

# How APL made me a better Python developer

by Rodrigo Girão Serrão



Rodrigo 🐍🚀  
🐦 @mathsppblog



Rodrigo 🐍🚀

🐦 @mathsppblog

Textualize  
[mathspp.com](http://mathspp.com)

[mathspp.com/talks](https://mathspp.com/talks)

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

— Alan J. Perlis

# What's APL?

# APL = A Programming Language

APL = A Programming Language  
Terrible name, right?

APL = A Programming Language

Terrible name, right?

Not really, it isn't a pun.

# Game of life



.apl

{↑1 ωv.∧3 4=+/ , ¯1 0 1∘.Θ¯1 0 1∘.φ⊂ω}

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.

Stating the obvious can be good.  
Example: pigeonhole principle.

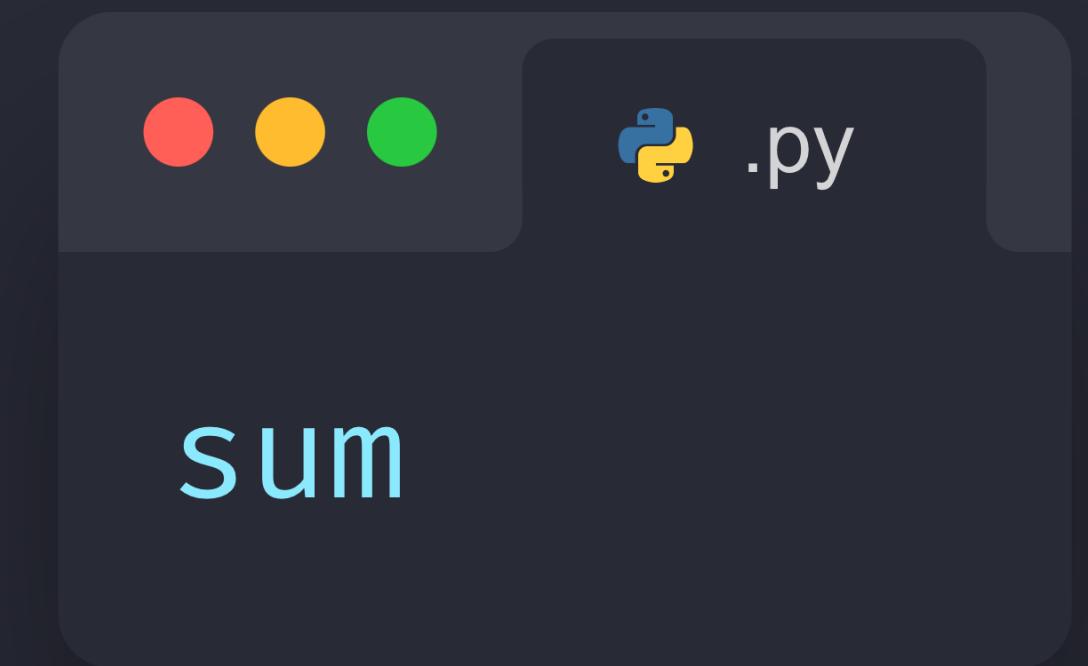
The LOC that  
changed everything



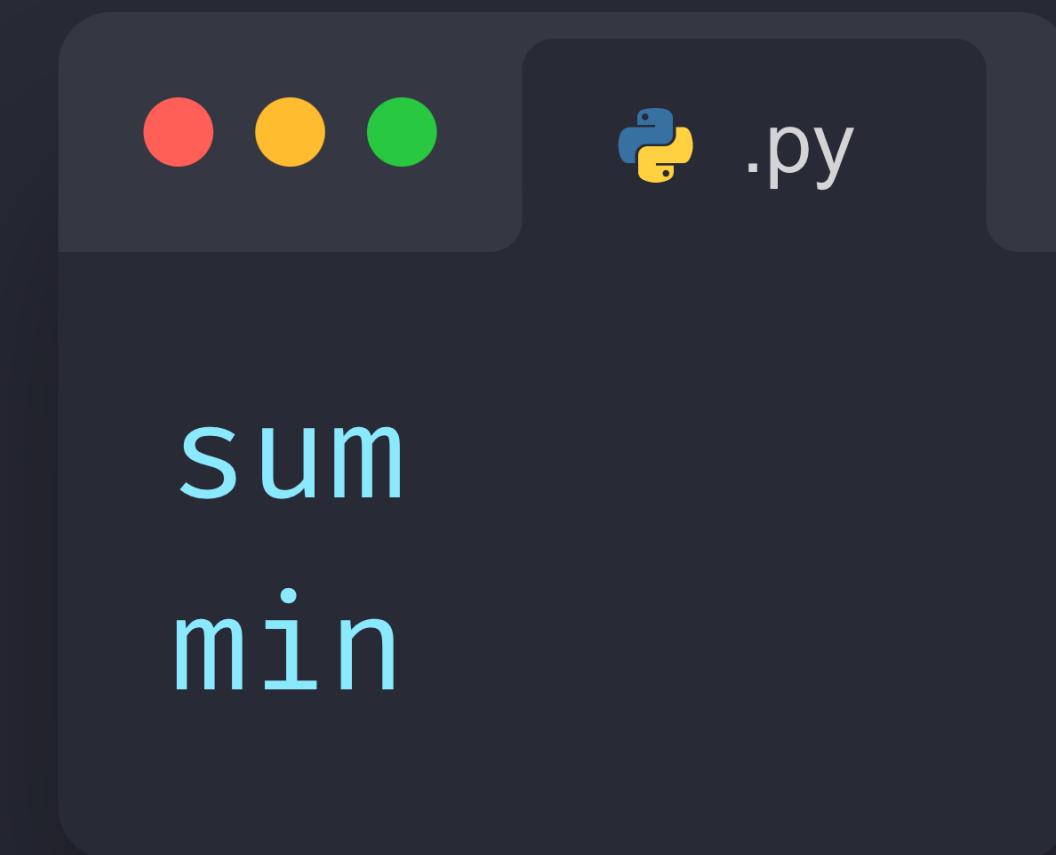
python .py

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

\*\*\*\*\* are  
everywhere



rodrigo@mathspp.com





python .py

sum

min

all

" ".join

math.prod

