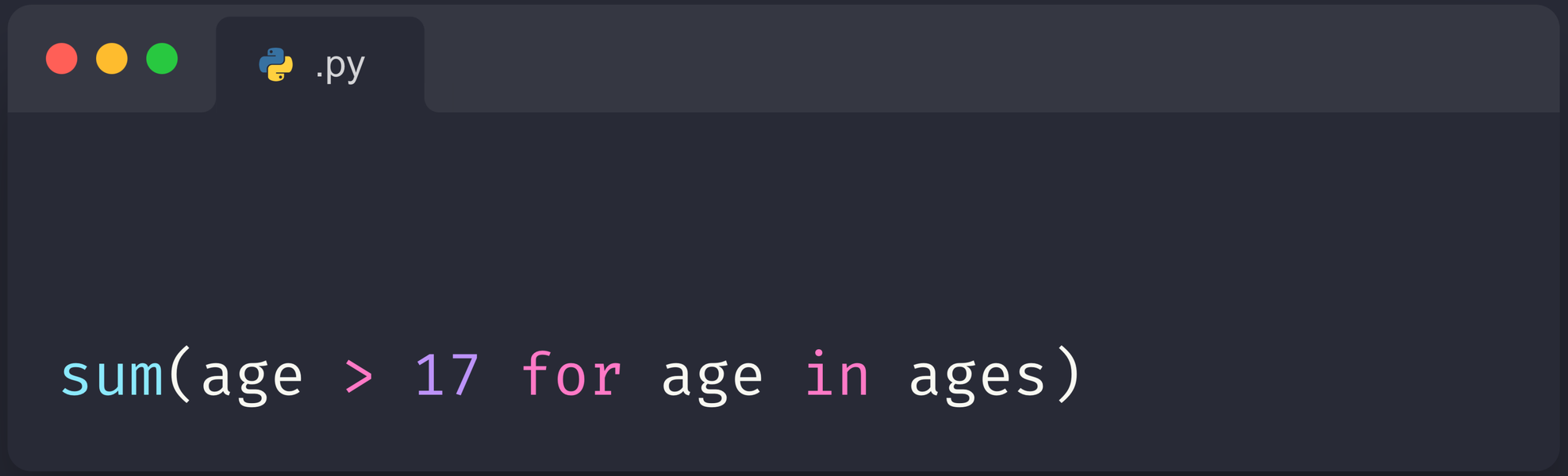
rodrigo@mathspp.com

Reductions are everywhere

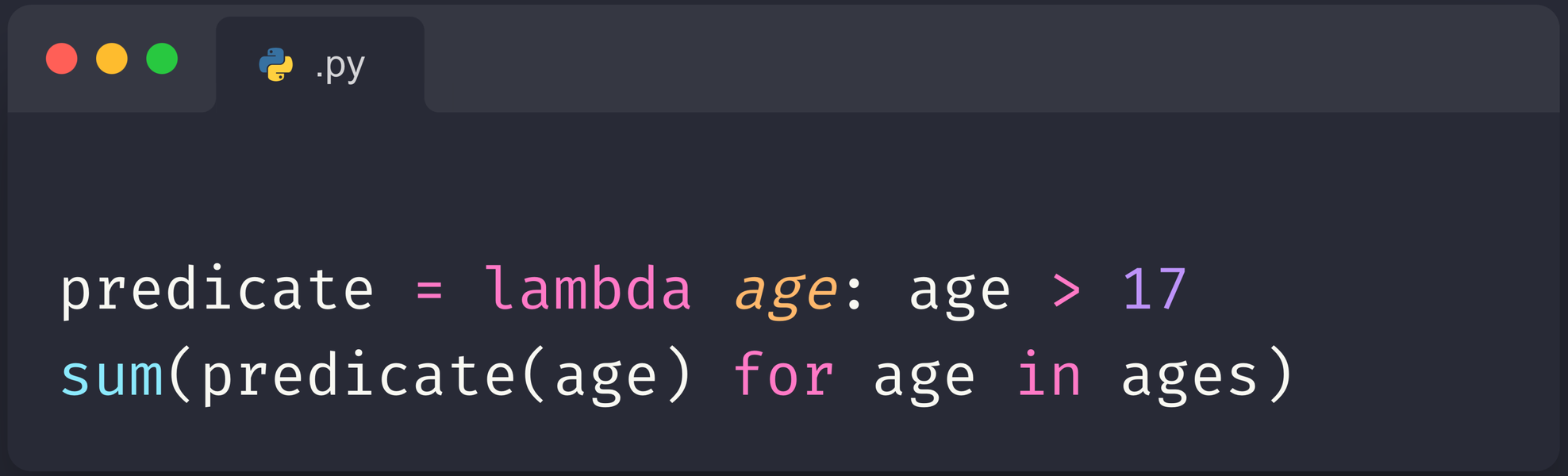
Scalar functions &
list comprehensions

Booleans, 0/1, and data-driven conditionals

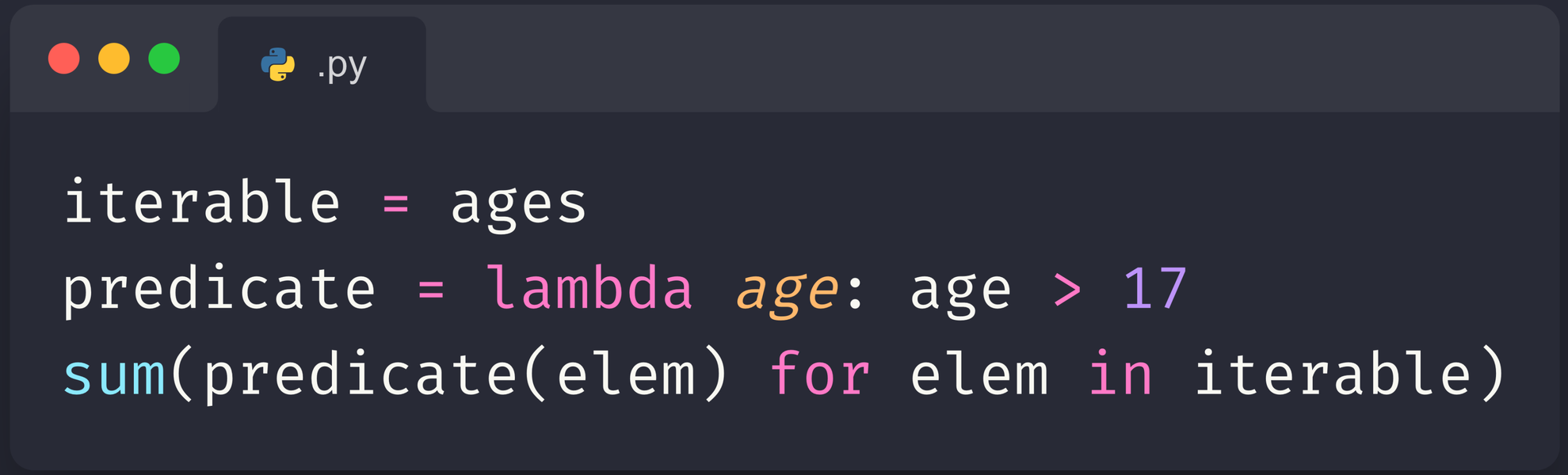
rodrigo@mathspp.com



rodrigo@mathspp.com



rodrigo@mathspp.com



rodrigo@mathspp.com

.py

```
iterable = ages
predicate = lambda age: age > 17
sum(predicate(elem) for elem in iterable)
```

Idiom: count elements that satisfy `predicate`.

rodrigo@mathspp.com

“[…] Accumulate idioms. […] The only difference between Shakespeare and you was the size of his idiom list […]”

— Alan J. Perlis

rodrigo@mathspp.com

presented with  snappify*

*no affiliation, they're just awesome

rodrigo@mathspp.com

rodrigo@mathspp.com

rodrigo@mathspp.com

mathspp.com/talks