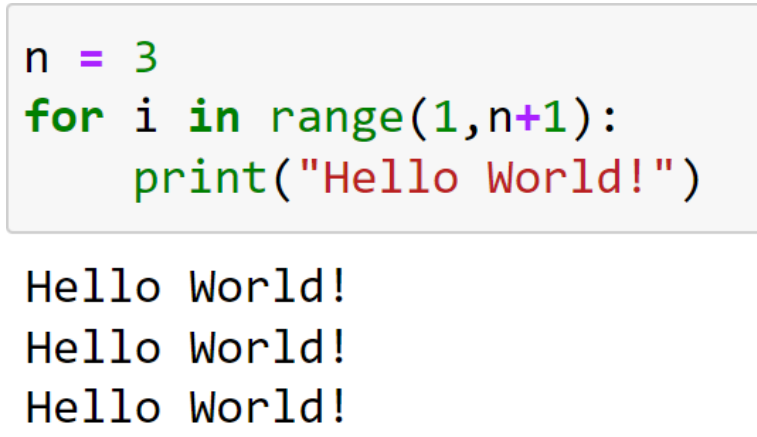
The Boolean Data Type

Programming and Algorithms

STEM Digital Academy

School of Science & Technology

www.city.ac.uk



```
n = 3
for i in range(1,n+1):
    print("Hello World!")
```

```
Hello World!
Hello World!
Hello World!
```

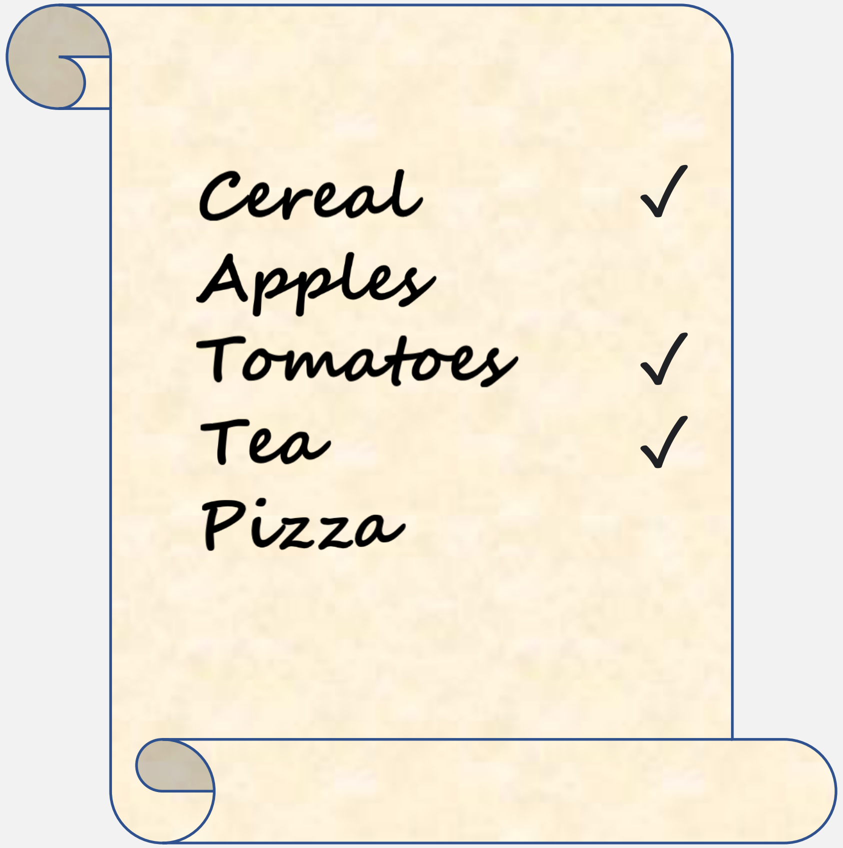
Lecture by

Dr Daniil Osudin

What will we Cover?