any

max



python .py

sum

math.prod

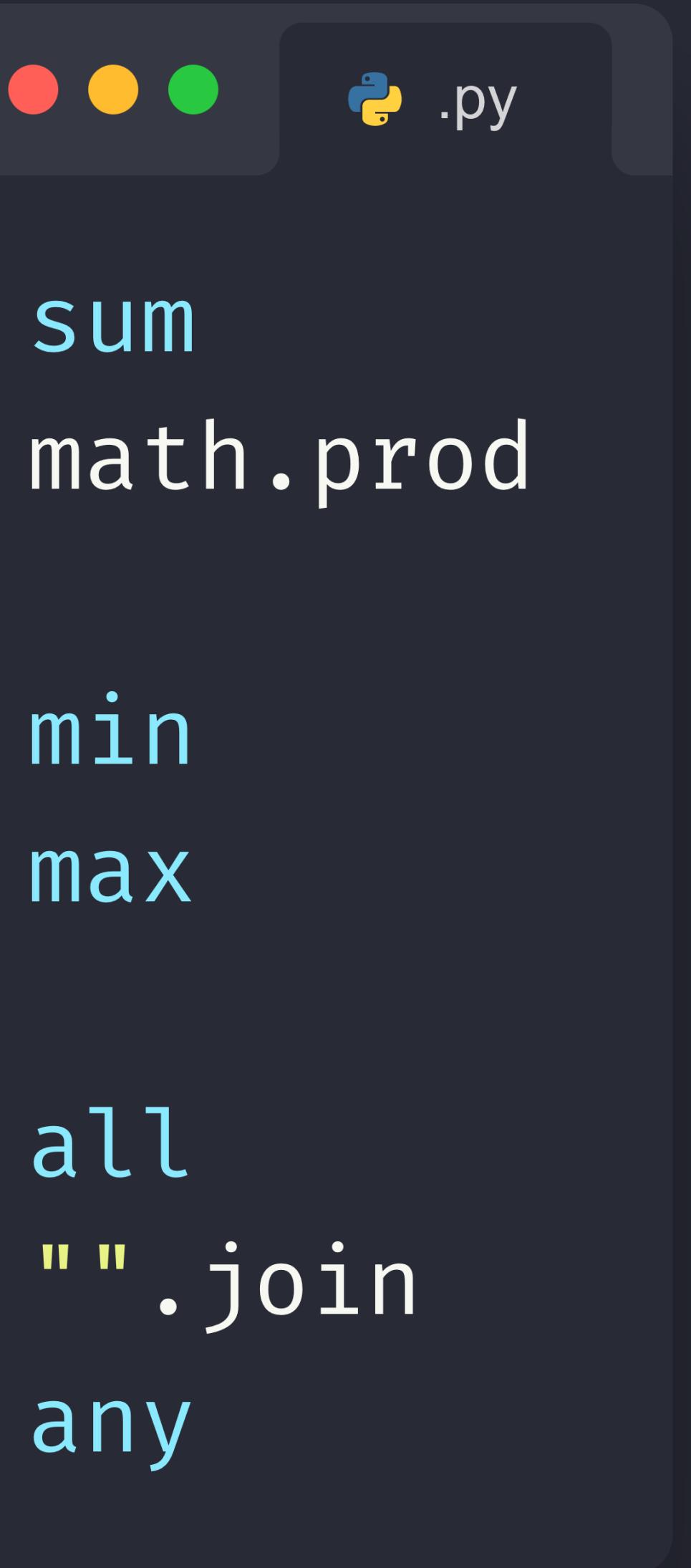
min

all

" ".join

any

max





python .py

sum

math.prod

min

max

all

any

" ".join



python .py

sum

math.prod

min

max

all

any

" ".join



python .py

sum

math.prod

min

max

all

any

" ".join



.apl

+ /



python .py

sum

math.prod

min

max

all

any

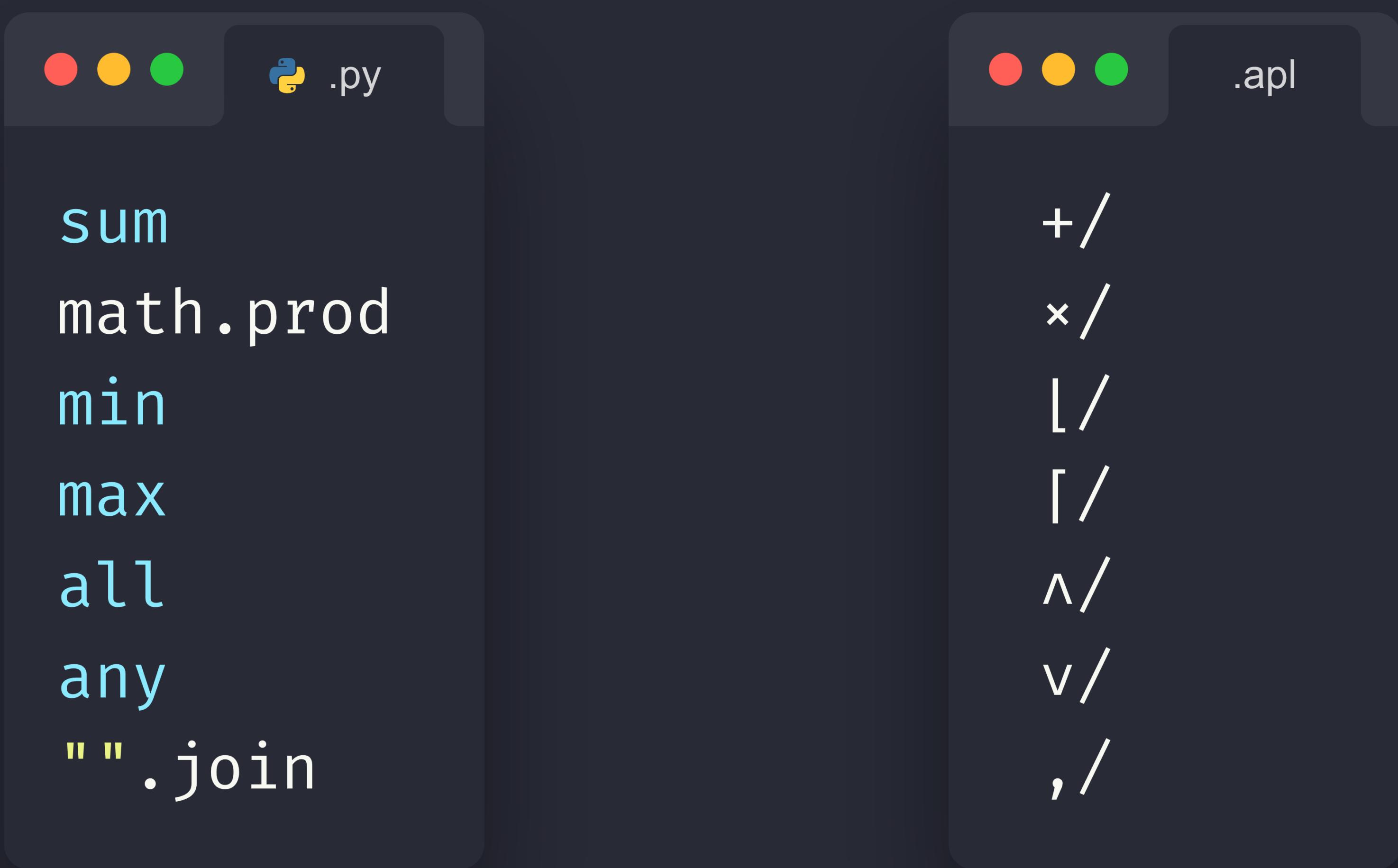
" ".join



.apl

+ /

× /





.py

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



.apl

+ /  
× /  
L /  
T /  
^ /  
∨ /  
, /



.py

sum

math.prod

min

max

all

any

" ".join



.apl

+ /

× /

⌊ /

⌈ /

⌃ /

⌄ /

, /

These are all the same thing!



.py

sum

math.prod

min

max

all

any

" ".join



.apl

+ /

× /

⌊ /

⌈ /

^ /

∨ /

, /

functools.reduce

# Data-driven conditionals



python .py

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



python .py

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



python .py

