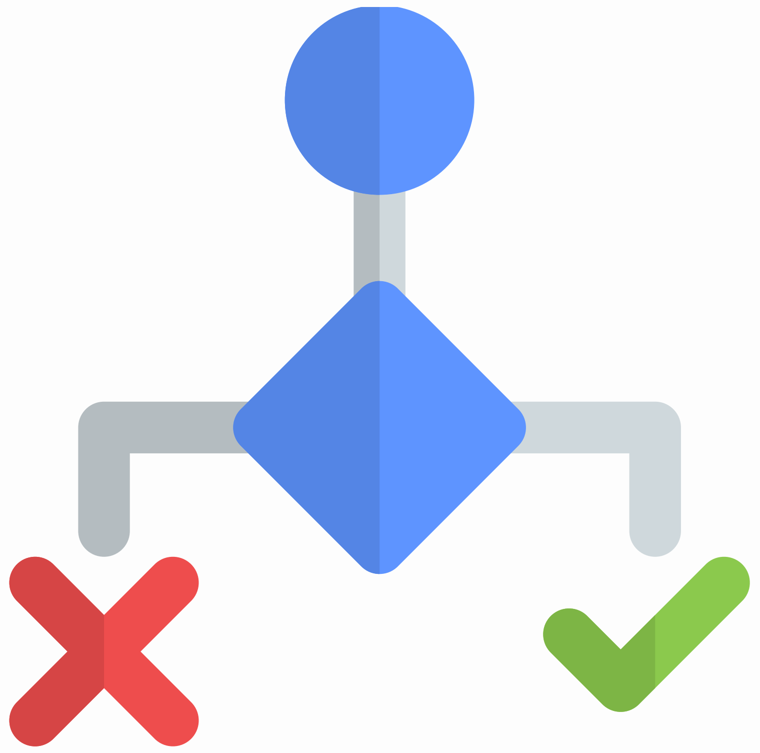
Python programming fundamentals

Contents

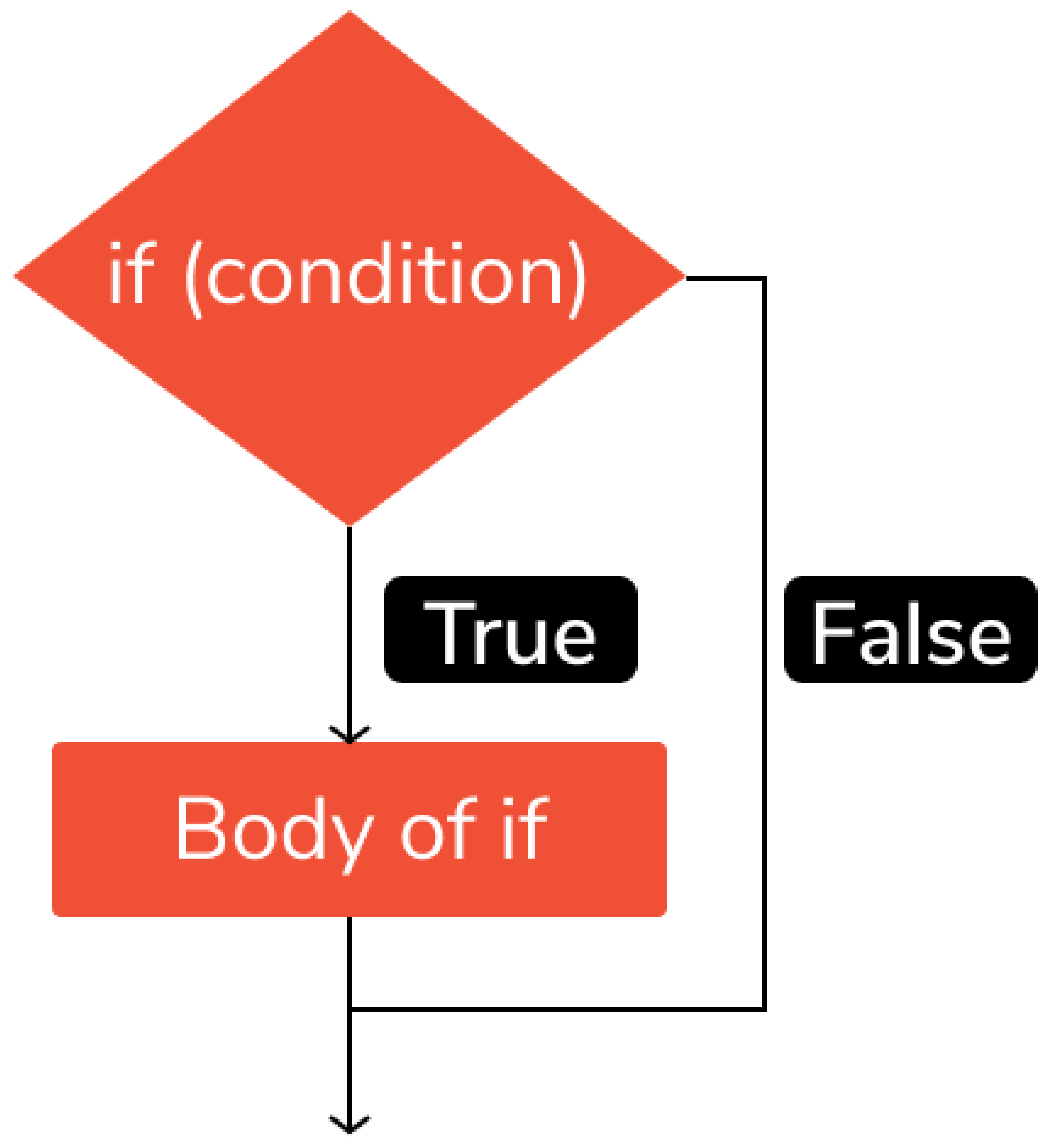
Condition and Branching

Condition

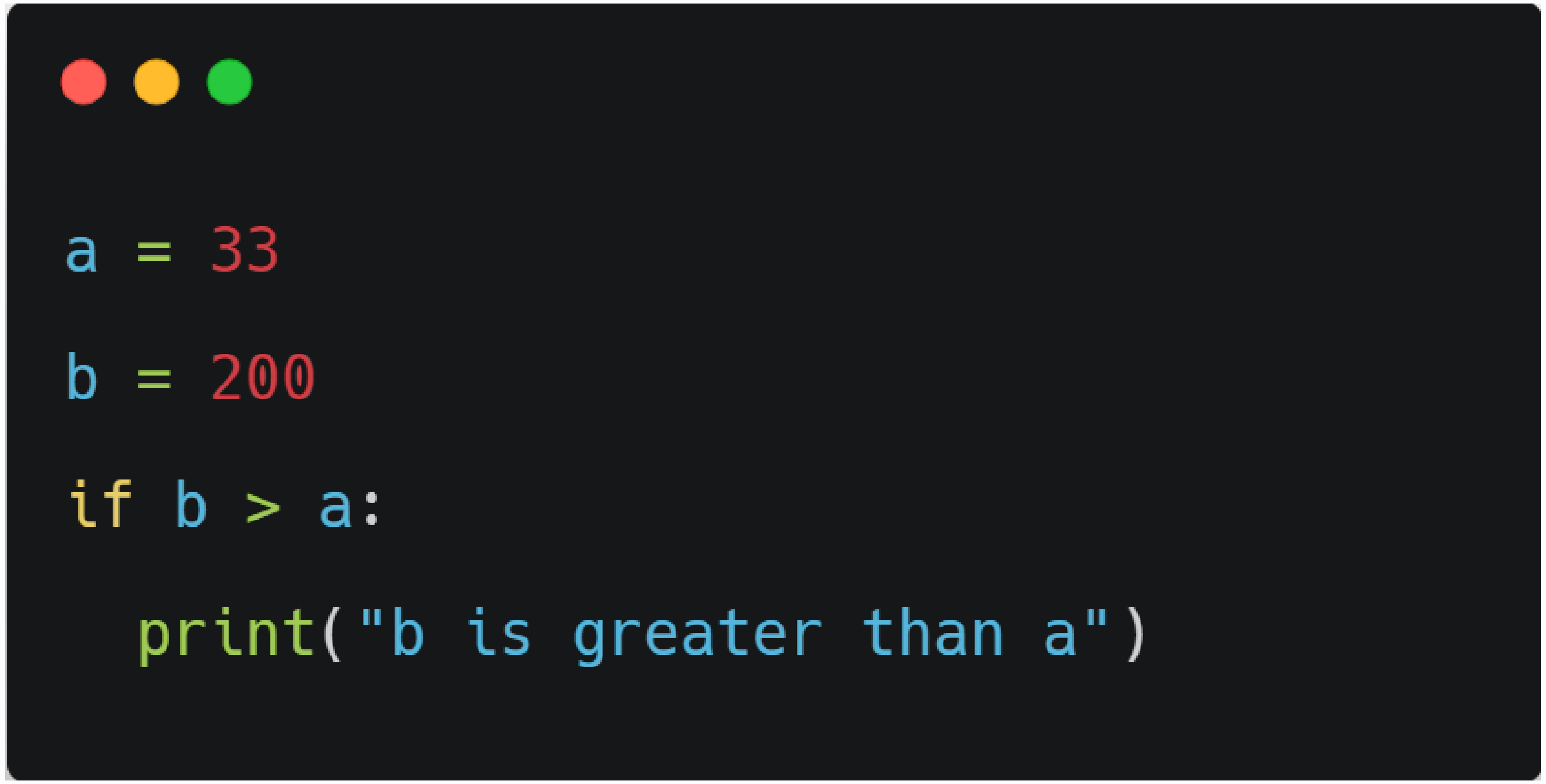


- Conditional statements in Python are used to control the flow of a program
- They allow the program to make decisions based on certain conditions

Condition

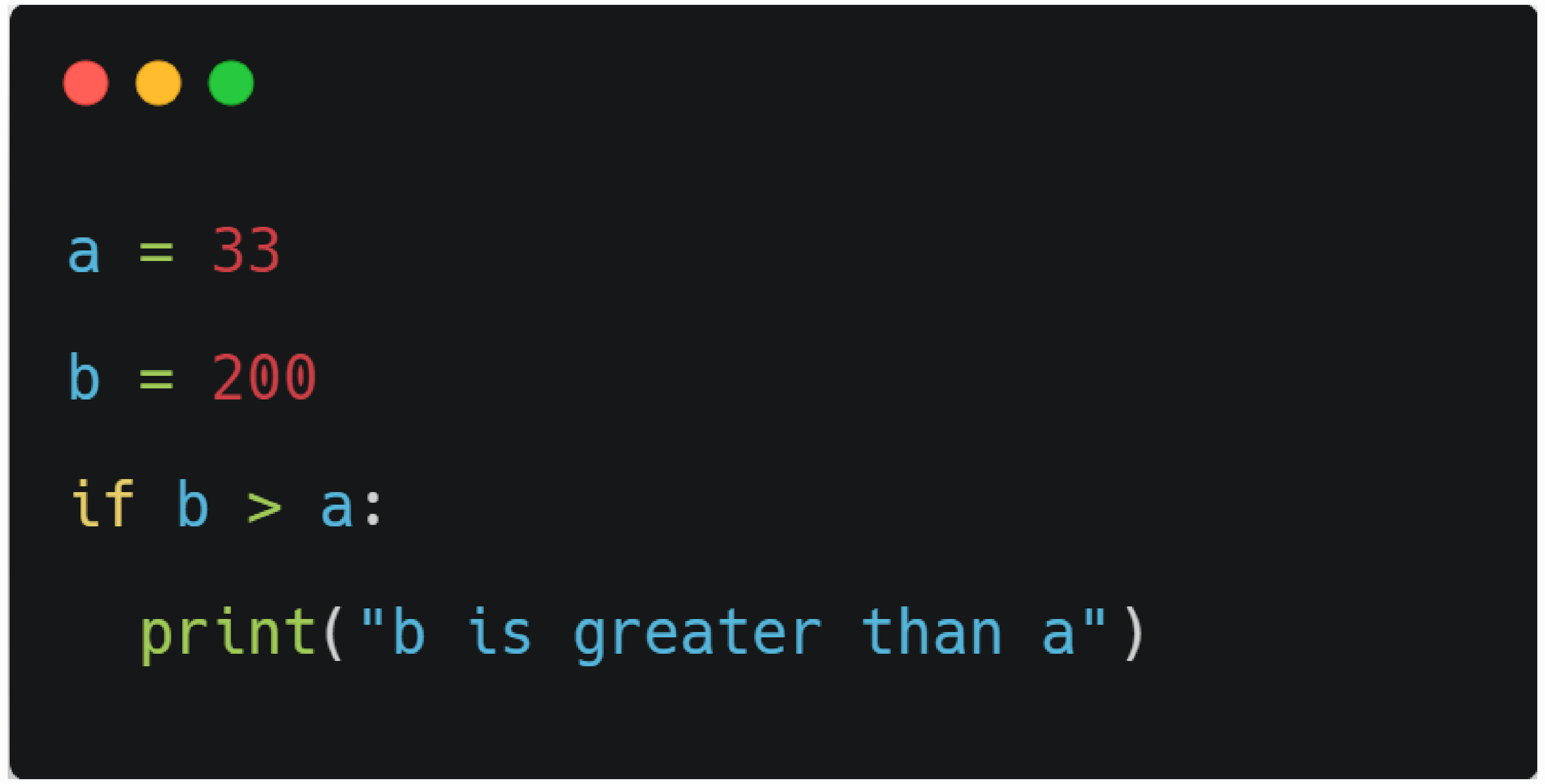


if statement



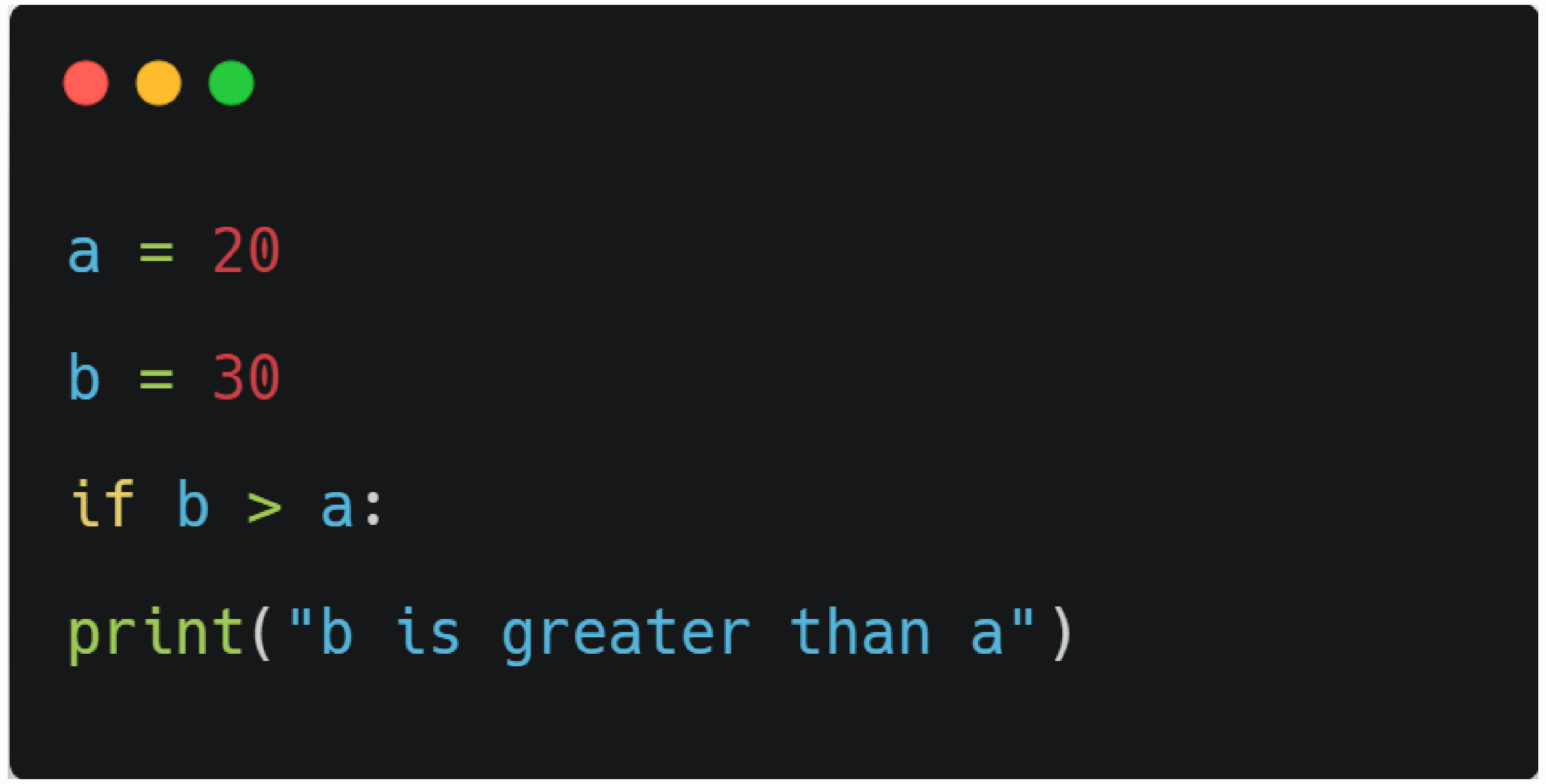
```
a = 33

b = 200

if b > a:

  print("b is greater than a")
```

