# Expressions and Syntax

Carol V. Alexandru-Funakoshi
Informatics I - 2024

# Expressions and Syntax

$$\sqrt{\frac{1}{2}}$$

$$\sum_{i=3}^{6} i^2$$

$x + y^2$

$$\sqrt{a^2+b^2}$$

$$m\,c^2$$

# What is an expression?

Something that can be evaluated by itself

Expressions:

1 + 3

True

"a".upper() + "pple"

Not expressions:

1 +

True or

"a".

# Values

True

1

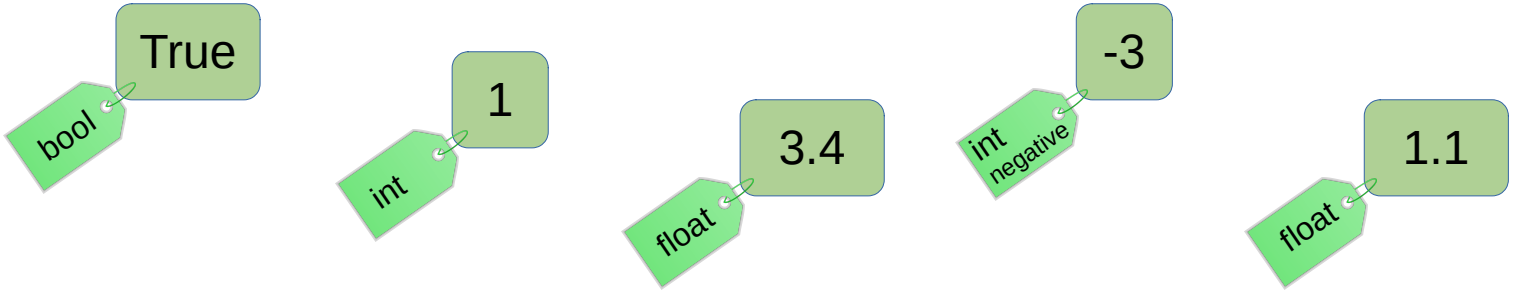
3.4

-3

1.1

Values are the simplest expressions

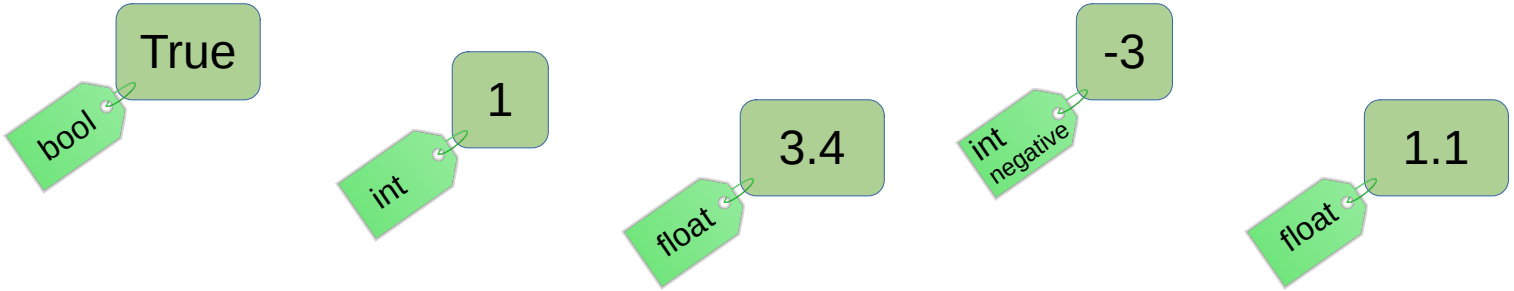
# Values




Each value has a type

# Values

$x + y^2$
in Python:
x + y**2



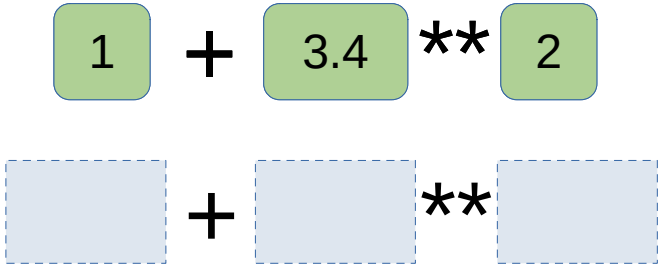

Values “fit in” where they have a compatible type:

Fit: 1 + 3.4 ** 2

No fit: 1 + “Hi”  3.4 .lower()

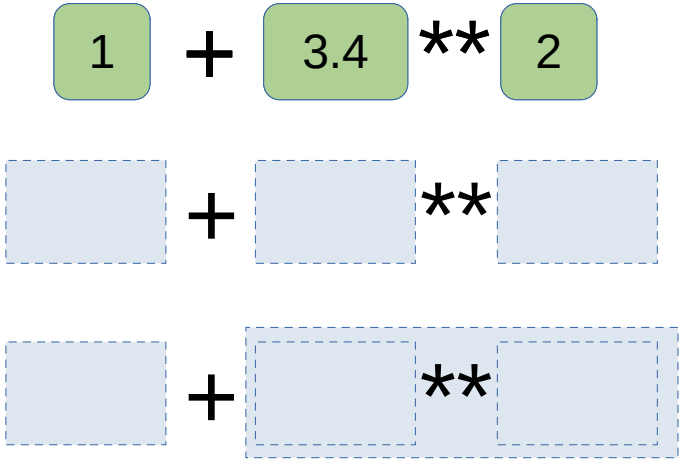
# Understanding an expression

Step 1: Recognize the boxes!



# Understanding an expression

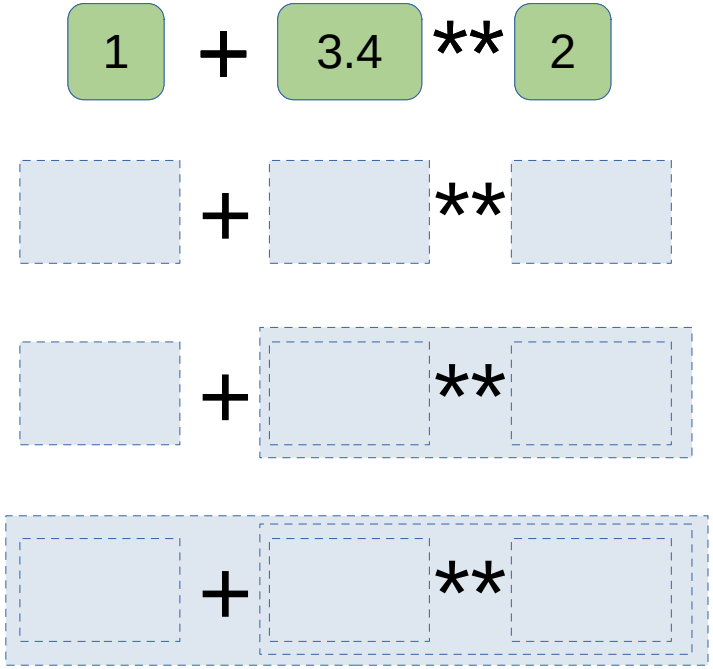
## Step 1: Recognize the boxes!




