Conditional Statements

Outline

- Boolean expressions
- Statements
- Flowcharts
- Conditional statements
  - Conditional execution
  - Alternative execution
  - Chained conditionals
  - Nested conditionals

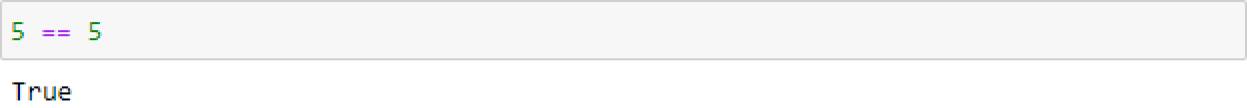
Data types

Name	Type	Description	Example
String	str	A sequence of characters	“hello”, “course”, “covid-19”, “2”
Integer	Int	Whole numbers	2, 4, 100, 4000
Float	float	Numbers containing one or more decimals	3.8, 50.9, 100.0
Booleans	bool	Logical value indicating TRUE or FALSE	True, False
List	list	Ordered sequence of objects	[“hello”, “world”,”2021”] [“hello, 5, 100.0]
Dictionary	dict	Key: value pairs	{“key1”: name1, “key2”:name2}
Tuples	tup	Ordered immutable sequence of objects	(10,20) (“hello”, “world”)

Sets	set	Unordered collection of unique objects	{2,4,6,8}
			{3,”hello”, 50.9}

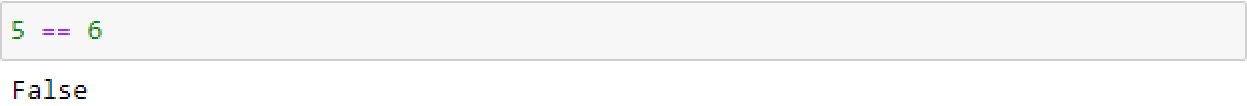
Boolean expression

- A boolean expression is an expression that is either true or false.
- The following examples use the comparison operator ==, which compares two objects and produces True if they are equal and False otherwise



```
5 == 5
```

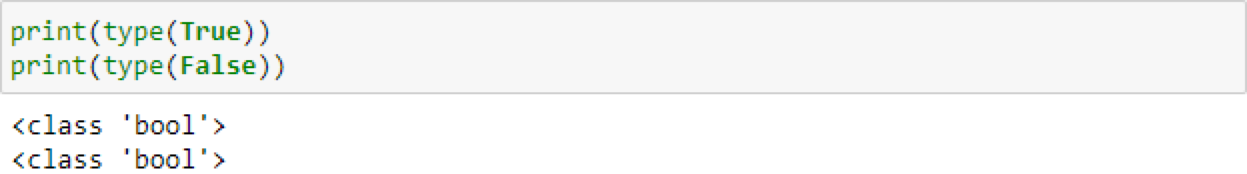
```
True
```



```
5 == 6
```

```
False
```

- True and False are special data type that belong to the type bool. They are not strings.



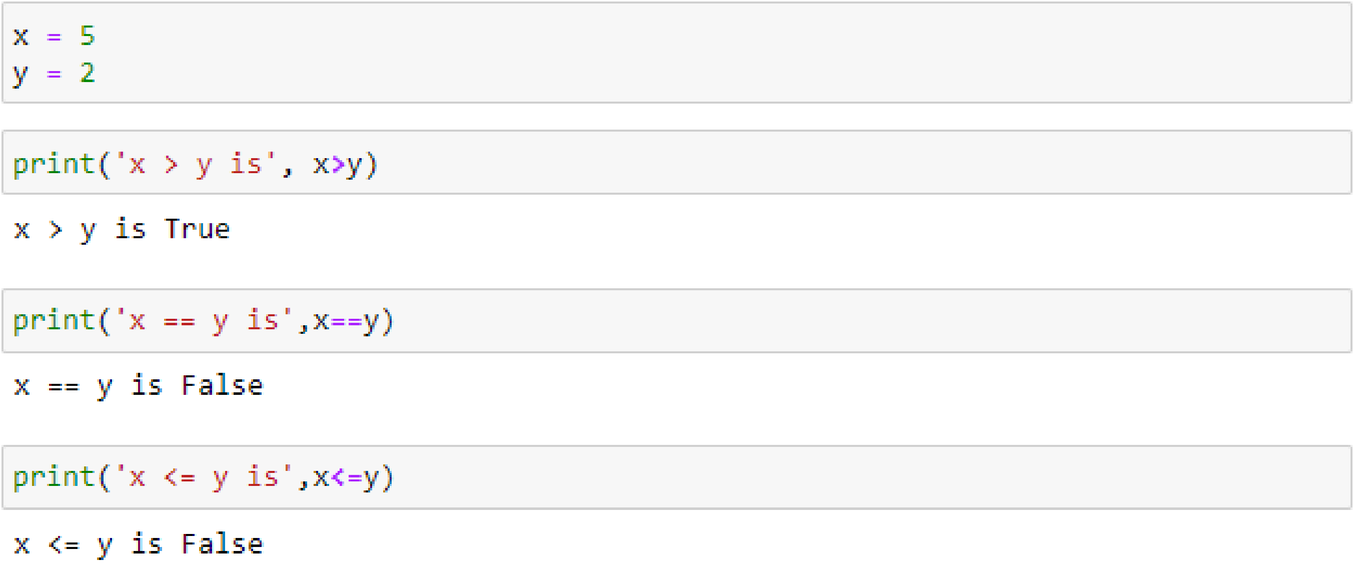
```
print(type(True))
print(type(False))
```

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

Boolean expression – comparison operators

- Comparison operators are used to compare values. It returns either True or False.

Comparison operator	Meaning
>	Greater than
<	Less than
==	Equal to
!=	Not equal to
>=	Greater than or equal to
<=	Less than or equal to division



```
x = 5
y = 2
```

```
print('x > y is', x>y)
```

```
x > y is True
```

```
print('x == y is',x==y)
```

```
x == y is False
```

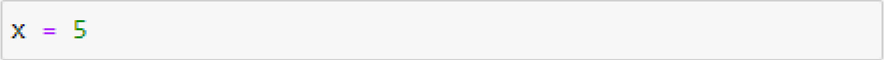
```
print('x <= y is',x<=y)
```

```
x <= y is False
```

Statements

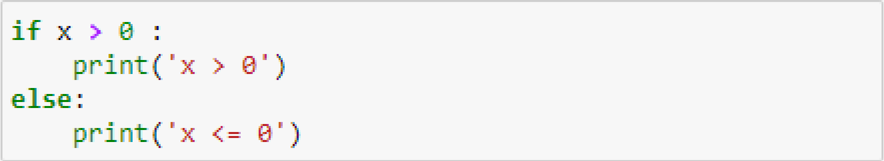
- A statement is an instruction that the Python interpreter can execute, which can be either one line or multiple lines.

Assignment statement



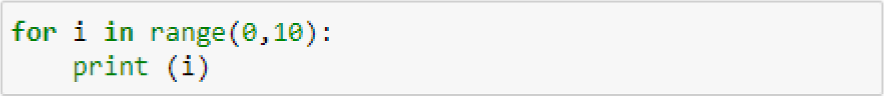
```
x = 5
```

Conditional statement



```
if x > 0 :
    print('x > 0')
else:
    print('x <= 0')
```

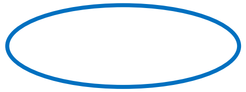
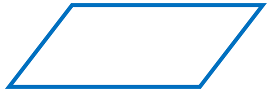
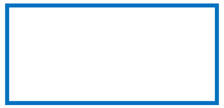
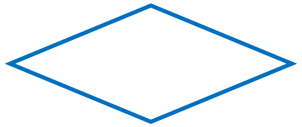
Loop statement



```
for i in range(0,10):
    print (i)
```

Flowcharts

- Flowchart : A diagram that graphically describes the steps in a program.