Intendation rule



```python
a = 33
b = 200
if b > a:
  print("b is greater than a")
```

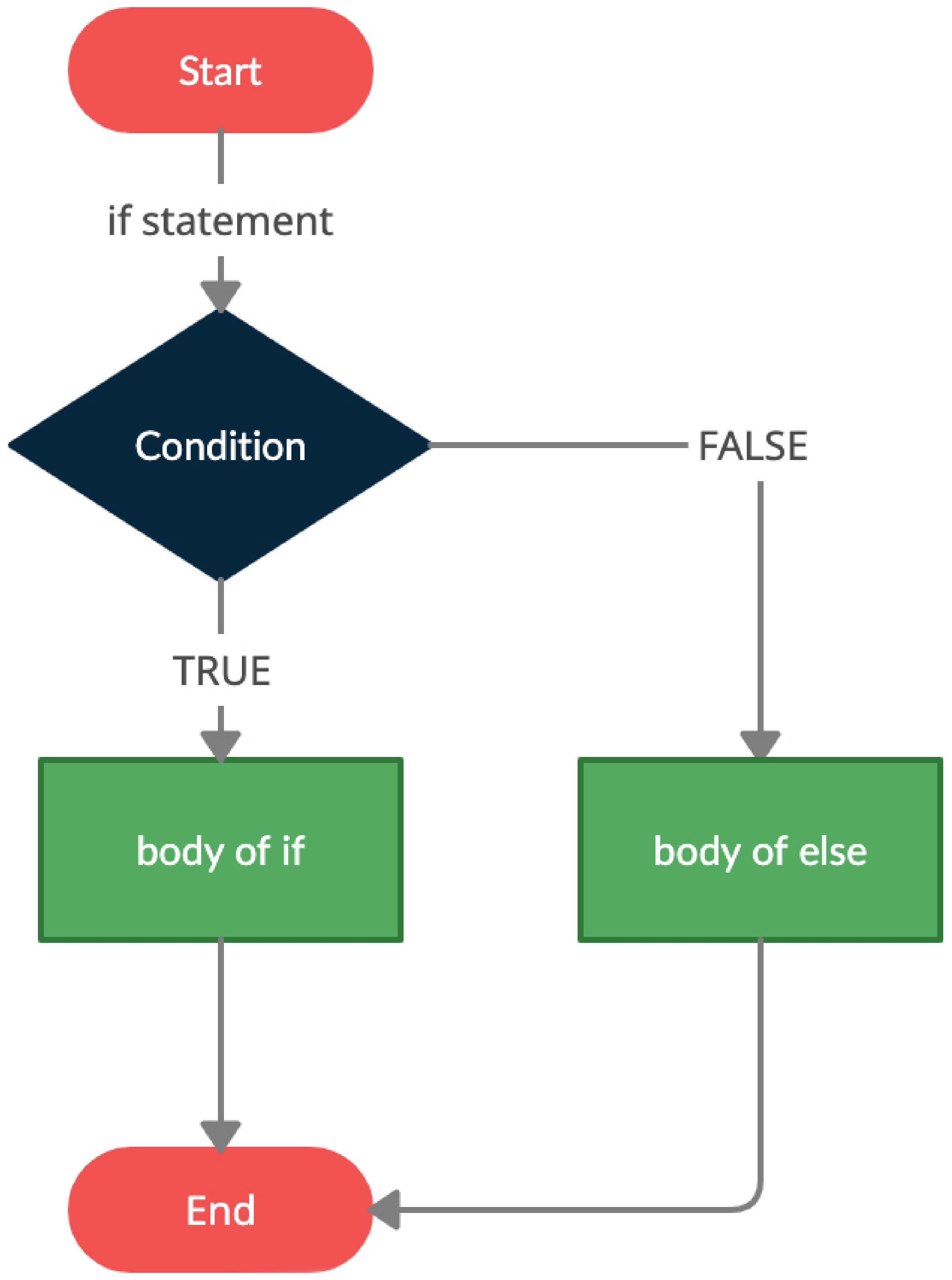
✓



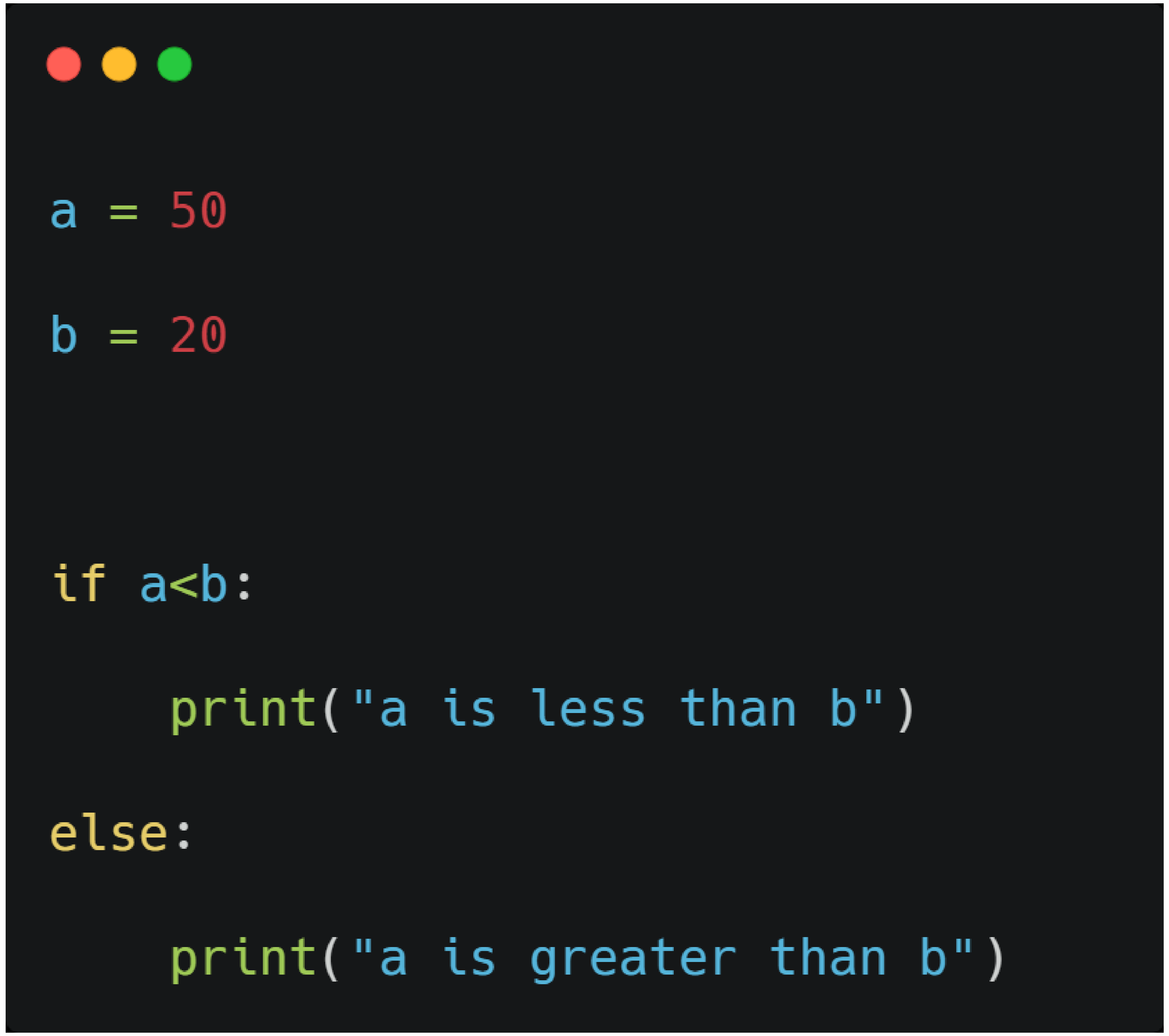
```python
a = 20
b = 30
if b > a:
print("b is greater than a")
```

✗

If else statements

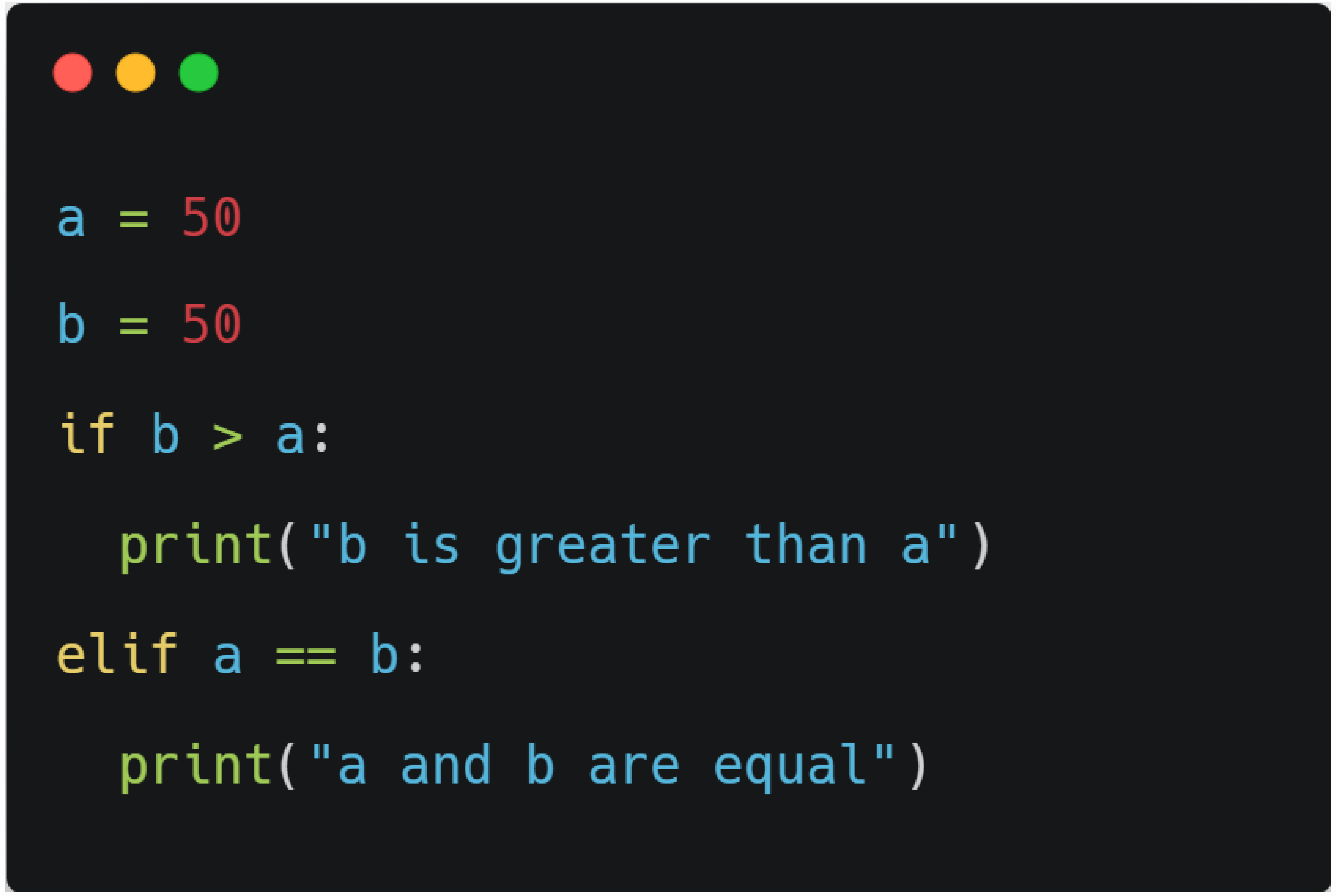


If else statements



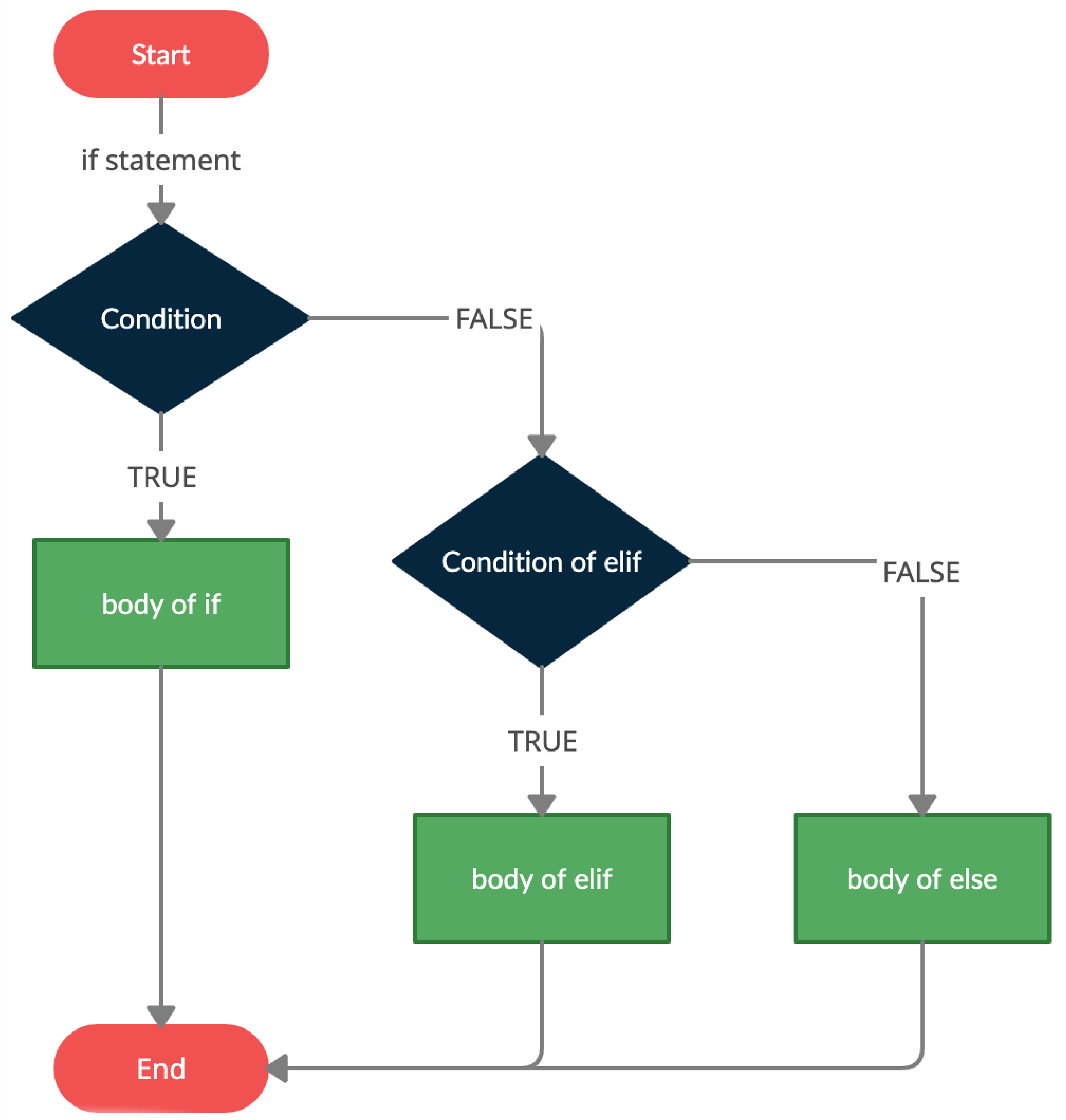
```
a = 50

b = 20


if a<b:

    print("a is less than b")

else:

    print("a is greater than b")
```