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



*Do we add?* 🤔



🐍 .py

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



python .py

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

“Useless” but correct



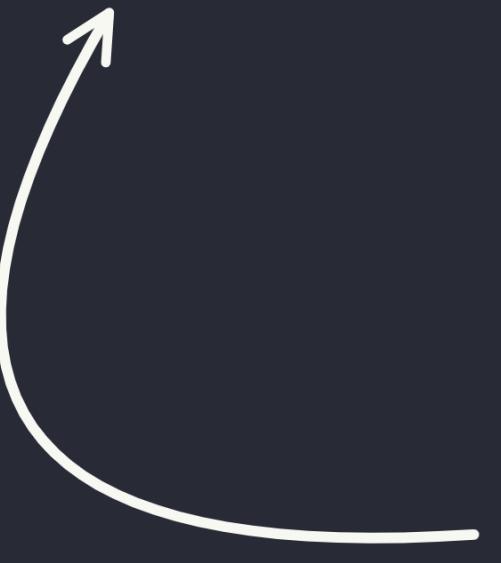
🐍 .py

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



python .py

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



*What do we add?* 🤔



python .py

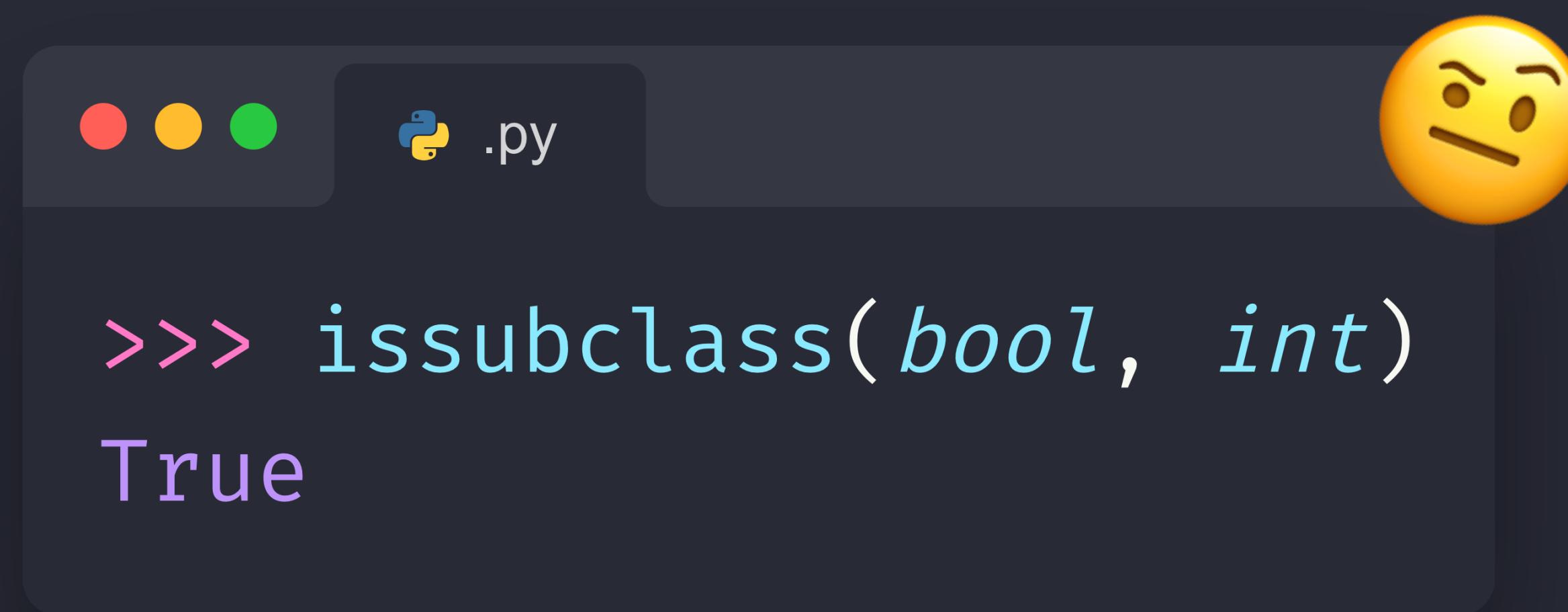
```
count = 0
for age in ages:
    count += 1 if age > 17 else 0
```



What do we add? 🤔

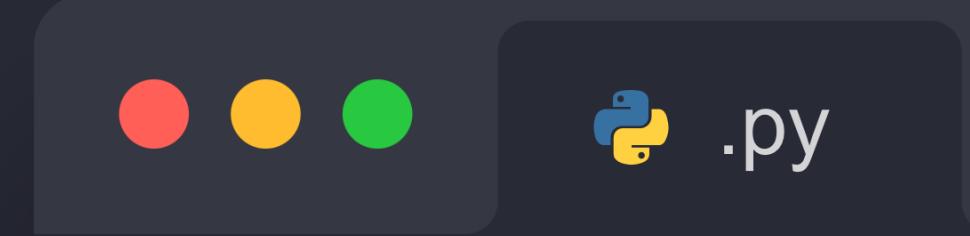
## Data-driven conditionals

# Booleans and 0/1



A screenshot of a terminal window with a dark background. At the top, there are three colored dots (red, yellow, green) and a Python logo icon followed by '.py'. To the right of the Python icon is a large, yellow, angry face emoji. The main area of the terminal shows the following text:

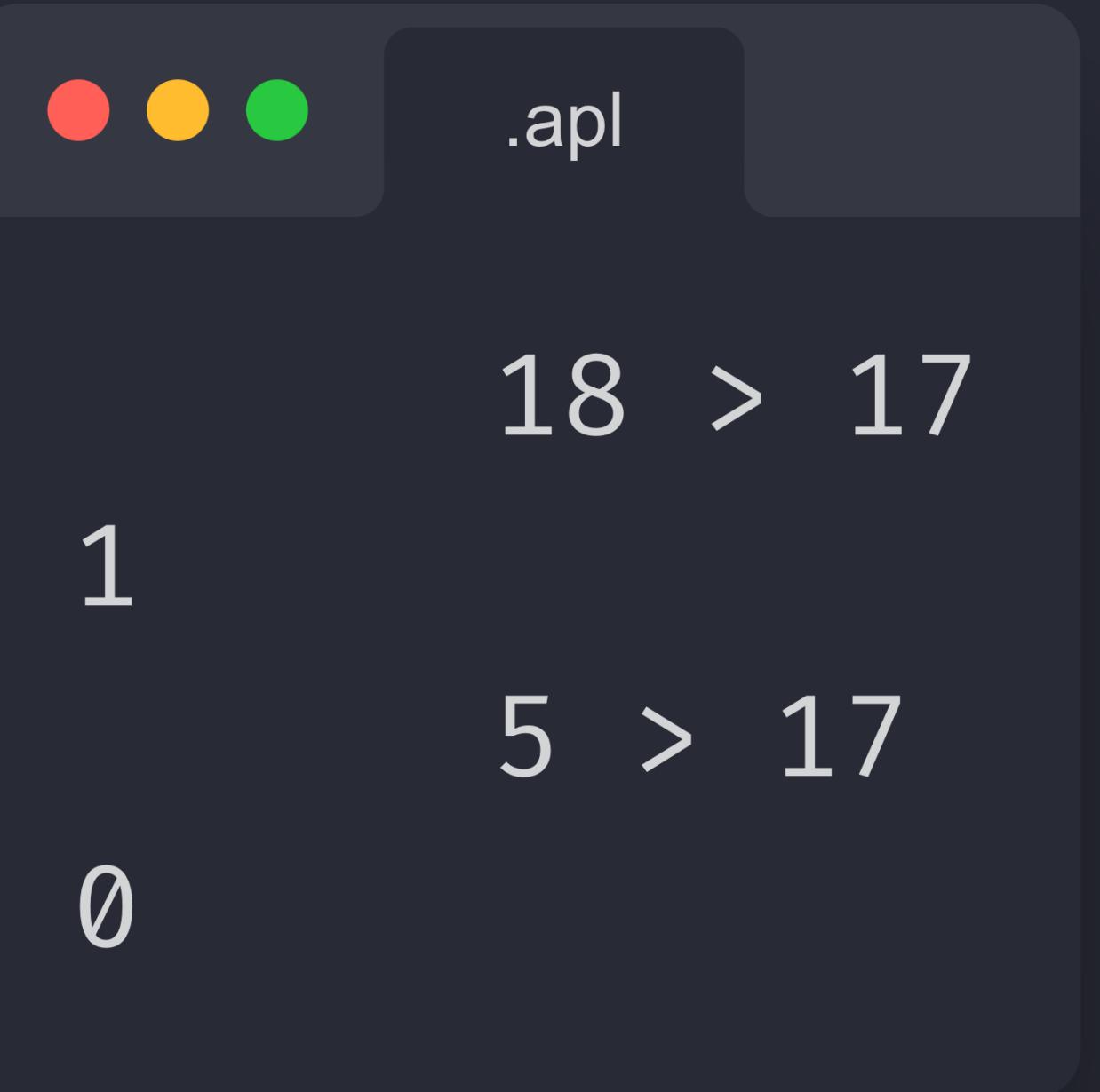
```
>>> issubclass(bool, int)
True
```



```
>>> issubclass(bool, int)
True

>>> True + True
2

>>> 3 * False
0
```