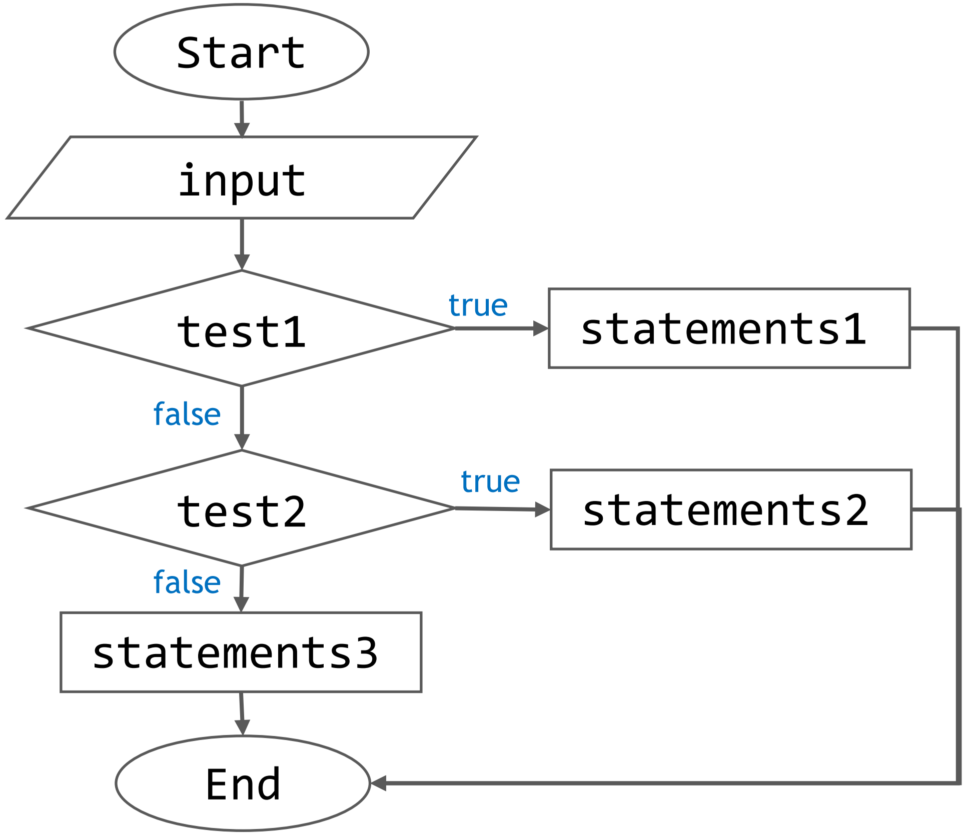
Symbol	Name
	Start/end
	Input/output
	Process (e.g., assign a value, calculate a value)
	Decision (determine whether an expression is true/false)

Conditional statements

- Conditional statements in Python perform different computations or actions depending on whether a specific Boolean constraint evaluates to true or false.

General format

```
if <test1 is true>:
        <statements1>
elif <test2 is true>:   #optional
        <statements2>
else:                   #optional
        <statements3>
```



Start

input

test1

true

statements1

false

test2

true

statements2

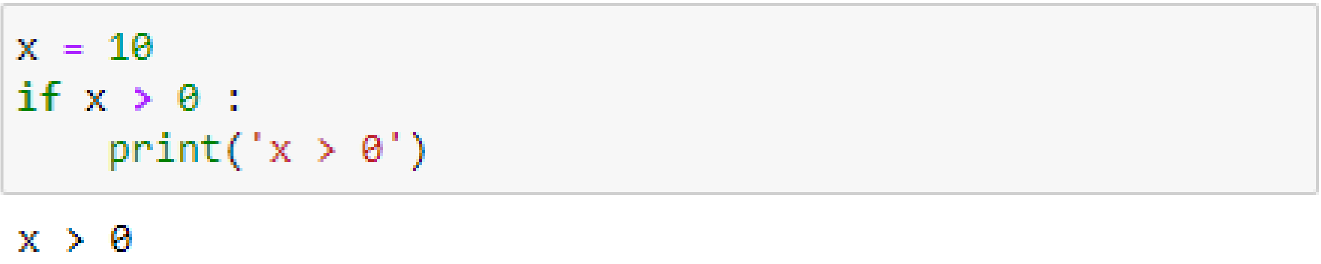
false

statements3

End

Conditional execution

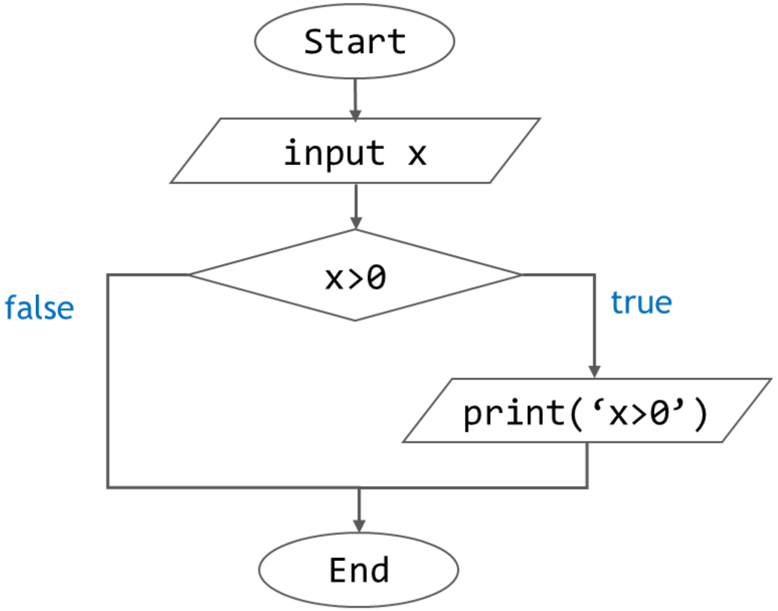
- Conditional statements are handled by the keyword **`if`** in Python.
  - The boolean expression after the **`if`** is called the condition.
  - Use colon character (:) after the Boolean expression.
  - The line(s) after the **`if`** are indented.



```
x = 10
if x > 0 :
    print('x > 0')
```

```
x > 0
```

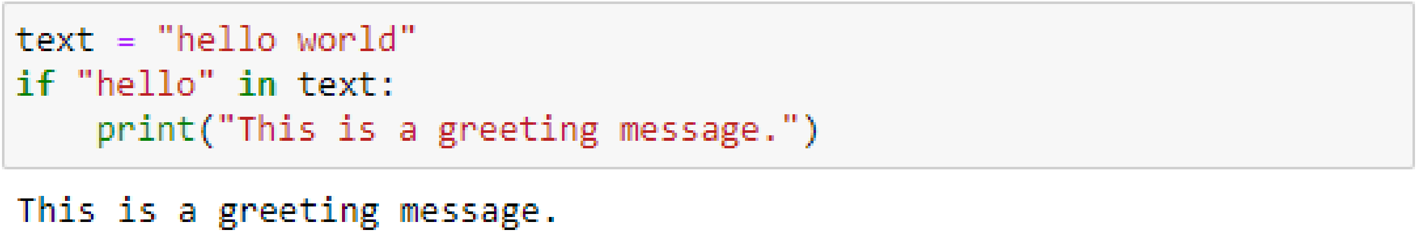
- If the condition is true (x>0), then the indented line gets executed.
- If the condition is false (x≤0), the indented line is skipped.