If elif statements

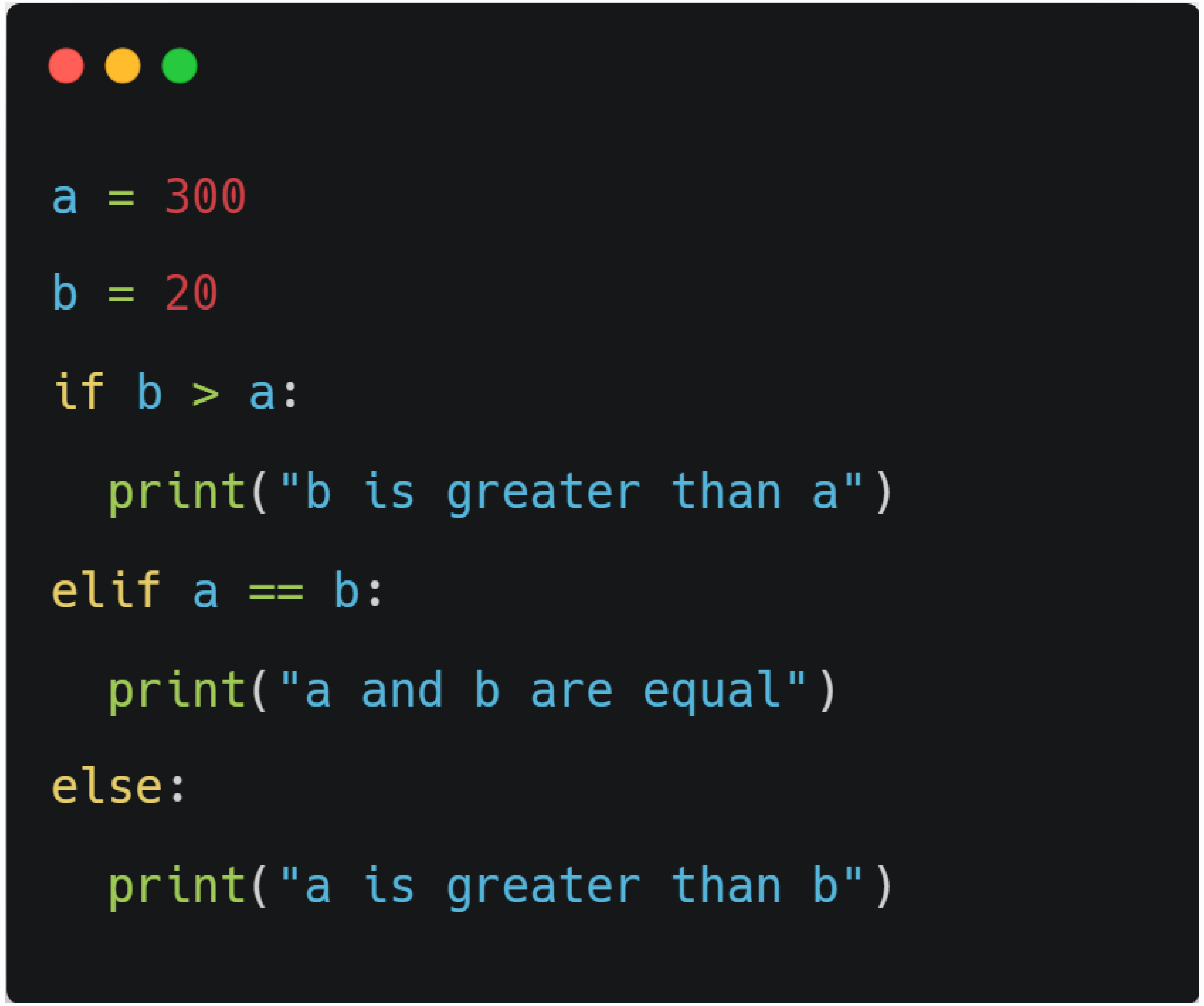


```
a = 50

b = 50

if b > a:

  print("b is greater than a")

elif a == b:

  print("a and b are equal")
```

If elif else statements

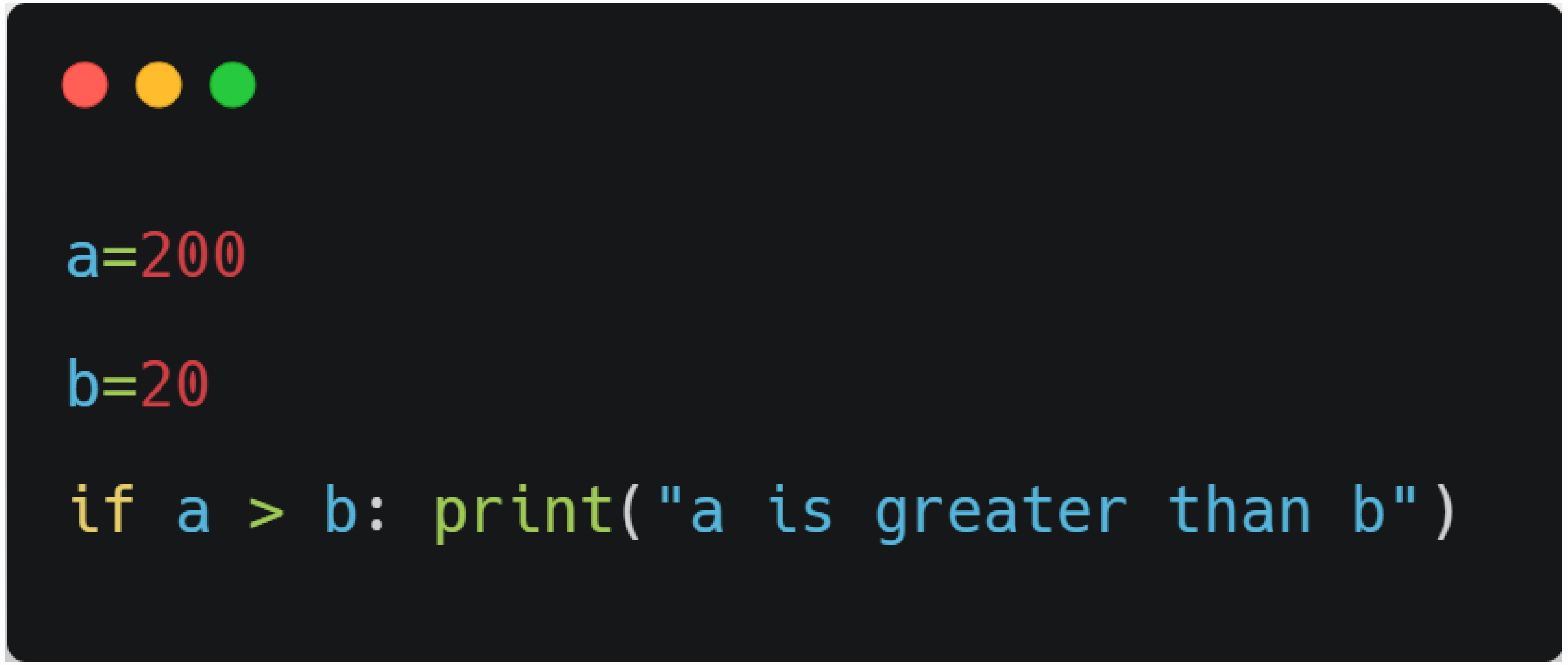


If elif else statements



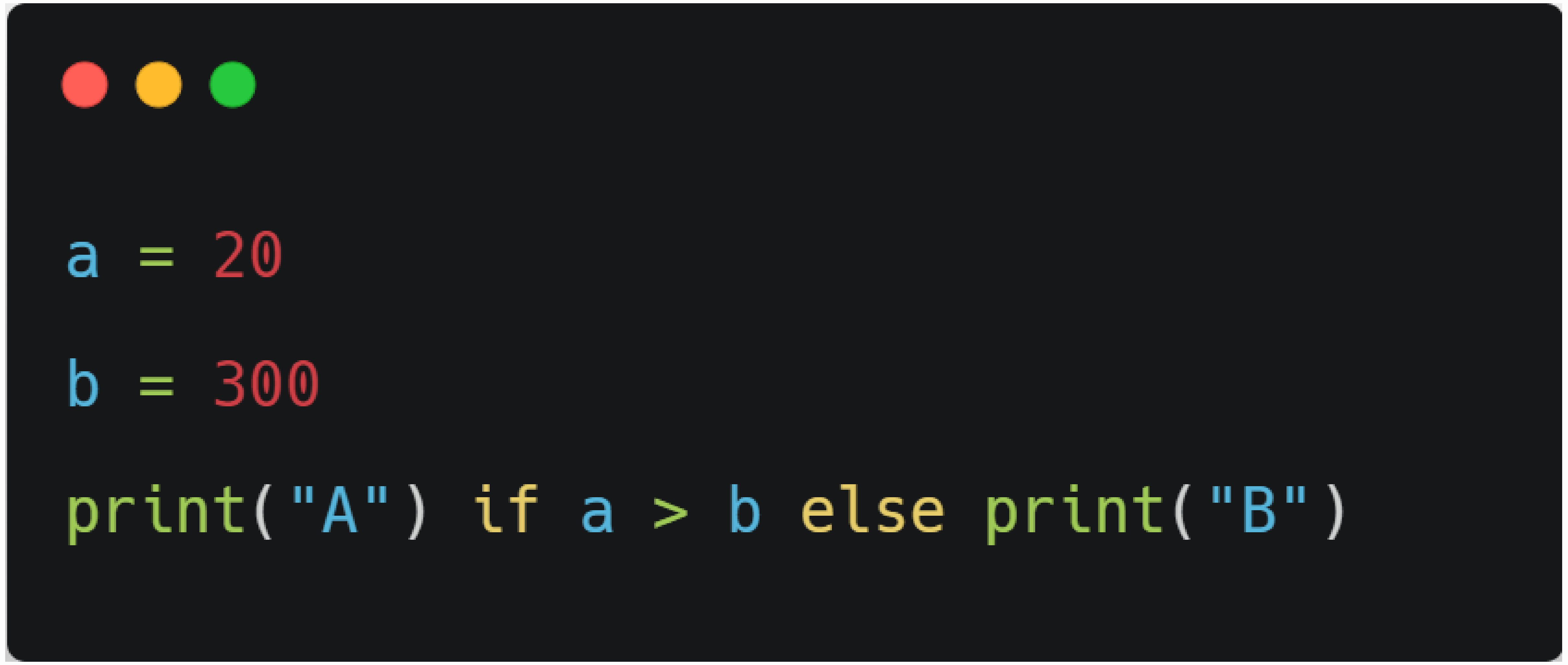
```
a = 300
b = 20
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

Short hand if



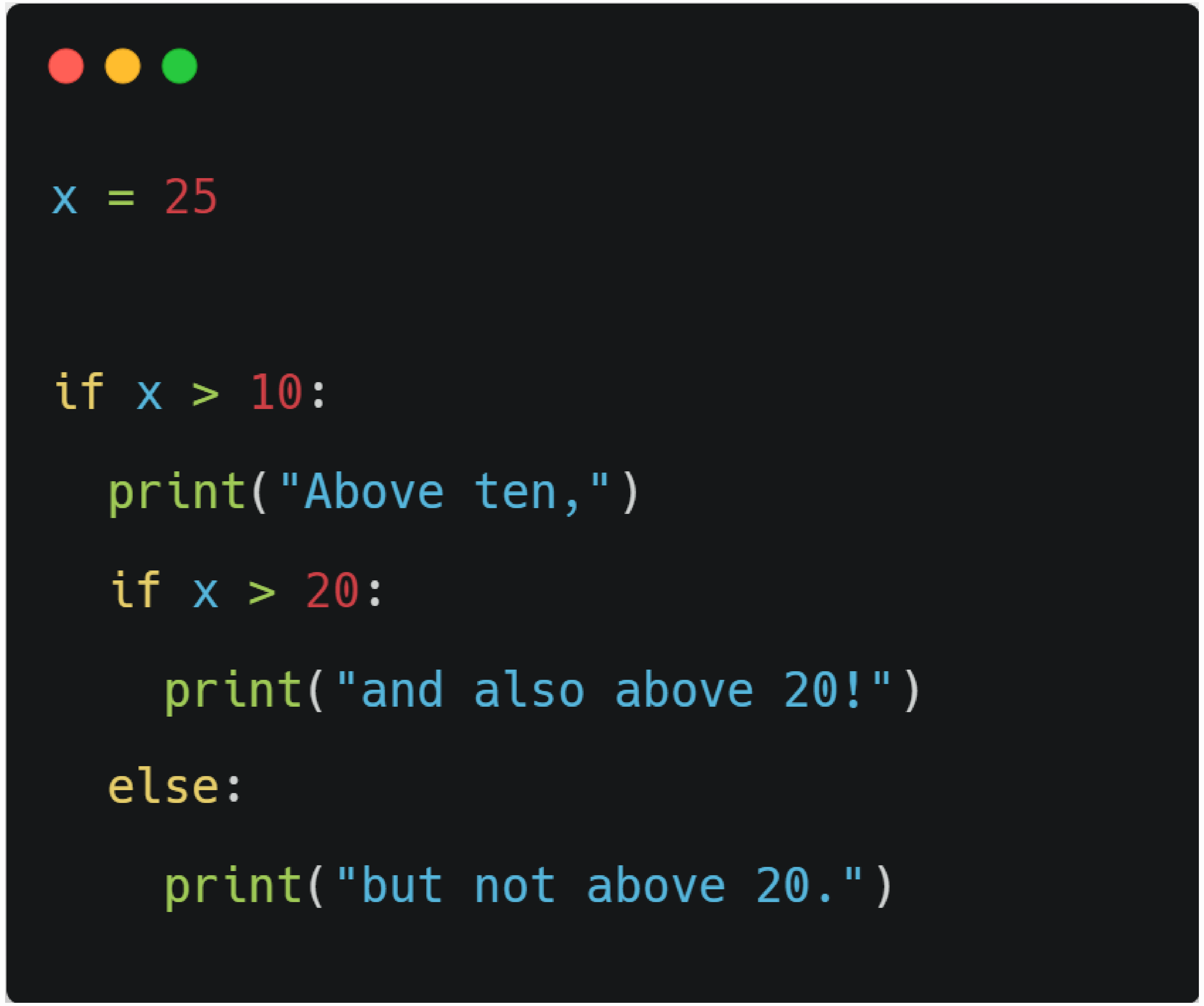
```
a=200

b=20

if a > b: print("a is greater than b")
```

Short hand if else



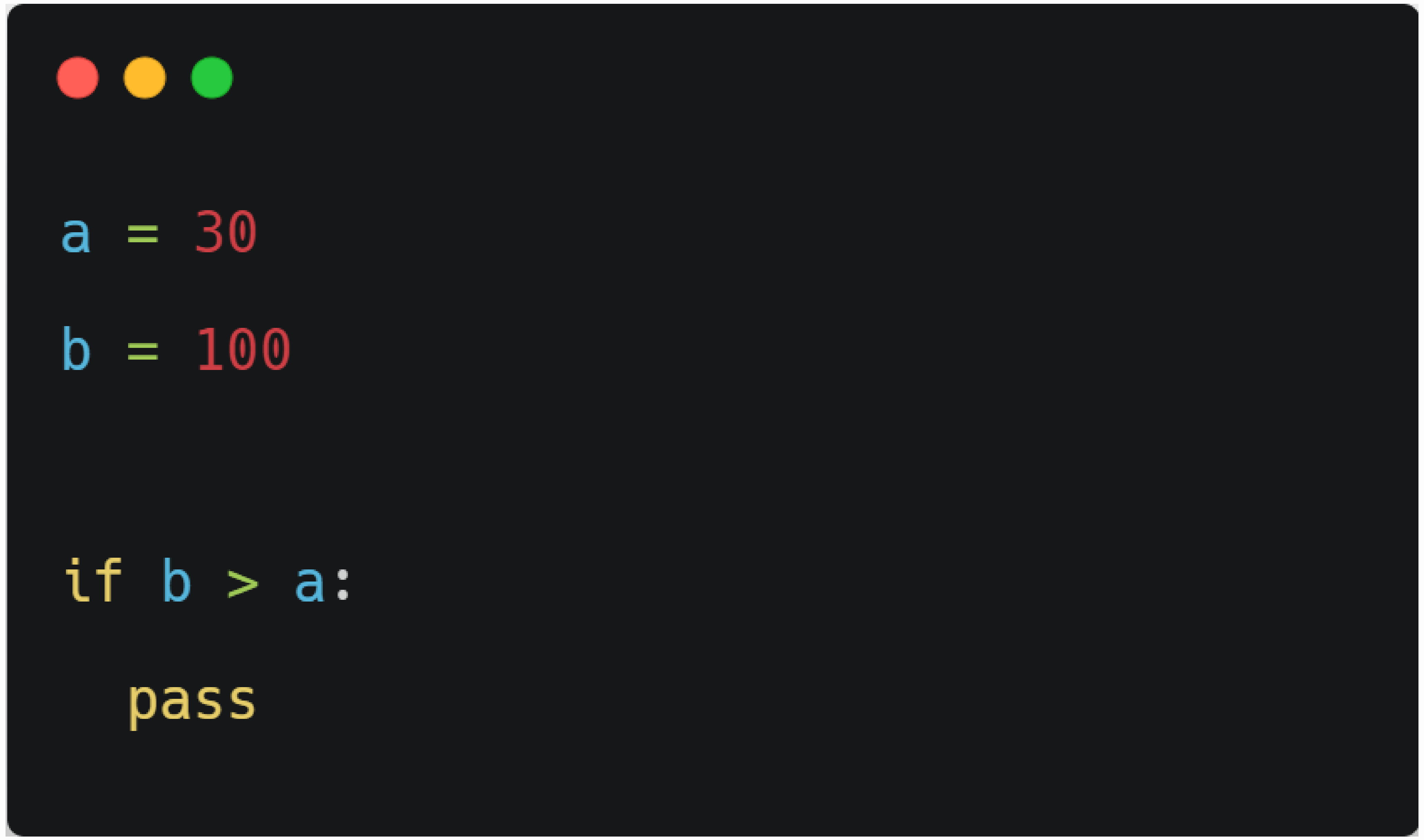
```
a = 20

b = 300

print("A") if a > b else print("B")
```

Nested if statements



```python
x = 25


if x > 10:
  print("Above ten,")
  if x > 20:
    print("and also above 20!")
  else:
    print("but not above 20.")
```

Pass statement