Operator precedence adds “invisible braces”

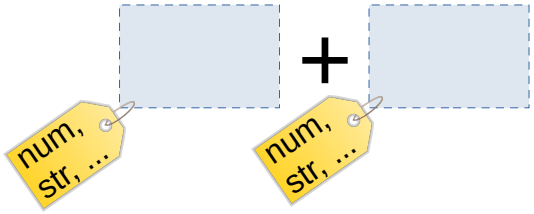
# Understanding an expression

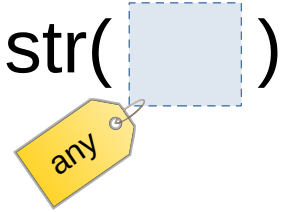
Step 1: Recognize the expressions (boxes)!



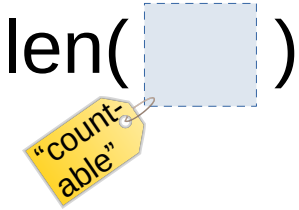
# Understanding an expression

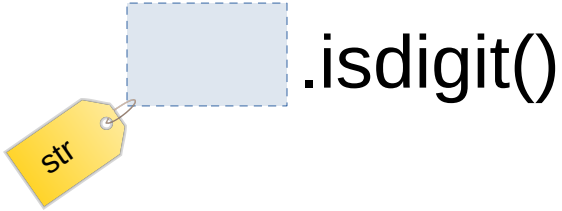
## Step 2: Recognize the possible input types!



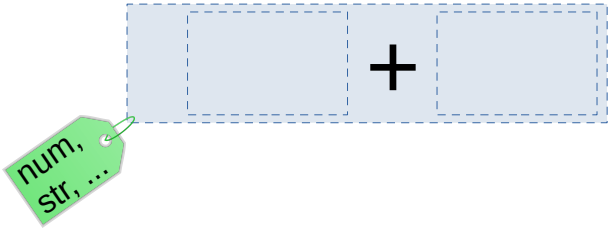


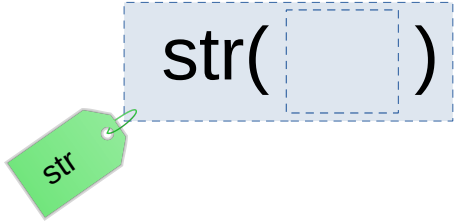



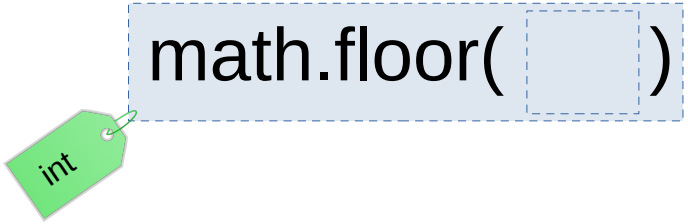



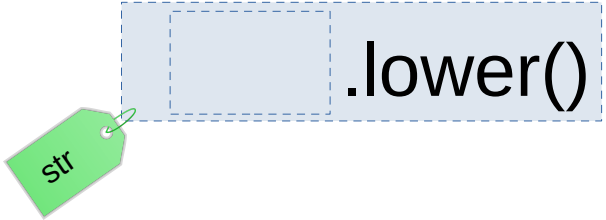
# Understanding an expression

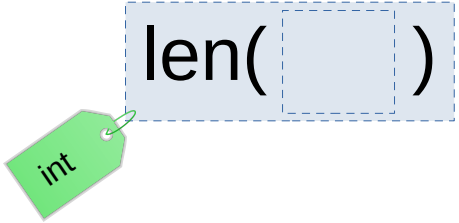
## Step 3: Recognize the resulting type!

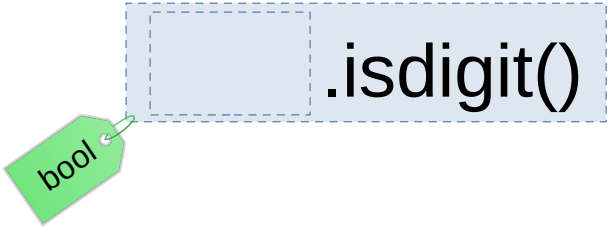


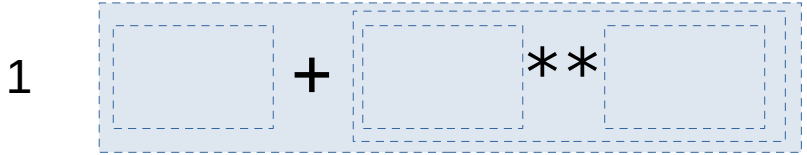



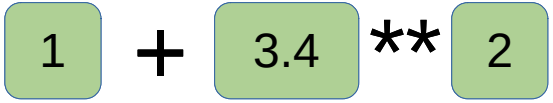



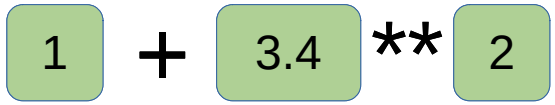
# Steps 1, 2 and 3:

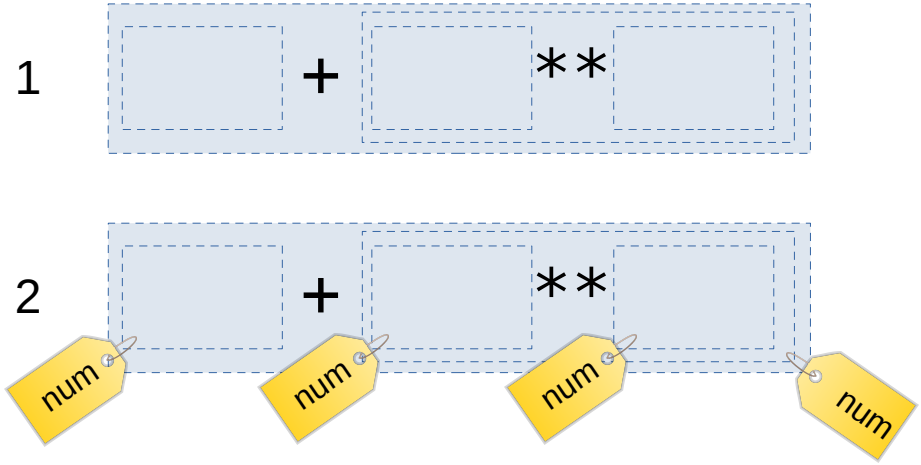



```
1 + 3.4 ** 2
```

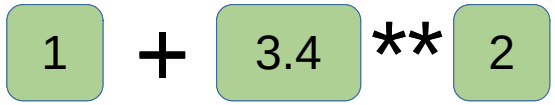
# Steps 1, 2 and 3:



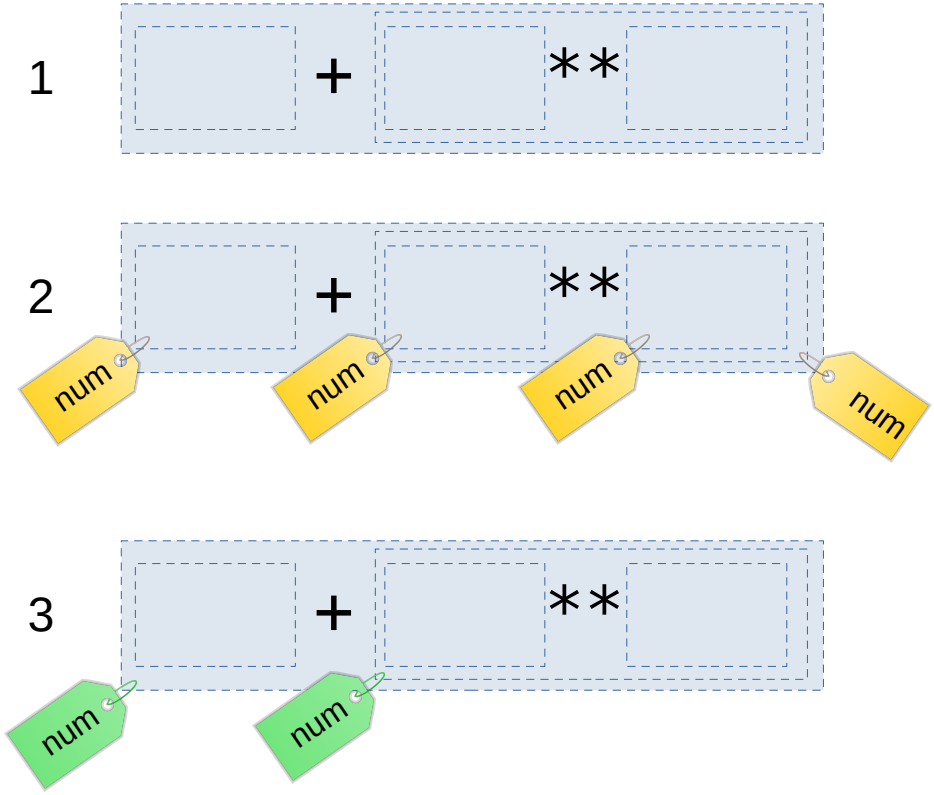




```
1 + 3.4 ** 2
```

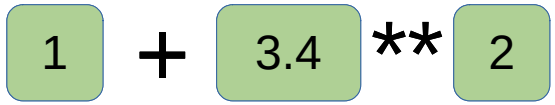
# Steps 1, 2 and 3:



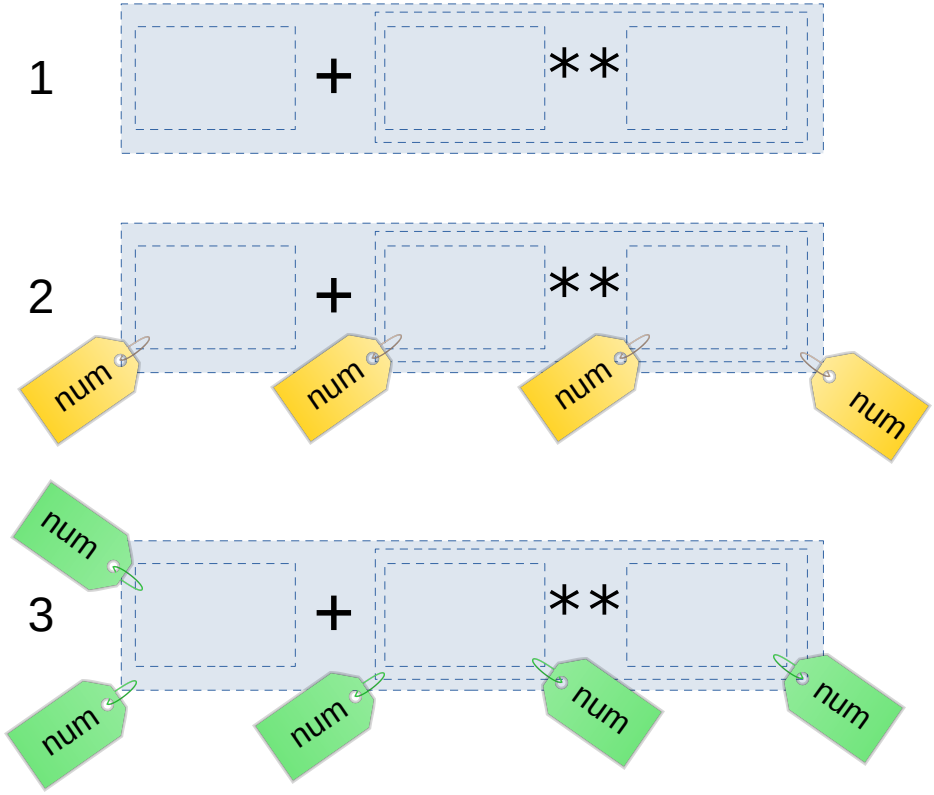

```
1 + 3.4 ** 2
```



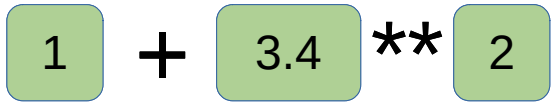
# Steps 1, 2 and 3:



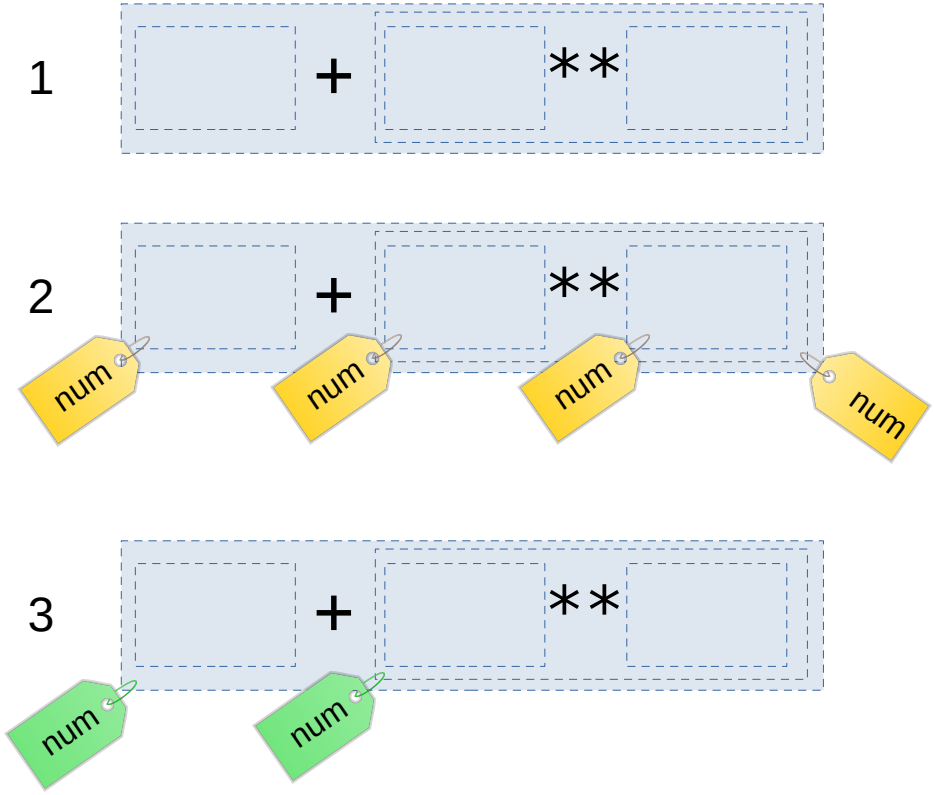

```
1 + 3.4 ** 2
```



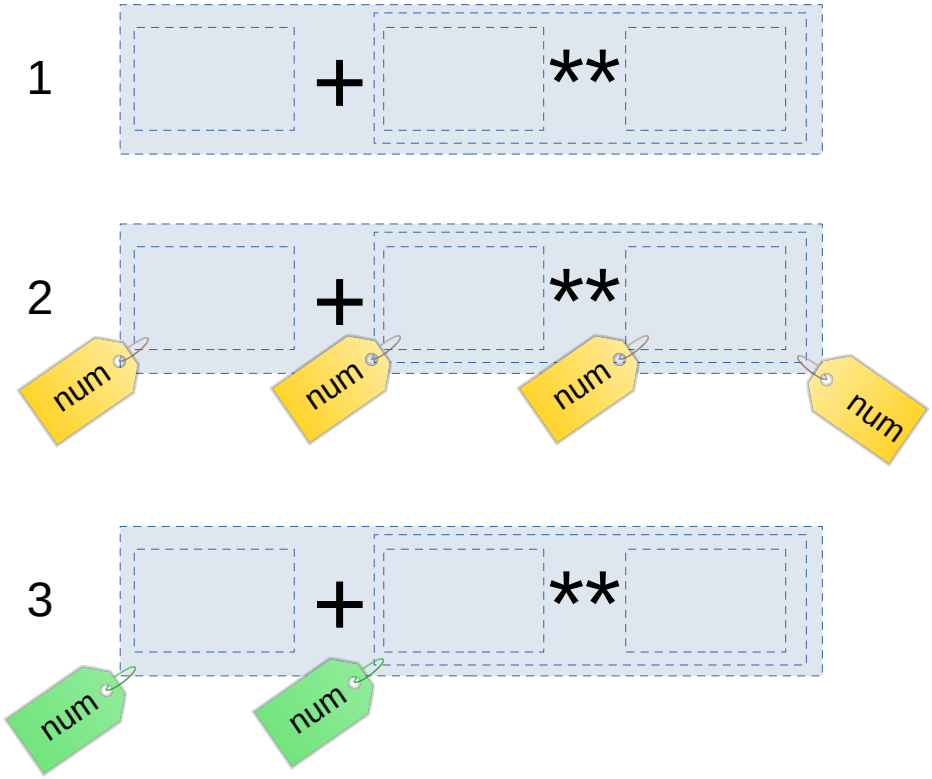
# Steps 1, 2 and 3:



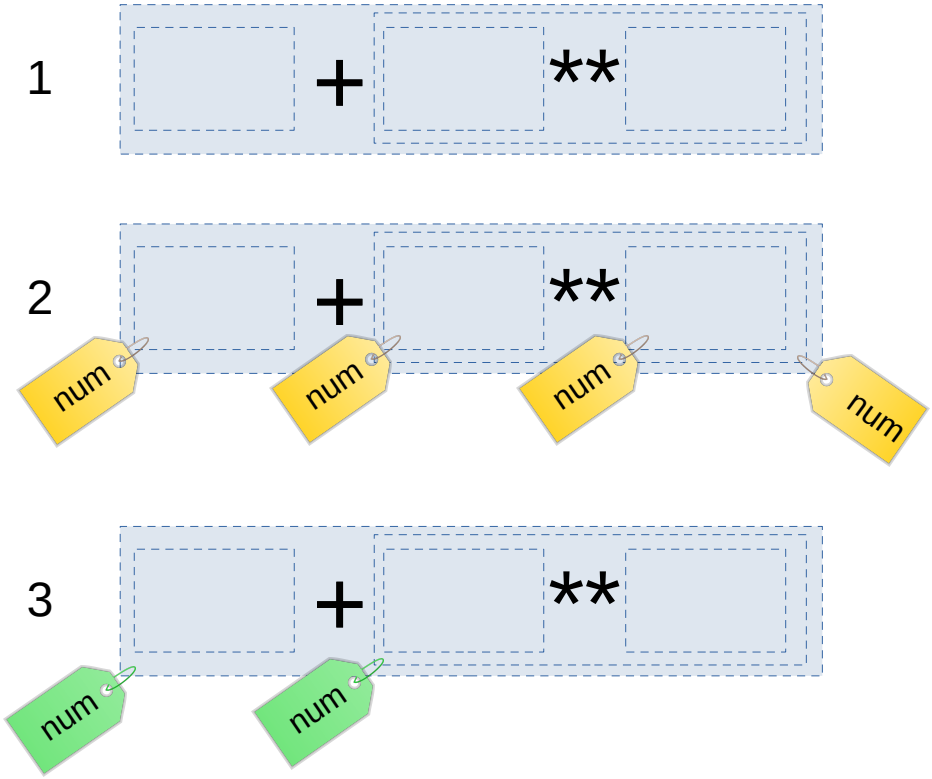

```
1 + 3.4 ** 2
```

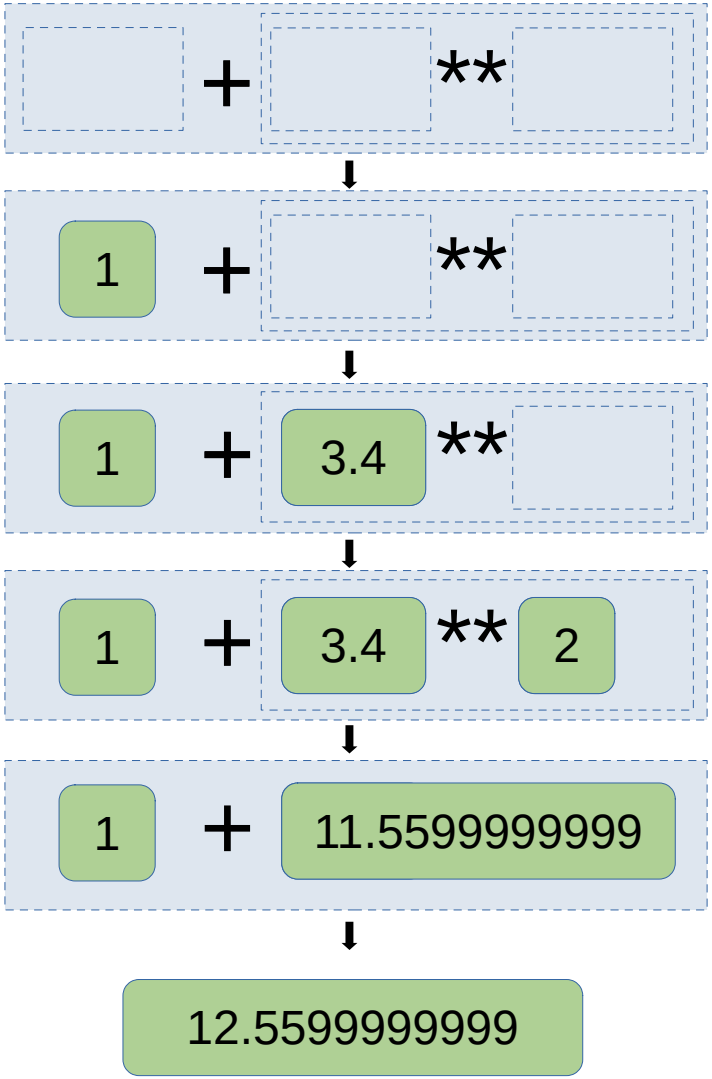


# Now we can compute



# Now we can compute

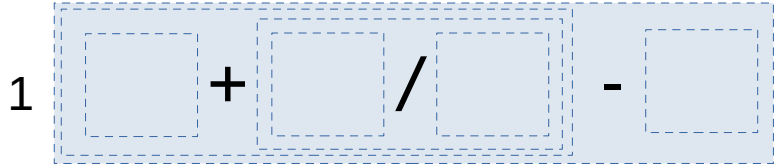


# Example 1

```
1 + 2 / 3 - 4
```

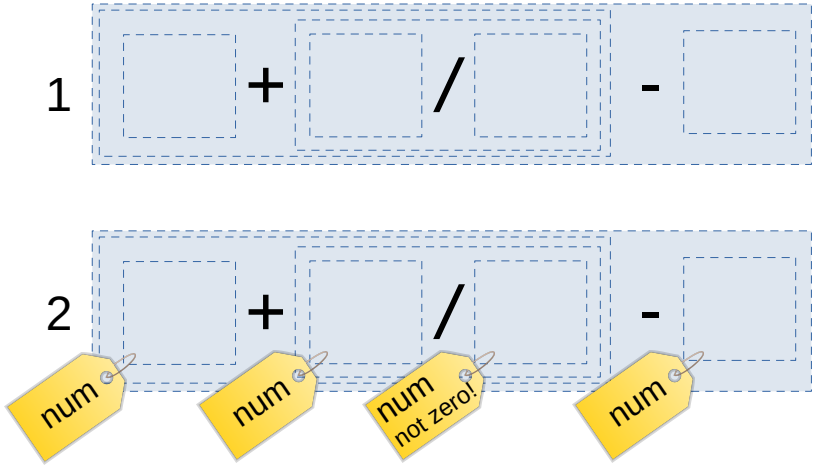
# Example 1




```
1 + 2 / 3 - 4
```

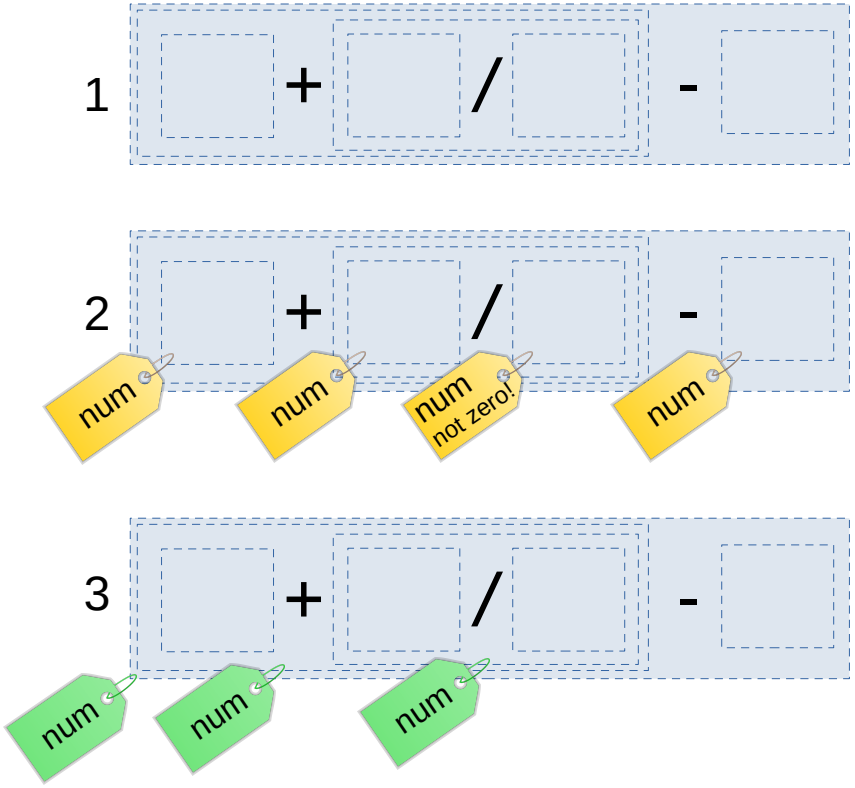
# Example 1

```
1 + 2 / 3 - 4
```



# Example 1

```
1 + 2 / 3 - 4
```



# Example 2

```
1 len(                    )
```

```
len("hello")
```

# Example 2

```
len("hello")
```

1 `len(  )`

2 `len(  )`

# Example 2

```
len("hello")
```

# Example 3

```
1 len([    ,    ])
```

```
len([22,"hi"])
```

# Example 3

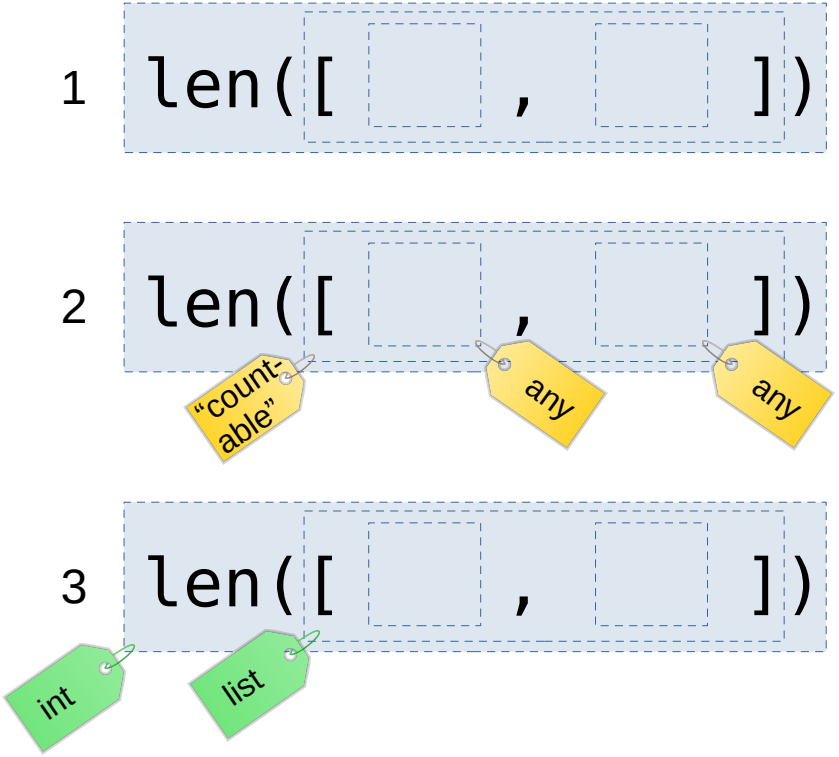
```
len([22,"hi"])
```

1 len([ , ])

