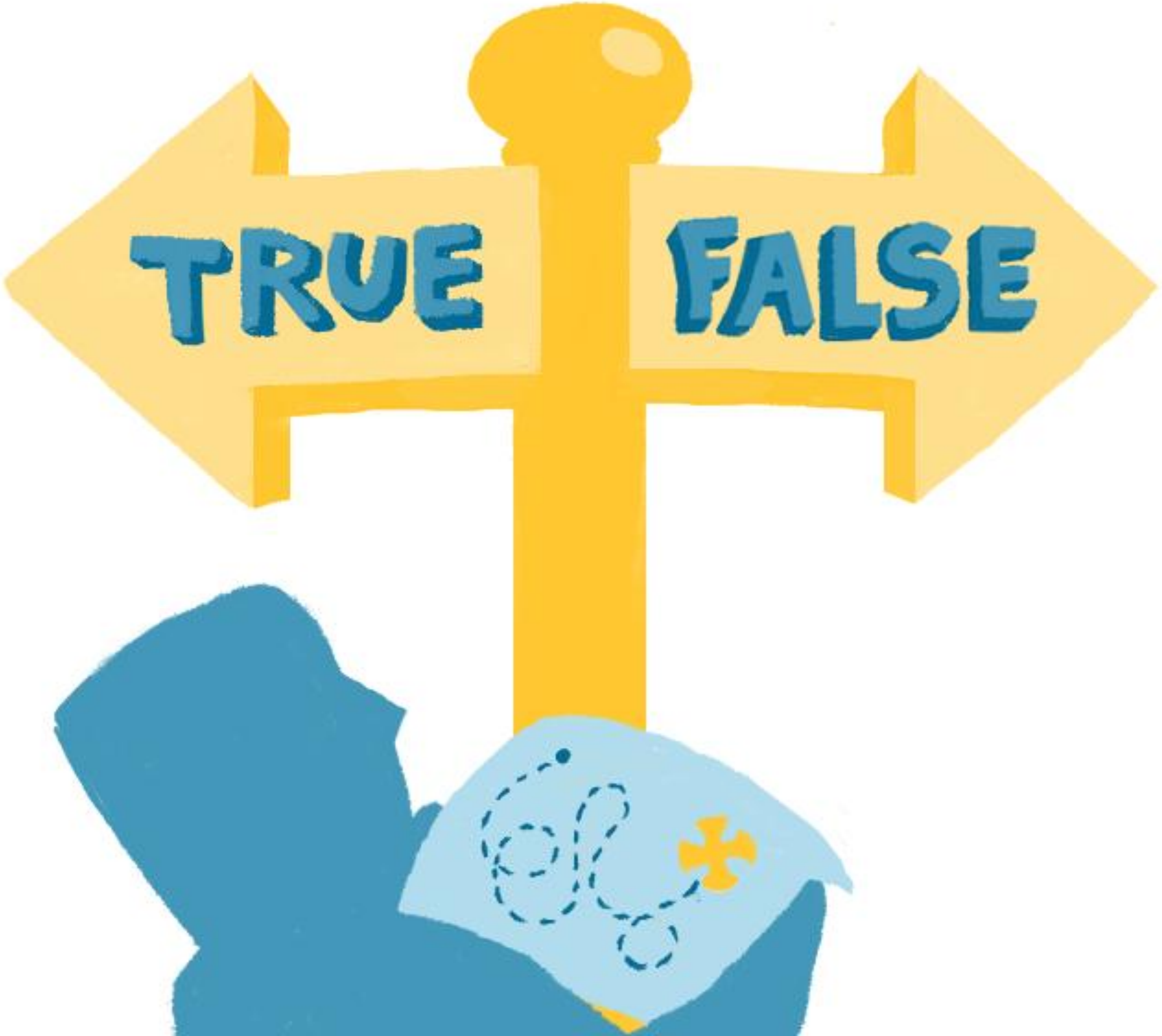
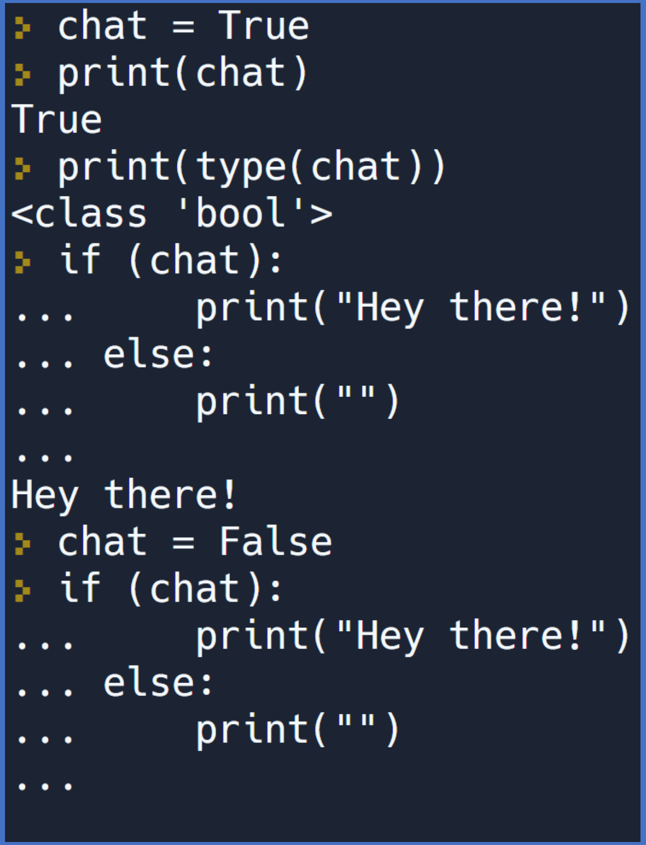
Programming for Social Scientists