```
a = 30
b = 100


if b > a:
  pass
```

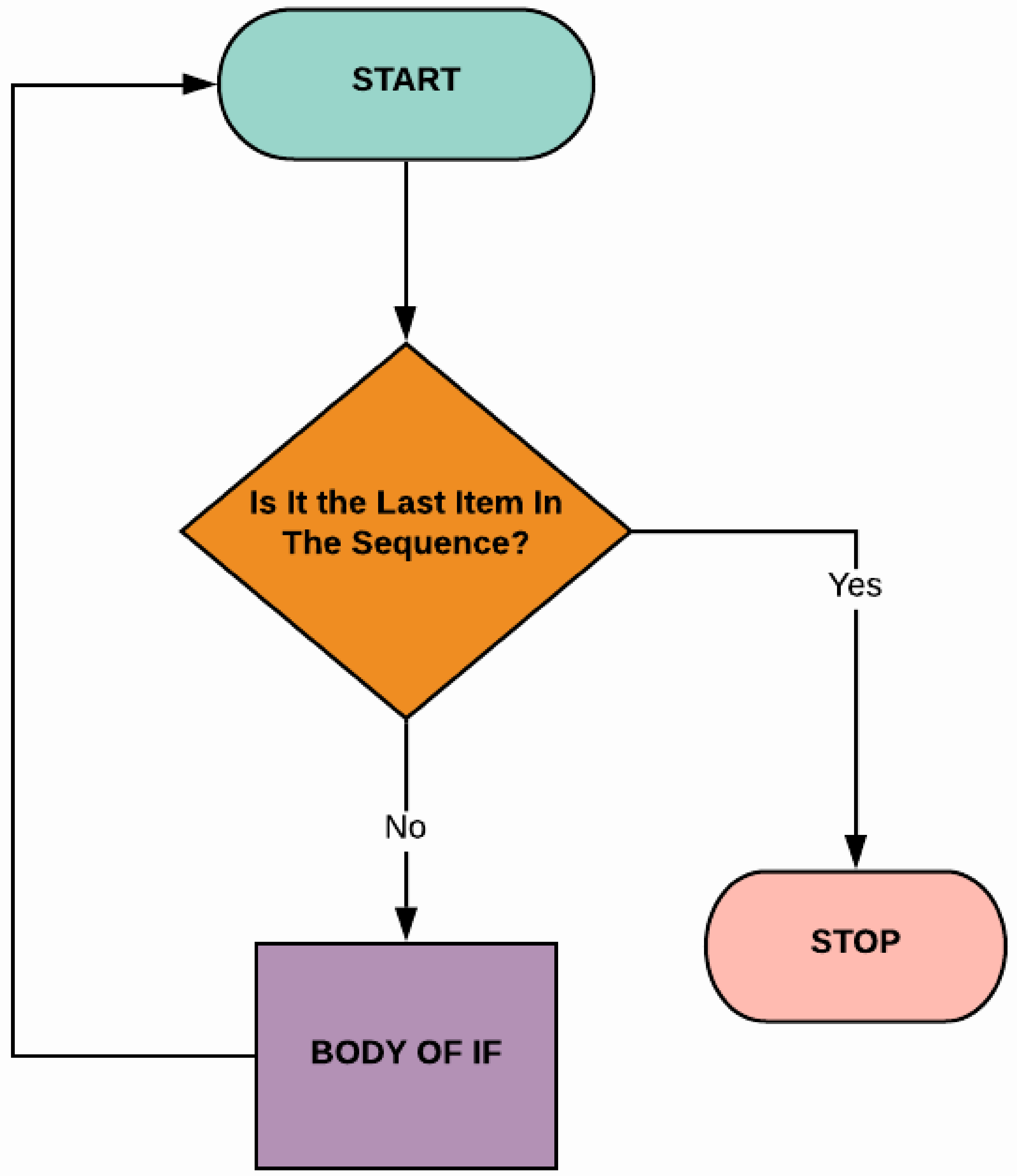
Loops

Loops

- For loop
- While loop

For Loop

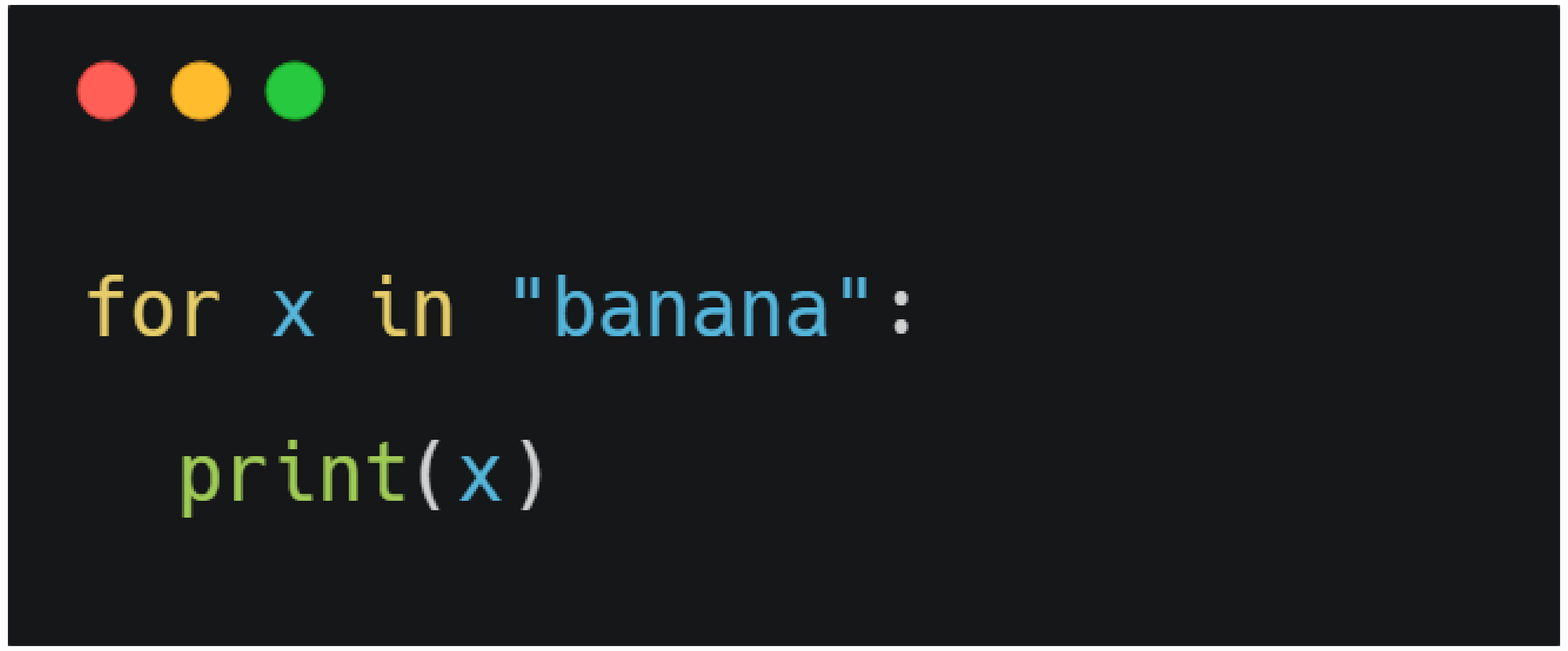
For loop



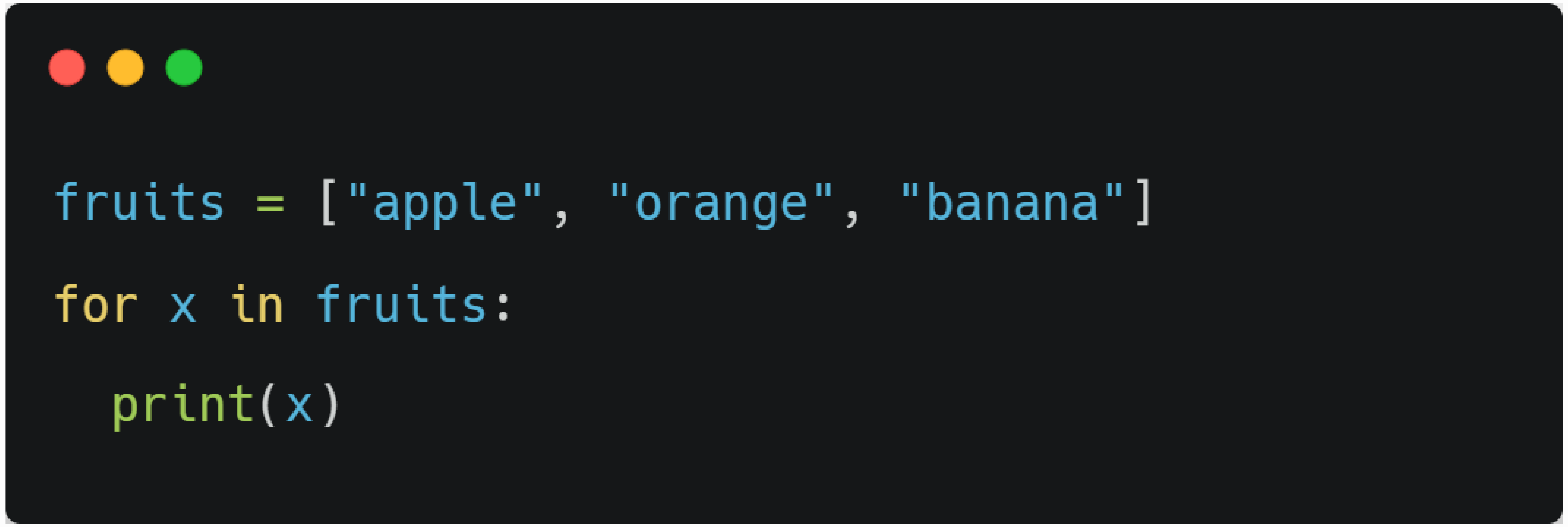
- For loops in Python are used to iterate over a sequence (list, tuple, or string) and perform a specific action for each item in the sequence

- Even strings are iterable objects, they contain a sequence of characters

For loop

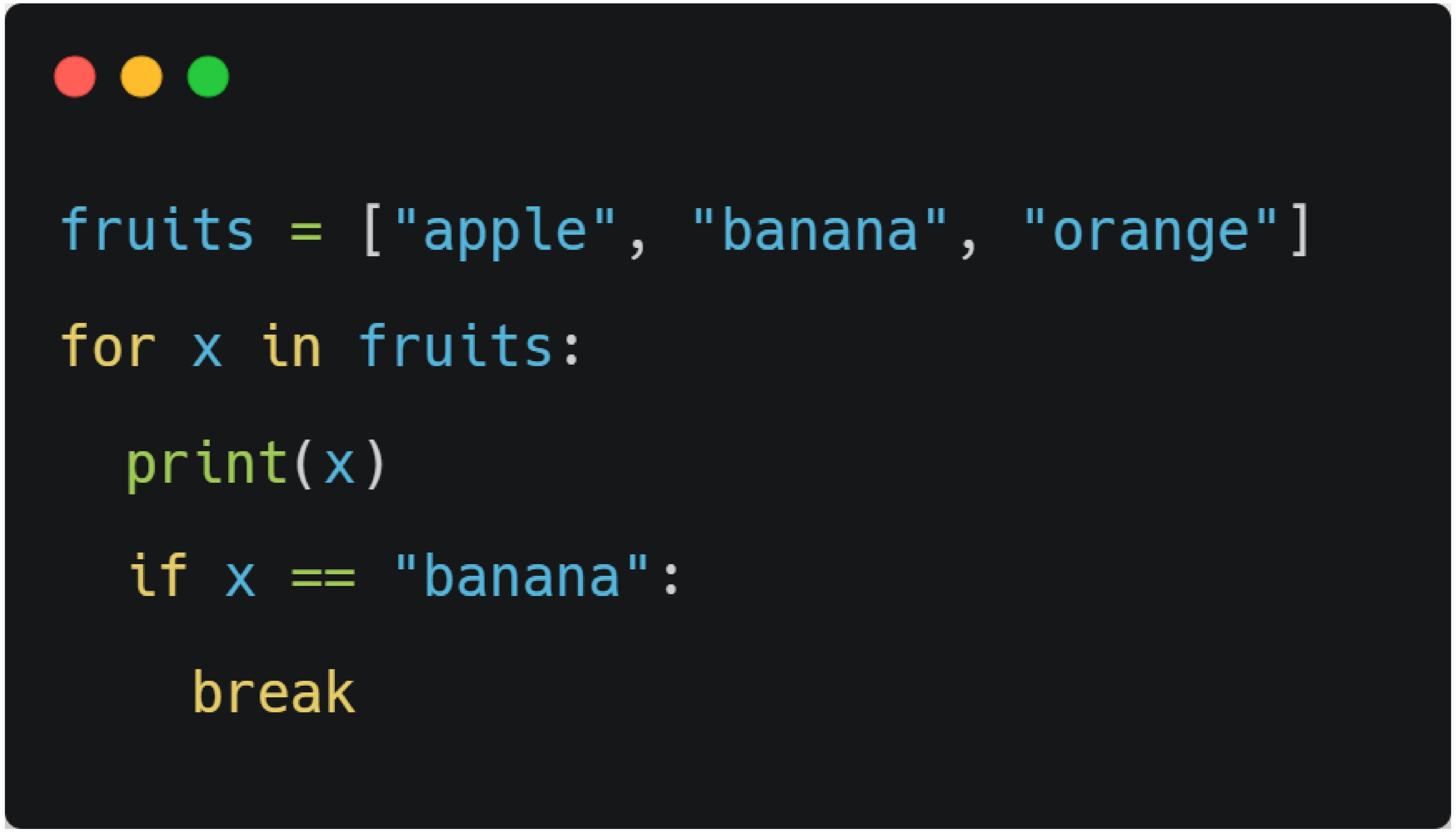