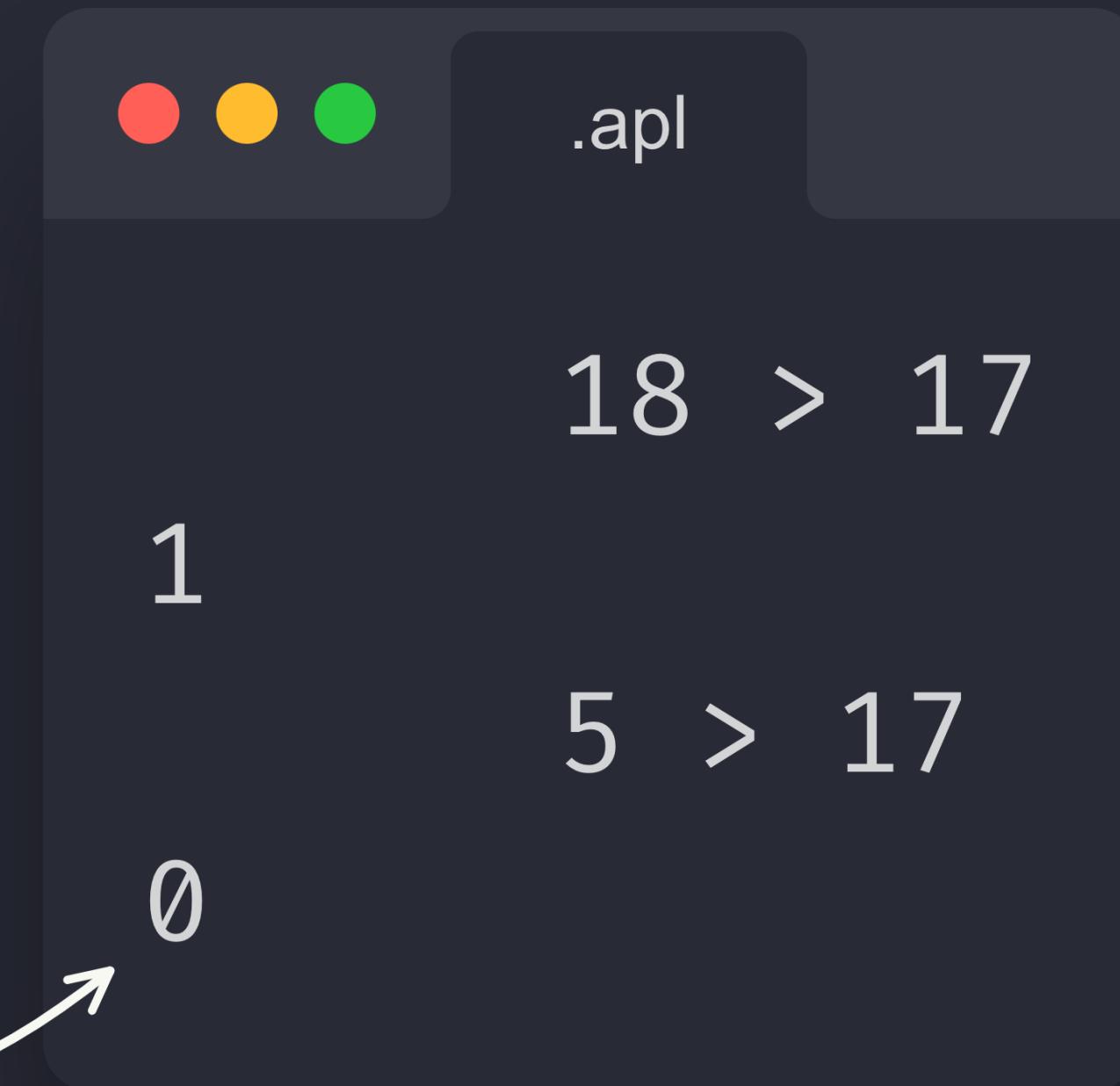
18 > 17

1

5 > 17

0

APL doesn't  
have Boolean  
values



.apl

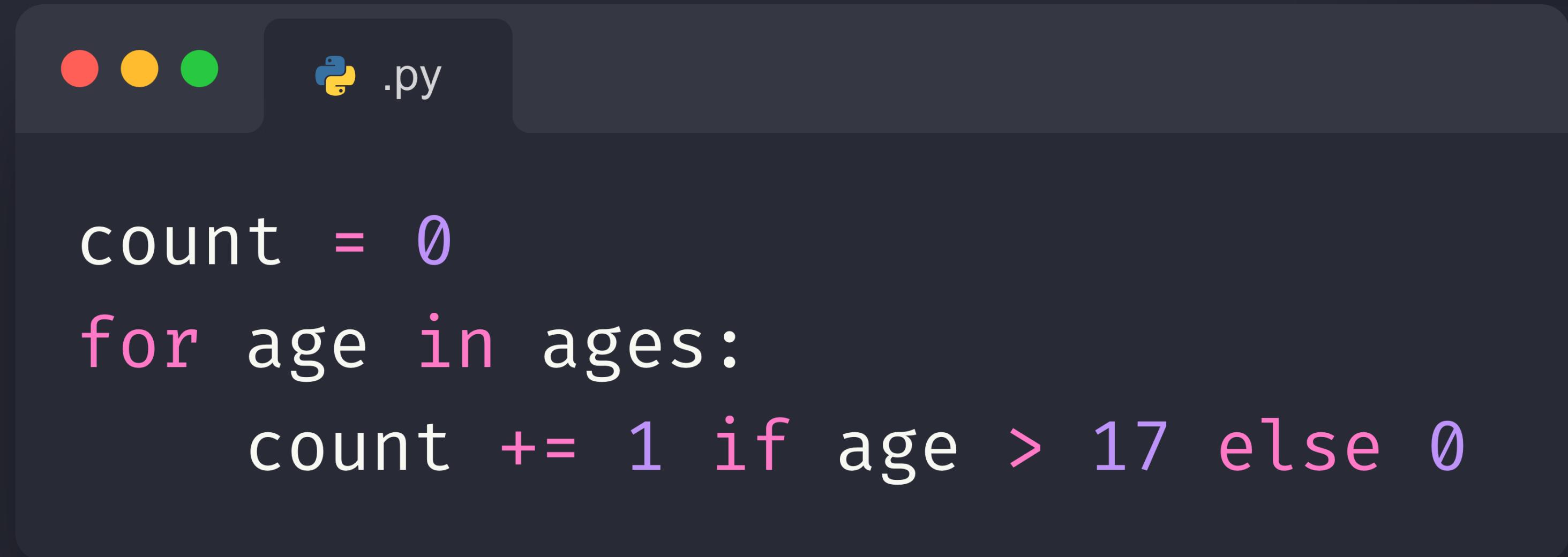
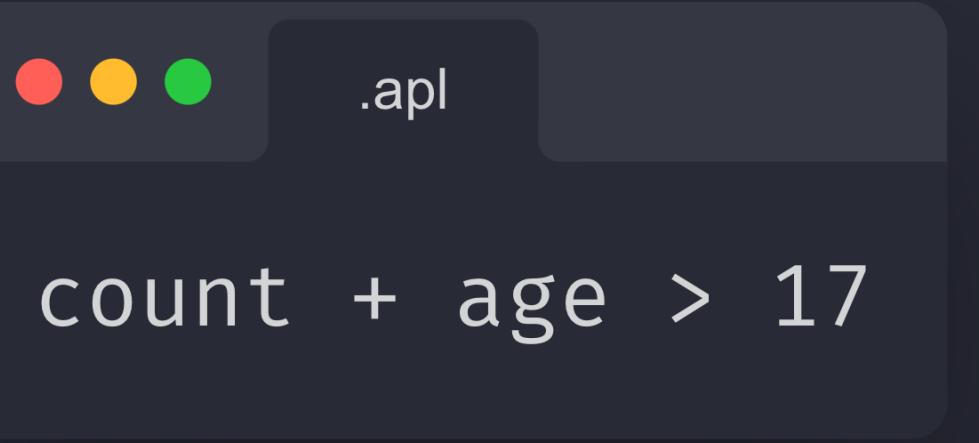
18 > 17

1

5 > 17

0





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

A screenshot of a dark-themed code editor window. The title bar shows three colored dots (red, yellow, green) and the file extension '.py'. The main editor area contains the following Python code:

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



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

Using the  
Boolean directly

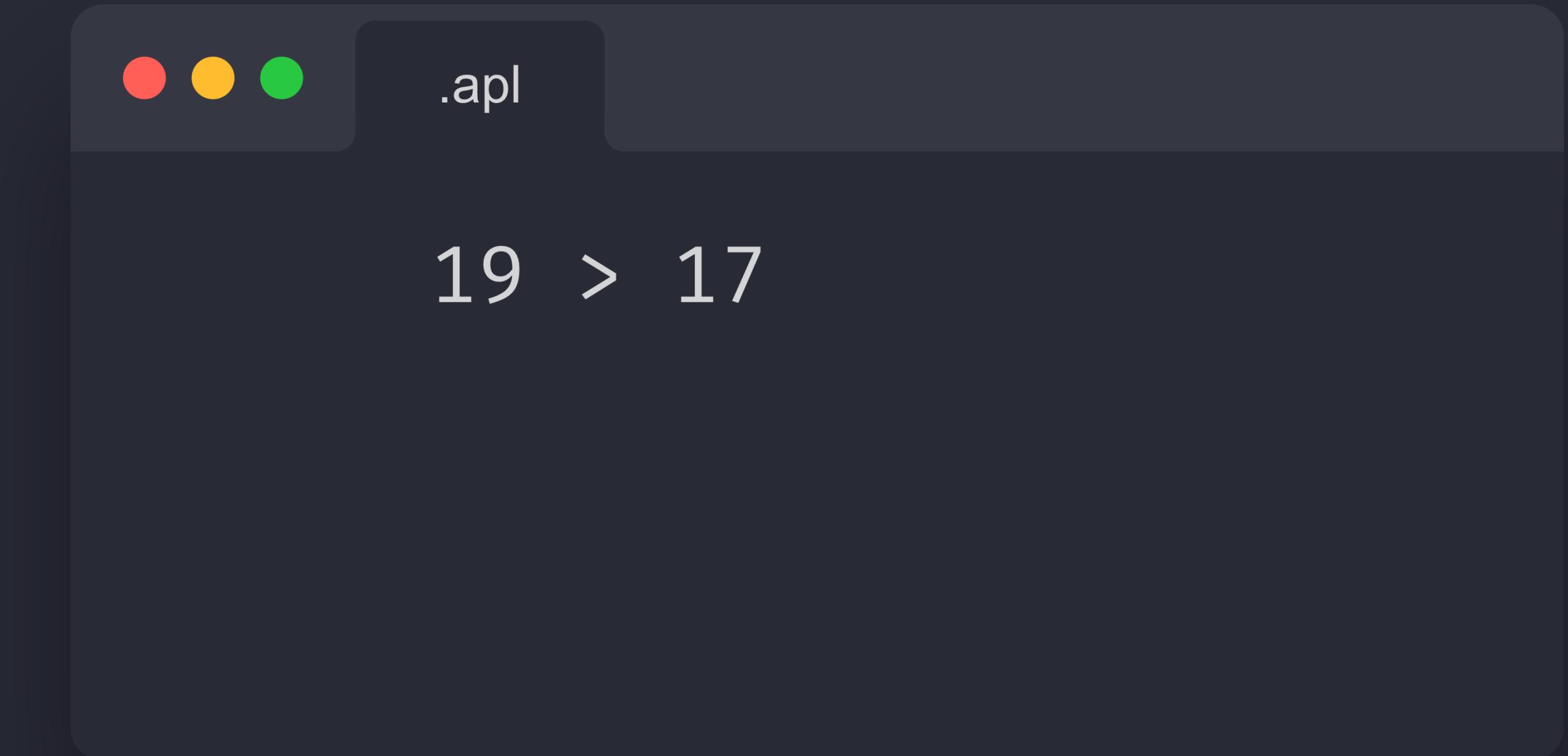
# Scalar functions & list comprehensions

Using the  
Boolean directly

scalar.py  
count = 0

for age in ages:  
 count += age > 17

.apl  
count + age > 17





.apl

19 > 17

1



.apl

19 > 17

1

19 15 42 73 4 6 > 17

1 0 1 1 0 0



.apl

19 > 17

1

sum( 19 15 42 73 4 6 > 17 )