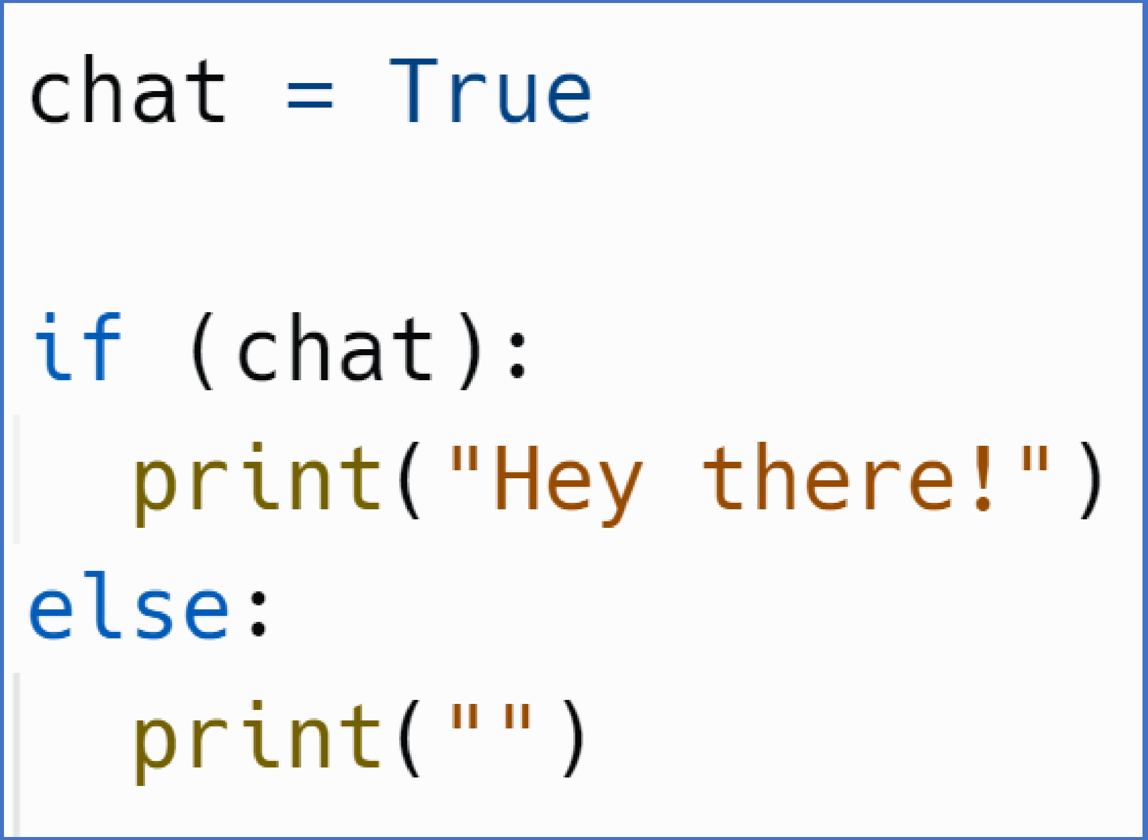
Conditional expressions

Johan A. Dornschneider-Elkink

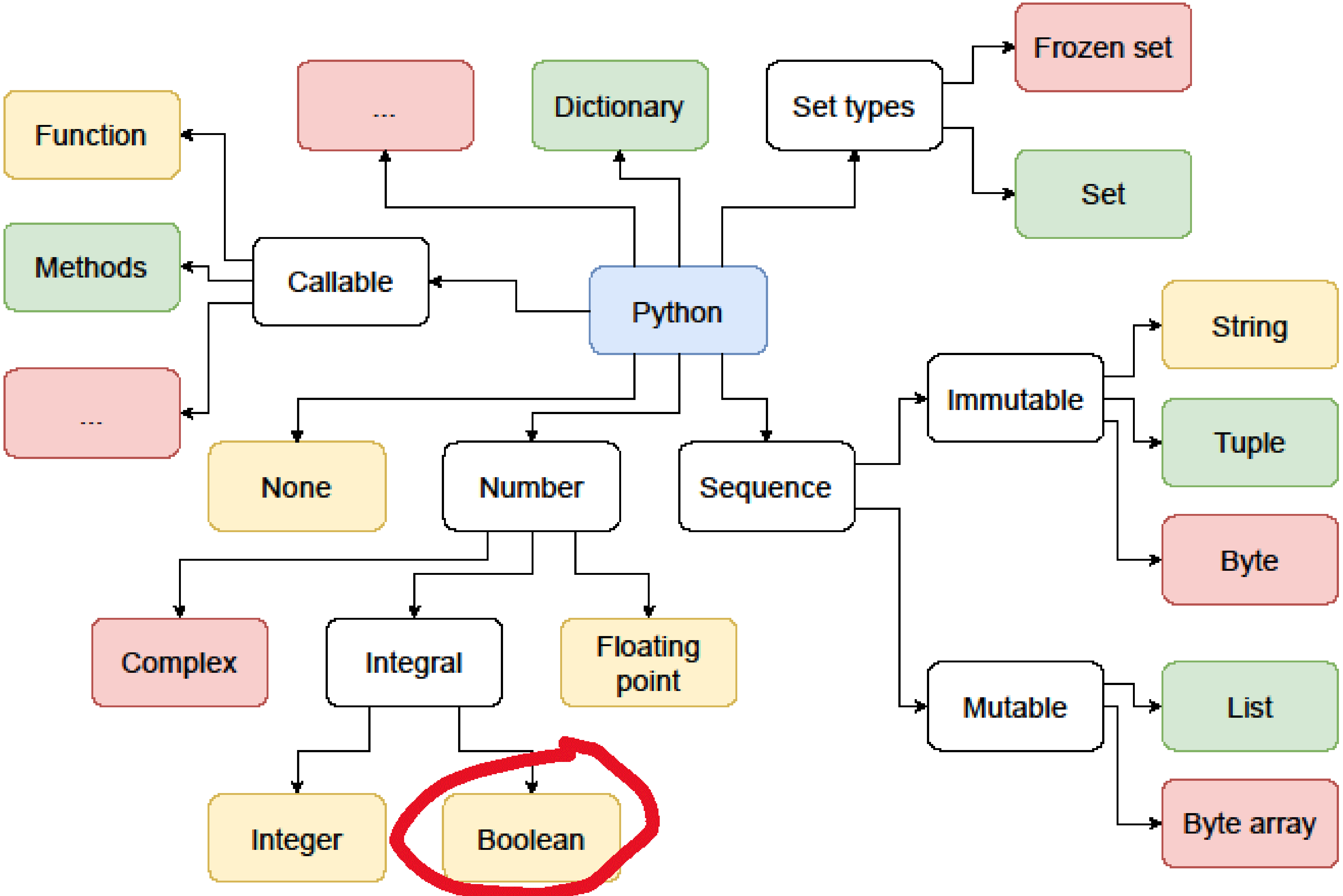

TRUE
FALSE



```
> chat = True
> print(chat)
True
> print(type(chat))
<class 'bool'>
> if (chat):
...     print("Hey there!")
... else:
...     print("")
...
Hey there!
> chat = False
> if (chat):
...     print("Hey there!")
... else:
...     print("")
...
```



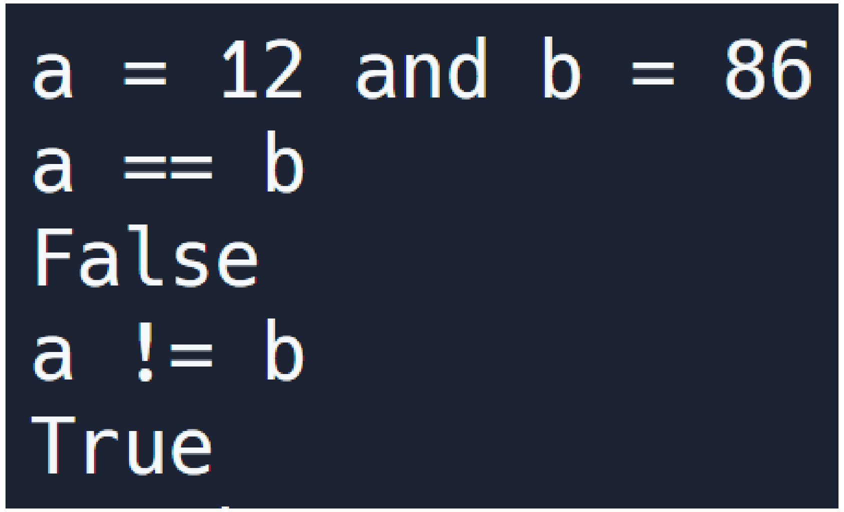
```python
chat = True

if (chat):
  print("Hey there!")
else:
  print("")
```


Frozen set
Function
...
Dictionary
Set types
Set
Methods
Callable
Python
String
...
Immutable
Tuple
None
Number
Sequence
Byte
Complex
Integral
Floating point
Mutable
List
Integer
Boolean
Byte array

==	equal	!=	not equal
>	greater than	<	less than
>=	greater than or equal	<=	less than or equal

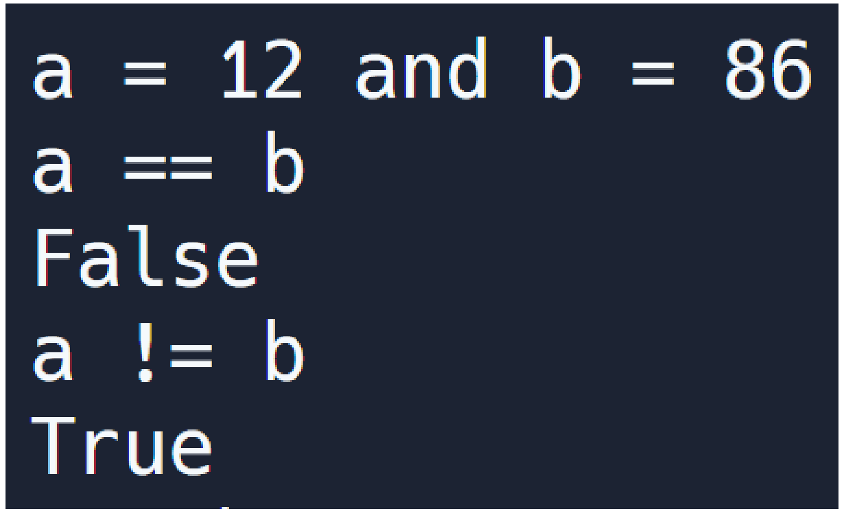
```python
import random

a = random.randint(0, 100)
b = random.randint(0, 100)

print("a = %d and b = %d" % (a, b))
```

```python
print("a == b")
print(a == b)

print("a != b")
print(a != b)
```



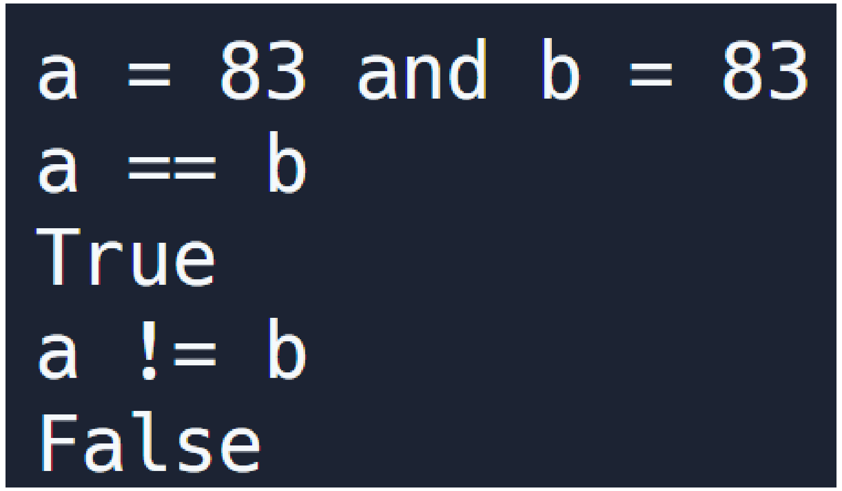
```
a = 12 and b = 86
a == b
False
a != b
True
```

```python
import random

a = random.randint(0, 100)
b = random.randint(0, 100)

print("a = %d and b = %d" % (a, b))
```