```python
for x in "banana":
  print(x)
```



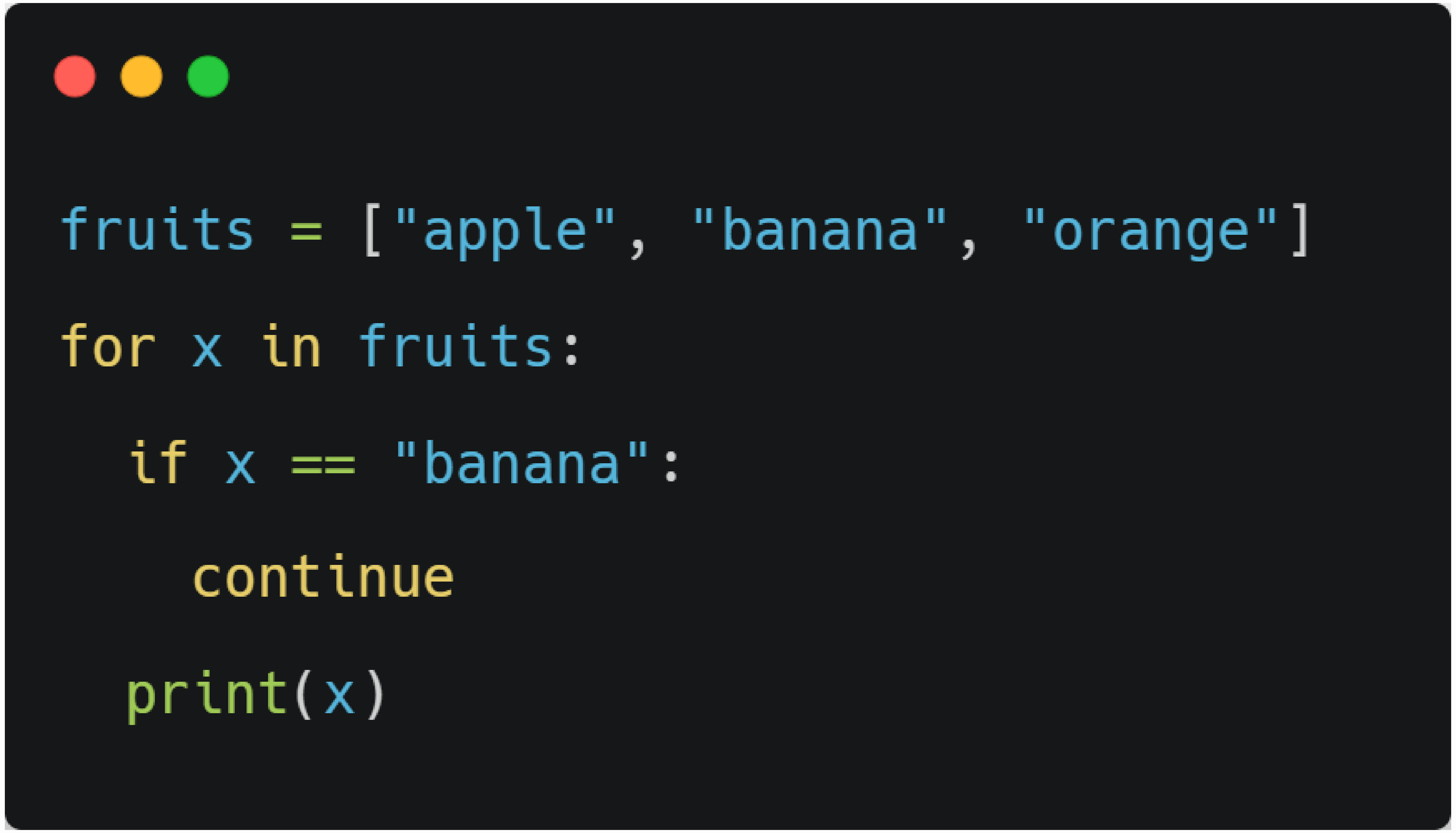
```python
fruits = ["apple", "orange", "banana"]
for x in fruits:
  print(x)
```

Break statement



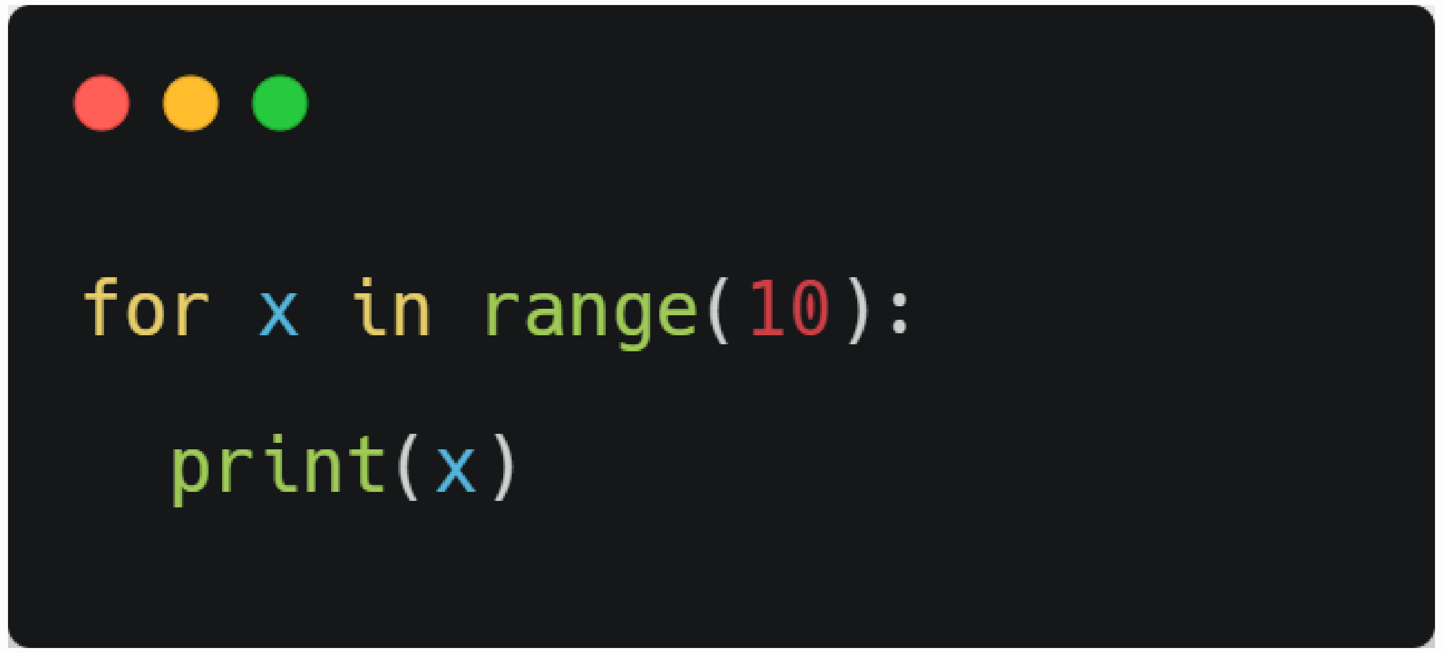
```
fruits = ["apple", "banana", "orange"]
for x in fruits:
  print(x)
  if x == "banana":
    break
```

Continue statement

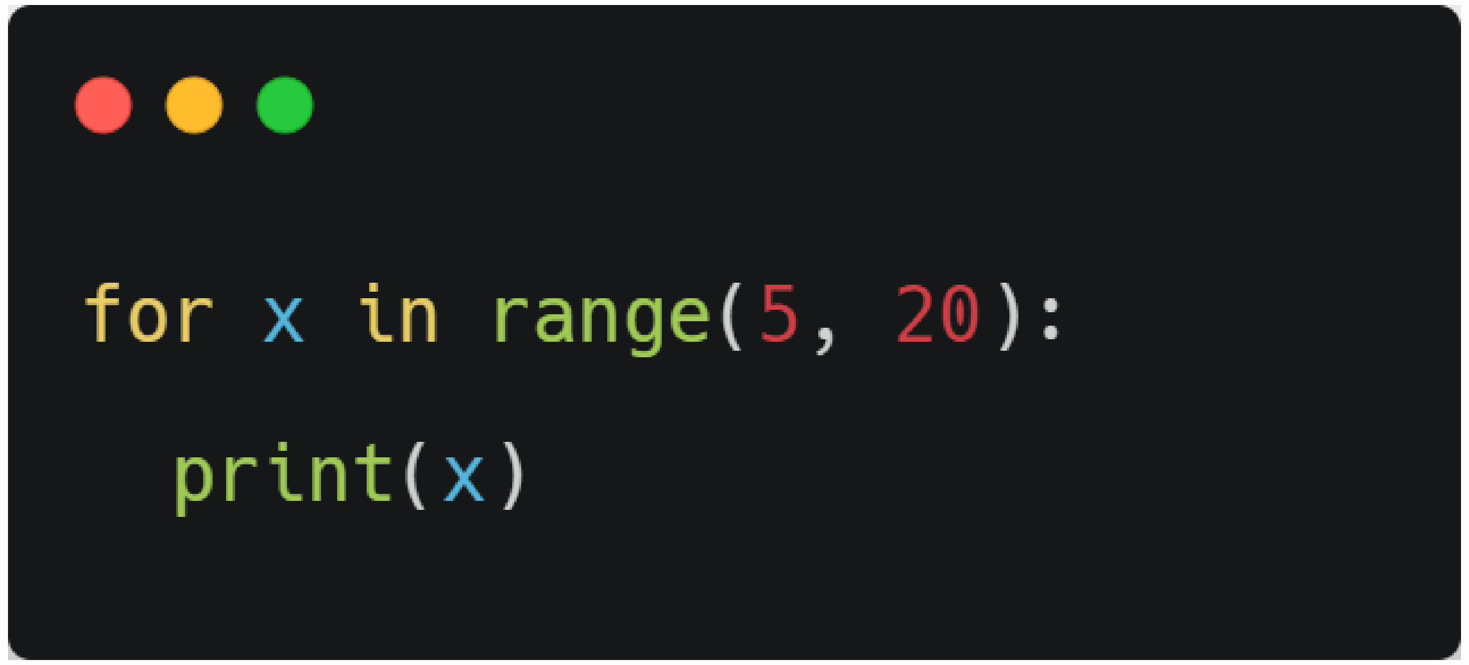


```python
fruits = ["apple", "banana", "orange"]
for x in fruits:
  if x == "banana":
    continue
  print(x)
```

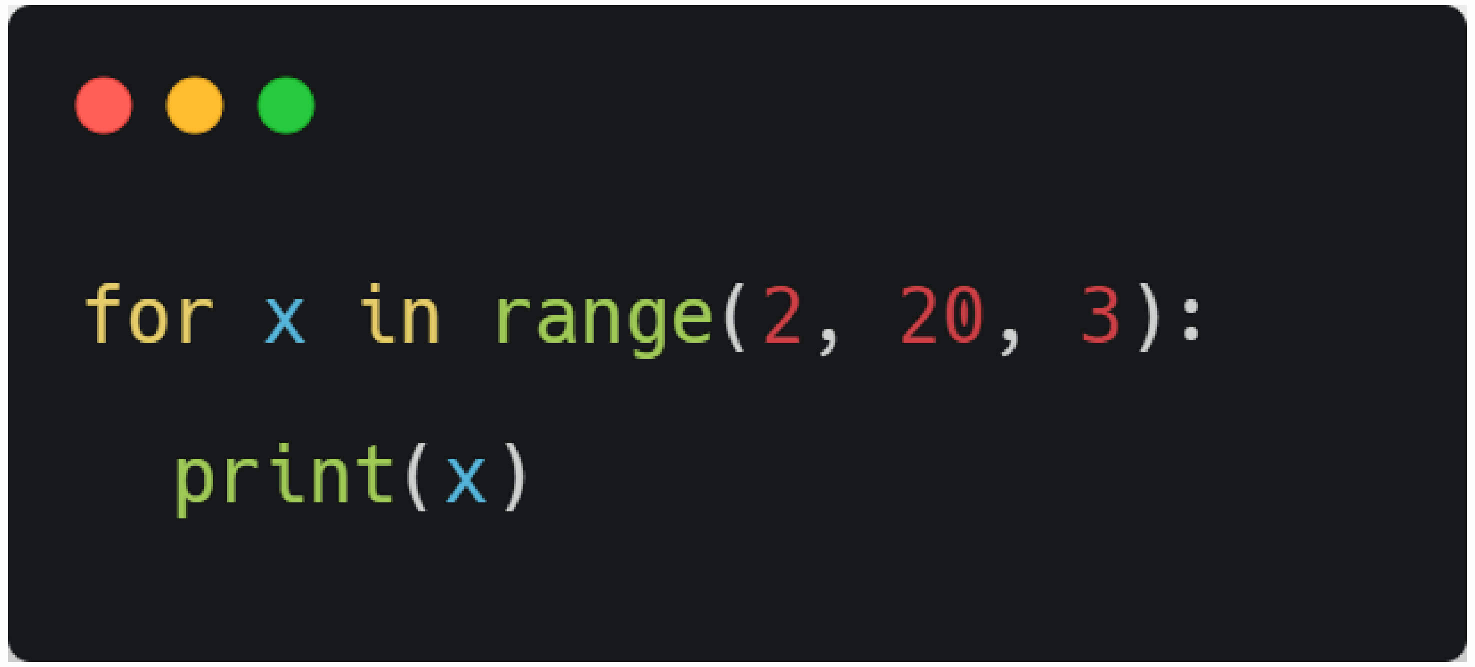
Range



```python
for x in range(10):
  print(x)
```

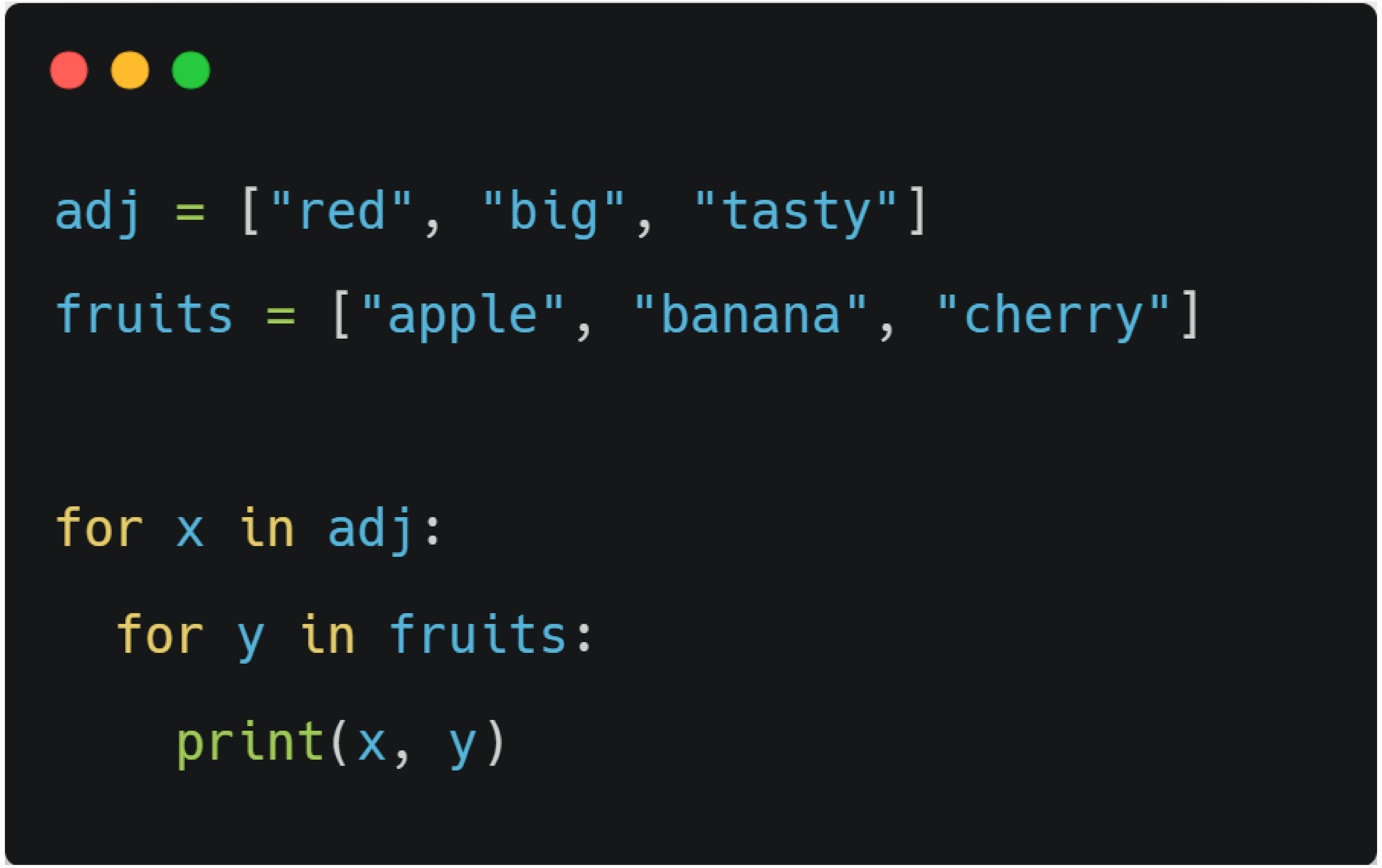


```python
for x in range(5, 20):
  print(x)
```



```python
for x in range(2, 20, 3):
  print(x)
```

Nested for loop

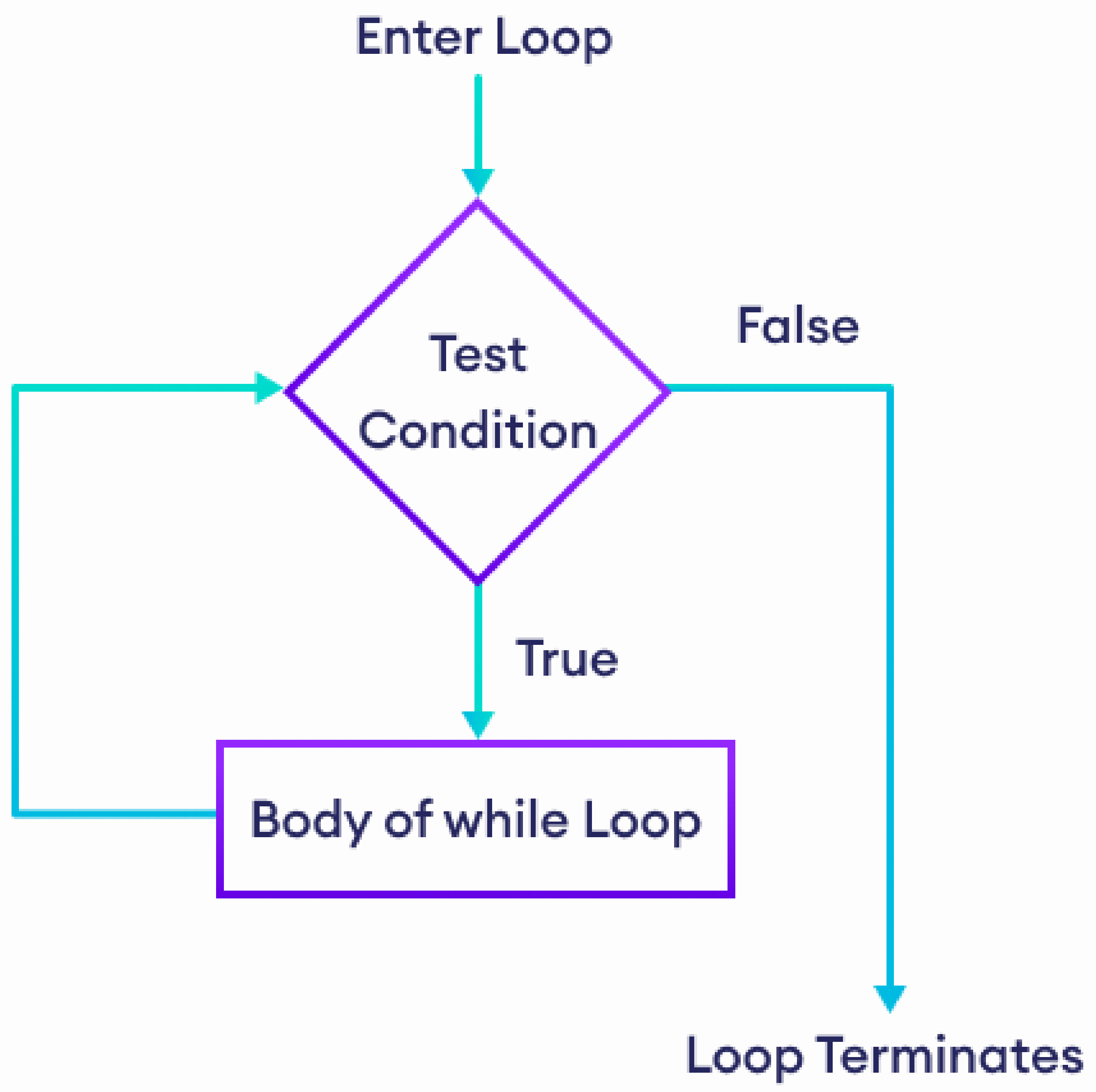