Python relies on indentation (default is 4 whitespaces) to define scope in the code.
Other programming languages often use curly-brackets { } for this purpose.

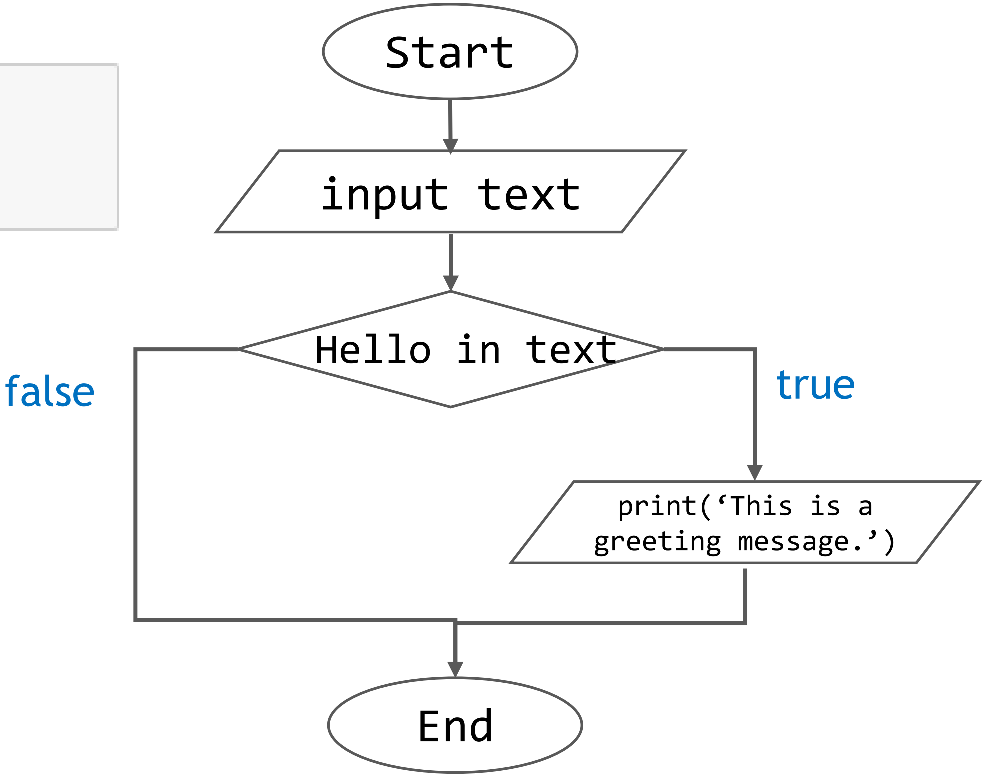
Conditional execution

- Example



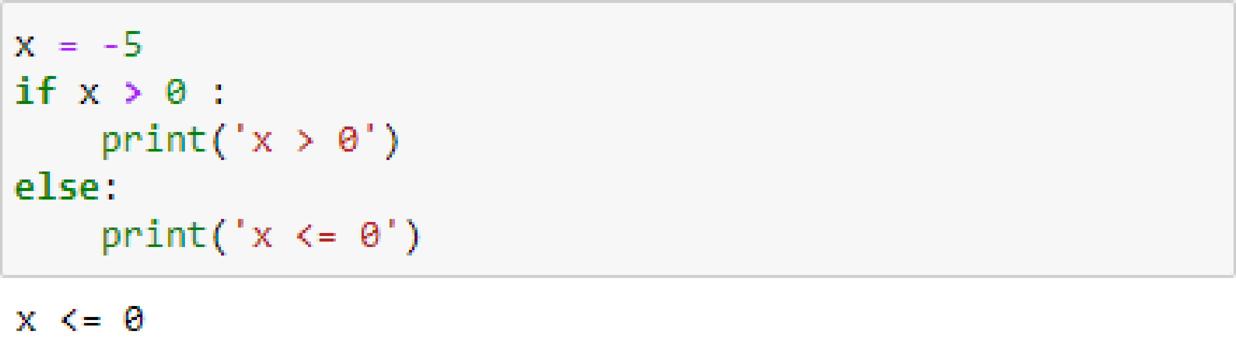
```
text = "hello world"
if "hello" in text:
    print("This is a greeting message.")
```

```
This is a greeting message.
```



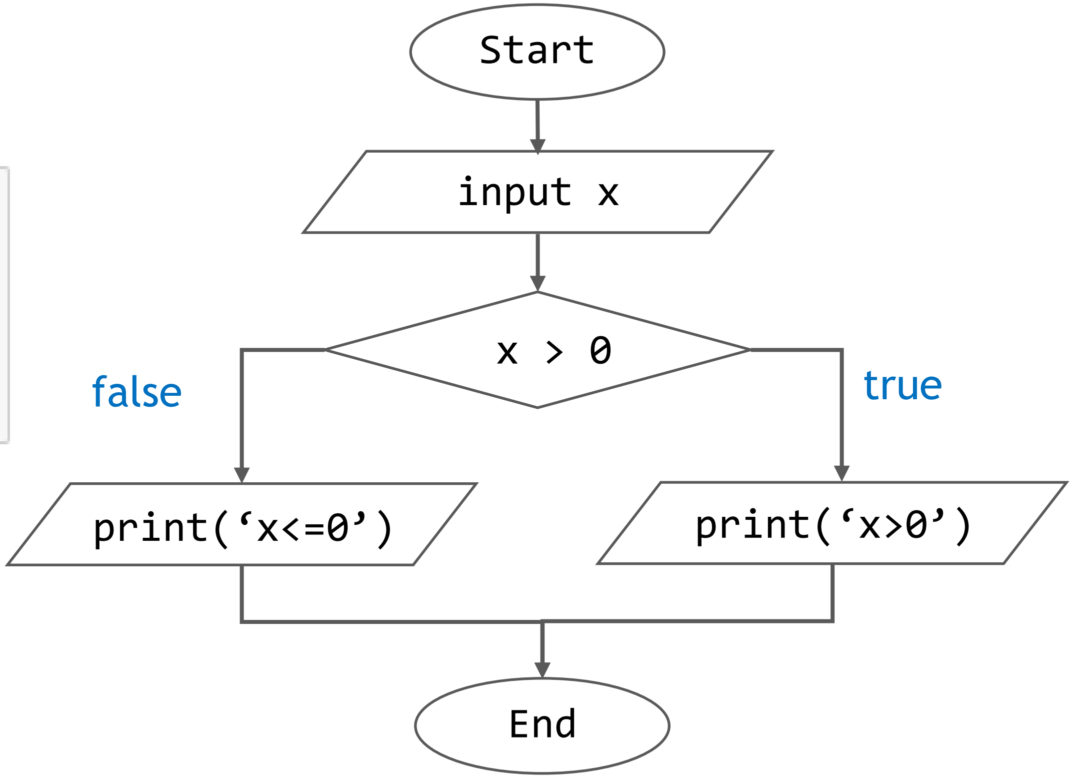
Alternative execution

- It is also possible to execute a statement when the condition is false.



```
x = -5
if x > 0 :
    print('x > 0')
else:
    print('x <= 0')
```

```
x <= 0
```