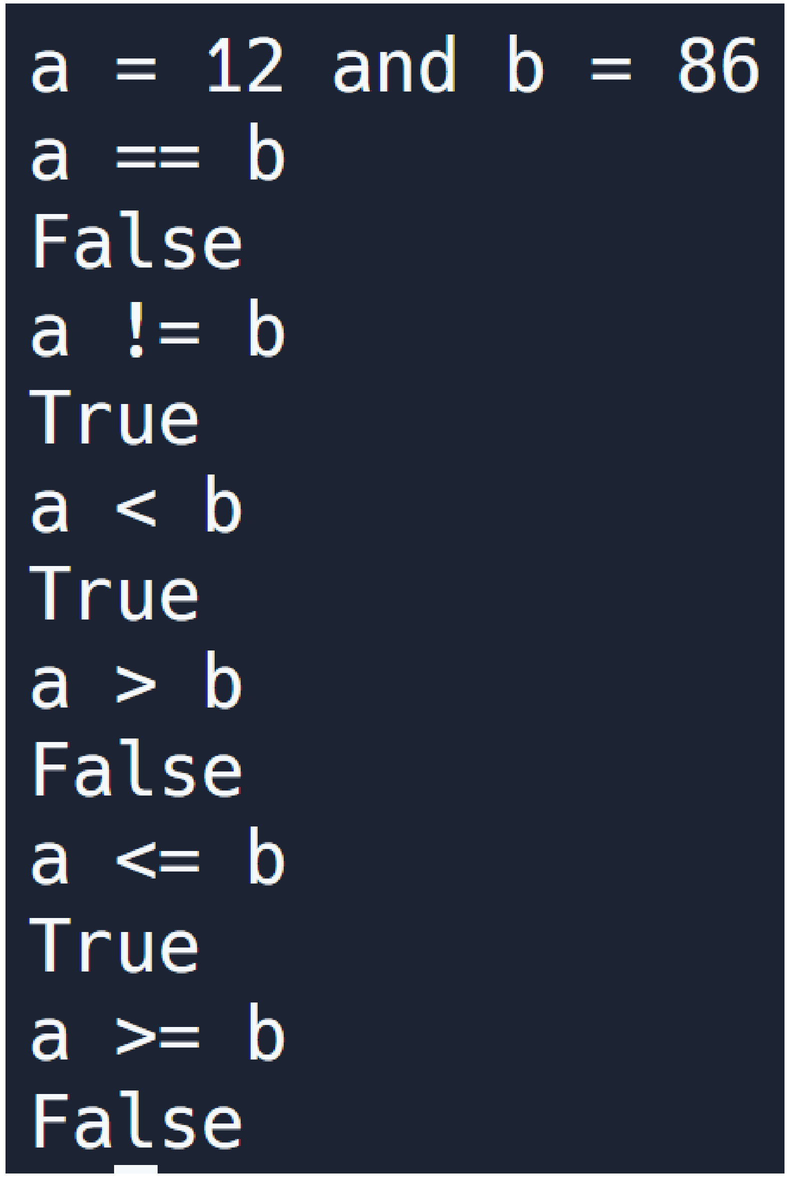


```
a = 12 and b = 86
a == b
False
a != b
True
```

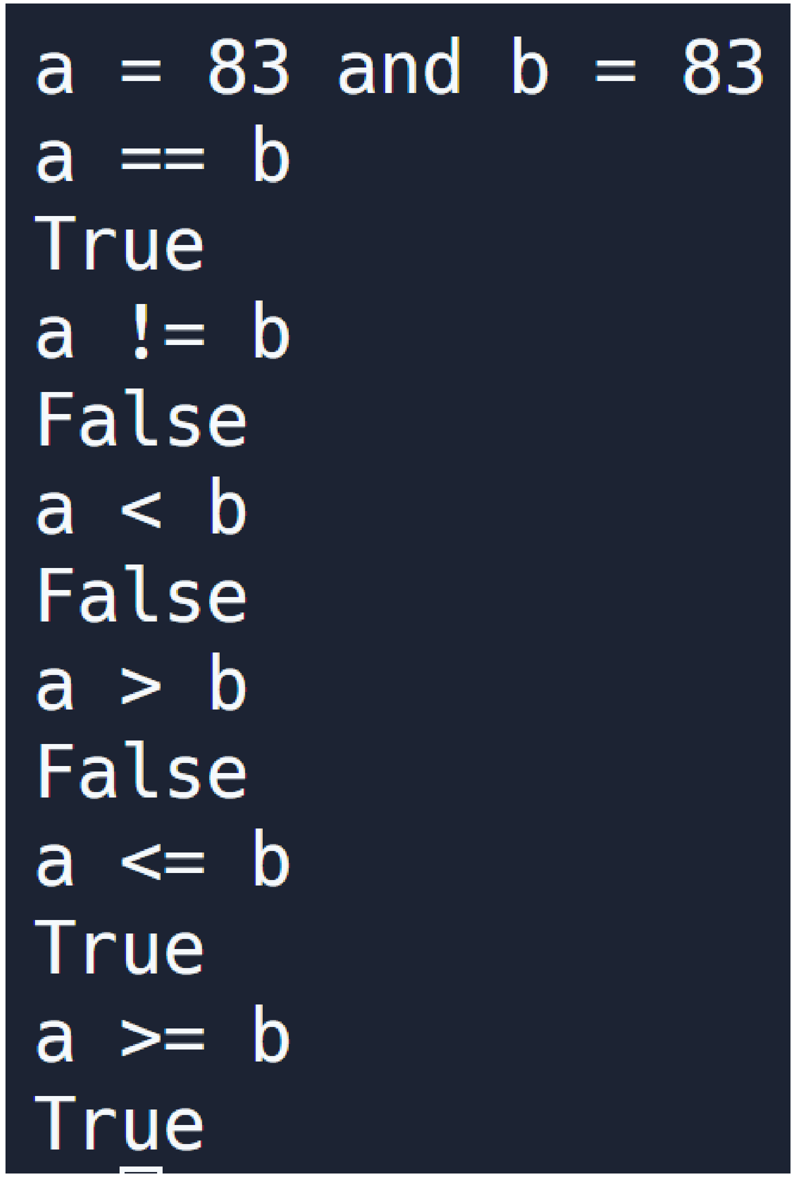
```python
print("a == b")
print(a == b)

print("a != b")
print(a != b)
```



```
a = 83 and b = 83
a == b
True
a != b
False
```



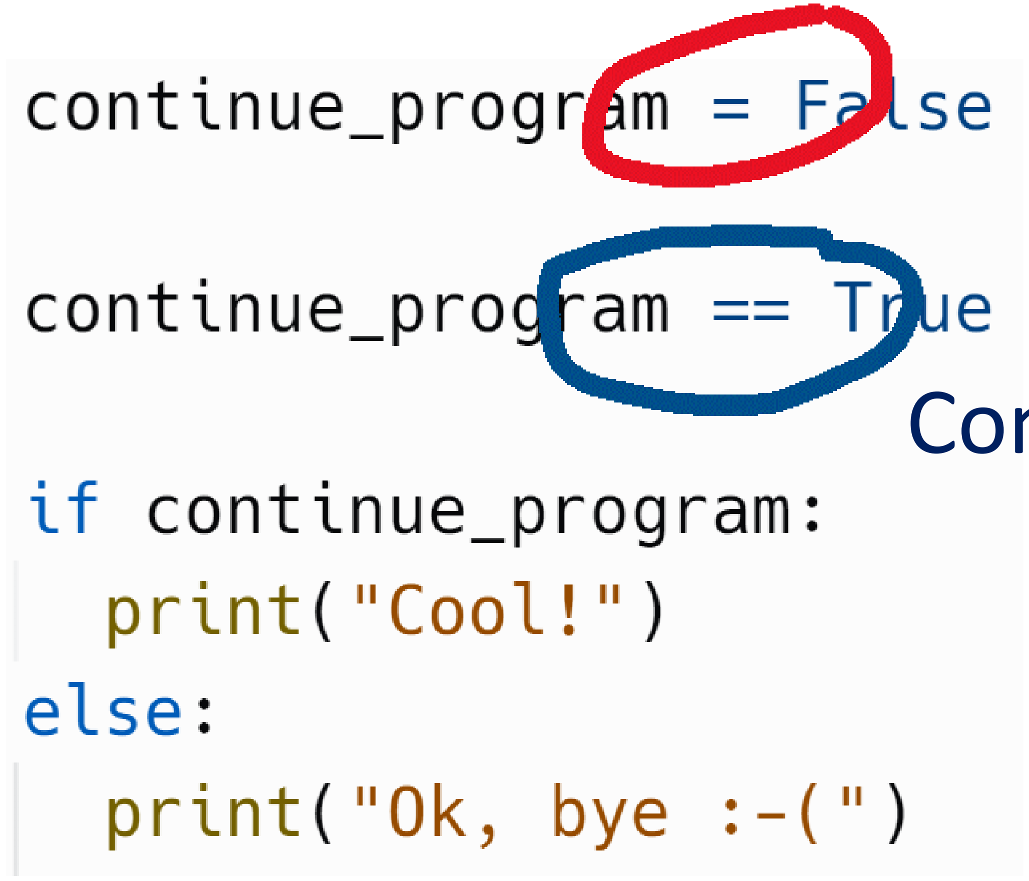
```
a = 12 and b = 86
a == b
False
a != b
True
a < b
True
a > b
False
a <= b
True
a >= b
False
```



```
a = 83 and b = 83
a == b
True
a != b
False
a < b
False
a > b
False
a <= b
True
a >= b
True
```