```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
  for y in fruits:
    print(x, y)
```

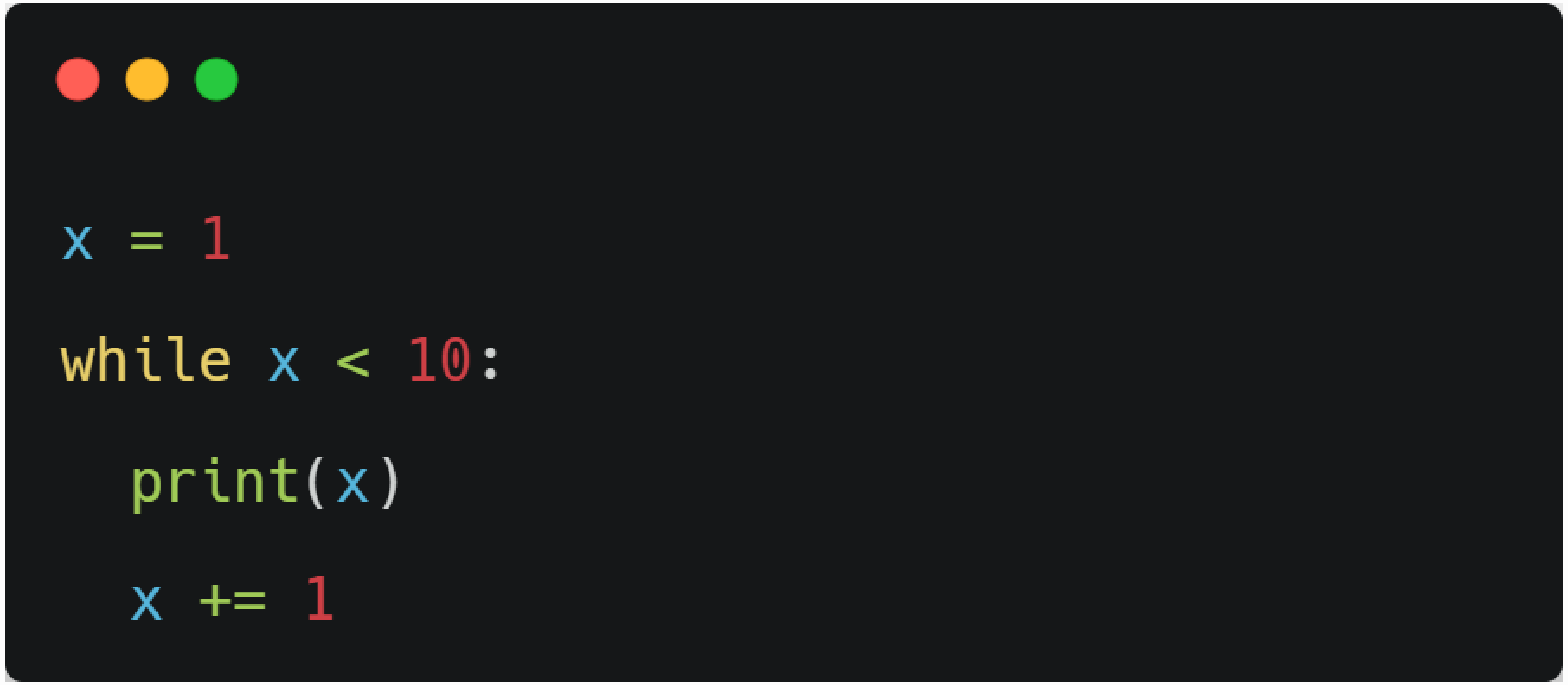
While Loop

While loop

With the while loop we can execute a set of statements as long as a condition is true

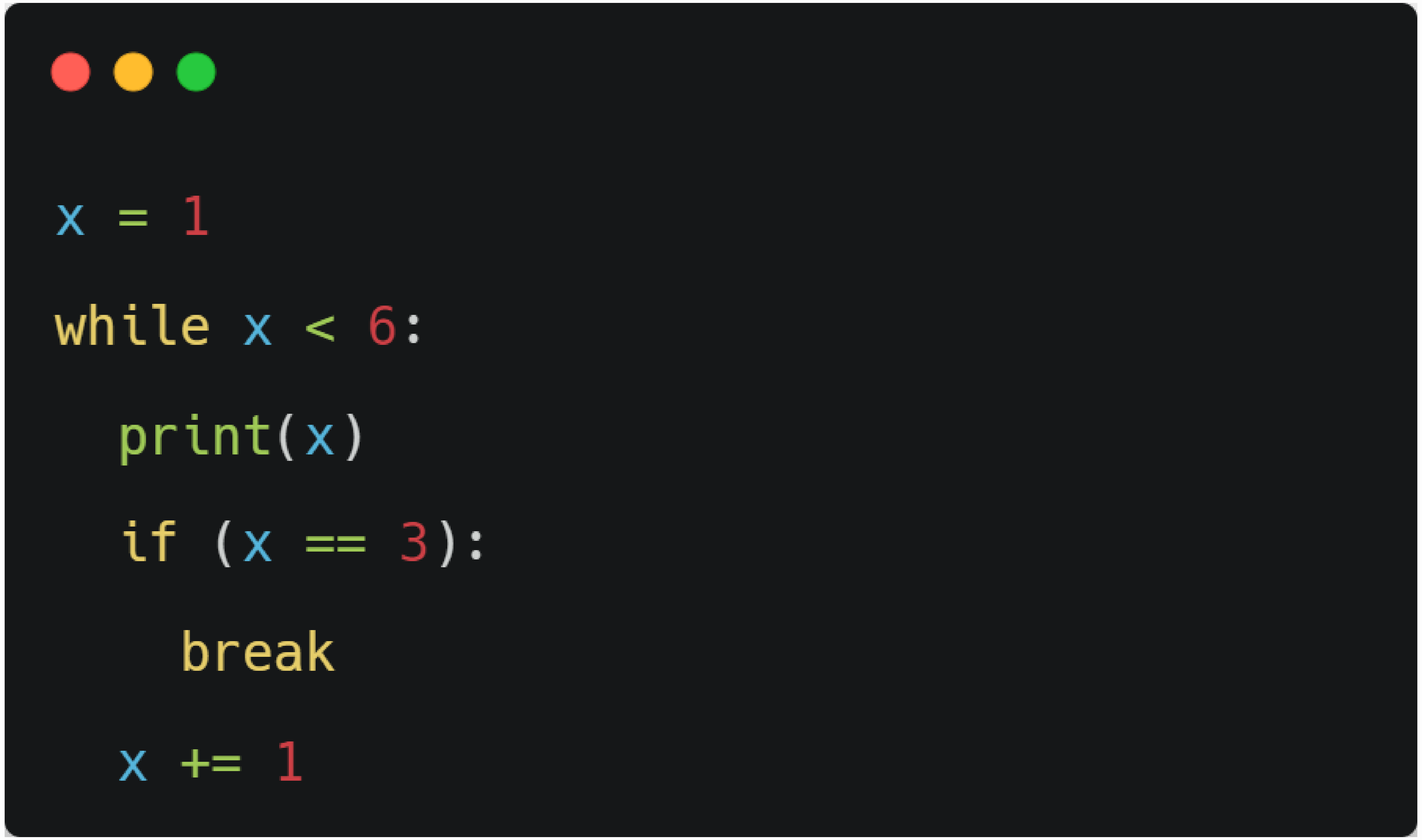


While loop



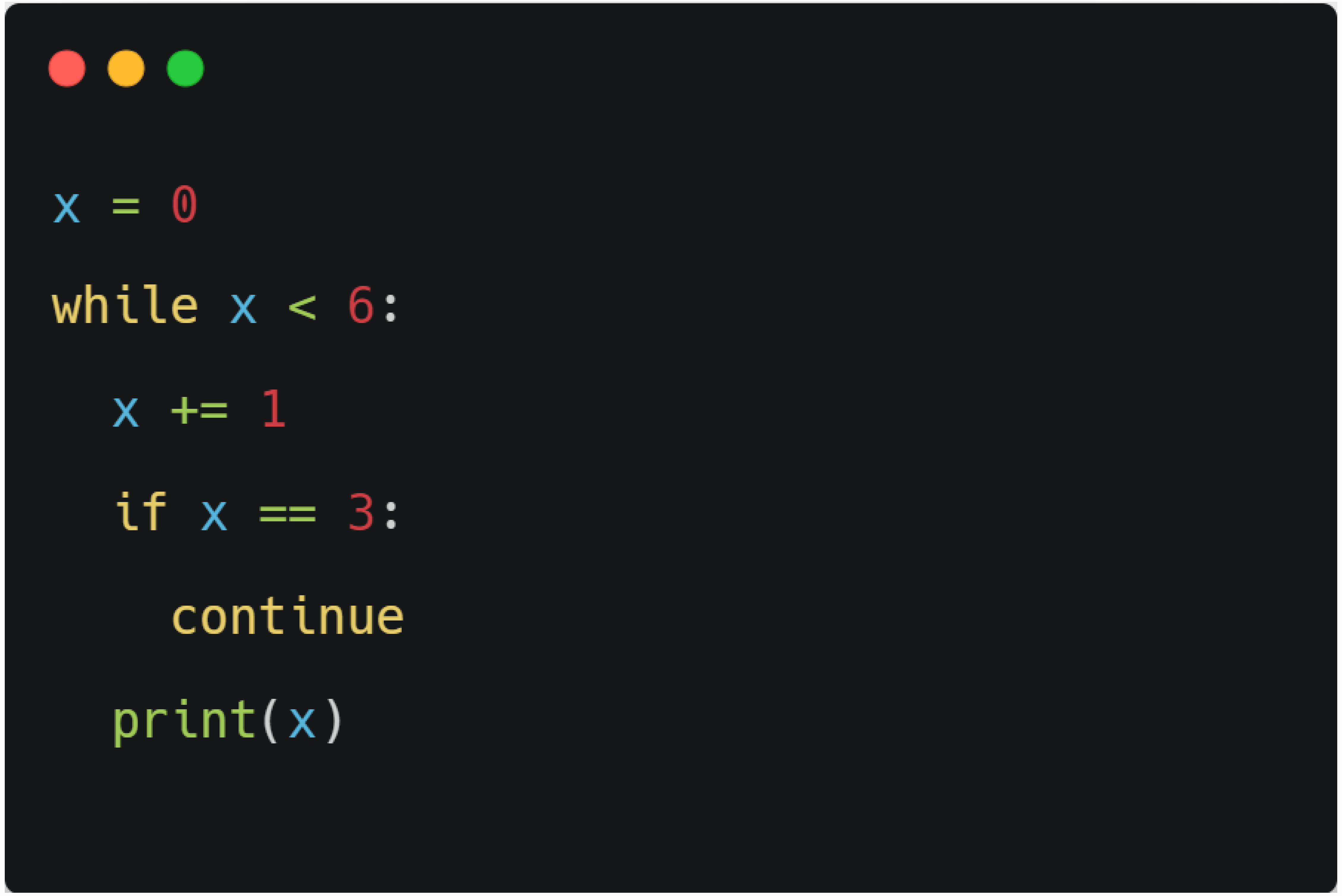
```
x = 1
while x < 10:
  print(x)
  x += 1
```

Break statement



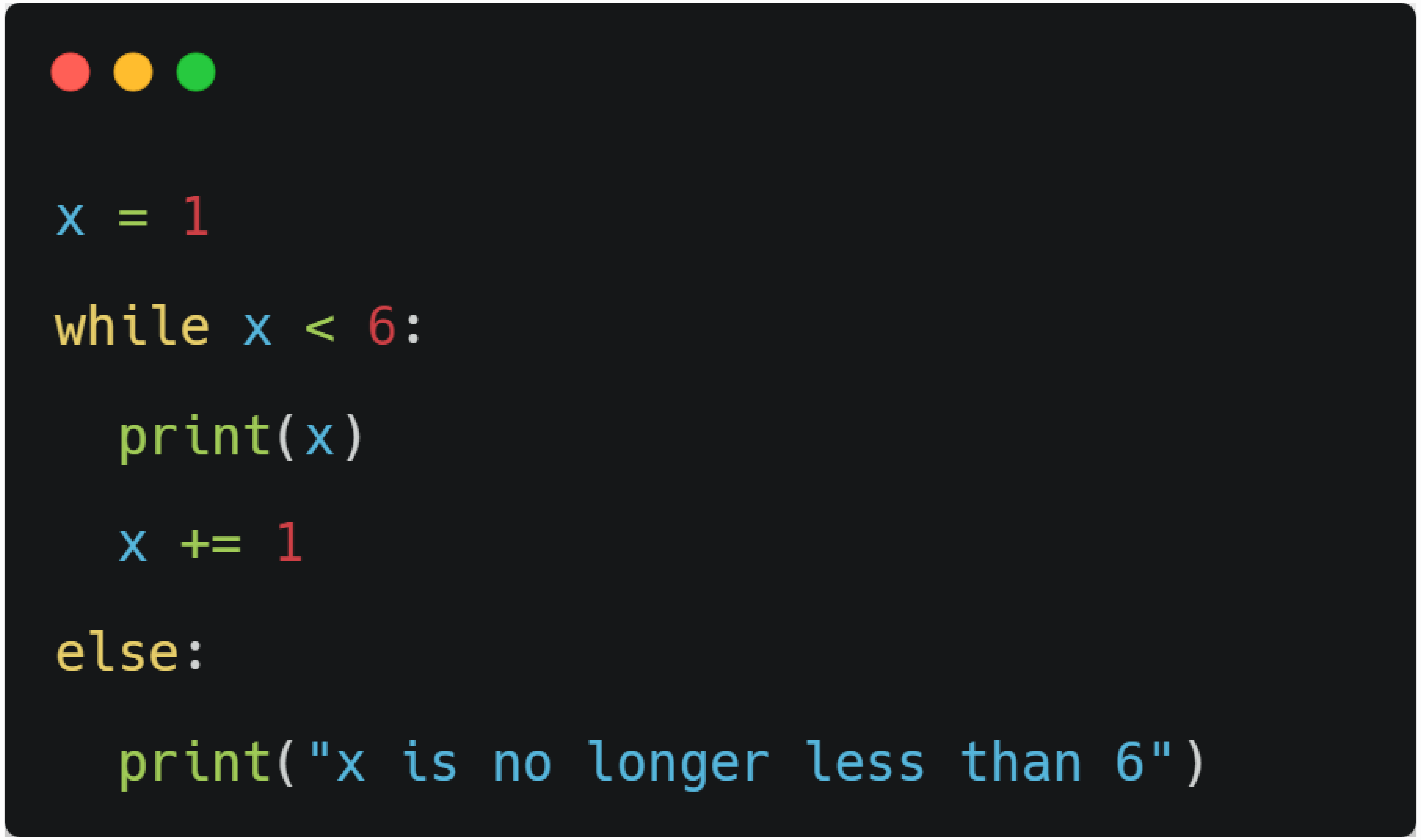
```
x = 1
while x < 6:
  print(x)
  if (x == 3):
    break
  x += 1
```

Continue statement