2 len([ , ])

# Example 3

```
len([22,"hi"])
```



# Example 4

```
stuff = (6,12,"hi")
stuff[2:]
```

1 `stuff[ : ]`

2 `stuff[ : ]`

int or nothing

int or nothing

3 `stuff[ : ]`

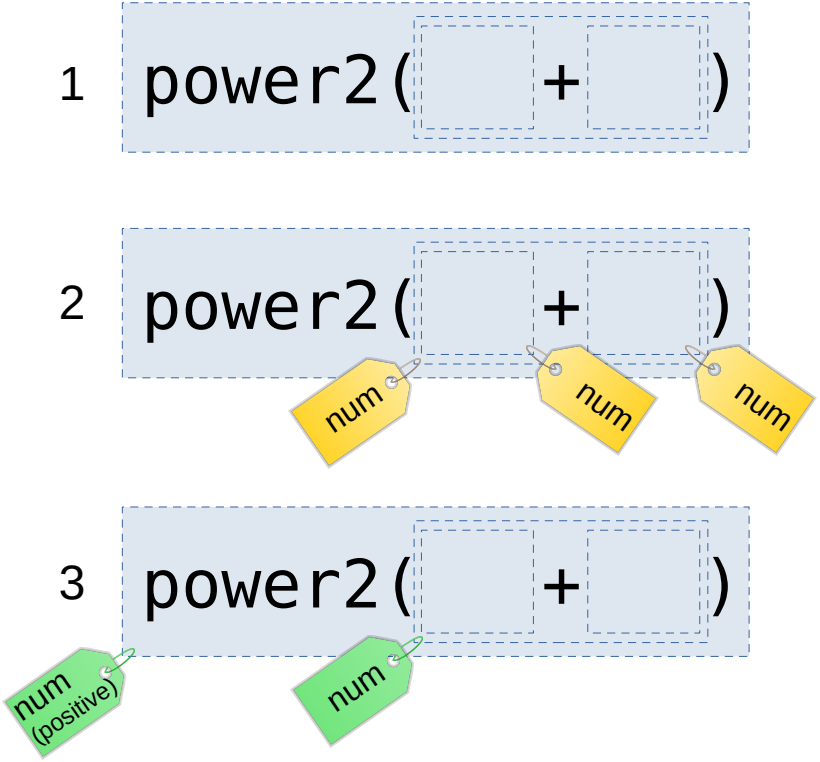
tuple

# Example 5

```
stuff = (6,12,"hi")
stuff[1+1:]
```

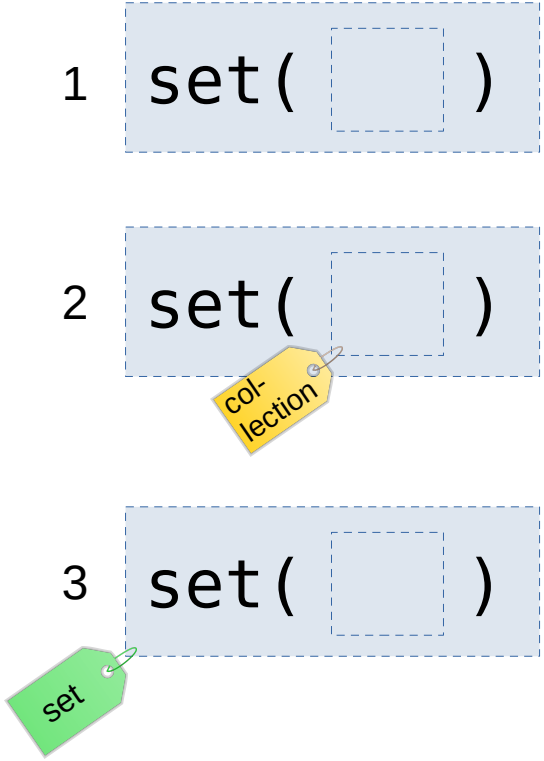
# Example 6

```
def power2(x):
    return x**2
power2(2+2)
```



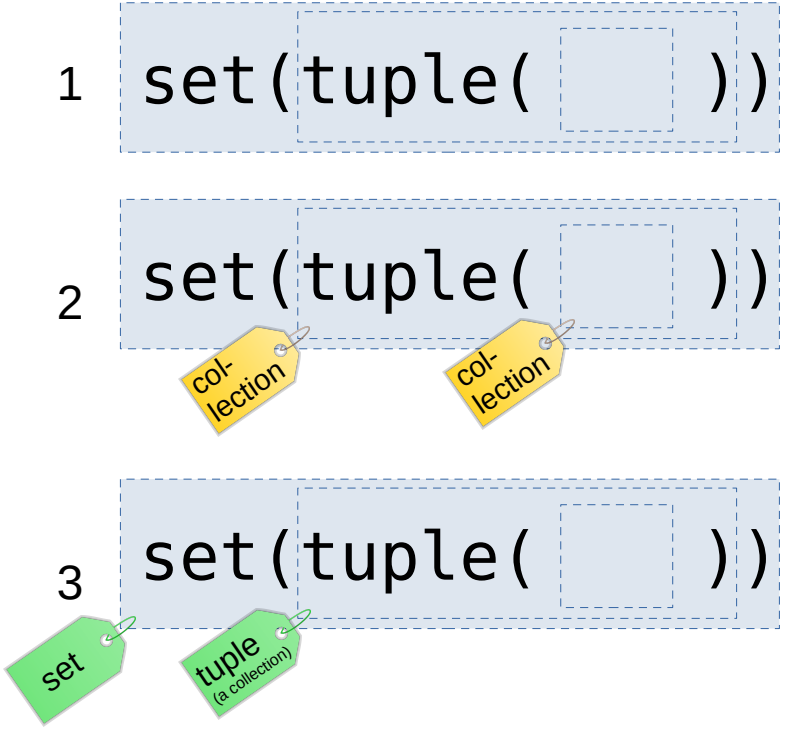
# Example 7

```
my_list = [1,1,"hi"]
t = tuple(my_list)
set(t)
```