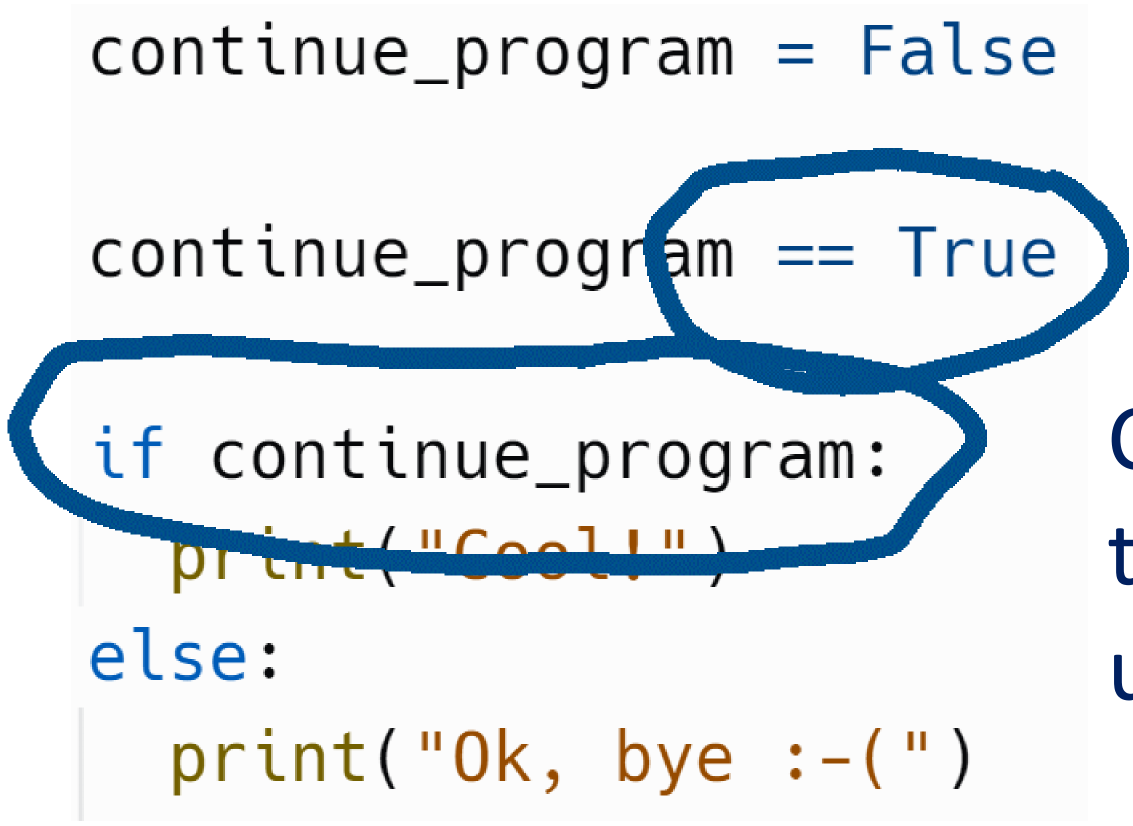
```python
continue_program = False

continue_program == True

if continue_program:
  print("Cool!")
else:
  print("Ok, bye :-(")
```

Assignment



```
continue_program = False

continue_program == True

if continue_program:
  print("Cool!")
else:
  print("Ok, bye :-(")
```

Comparison



```
continue_program = False

continue_program == True

if continue_program:
  print("Cool!")
else:
  print("Ok, bye :-(")
```

Comparison to True is unnecessary

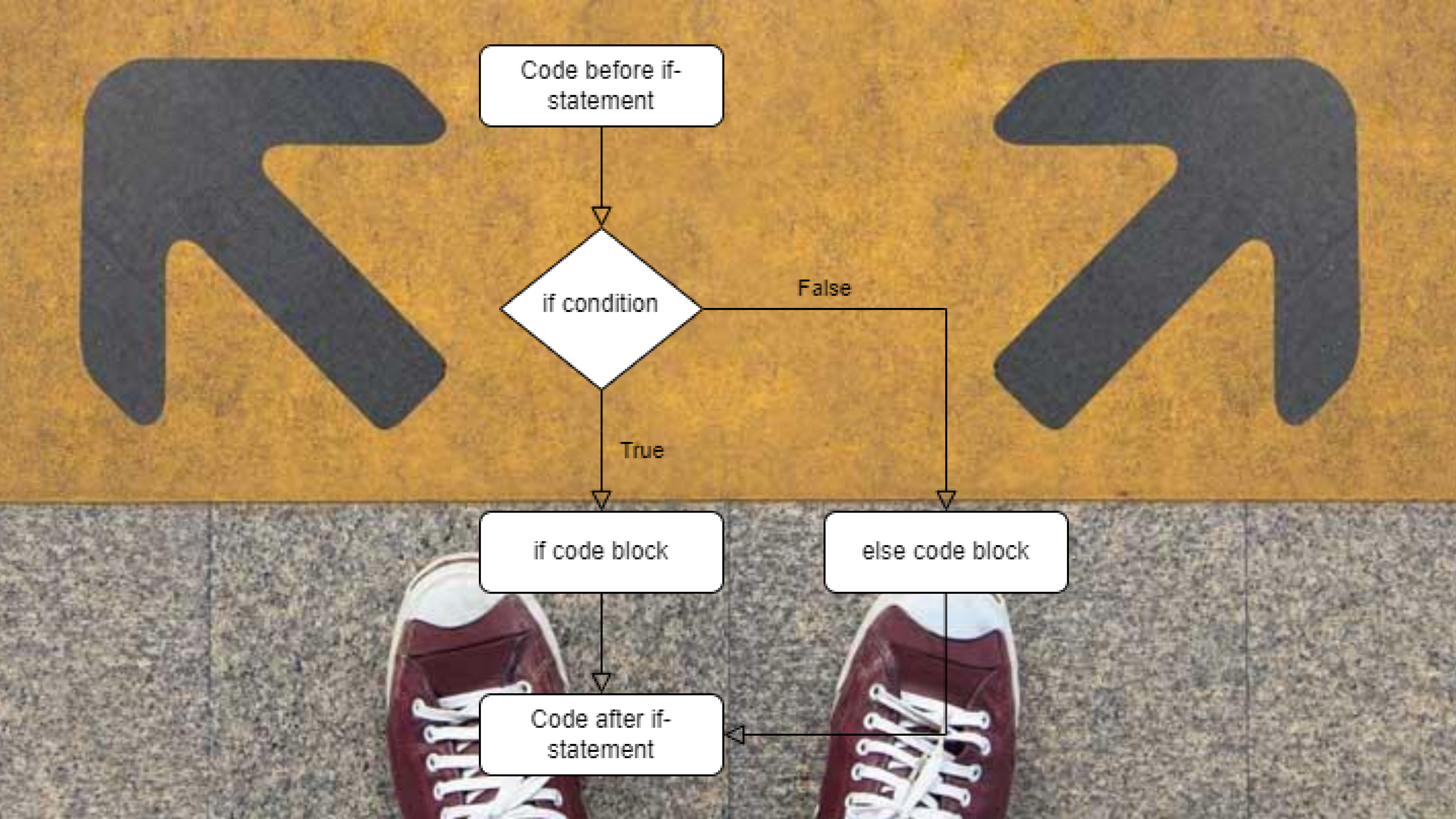
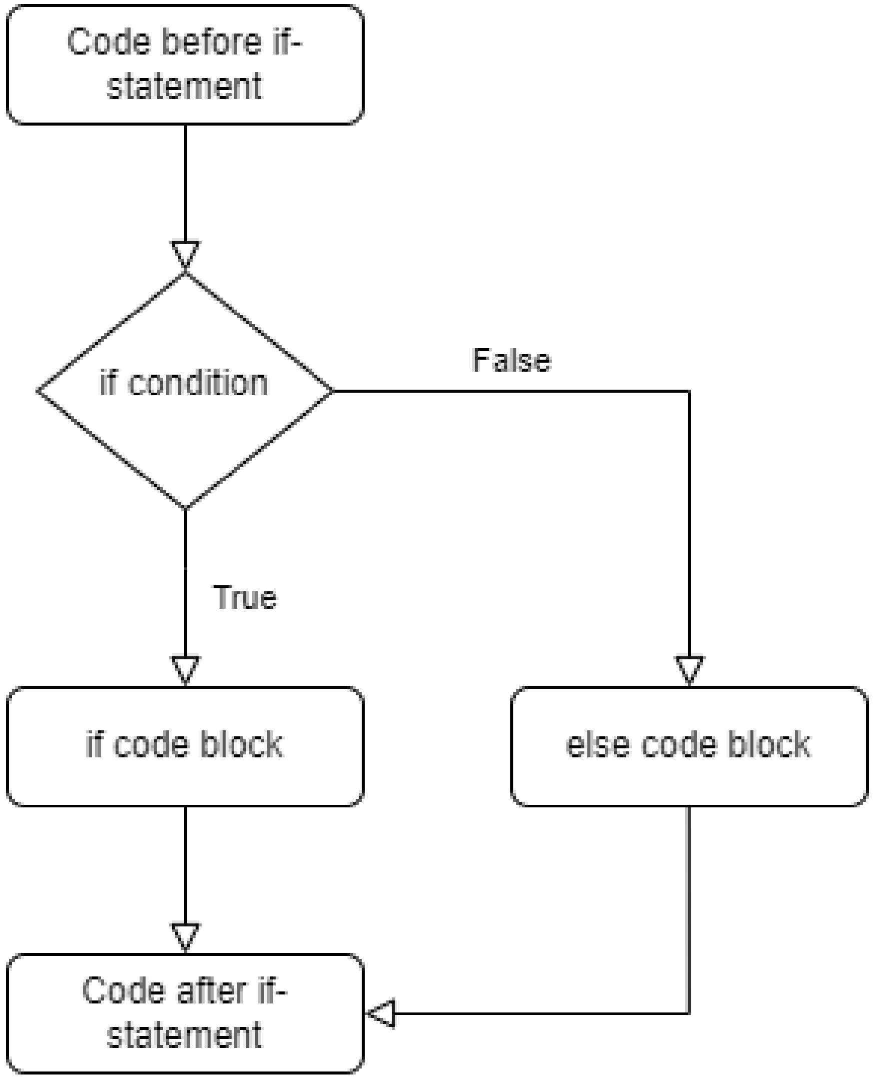