- The Boolean data type
- Comparison operators
- Evaluation of Boolean expressions

Example



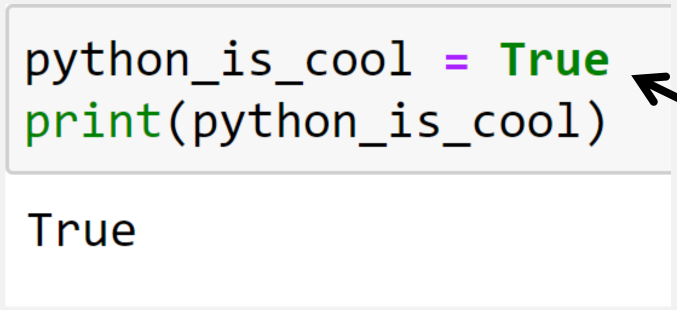
Using your shopping list, keep track of the items you found:

- Tick the items you placed into your cart (`true`)
- Leave the other items not ticked (`false`)

The Boolean Data Type

- Has only two values – `True`, `False`
- Variables of type Boolean can be assigned one of these values
- A Boolean value can result from a logical expression which evaluates to `True` or `False`.
- Expression '`a` is equal to `b`' will be
  - `True` if `a` and `b` are equal
  - `False` if `a` and `b` are not equal

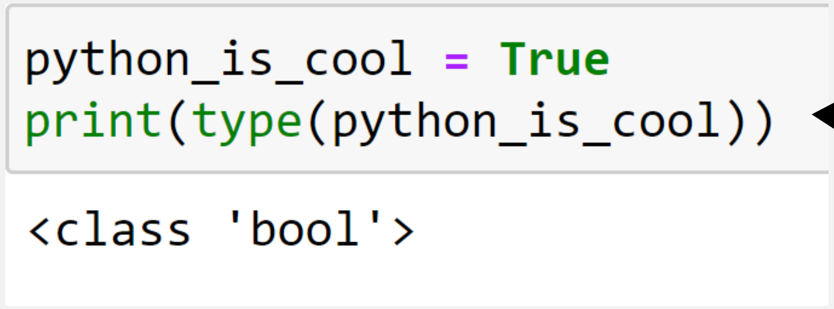
Examples I



```
python_is_cool = True
print(python_is_cool)
```

```
True
```

Create a Boolean variable 'python_is_cool' and assign a value `True`



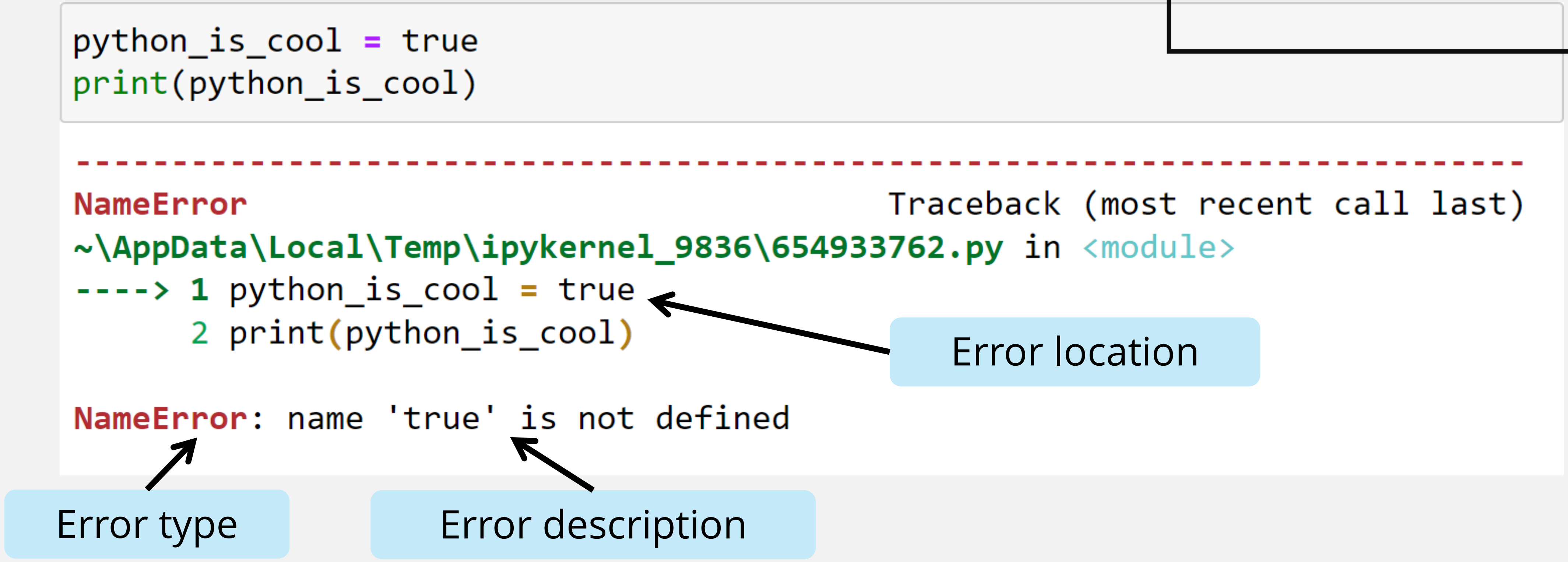
```
python_is_cool = True
print(type(python_is_cool))
```

```
<class 'bool'>
```