Start

input x

x > 0

false

true

print(‘x<=0’)

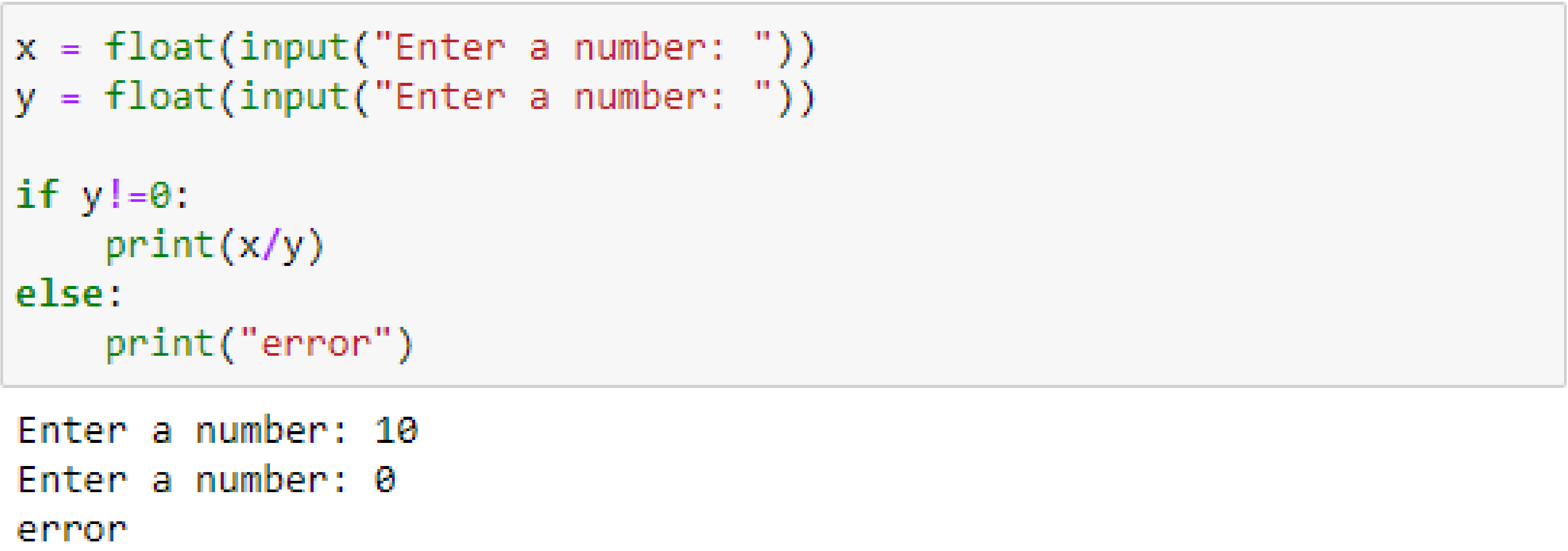
print(‘x>0’)

End

- The alternatives are called branches, because they are branches in the flow of execution.

Alternative execution

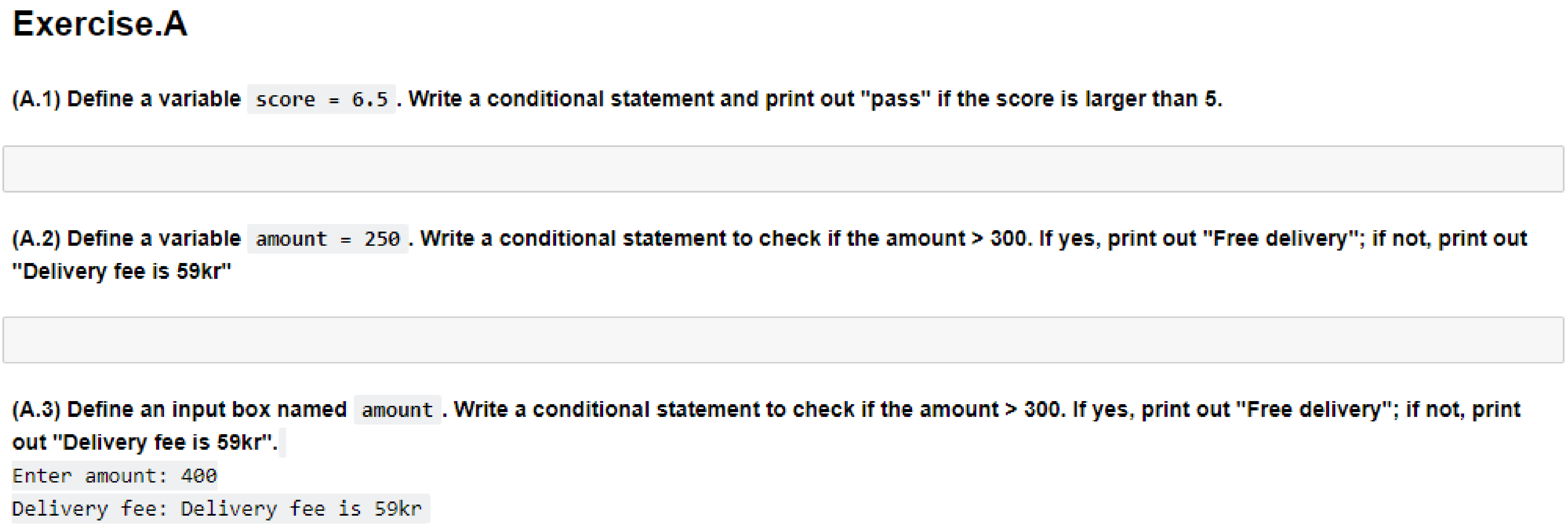
- Example



```
x = float(input("Enter a number: "))
y = float(input("Enter a number: "))

if y!=0:
    print(x/y)
else:
    print("error")
```

```
Enter a number: 10
Enter a number: 0
error
```

Exercise



Exercise.A

(A.1) Define a variable `score = 6.5`. Write a conditional statement and print out "pass" if the score is larger than 5.

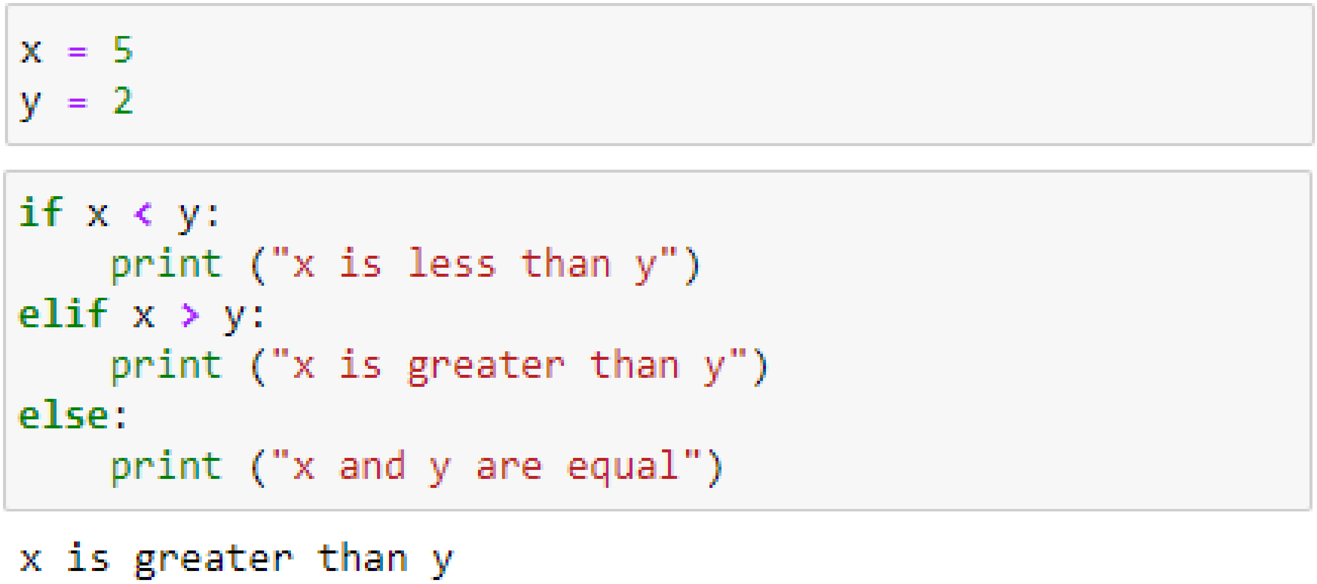
(A.2) Define a variable `amount = 250`. Write a conditional statement to check if the amount > 300. If yes, print out "Free delivery"; if not, print out "Delivery fee is 59kr"