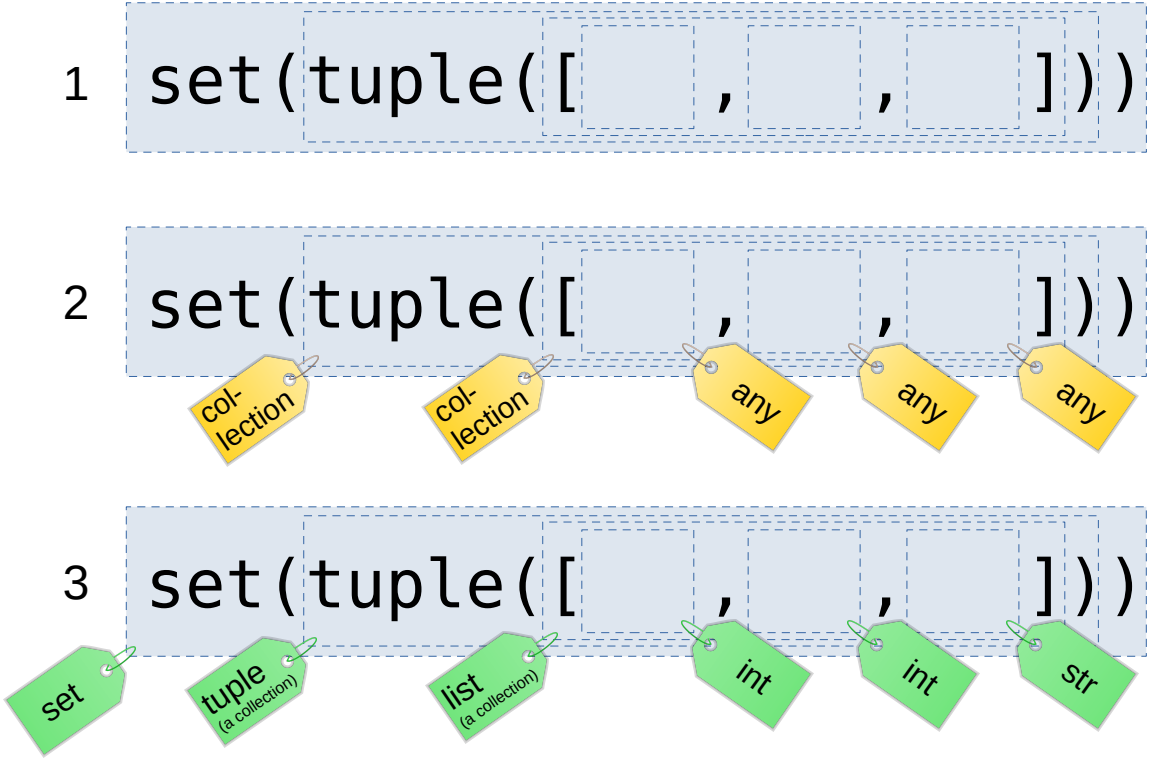
# Example 7 cont.

```
my_list = [1,1,"hi"]
set(tuple(my_list))
```



# Example 7 cont.

```
set(tuple([1,1,"hi"]))
```



In the lecture, you've heard about tuples, lists and sets, and how to create them:

```
my_tuple = (1, "a", 1, 2)  # my_tuple is going to be (1, 'a', 1, 2)
my_list = [1, "a", 1, 2]   # my_list is going to be  [1, 'a', 1, 2]
my_set = {1, "a", 1, 2}    # my_set is going to be   {'a', 1, 2}
```

These data structures can also be created from each other using the `tuple`, `list` and `set` functions:

```
another_set = set(my_tuple)    # another_set is going to be   {1, 'a', 2}
another_list = list(my_tuple)  # another_list is going to be  [1, 'a', 1, 2]
another_tuple = tuple(my_set)  # another_tuple is going to be ('a', 1, 2)
```

With this new knowledge, implement a function `list_unique` which takes a **tuple** of arbitrary elements `elements` as the only parameter and returns a **list** containing only the unique elements contained in `elements`. The ordering of the elements in the return value does not matter.

No ‘for’ loops needed...

# Example 8

```
"hi".upper().endswith("I")
```

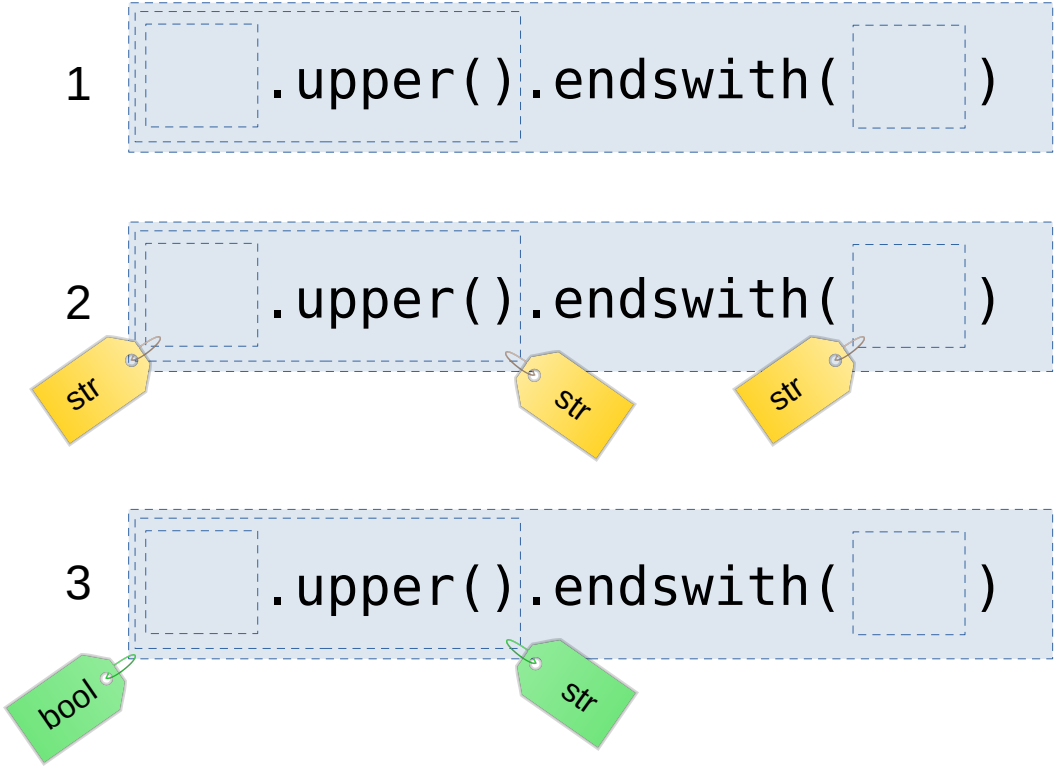
# Example 8

```
1   .upper().endswith(  )
```

```
"hi".upper().endswith("I")
```

