Conditional Statements

Outline

- Boolean expressions
- Flowcharts
- Conditional statements
  - if
  - if, else
  - Chained conditionals
  - Nested conditionals
  - Conditional statements with logical operators
- Exception handling

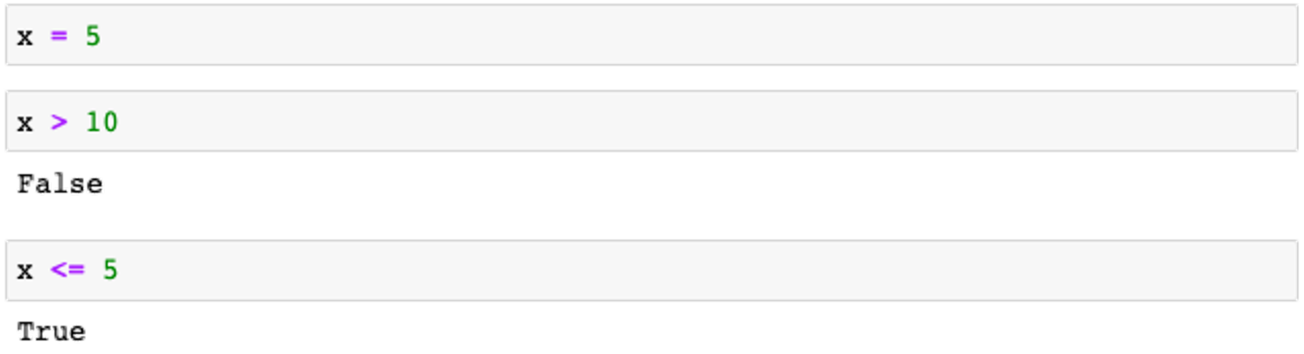
Data types

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 evaluates to either <u>true</u> or <u>false</u>.
- Boolean expressions are often used to make decisions in programming.



```
x = 5
```

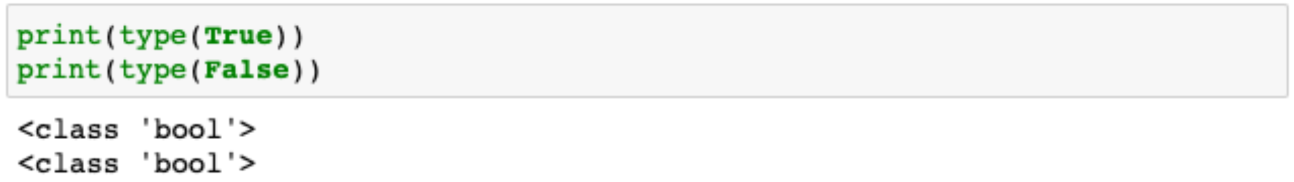
```
x > 10
```

```
False
```

```
x <= 5
```

```
True
```

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



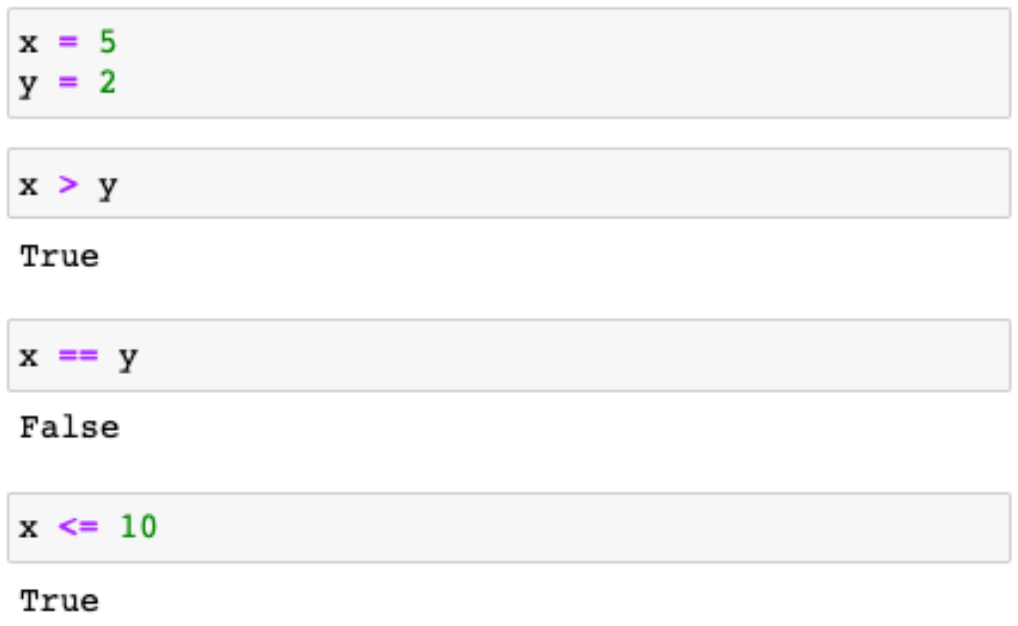
```
print(type(True))
print(type(False))
```

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

Boolean expression – comparison operators

- Boolean expressions often involve comparison operators.