Print the data type of the variable 'python_is_cool'

Common Errors in Python

- `True` must use a capital `T`



```
python_is_cool = true
print(python_is_cool)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_9836\654933762.py in <module>
----> 1 python_is_cool = true
      2 print(python_is_cool)

NameError: name 'true' is not defined
```

Comparison Operators I

The table below shows comparison operations that can be performed between two data items. Consider
`x = 3, y = 5`

Mathematical operator	Python notation	Description	Examples	Result

=	`==`	Equal to
		(are the two values the same)
			`x == 3`
			`x == 5`
			`x == y`
				`True`
				`False`
				`False`

≠	`!=`	Not equal to
		(are the two values different)
			`x != 3`
			`x != 5`
			`x != y`
				`False`
				`True`
				`True`

Note: equality operator contains two == characters as opposed to the assignment operator =

Comparison Operators II

The table below shows more comparison operations that can be performed between two data items.
Consider `x = 3, y = 5`

Mathematical operator	Python notation	Description	Examples	Result

<	`<`	Less than	`x < 3`
			`x < 7`
			`x < 1`
			`x < y`
				`False`
				`True`
				`False`
				`True`

>	`>`	Greater than	`x > 3`
			`x > 7`
			`x > 1`
			`x > y`
				`False`
				`False`
				`True`
				`False`

Comparison Operators III

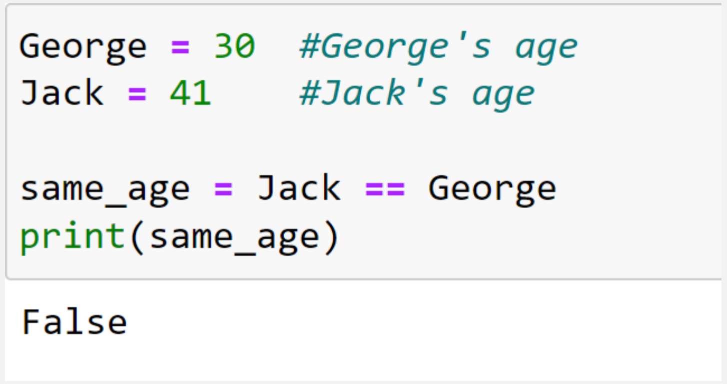
The table below shows more comparison operations that can be performed between two data items.
Consider `x = 3, y = 5`

Mathematical operator	Python notation	Description	Examples	Result

≤	`<=`	Less than or equal to	`x <= 3`
			`x <= 7`
			`x <= 1`
			`x <= y`
				`True`
				`True`
				`False`
				`True`

≥	`>=`	Greater than or equal to	`x >= 3`
			`x >= 7`
			`x >= 1`
			`x >= y`
				`True`
				`False`
				`True`
				`False`

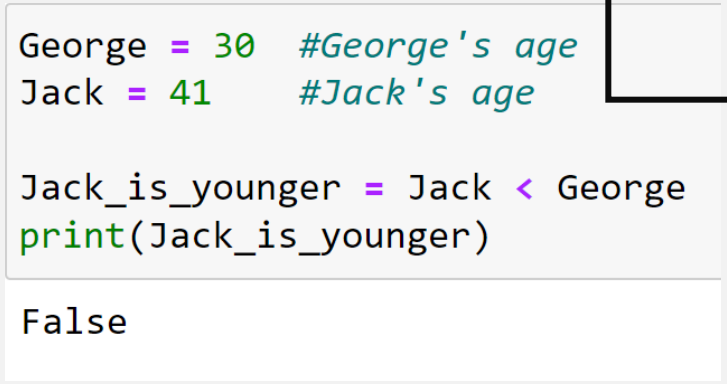
Examples II



```python
George = 30  #George's age
Jack = 41    #Jack's age

same_age = Jack == George
print(same_age)
```