# Example 8

```
"hi".upper().endswith("I")
```

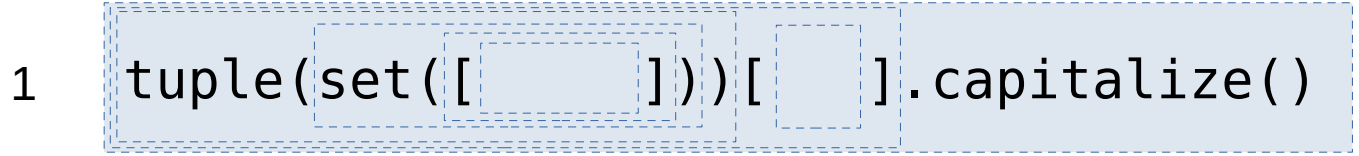


# Example 9

```
tuple(set(["bob"]))[0].capitalize()
```

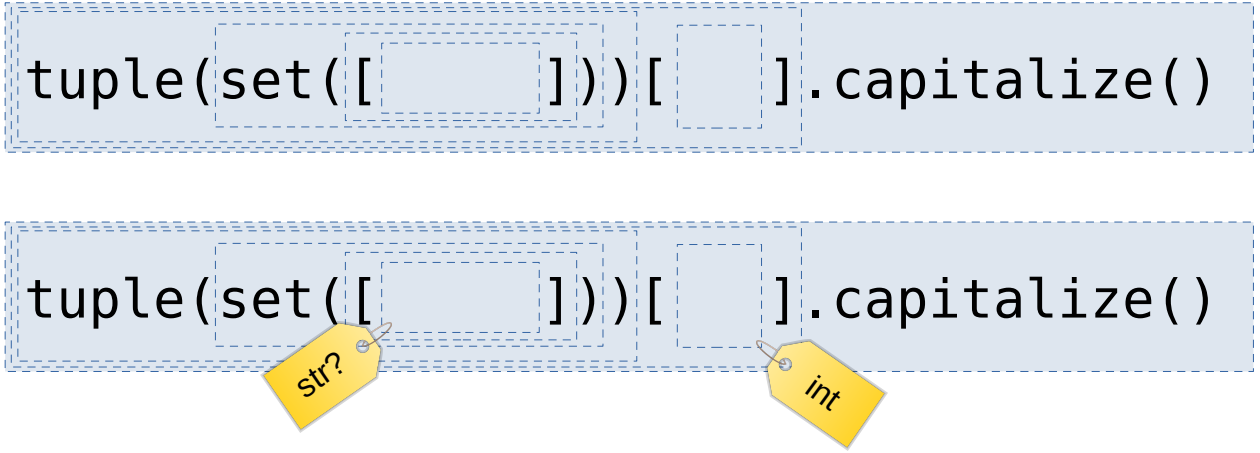
# Example 9

```
tuple(set(["bob"]))[0].capitalize()
```



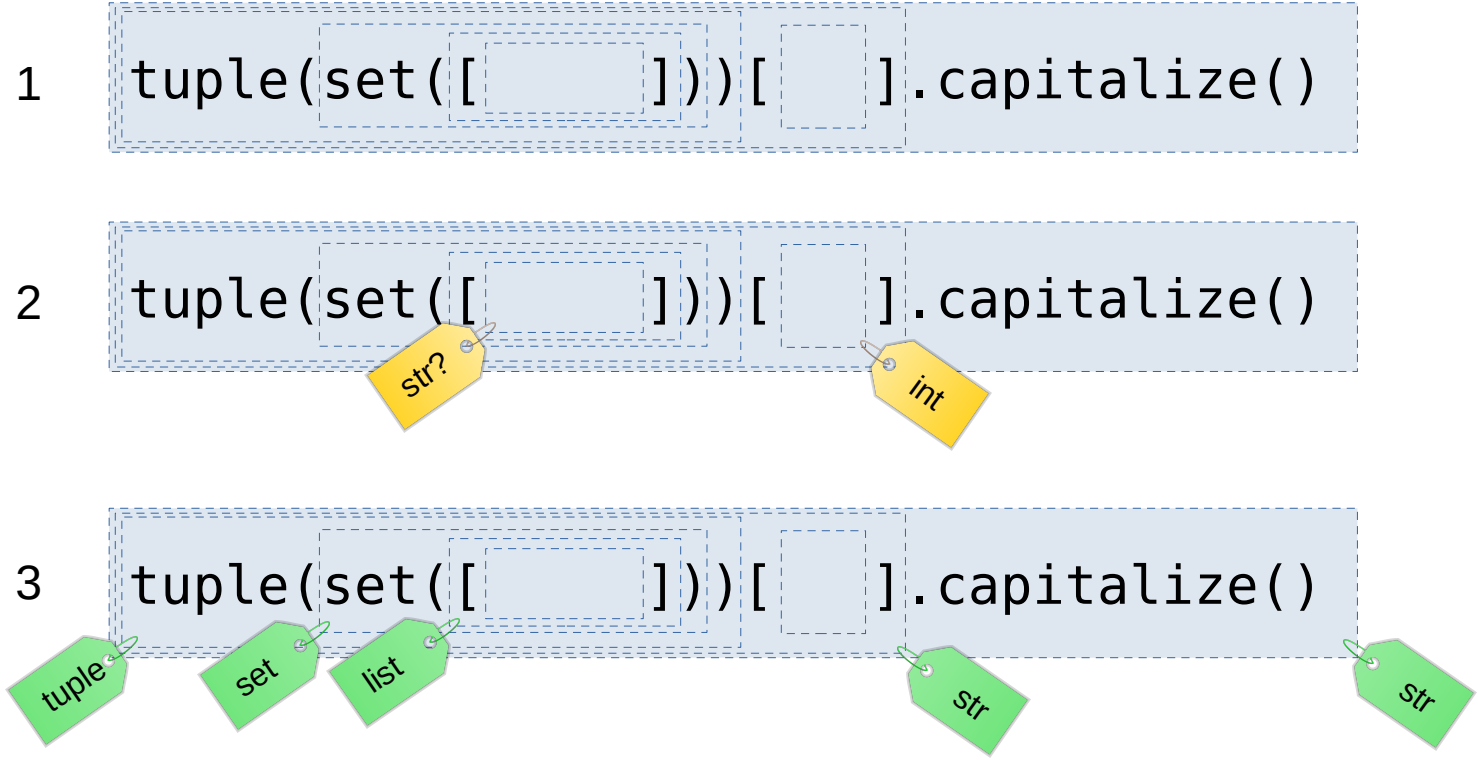
# Example 9

```
tuple(set(["bob"]))[0].capitalize()
```



# Example 9

```
tuple(set(["bob"]))[0].capitalize()
```

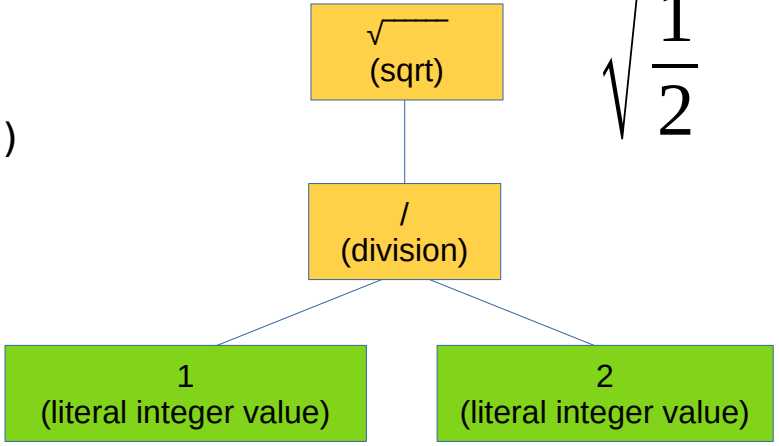


# You already know all of this!

Now you're just learning some new syntax and functionality

$$\sum_{i=3}^{6} i^2$$

```
sum([i**2 for i in range(3,7)])
```




$$\sqrt{\frac{1}{2}}$$