(A.3) Define an input box named `amount`. Write a conditional statement to check if the amount > 300. If yes, print out "Free delivery"; if not, print out "Delivery fee is 59kr".

```
Enter amount: 400
Delivery fee: Delivery fee is 59kr
```

Chained conditionals

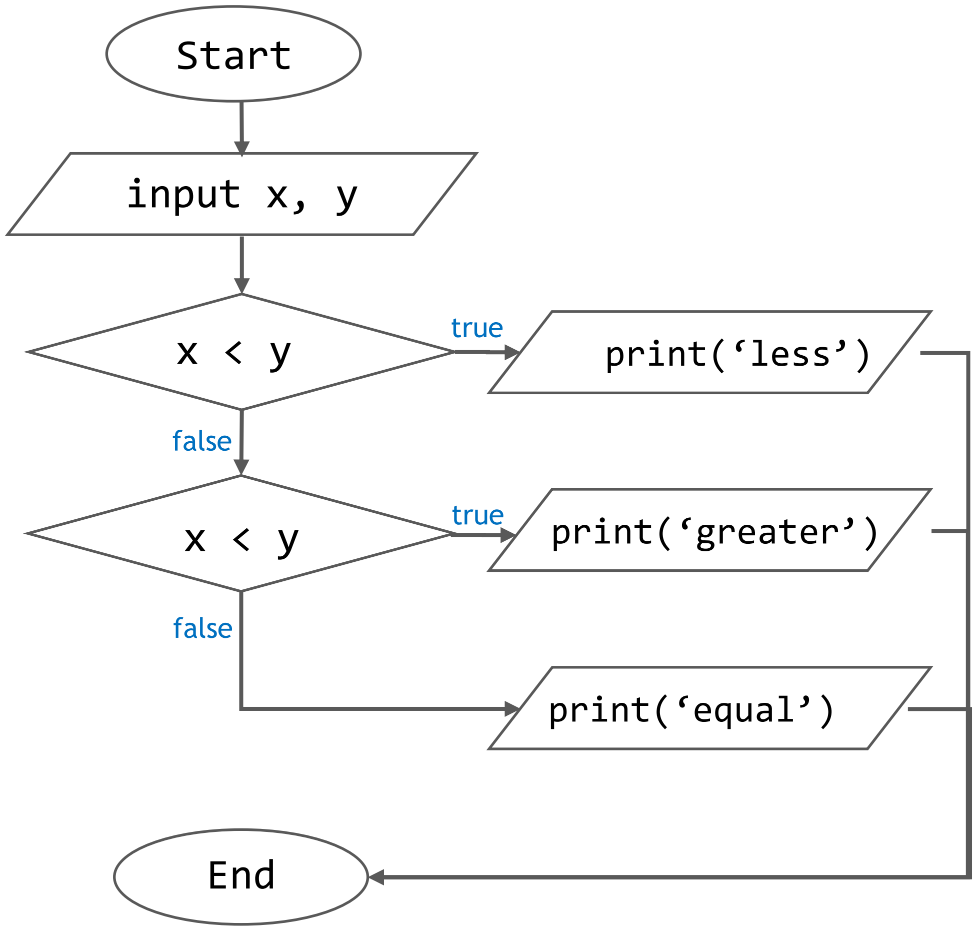
- Sometimes there are more than two possibilities, and we need more than two branches.
- Write a chained conditional statement by using **elif**.



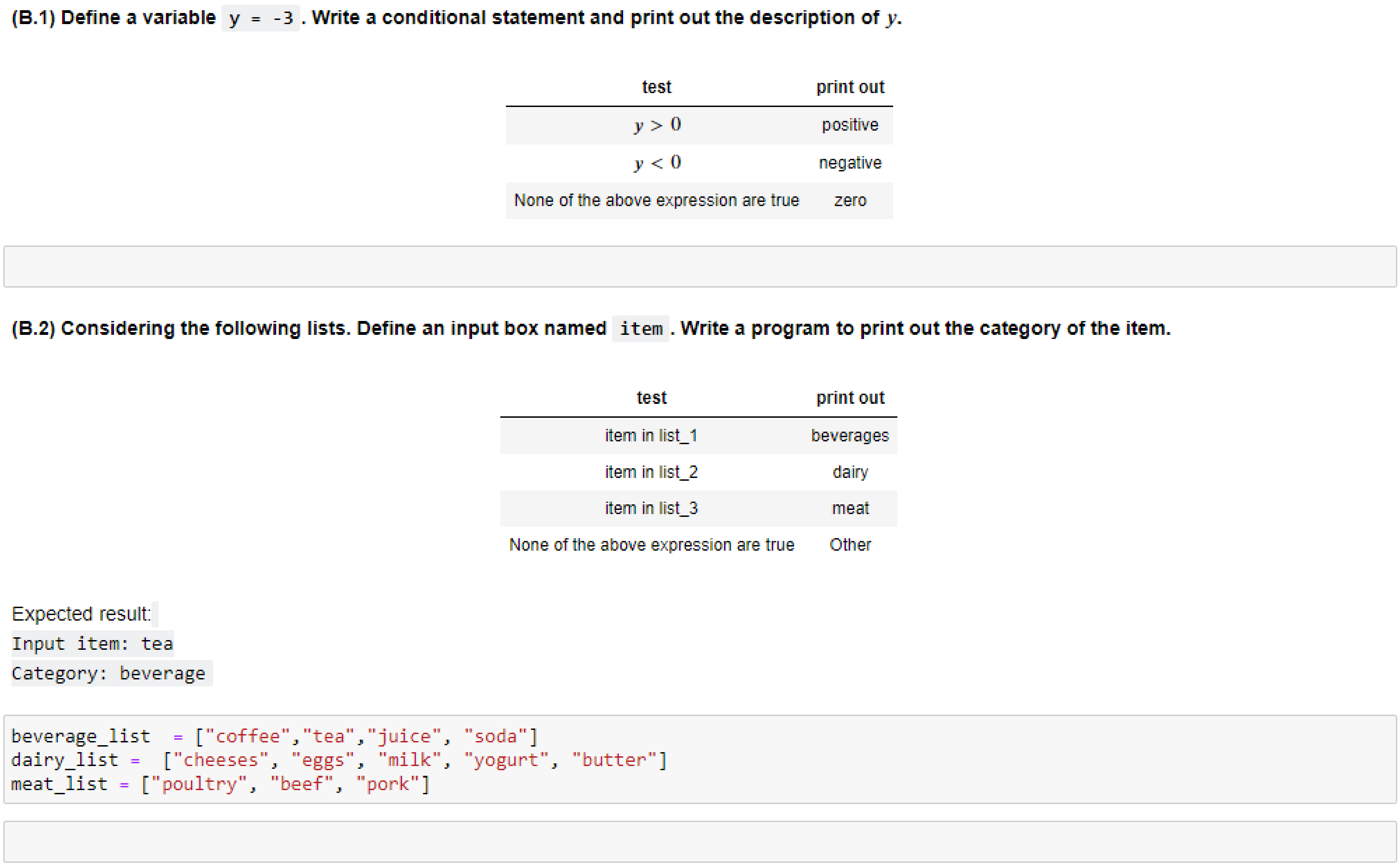
```
x = 5
y = 2
```

```
if x < y:
    print ("x is less than y")
elif x > y:
    print ("x is greater than y")
else:
    print ("x and y are equal")
```

```
x is greater than y
```