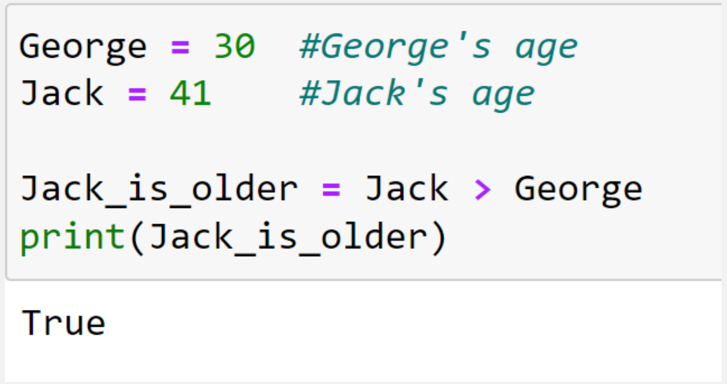
```
False
```



```python
George = 30  #George's age
Jack = 41    #Jack's age

Jack_is_younger = Jack < George
print(Jack_is_younger)
```

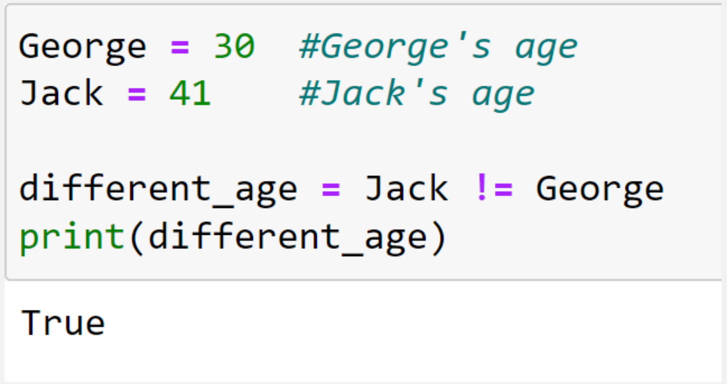
```
False
```



```python
George = 30  #George's age
Jack = 41    #Jack's age

Jack_is_older = Jack > George
print(Jack_is_older)
```

```
True
```

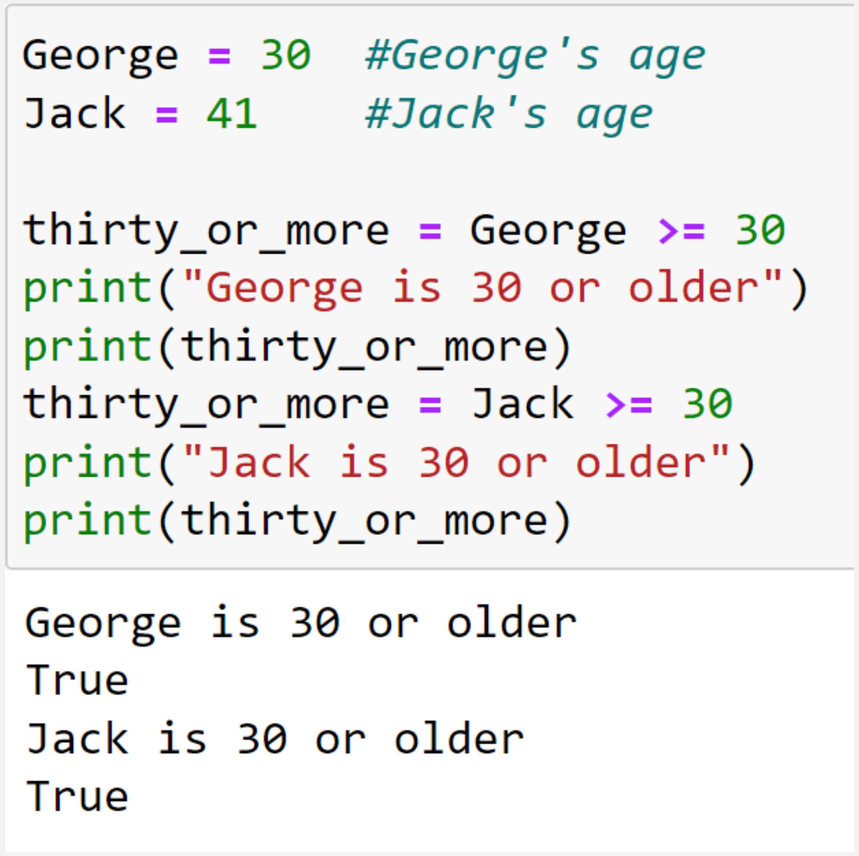


```python
George = 30  #George's age
Jack = 41    #Jack's age

different_age = Jack != George
print(different_age)
```

```
True
```