3



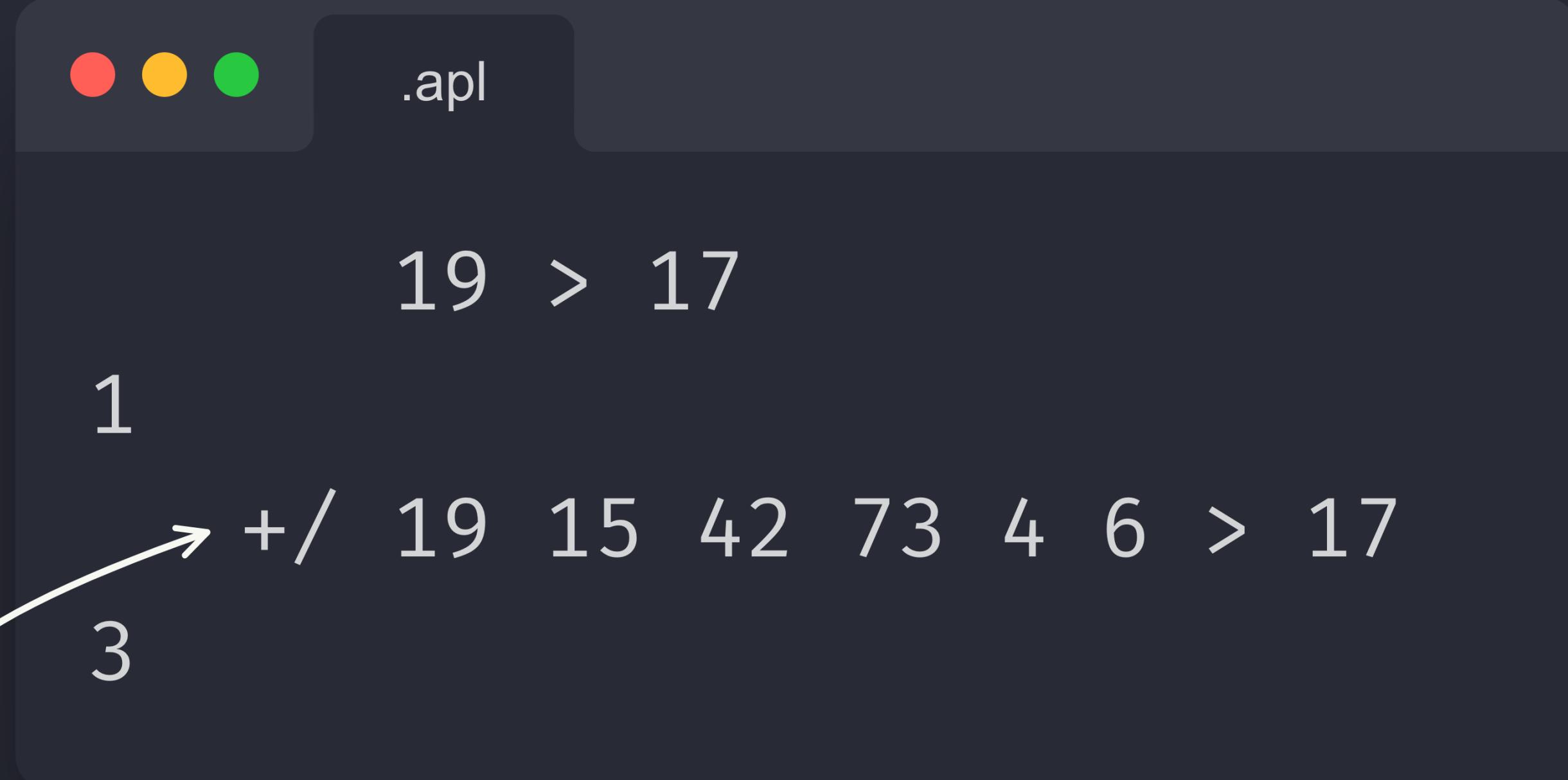
.apl

19 > 17

1

+/ 19 15 42 73 4 6 > 17

3



A screenshot of an APL session window. The title bar shows three colored dots (red, yellow, green) and the file extension '.apl'. The main area displays the following APL code:

```
19 > 17  
1  
→ +/ 19 15 42 73 4 6 > 17
```

The number '1' is positioned above the first line of code. An arrow points from the number '3' to the plus sign '+' in the second line of code.

APL way of  
counting things



.apl

19 > 17

1

+/ ages > 17

3



.apl

+ / ages > 17



 .py

sum(age > 17 for age in ages)



.apl

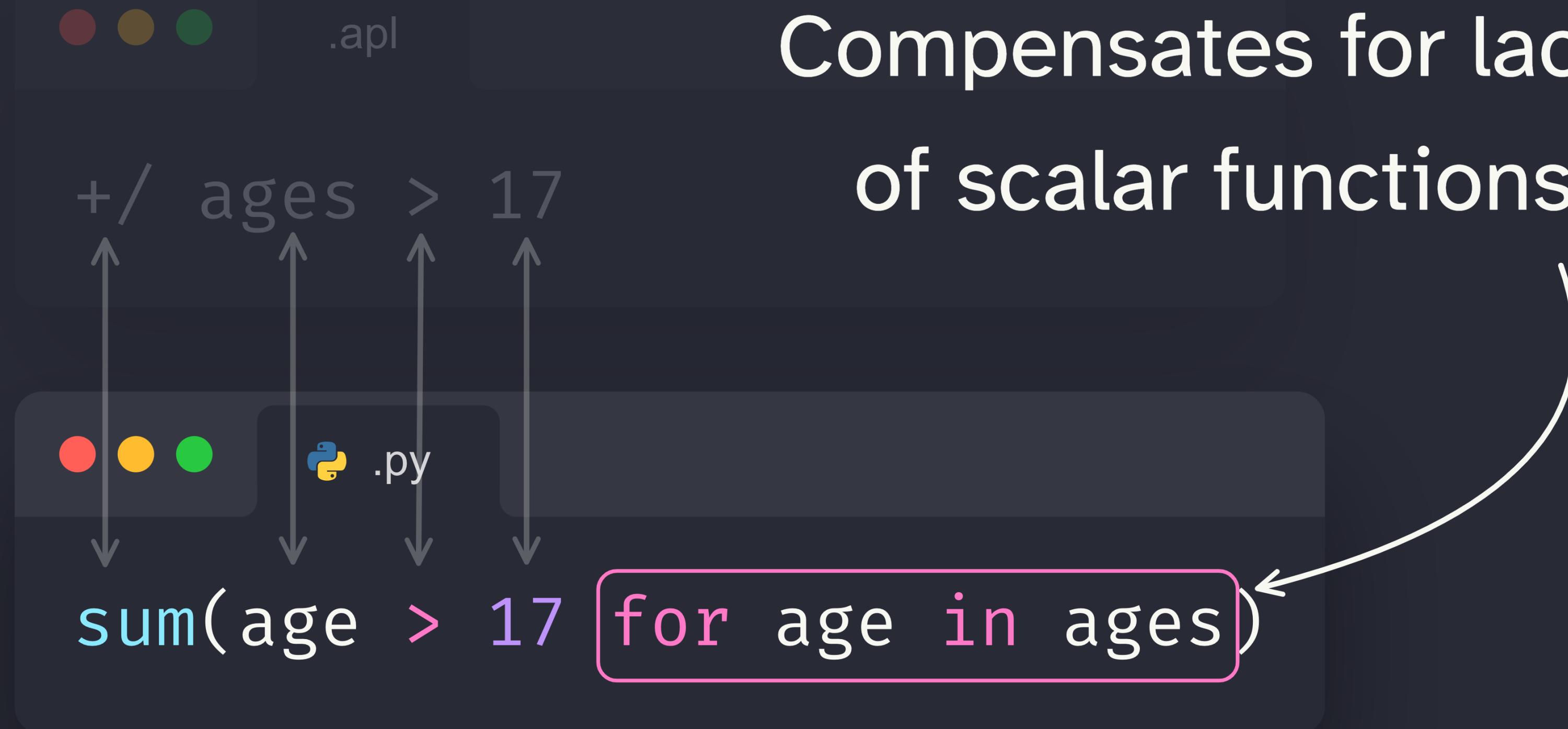
+/ ages > 17



.py

sum(age > 17 for age in ages)

Compensates for lack  
of scalar functions



# Expression that matters comes first

The diagram illustrates the execution flow of a Python code snippet. At the top, three colored dots (red, yellow, and green) are shown above a dark gray rounded rectangle. Inside the rectangle, there is a Python logo icon and the suffix '.py'. Four arrows point downwards from the text 'ages > 17' at the top left to the 'sum' function call at the bottom. A bracket on the right side of the 'sum' call groups the condition 'age > 17'. A white arrow points from the word 'for' to the opening parenthesis of the 'for' loop.

```
+/\ ages > 17
sum(age > 17 for age in ages)
```