Code before if-statement
if condition
False
True
if code block
else code block
Code after if-statement

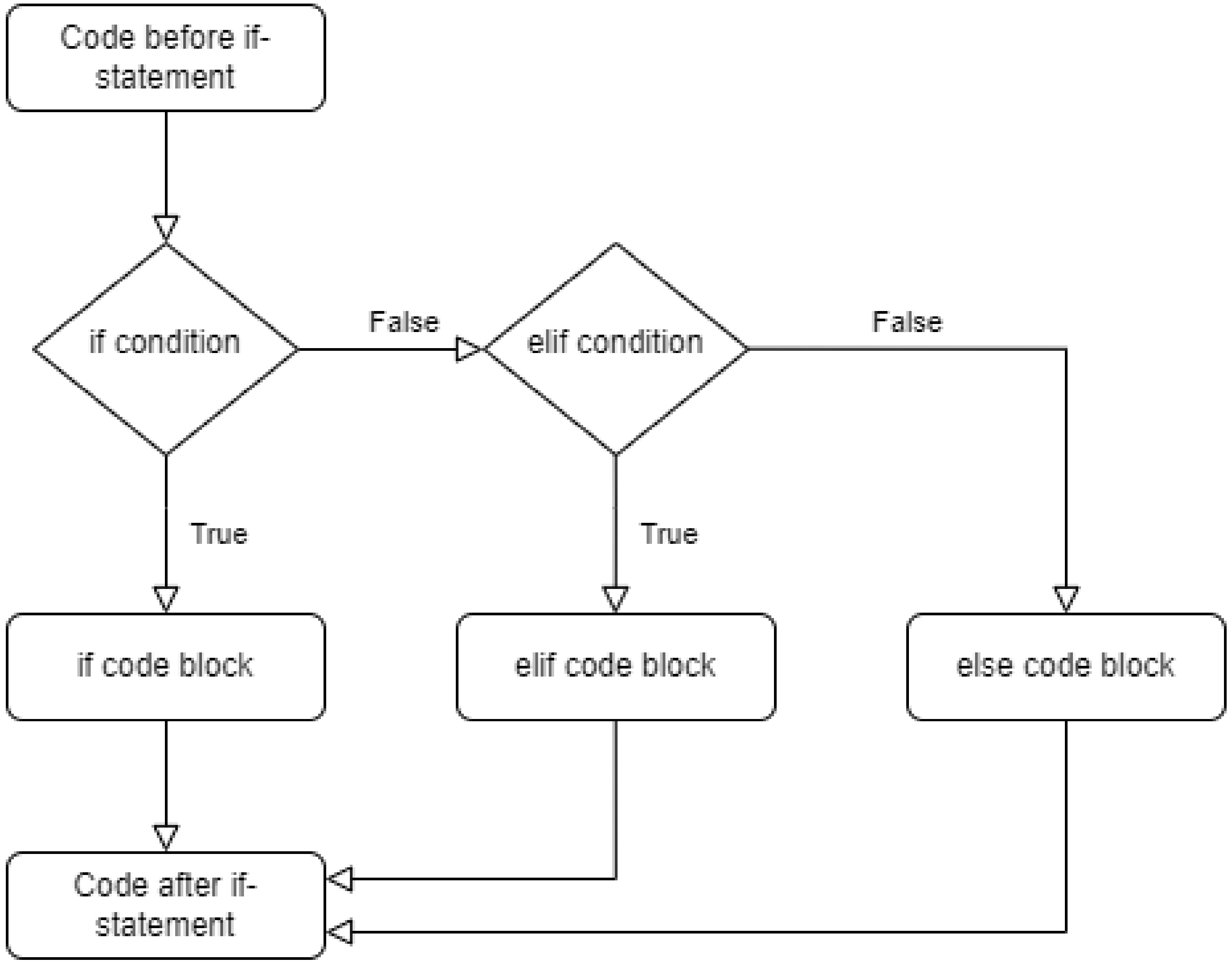
Photo by Gage Skidmore

```python
import random

perc_trump = random.random() * 100
perc_biden = 100 - perc_trump

if perc_trump > perc_biden:
  print("Trump wins the election!")
else:
  print("Biden wins the election!")
```

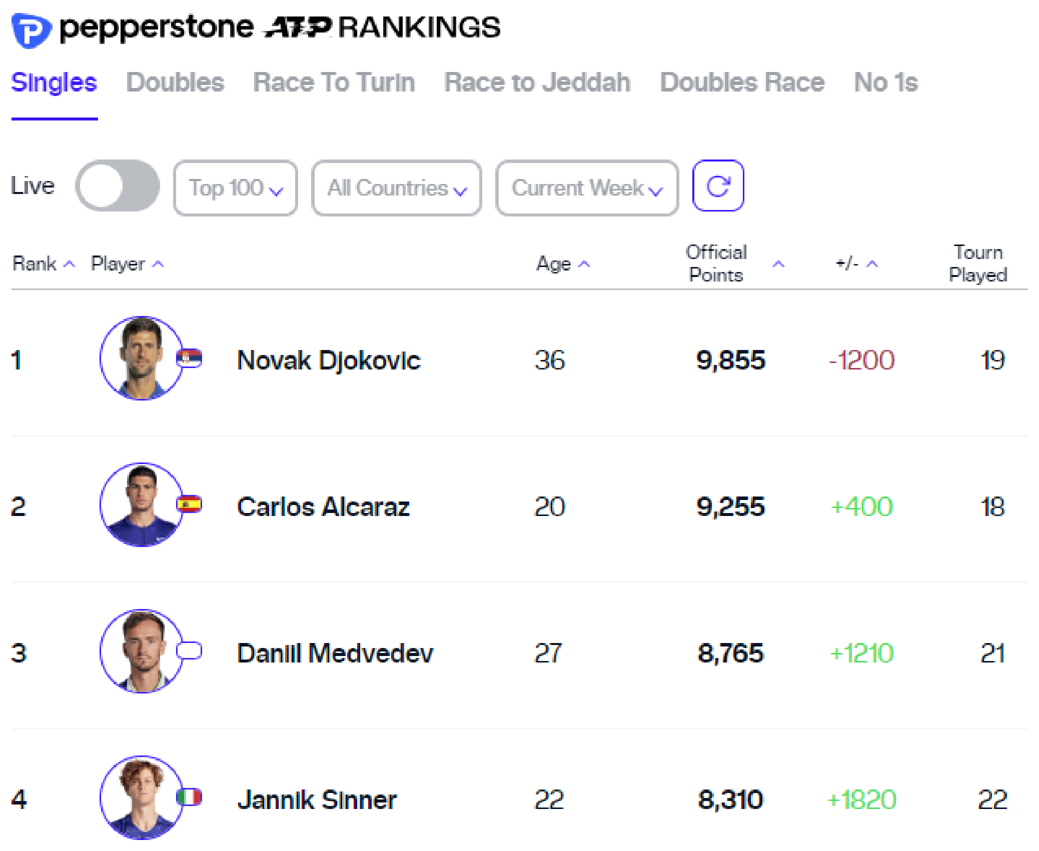
Photo by Gage Skidmore

```python
import random

perc_trump = random.random() * 100
perc_biden = 100 - perc_trump

if perc_trump > perc_biden:
  print("Trump wins the election!")
  winner = "Trump"
else:
  print("Biden wins the election!")
  winner = "Biden"

print("So the winner is %s" % winner)
```


Code before if-statement
if condition
False
True
if code block
else code block
Code after if-statement


Code before if-statement
if condition
False
elif condition
False
True
True
if code block
elif code block
else code block
Code after if-statement

```python
if first == "Djokovic":
  print("Novak is first!")
elif first == "Alcaraz":
  print("Carlos is first!")
else:
  print("Neither Novak nor Carlos is first ...")
```