Exercise



(B.1) Define a variable `y = -3`. Write a conditional statement and print out the description of y.

test	print out
$y > 0$	positive
$y < 0$	negative
None of the above expression are true	zero

(B.2) Considering the following lists. Define an input box named `item`. Write a program to print out the category of the item.

test	print out
item in list_1	beverages
item in list_2	dairy
item in list_3	meat
None of the above expression are true	Other

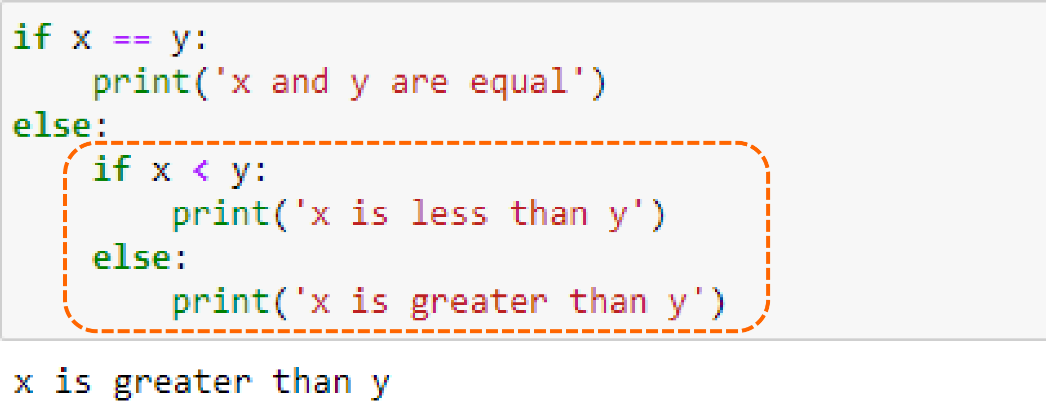
Expected result:

```
Input item: tea
Category: beverage
```

```
beverage_list  = ["coffee","tea","juice", "soda"]
dairy_list =  ["cheeses", "eggs", "milk", "yogurt", "butter"]
meat_list = ["poultry", "beef", "pork"]
```

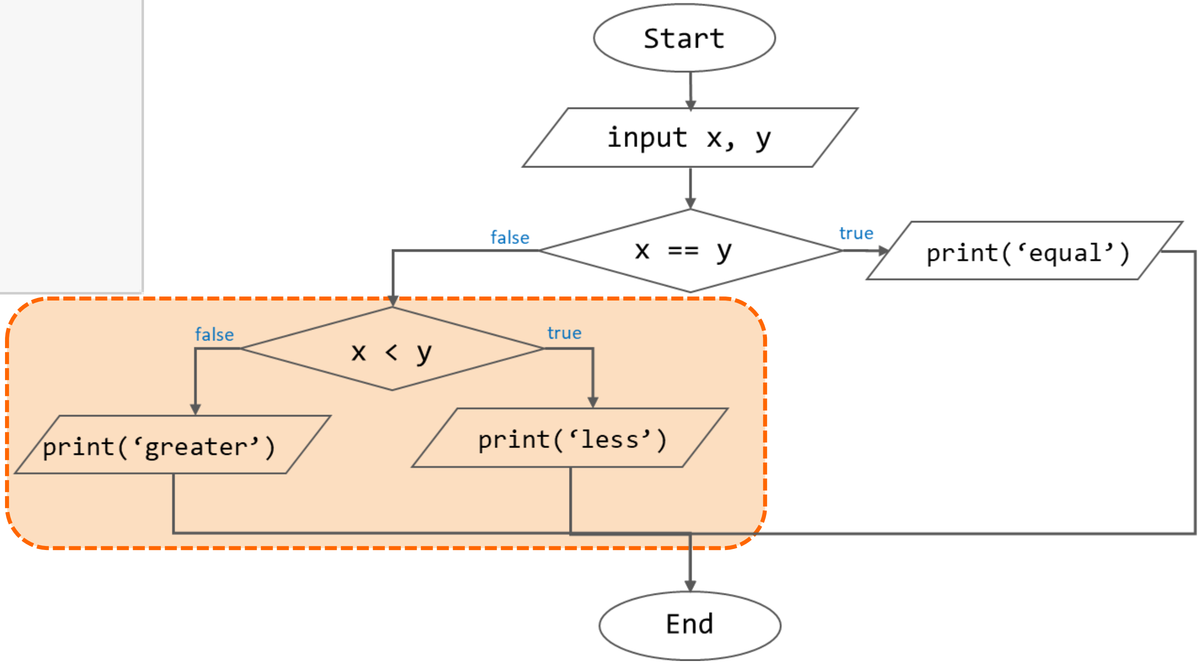
Nested conditionals

- One conditional can also be nested within another.

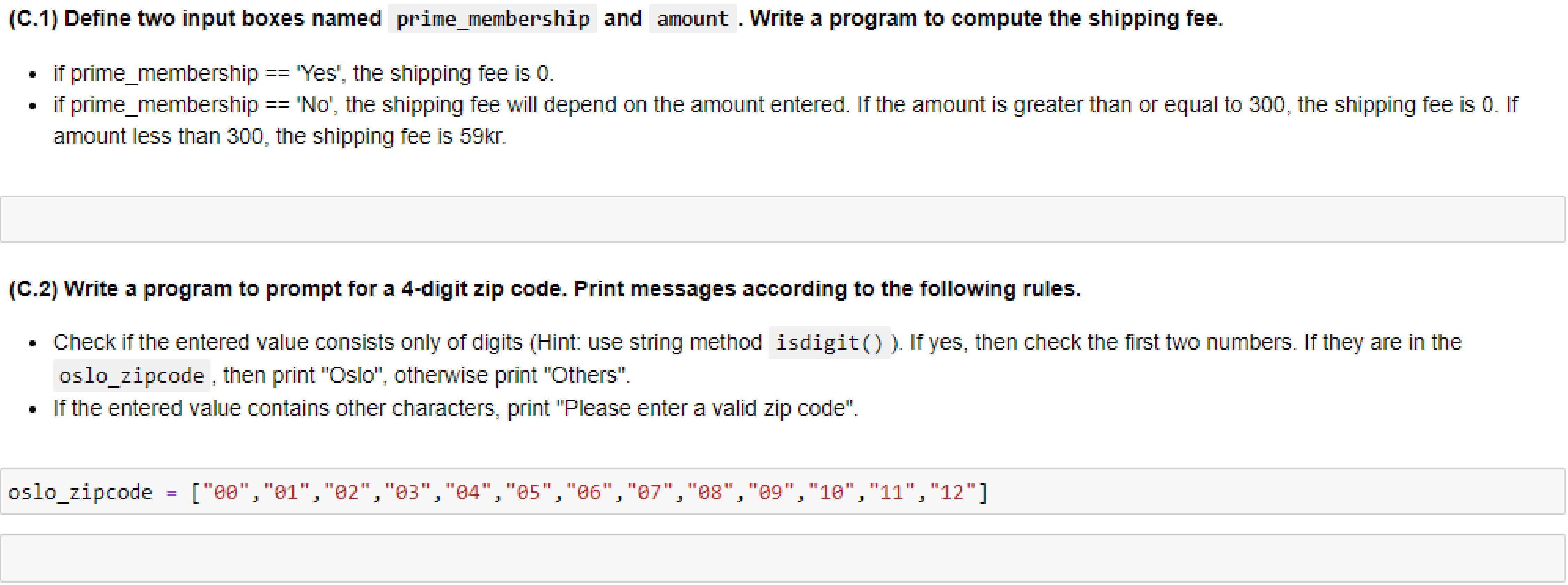


```
if x == y:
    print('x and y are equal')
else:
    if x < y:
        print('x is less than y')
    else:
        print('x is greater than y')
```

```
x is greater than y
```