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)
```



A screenshot of a dark-themed code editor window. At the top left, there are three small colored circles (red, yellow, green) followed by a Python logo icon and the file extension ".py". The main area contains the following Python code:

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

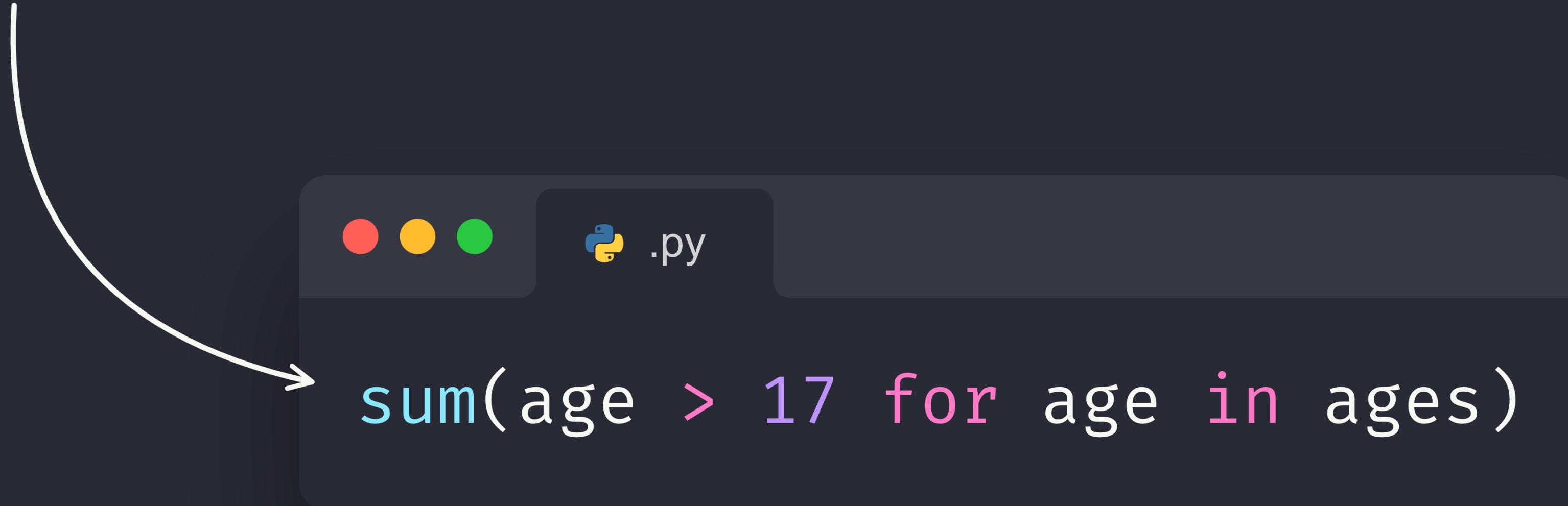


```
● ● ● .py
```

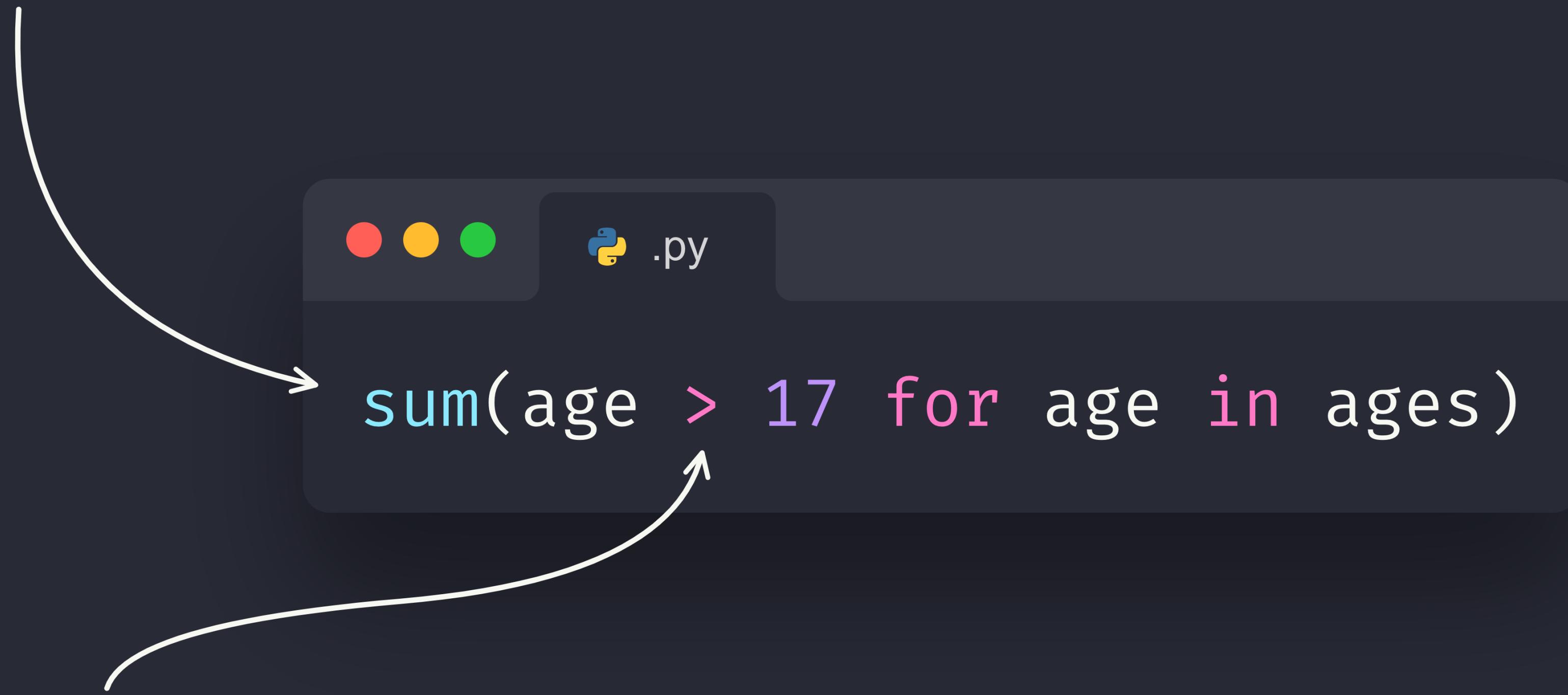
sum(age > 17 for age in ages)

# Recap

# Reductions are everywhere



# Reductions are everywhere



Booleans, 0/1, and data-driven conditionals

# Reductions are everywhere

# Scalar functions & list comprehensions



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

Booleans, 0/1, and data-driven conditionals



A screenshot of a dark-themed code editor window. In the top left corner, there are three small colored circles (red, yellow, green). To the right of them is a Python logo icon followed by '.py'. The main area of the editor contains the following Python code:

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



.py

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



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

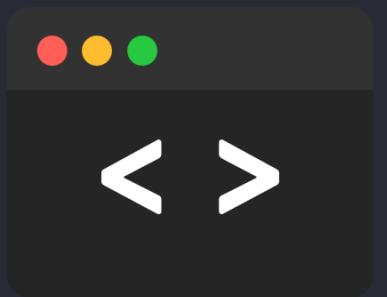


Idiom: count elements that satisfy predicate.

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

— Alan J. Perlis

presented with



snappify\*

\*no affiliation, they're just awesome

rodrigo@mathspp.com

[mathspp.com/talks](https://mathspp.com/talks)