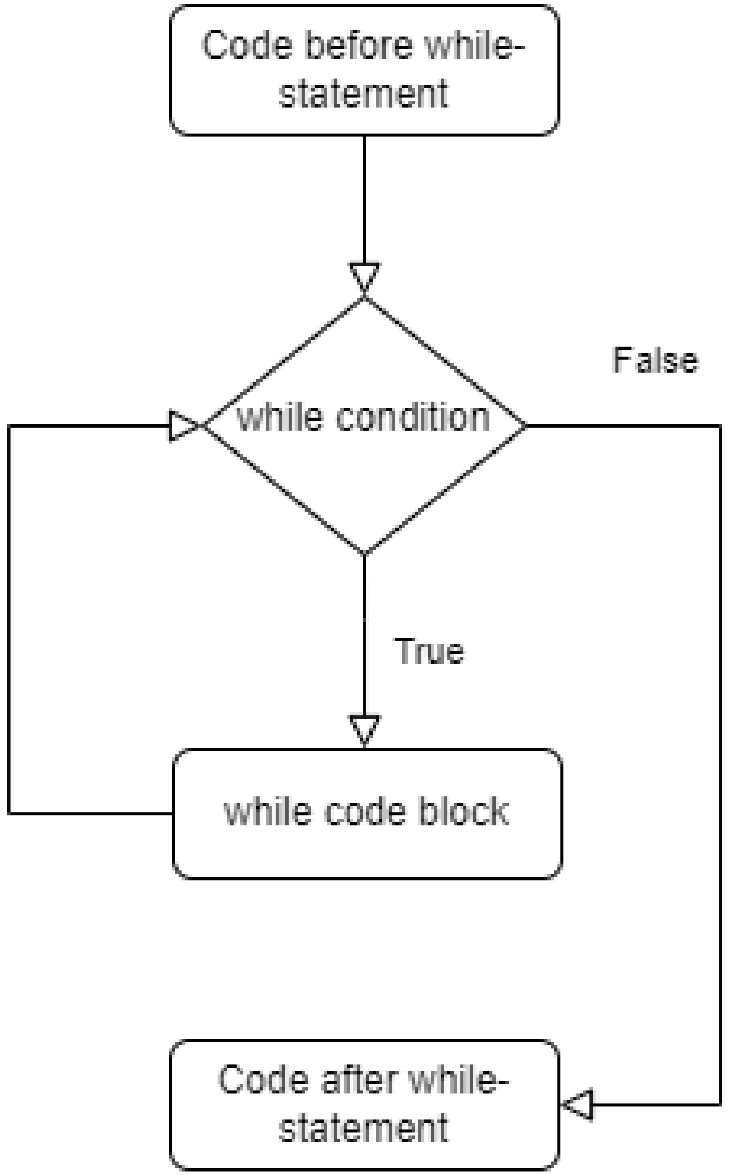


pepperstone ATP RANKINGS

Singles | Doubles | Race To Turin | Race to Jeddah | Doubles Race | No 1s

Live | Top 100 | All Countries | Current Week

Rank	Player	Age	Official Points	+/-	Tourn Played
1	Novak Djokovic	36	9,855	-1200	19
2	Carlos Alcaraz	20	9,255	+400	18
3	Daniil Medvedev	27	8,765	+1210	21
4	Jannik Sinner	22	8,310	+1820	22



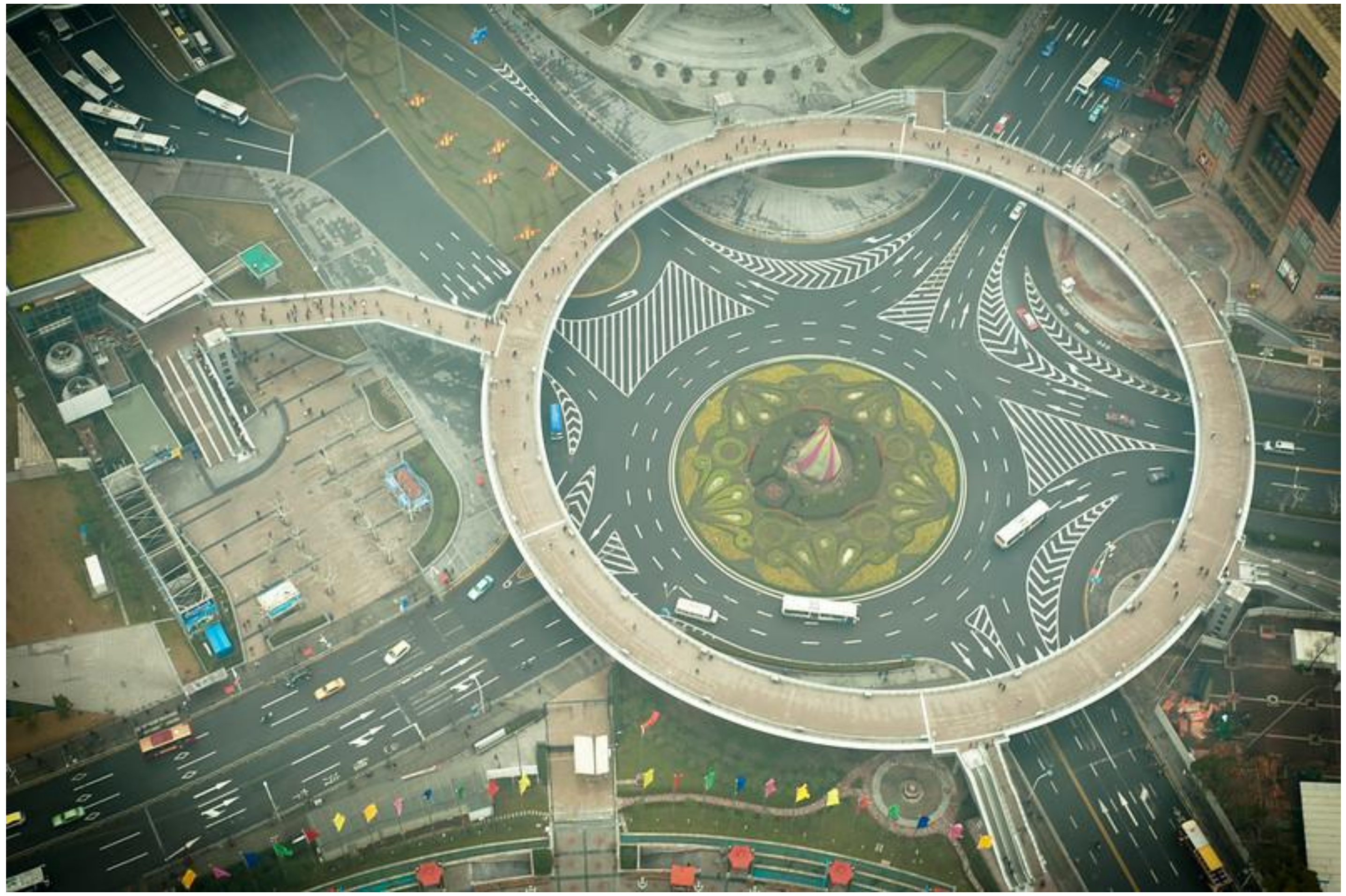
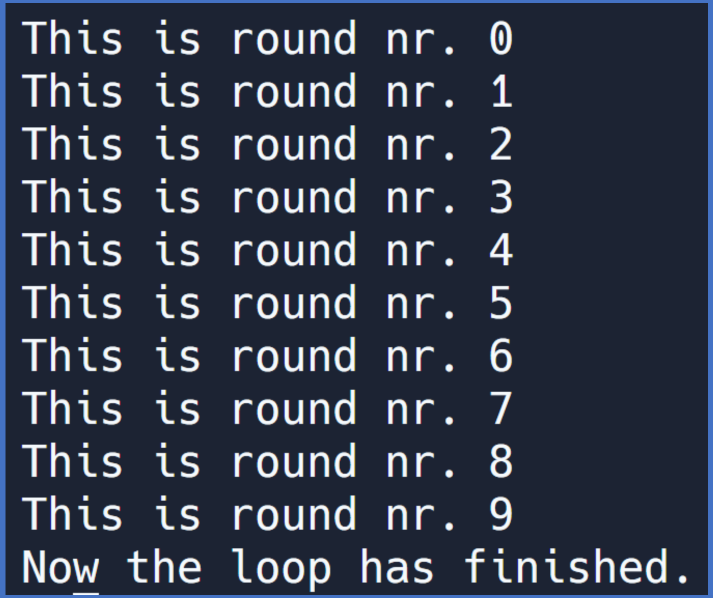


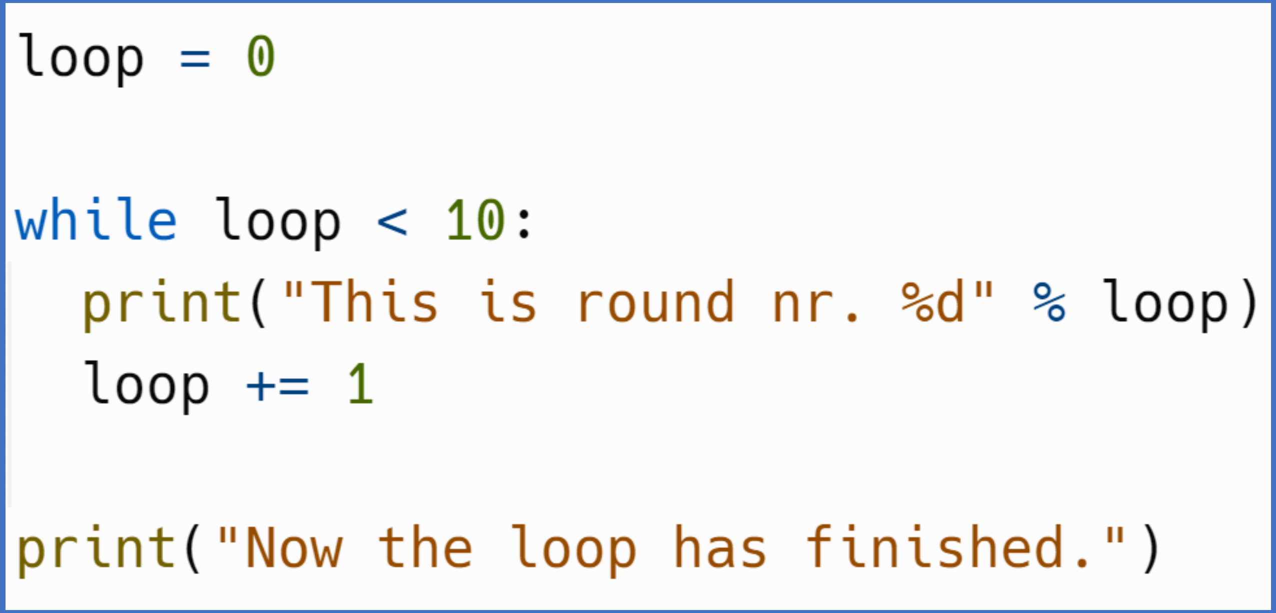
Photo by Tauno Tohk

```python
import random

perc_yes = 0
nr_referendums = 0

while perc_yes < 50:
  perc_yes = random.random() * 100
  print("Yes vote: %.1f" % perc_yes)
  nr_referendums += 1

print("This required %d referendums"
% nr_referendums)
```