Exercise



(C.1) Define two input boxes named `prime_membership` and `amount`. Write a program to compute the shipping fee.

- if prime_membership == 'Yes', the shipping fee is 0.
- if prime_membership == 'No', the shipping fee will depend on the amount entered. If the amount is greater than or equal to 300, the shipping fee is 0. If amount less than 300, the shipping fee is 59kr.

(C.2) Write a program to prompt for a 4-digit zip code. Print messages according to the following rules.

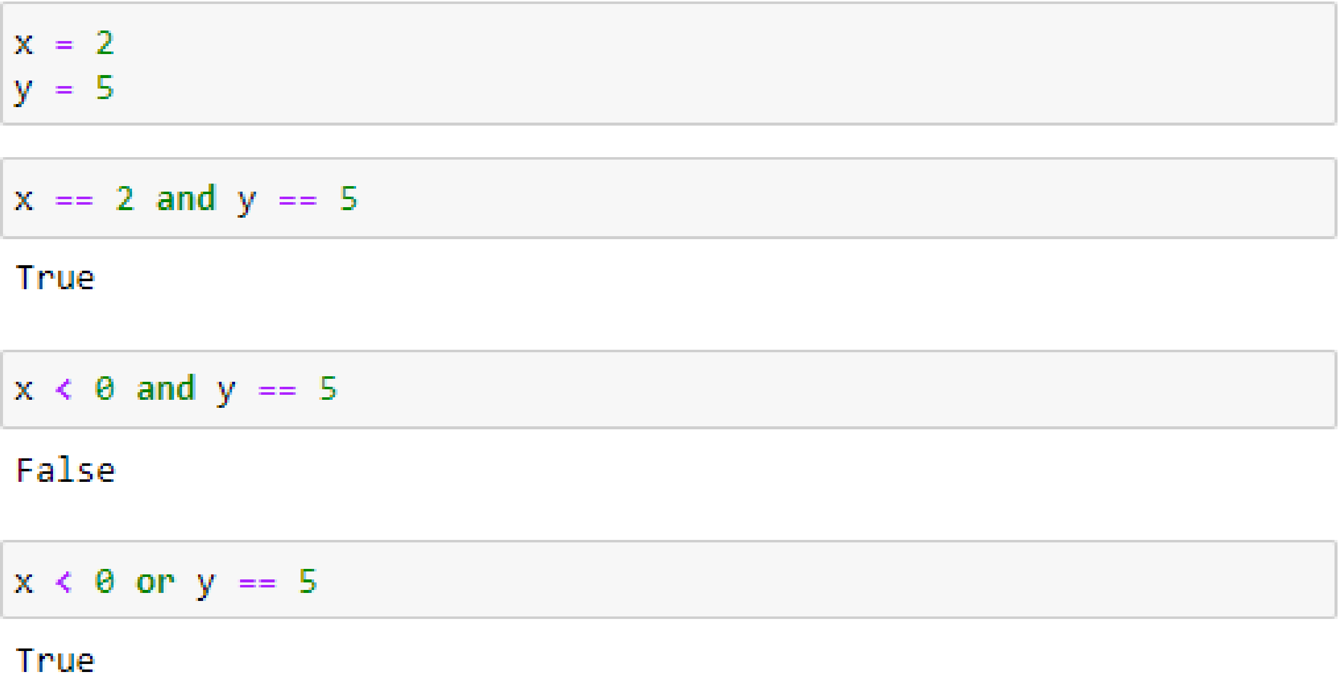
- Check if the entered value consists only of digits (Hint: use string method `isdigit()`). If yes, then check the first two numbers. If they are in the `oslo_zipcode`, then print "Oslo", otherwise print "Others".
- If the entered value contains other characters, print "Please enter a valid zip code".

```
oslo_zipcode = ["00","01","02","03","04","05","06","07","08","09","10","11","12"]
```

Boolean expression – logical operators

- Logical operators are used to combine boolean constraints.

Logical operator	Meaning
and	and
or	or
not	not



```
x = 2
y = 5
```

```
x == 2 and y == 5
```

```
True
```

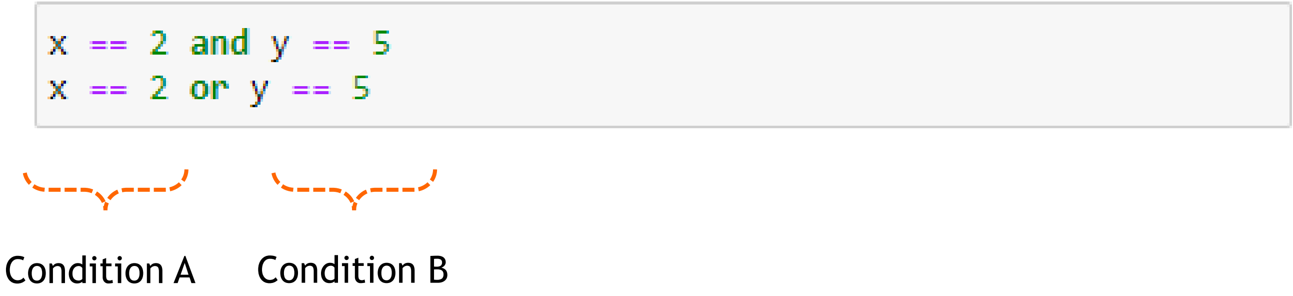
```
x < 0 and y == 5
```

```
False
```

```
x < 0 or y == 5
```

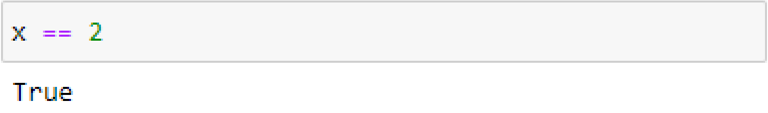
```
True
```

Boolean expression – logical operators



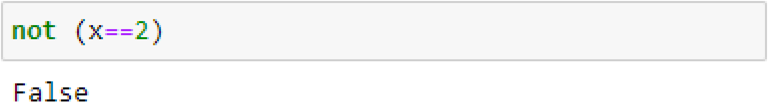
```
x == 2 and y == 5
x == 2 or y == 5
```