```
x = 0
while x < 6:
  x += 1
  if x == 3:
    continue
  print(x)
```

Else statement



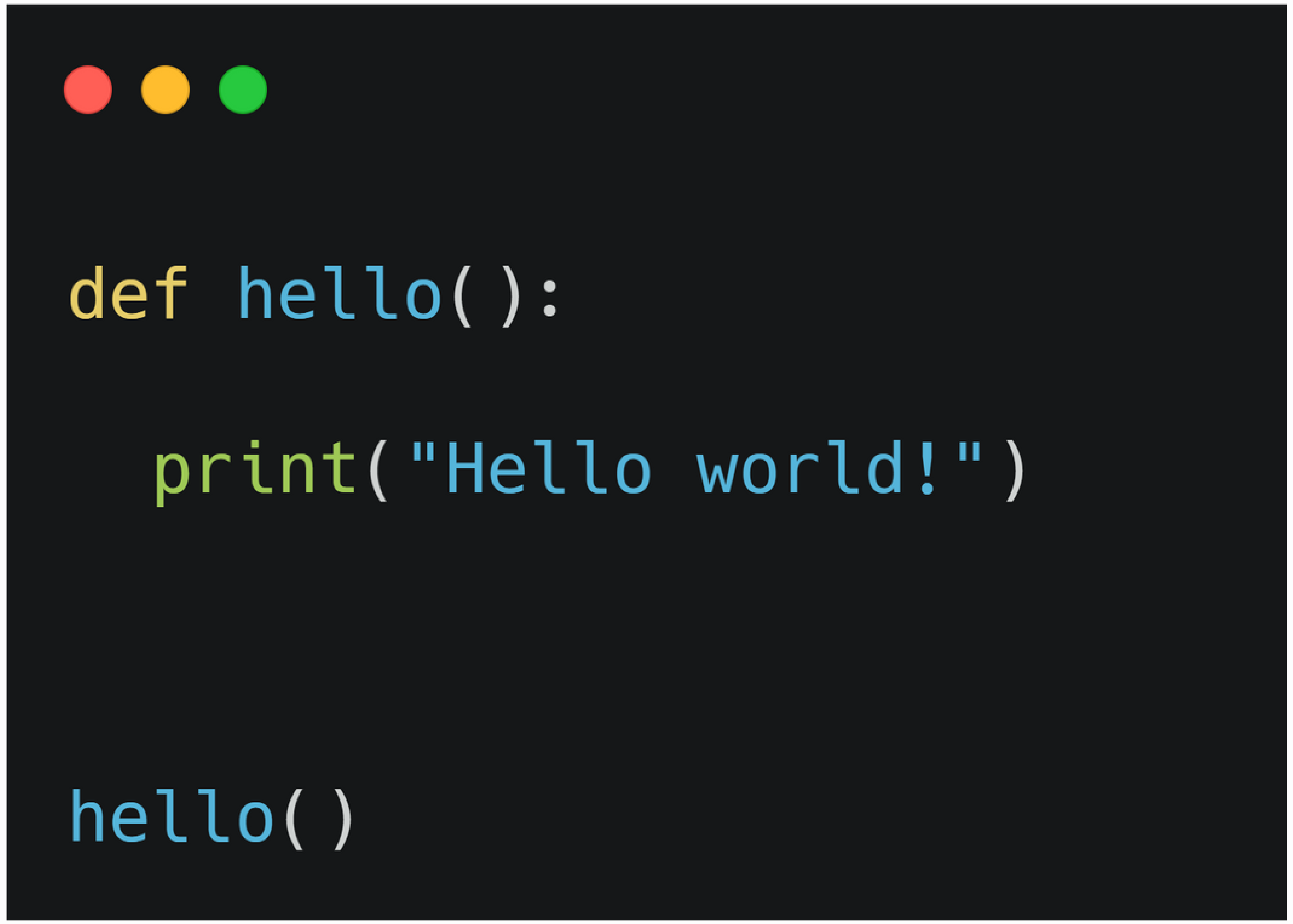
```
x = 1
while x < 6:
  print(x)
  x += 1
else:
  print("x is no longer less than 6")
```

Functions

Functions

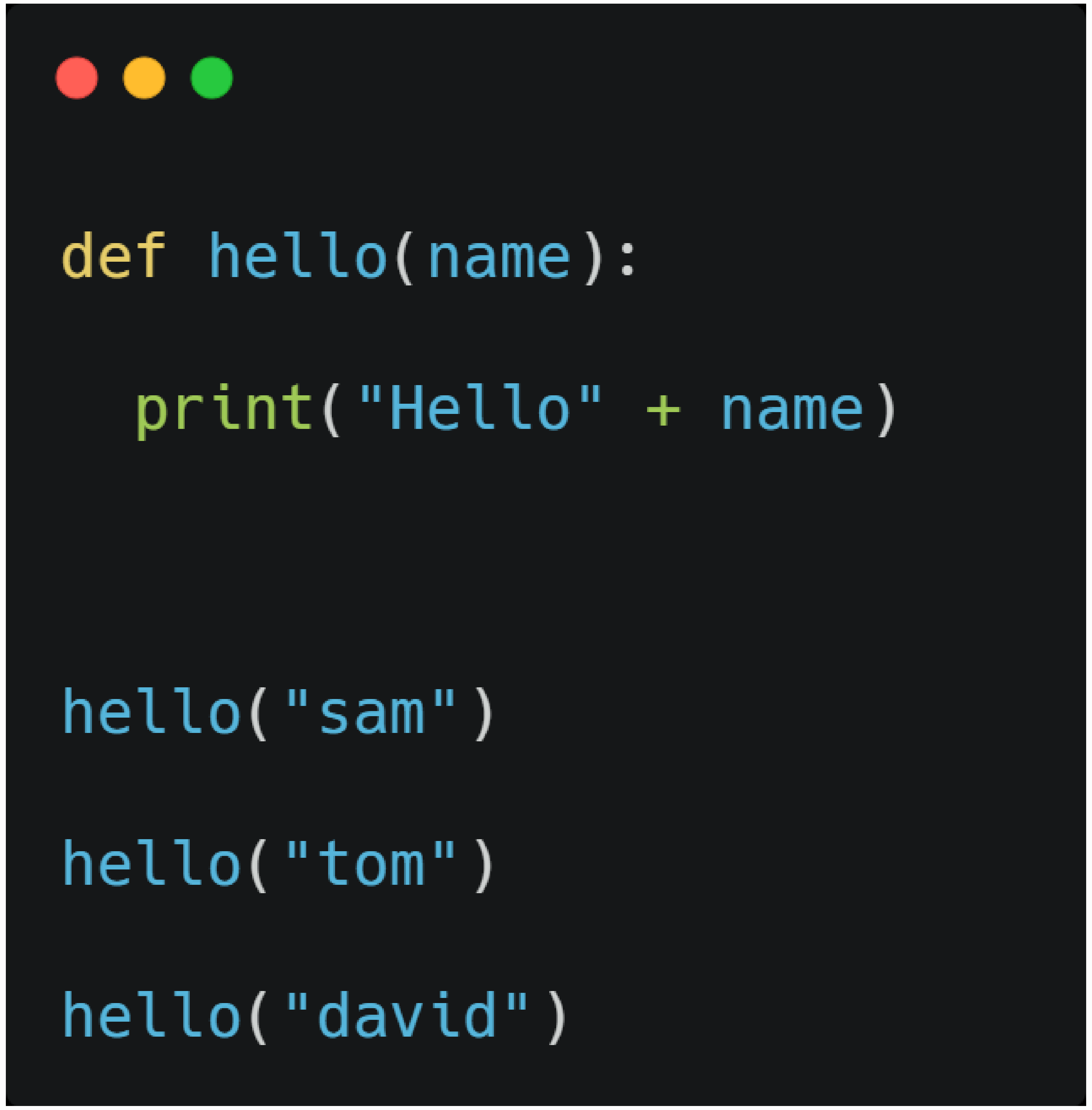
- Functions in Python are blocks of reusable code that perform specific tasks
- A function runs only when it is called
- They are defined using the "def" keyword
- They can be called multiple times

Functions



```
def hello():
  print("Hello world!")


hello()
```

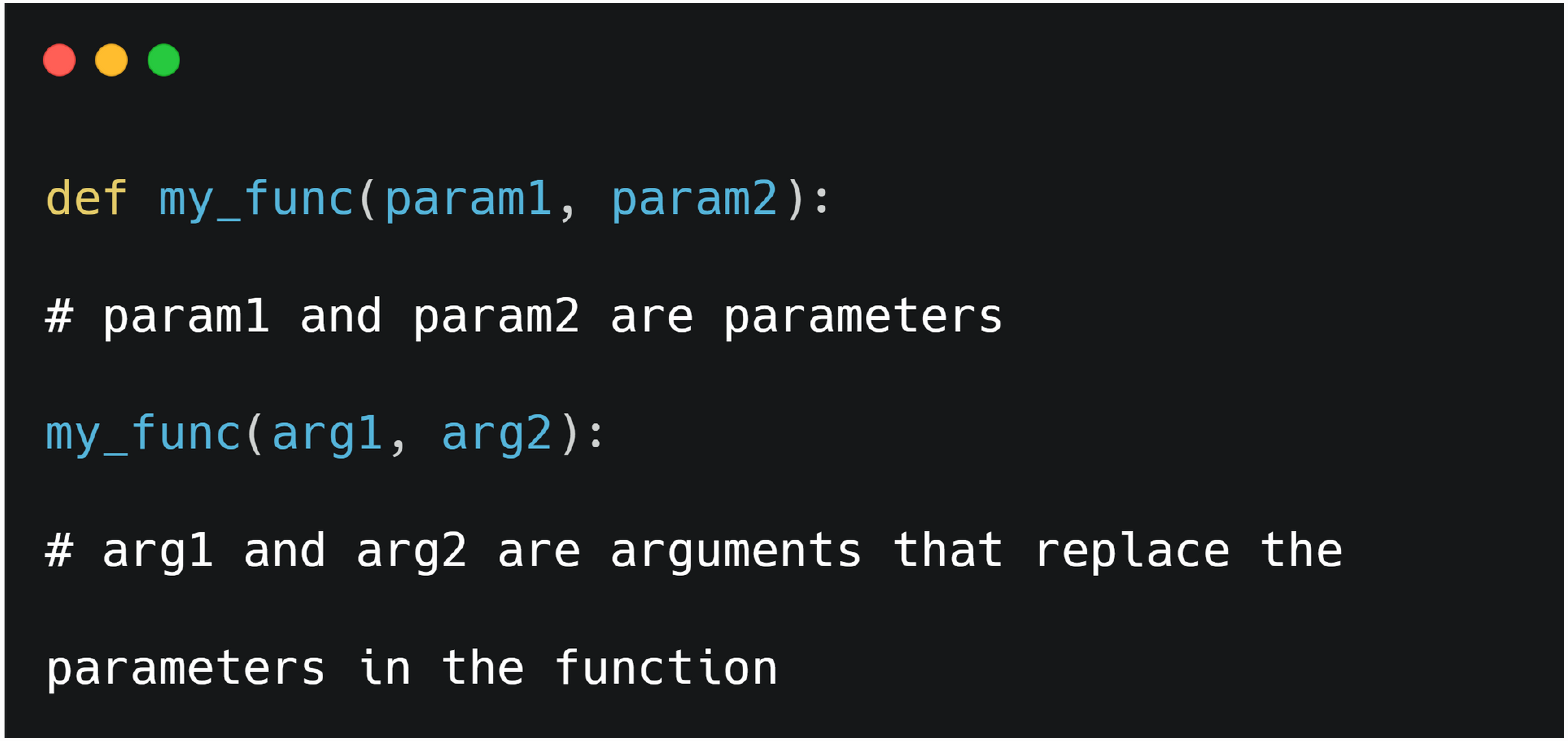
Parameters and arguments



```python
def hello(name):
  print("Hello" + name)


hello("sam")
hello("tom")
hello("david")
```

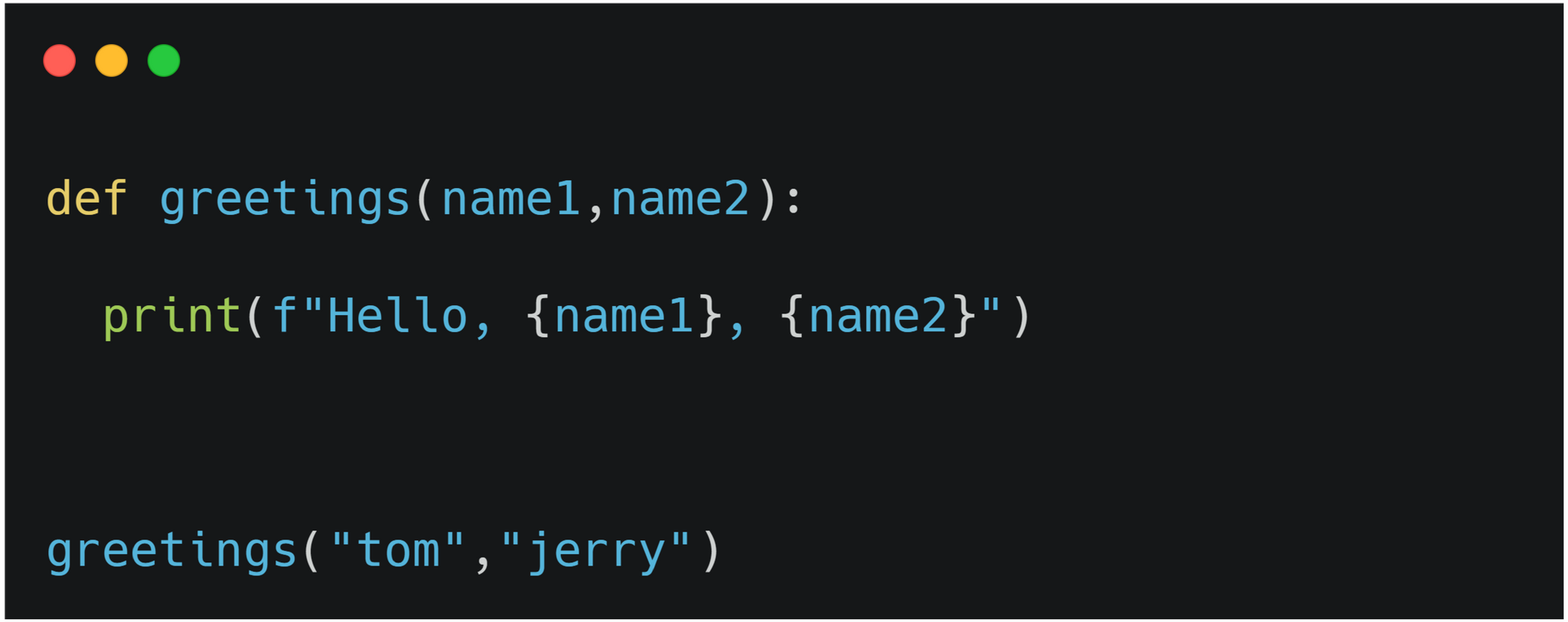
- Information can be passed into functions as arguments
- Arguments are specified after the function name, inside the parentheses
- You can add as many arguments as you want, just separate them with a comma

Parameters and arguments



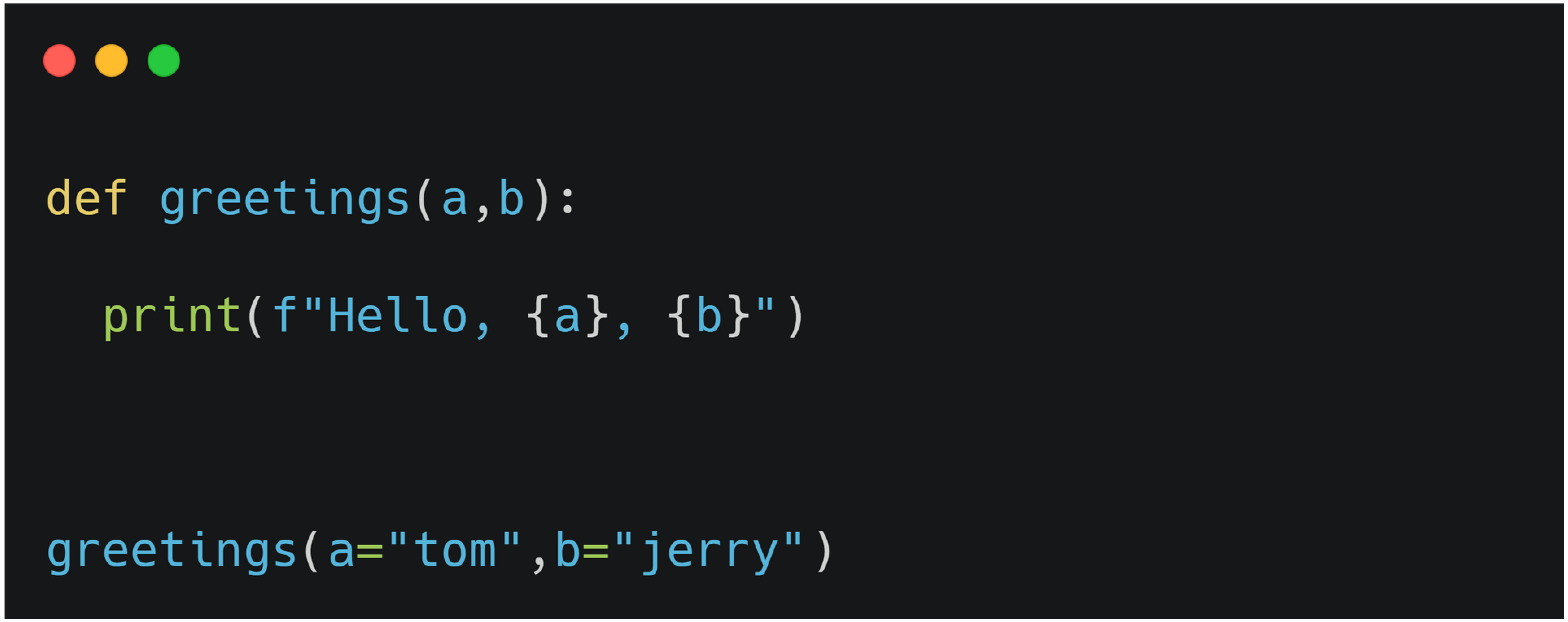
```
def my_func(param1, param2):