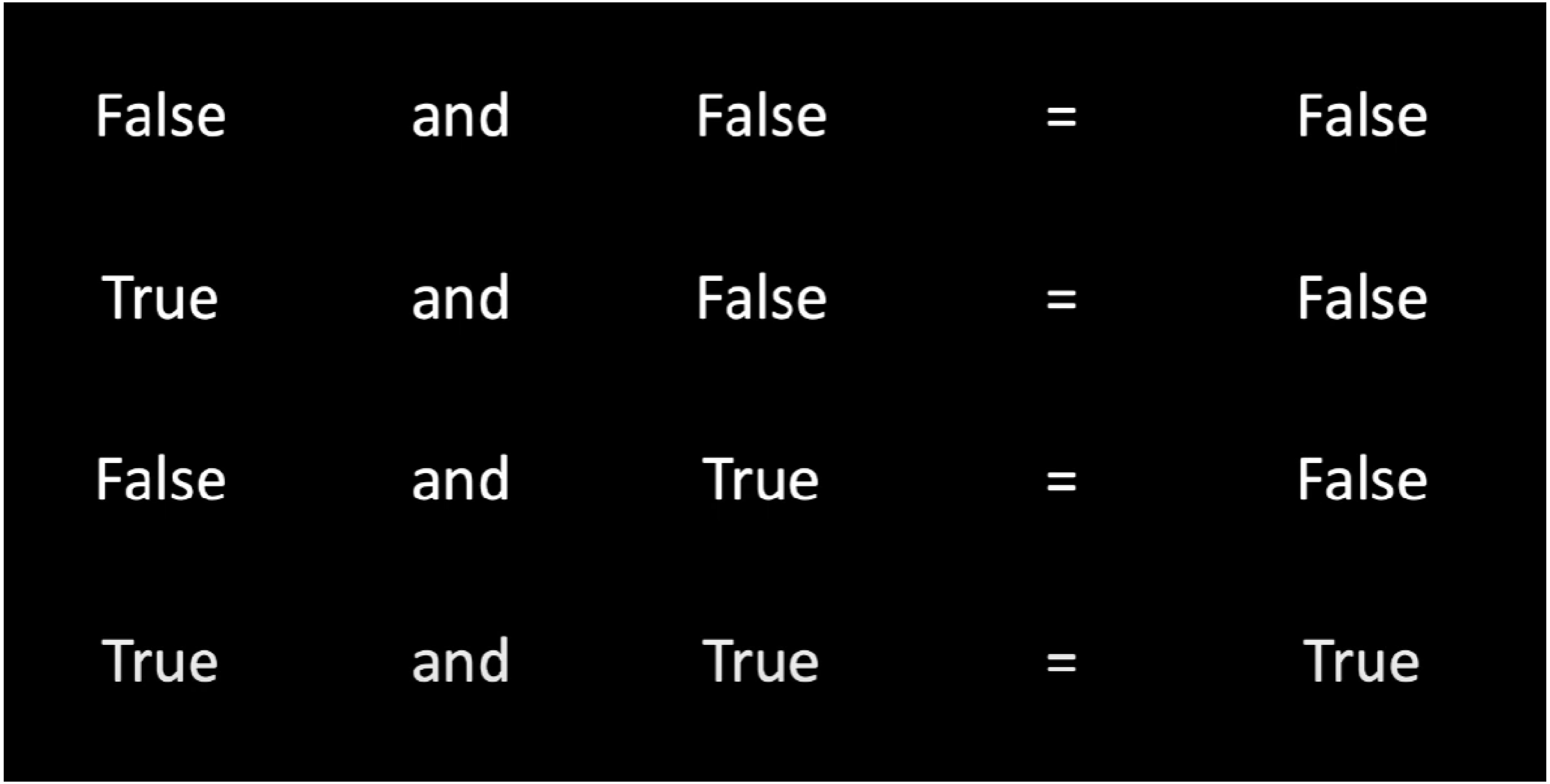
```
Yes vote: 31.3
Yes vote: 29.6
Yes vote: 66.0
This required 3 referendums
```



```
This is round nr. 0
This is round nr. 1
This is round nr. 2
This is round nr. 3
This is round nr. 4
This is round nr. 5
This is round nr. 6
This is round nr. 7
This is round nr. 8
This is round nr. 9
Now the loop has finished.
```



```python
loop = 0

while loop < 10:
  print("This is round nr. %d" % loop)
  loop += 1

print("Now the loop has finished.")
```



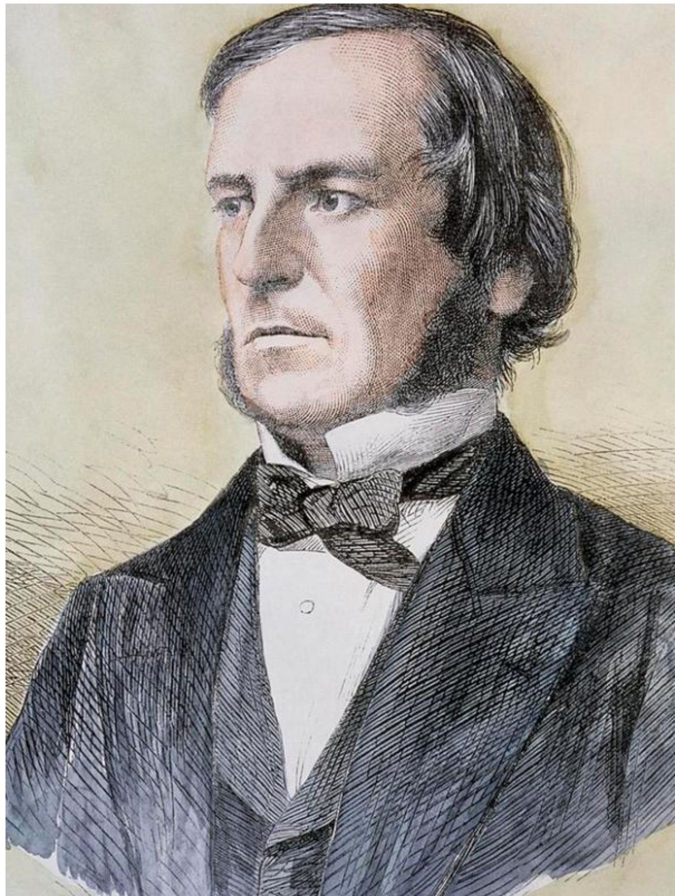
False	and	False	=	False
True	and	False	=	False
False	and	True	=	False
True	and	True	=	True

False	or	False	=	False
True	or	False	=	True
False	or	True	=	True
True	or	True	=	True