Examples III

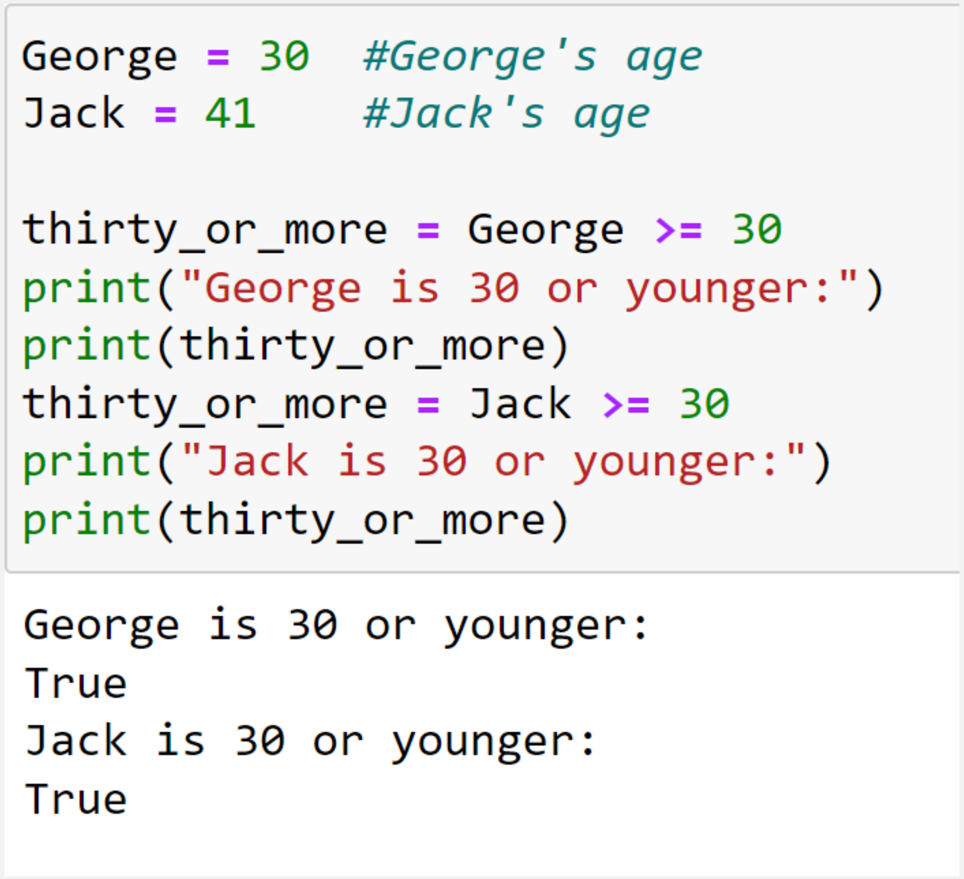


```python
George = 30  #George's age
Jack = 41    #Jack's age

thirty_or_more = George >= 30
print("George is 30 or older")
print(thirty_or_more)
thirty_or_more = Jack >= 30
print("Jack is 30 or older")
print(thirty_or_more)
```

```
George is 30 or older
True
Jack is 30 or older
True
```

Examples IV



```python
George = 30  #George's age
Jack = 41    #Jack's age

thirty_or_more = George >= 30
print("George is 30 or younger:")
print(thirty_or_more)
thirty_or_more = Jack >= 30
print("Jack is 30 or younger:")
print(thirty_or_more)
```

```
George is 30 or younger:
True
Jack is 30 or younger:
True
```

Logical Operators I

- Used to combine logical expressions
- **Operators:** `and`, `or`, `not`
- Examples
  - Defining a numeric range
    - Between the ages of 25 and 35
  - Satisfying multiple conditions
    - The biggest size of a shelf that will fit is 120cm wide and 40cm deep
  - Accepting either of two options
    - Nick can schedule a meeting for 9:00am or 2:00pm

Logical Operators II

The table below shows the description and examples of the `and` logical operator. Consider `x = 3, y = 5`

Operator	Description	Examples	Result

`and`	Evaluates to `True` **if both** left and right values and/or expressions are `True`	`(x < 4) and (y < 7)`
		`True and True`
			`True`

		`(x > 4) and (y < 7)`
		`False and True`
			`False`

		`(x < 4) and (y > 7)`
		`True and False`
			`False`

		`(x > 4) and (y > 7)`
		`False and False`
			`False`

Logical Operators III

The table below shows the description and examples of the `or` logical operator. Consider `x = 3, y = 5`

Operator	Description	Examples	Result

`or`	Evaluates to `True` **if either (or both)** left or right value and/or expression is `True`	`(x < 4) or (y < 7)`
		`True or True`
			`True`