Condition A   Condition B



```
x == 2
```

```
True
```



```
not (x==2)
```

```
False
```

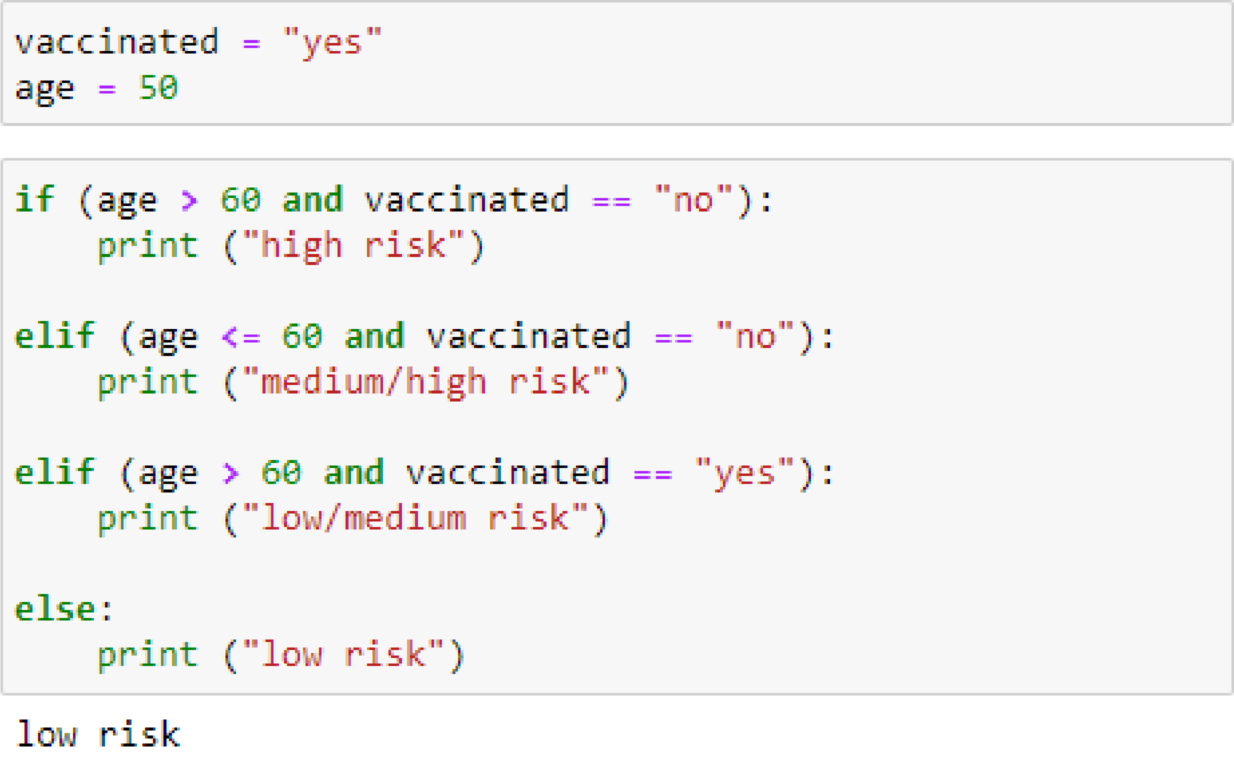
A	B	A and B
True	True	True
True	False	False
False	True	False
False	False	False

A	B	A or B
True	True	True
True	False	True
False	True	True
False	False	False

A	not A
True	False
False	True

Conditional statements with Logical operators

Risk	vaccinated = no	vaccinated = yes
Age >60	high	low/medium
Age <=60	medium/high	low

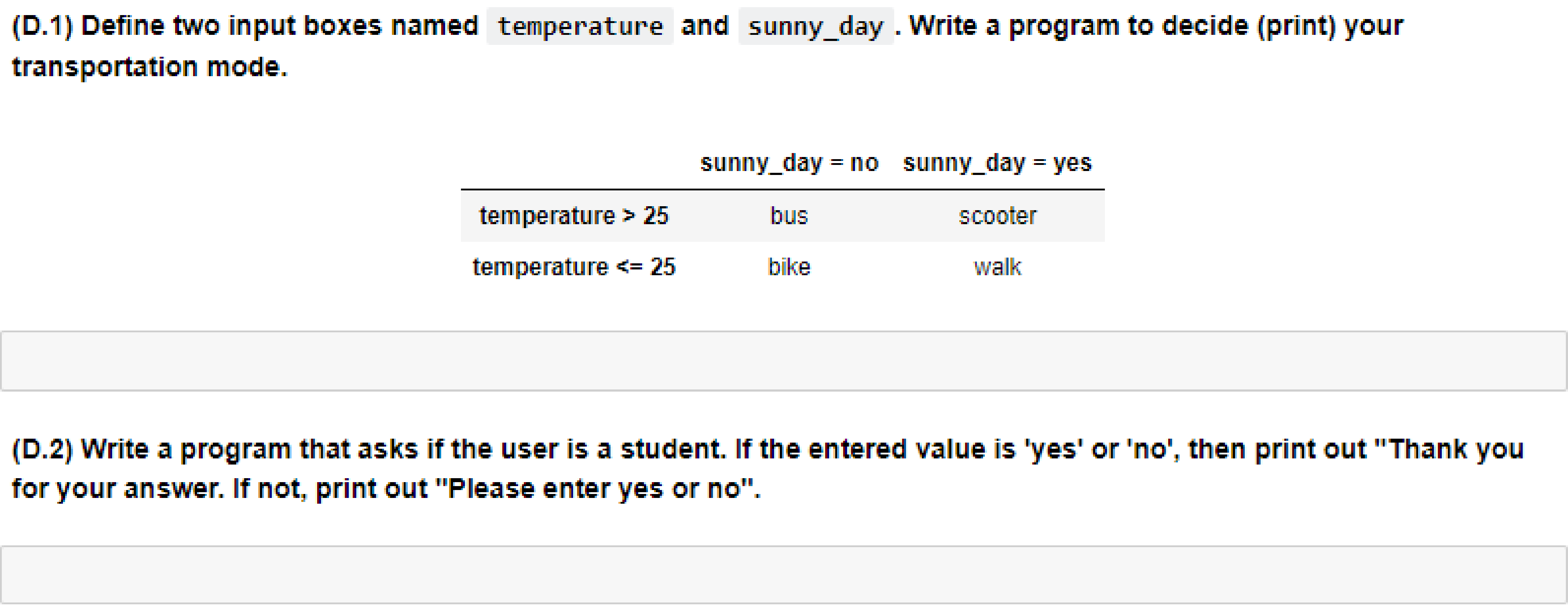


```python
vaccinated = "yes"
age = 50
```

```python
if (age > 60 and vaccinated == "no"):
    print ("high risk")

elif (age <= 60 and vaccinated == "no"):
    print ("medium/high risk")

elif (age > 60 and vaccinated == "yes"):
    print ("low/medium risk")

else:
    print ("low risk")
```

```
low risk
```

Exercise



(D.1) Define two input boxes named `temperature` and `sunny_day`. Write a program to decide (print) your transportation mode.

	sunny_day = no	sunny_day = yes
temperature > 25	bus	scooter
temperature <= 25	bike	walk

(D.2) Write a program that asks if the user is a student. If the entered value is 'yes' or 'no', then print out "Thank you for your answer. If not, print out "Please enter yes or no".