not	False	=	True
not	True	=	False

((A or C) and ((A and D) or (A and not D))) or (A and C) or C

((A or C) and ((A and D) or (A and not D))) or (A and C) or C



((A or C) and ((A and D) or (A and not D))) or (A and C) or C =

((A or C) and A and (D or not D)) or (A and C) or C =

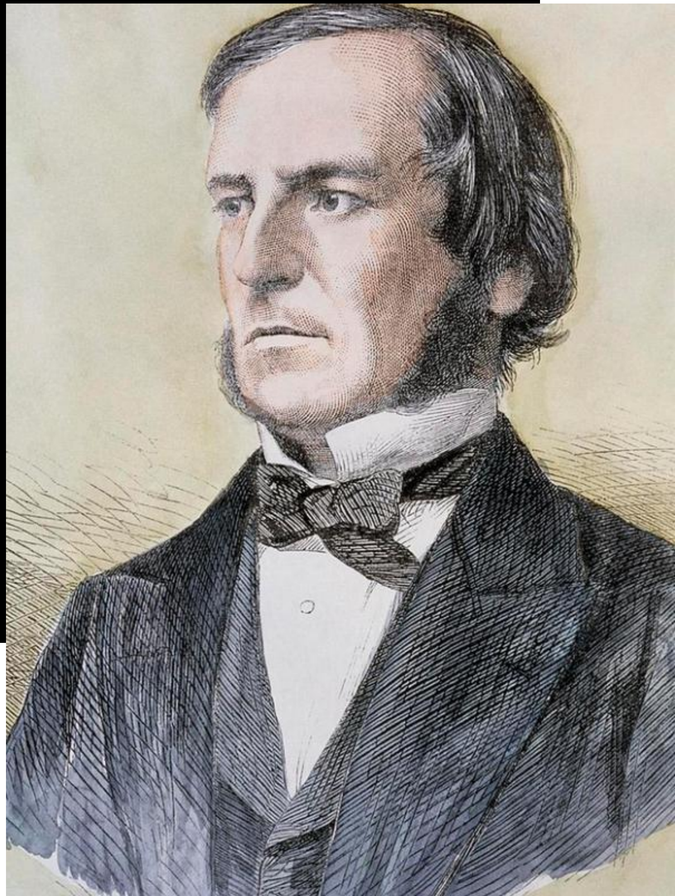
((A or C) and A) or (A and C) or C =

(A and ((A or C) or C)) or C =

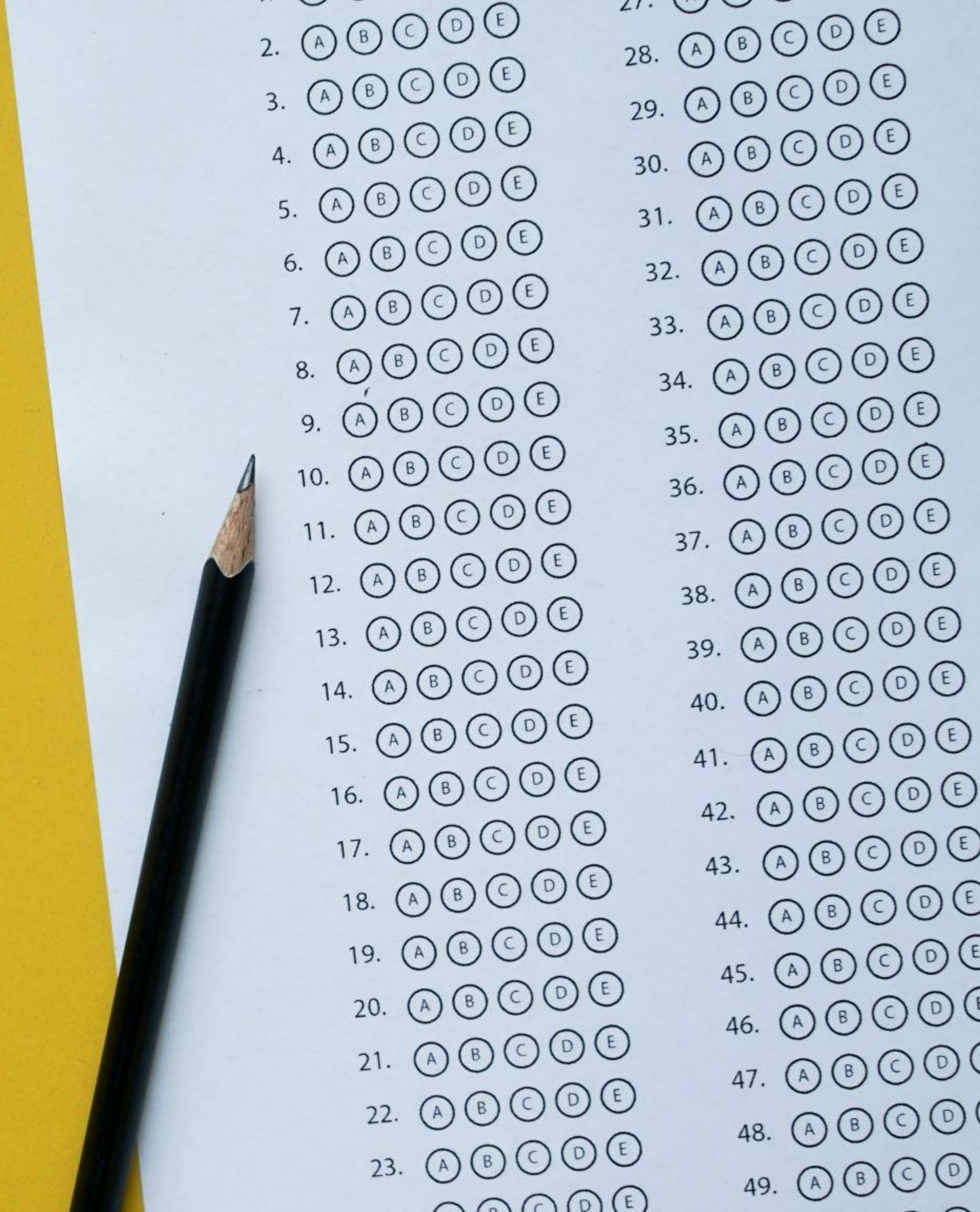
(A and (A or C)) or C =

(A and A) or (A and C) or C =

A or C

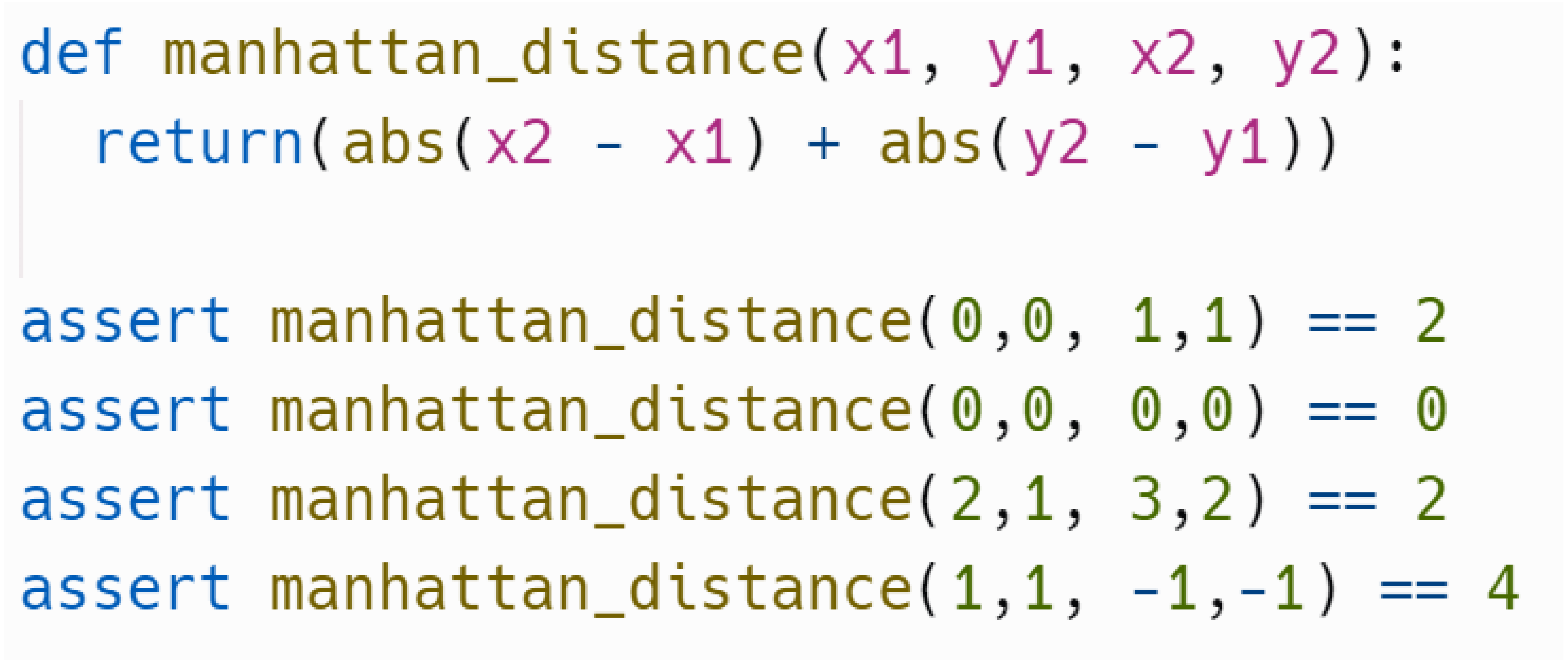


```python
def manhattan_distance(x1, y1, x2, y2):
  return(abs(x2 - x1) + abs(y2 - y1))

assert manhattan_distance(0,0, 1,1) == 2
assert manhattan_distance(0,0, 0,0) == 0
assert manhattan_distance(2,1, 3,2) == 2
assert manhattan_distance(1,1, -1,-1) == 4
```



Test-Driven Development

1. Write a test that defines the desired behavior of a small piece of functionality.
2. Run the test (it should fail because the functionality hasn't been implemented yet).
3. Write the minimum amount of code necessary to pass the test.
4. Run the test again (it should pass now).
5. Refactor the code if necessary while ensuring that all tests still pass.



2

```python
def manhattan_distance(x1, y1, x2, y2):
    return(abs(x2 - x1) + abs(y2 - y1))
```

1

```python
assert manhattan_distance(0,0, 1,1) == 2
assert manhattan_distance(0,0, 0,0) == 0
assert manhattan_distance(2,1, 3,2) == 2
assert manhattan_distance(1,1, -1,-1) == 4
```

Test-driven development: Write the tests first, the function second.