		`(x > 4) or (y < 7)`
		`False or True`
			`True`

		`(x < 4) or (y > 7)`
		`True or False`
			`True`

		`(x > 4) or (y > 7)`
		`False or False`
			`False`

Logical Operators IV

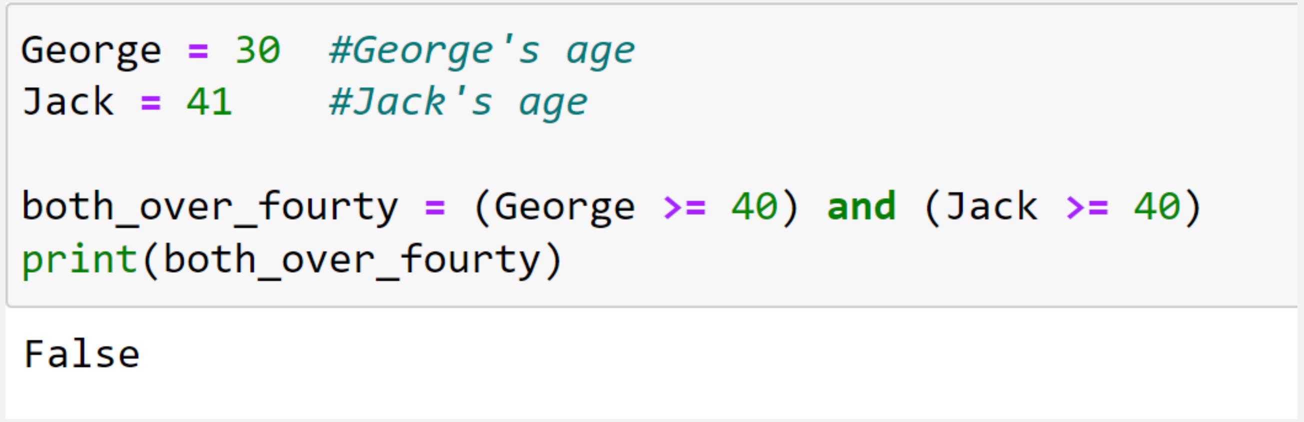
The table below shows the description and examples of the `not` logical operator. Consider `x = 3`

Operator	Description	Examples	Result

`not`	Evaluates to `True` if the value or the logical expression being checked is `False`	`not (x > 4)`
		`not    False`
			`True`

		`not (x < 4)`
		`not     True`
			`False`

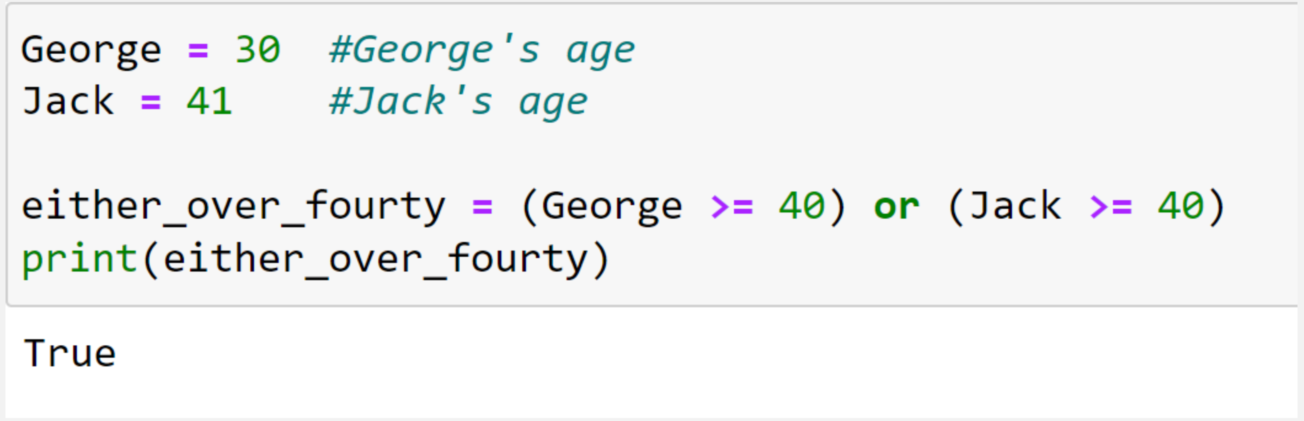
Examples V



```
George = 30  #George's age
Jack = 41    #Jack's age

both_over_fourty = (George >= 40) and (Jack >= 40)
print(both_over_fourty)
```

```
False
```