# param1 and param2 are parameters

my_func(arg1, arg2):

# arg1 and arg2 are arguments that replace the

parameters in the function
```

Positional Arguments



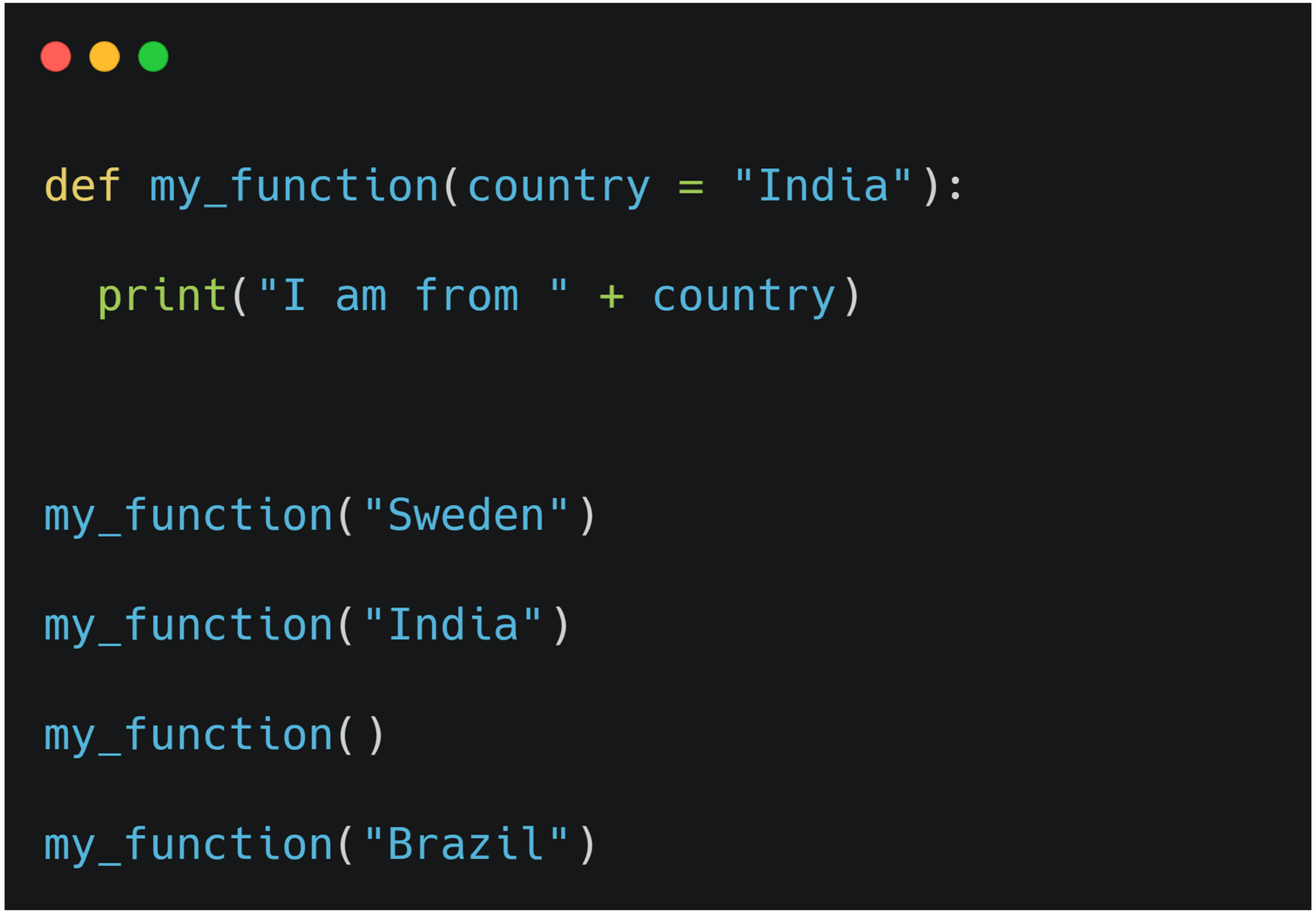
```
def greetings(name1,name2):
  print(f"Hello, {name1}, {name2}")


greetings("tom","jerry")
```

Keyword Arguments



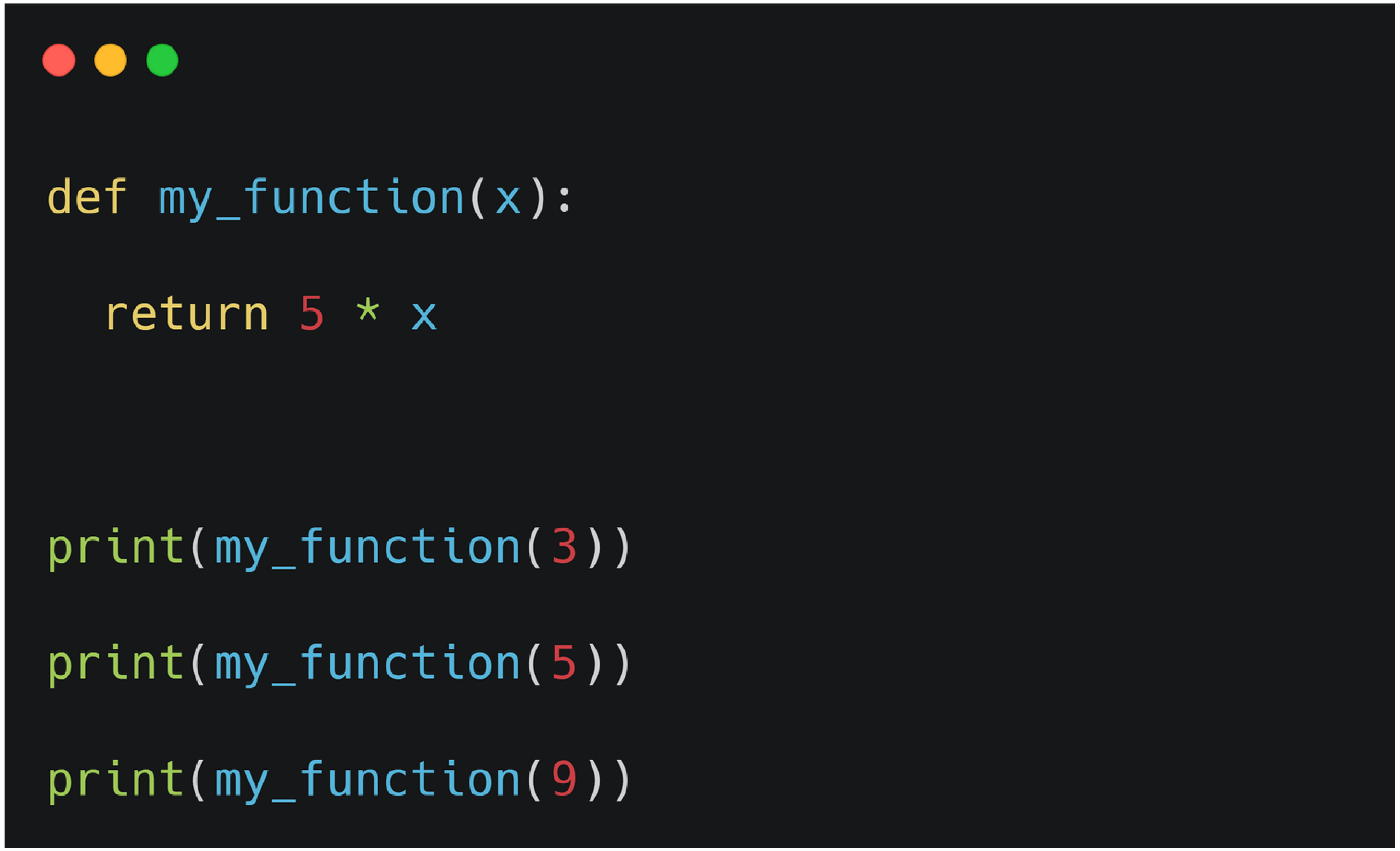
```python
def greetings(a,b):

  print(f"Hello, {a}, {b}")



greetings(a="tom",b="jerry")
```

Default parameter value



```
def my_function(country = "India"):
  print("I am from " + country)


my_function("Sweden")

my_function("India")

my_function()

my_function("Brazil")
```

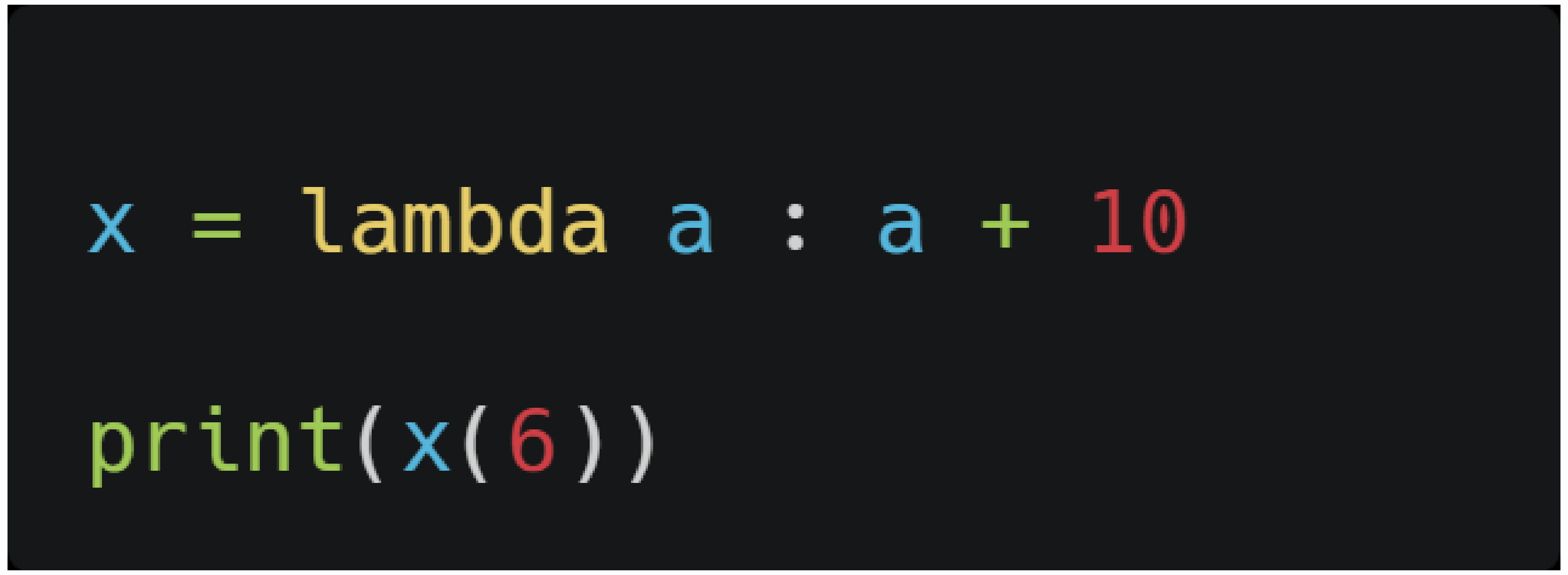
Return values



```python
def my_function(x):
  return 5 * x


print(my_function(3))
print(my_function(5))
print(my_function(9))
```

Lambda functions

Lambda

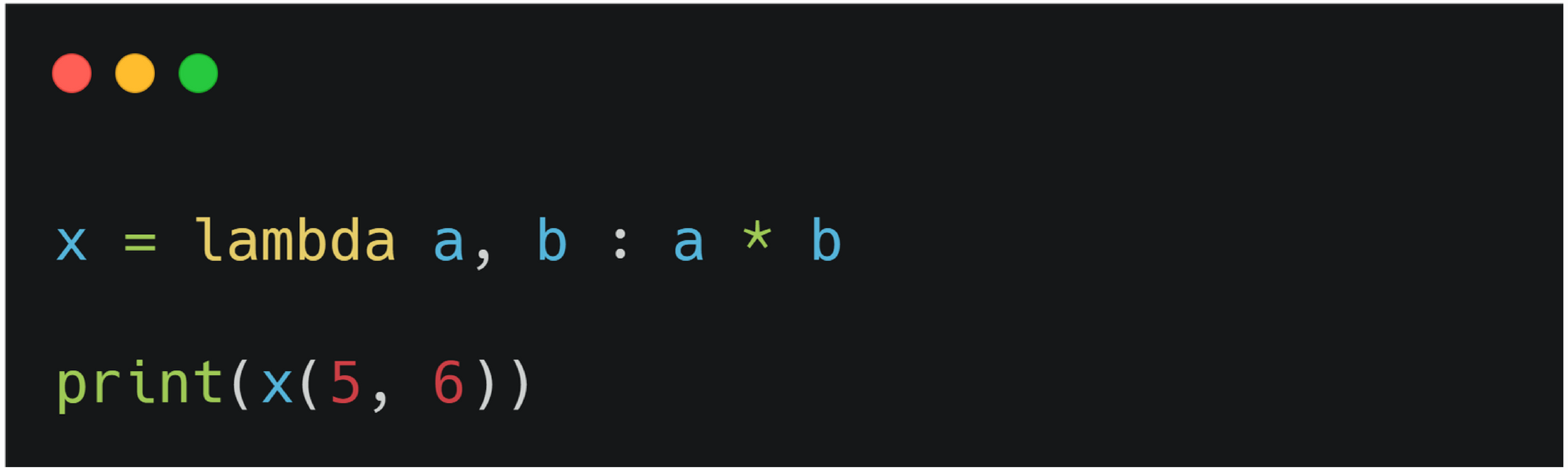


```
x = lambda a : a + 10

print(x(6))
```

- Lambda functions in Python are small, anonymous, single-expression functions that are defined using the lambda keyword

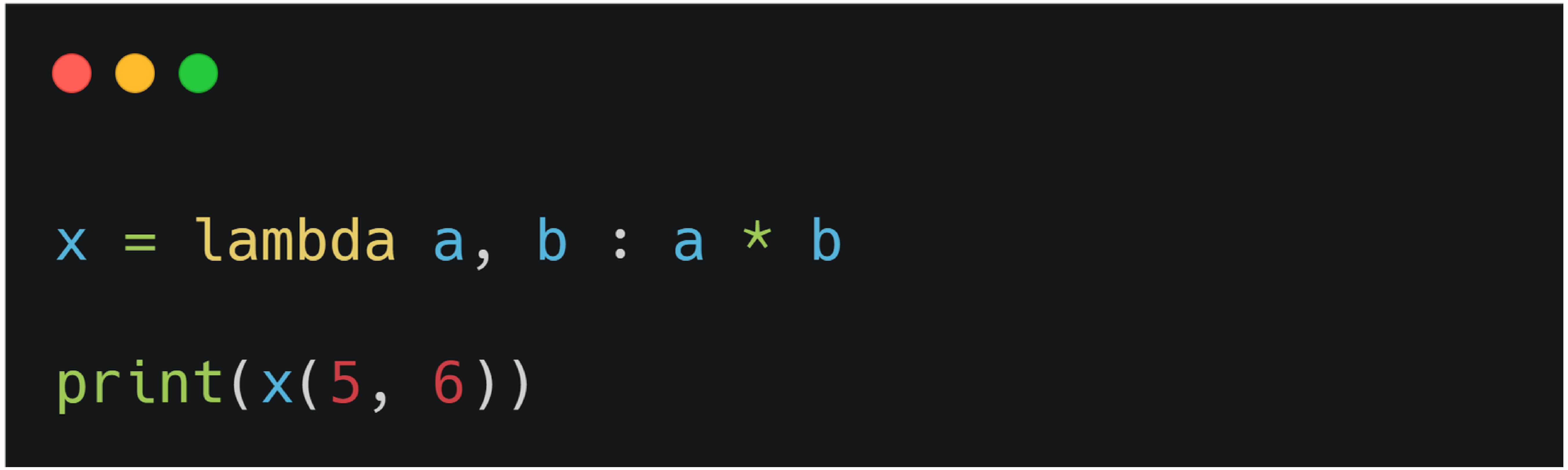
- They are used for quick, throw-away functions that are needed for a short period of time

Lambda



```
x = lambda a, b : a * b

print(x(5, 6))
```

Lambda



```
x = lambda a, b : a * b

print(x(5, 6))
```