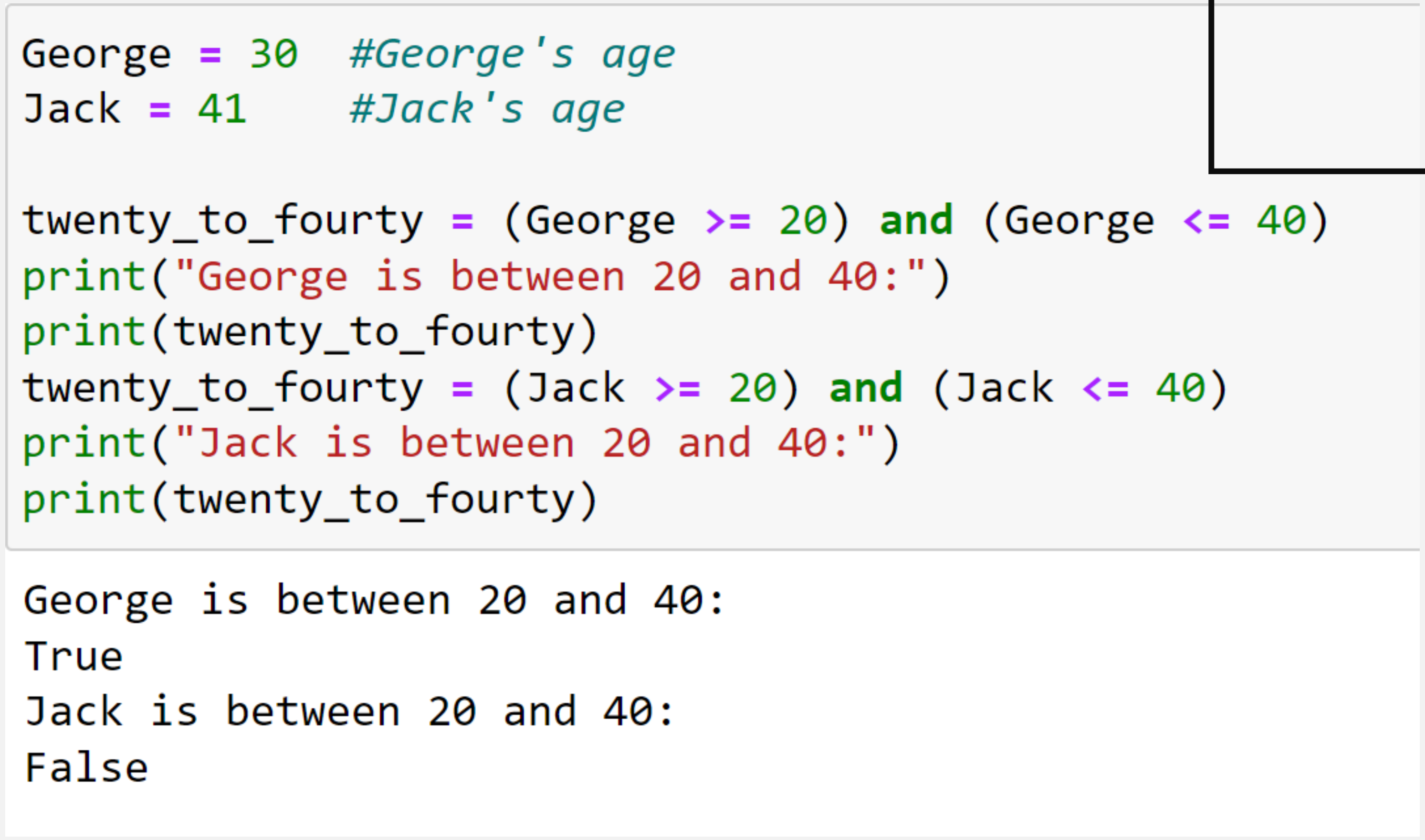


```
George = 30  #George's age
Jack = 41    #Jack's age

either_over_fourty = (George >= 40) or (Jack >= 40)
print(either_over_fourty)
```

```
True
```

Examples VI

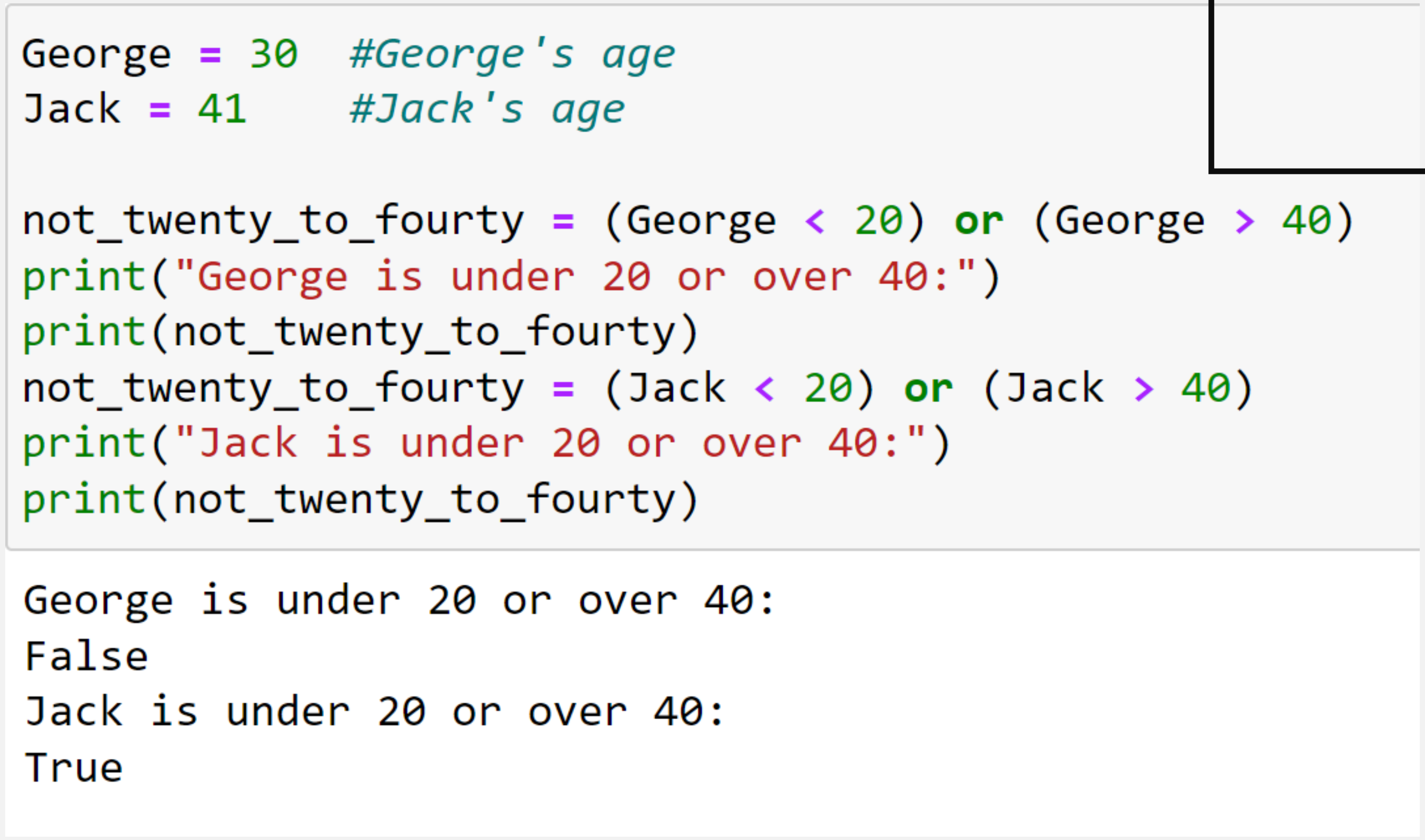


```python
George = 30  #George's age
Jack = 41    #Jack's age

twenty_to_fourty = (George >= 20) and (George <= 40)
print("George is between 20 and 40:")
print(twenty_to_fourty)
twenty_to_fourty = (Jack >= 20) and (Jack <= 40)
print("Jack is between 20 and 40:")
print(twenty_to_fourty)
```

```
George is between 20 and 40:
True
Jack is between 20 and 40:
False
```

Examples VII



```python
George = 30  #George's age
Jack = 41    #Jack's age

not_twenty_to_fourty = (George < 20) or (George > 40)
print("George is under 20 or over 40:")
print(not_twenty_to_fourty)
not_twenty_to_fourty = (Jack < 20) or (Jack > 40)
print("Jack is under 20 or over 40:")
print(not_twenty_to_fourty)
```

```
George is under 20 or over 40:
False
Jack is under 20 or over 40:
True
```

Try It Yourself I

Enter and run the following statements in the python environment:

```
x = 5
y = 7
print(x == y)
```

```
x = 5
y = 7
print(x != y)
```

```
x = 5
y = 7
print(x > y)
```

```
x = 5
y = 7
print(x < y)
```

Try It Yourself II

Enter and run the following statements in the python environment:

```
a = True
b = False
print(a or b)
```

```
a = True
b = False
print((not a) or b)
```

```
a = True
b = False
print(a and b)
```

```
a = True
b = False
print(a and (not b))
```