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



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

```
x > y
```

```
True
```

```
x == y
```

```
False
```

```
x <= 10
```

```
True
```

Flowcharts

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

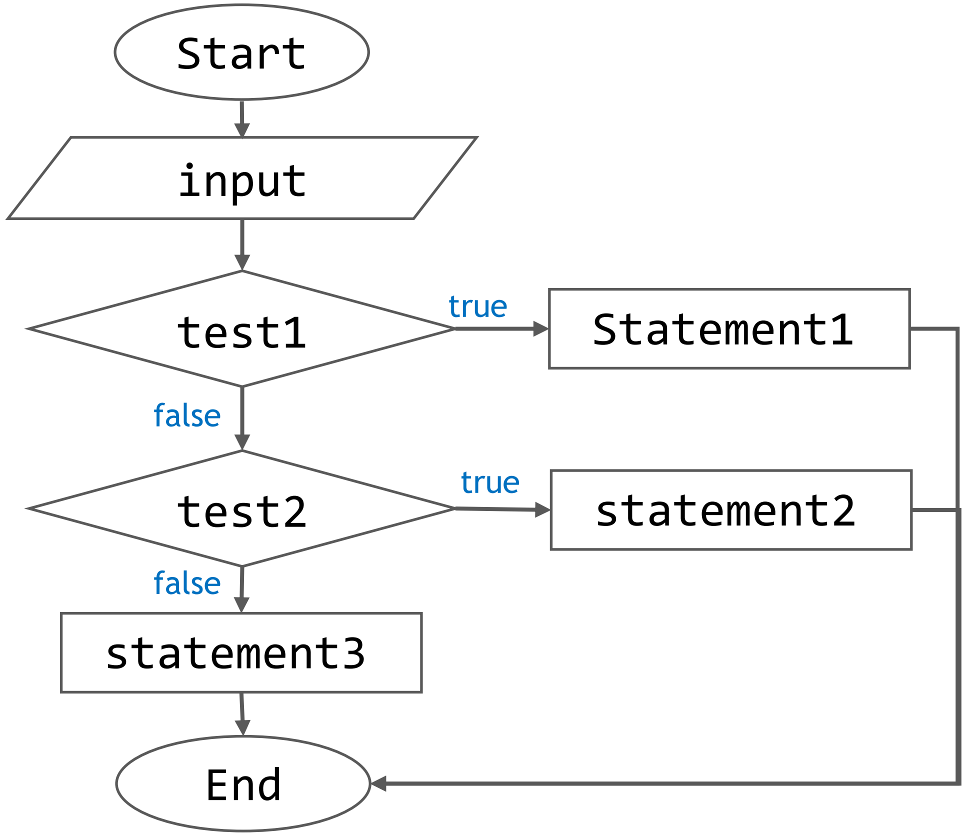
Symbol	Name
⬭	Start/end
▱	Input/output
▭	Process (e.g., assign a value, calculate a value)
◇	Decision (determine whether an expression is true/false)

Conditional statements

- Conditional statements perform different computations or actions depending on whether a specific boolean expression evaluates to true or false.

General format

```
if <test1 is true>:
      <statement1>
elif <test2 is true>:   #optional
      <statement2>
else:                   #optional
      <statement3>
```



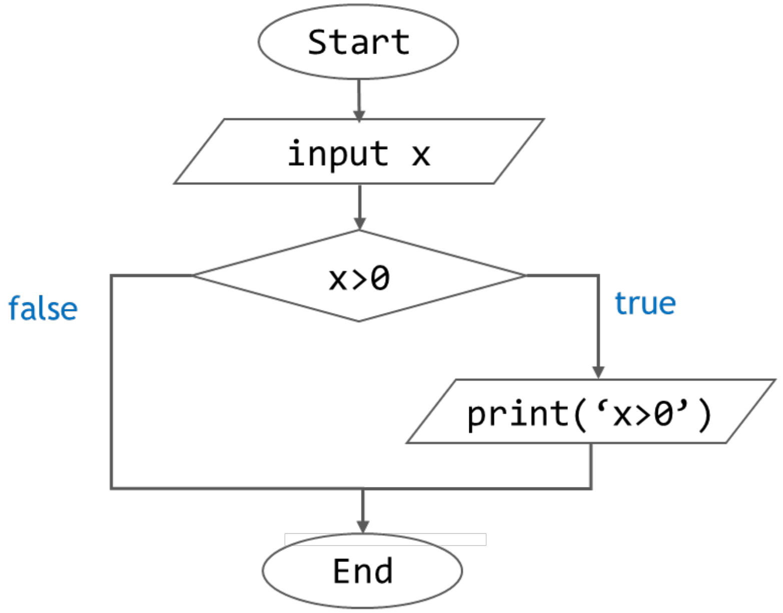
Conditional statements - if

- 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.

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

```
x > 0
```

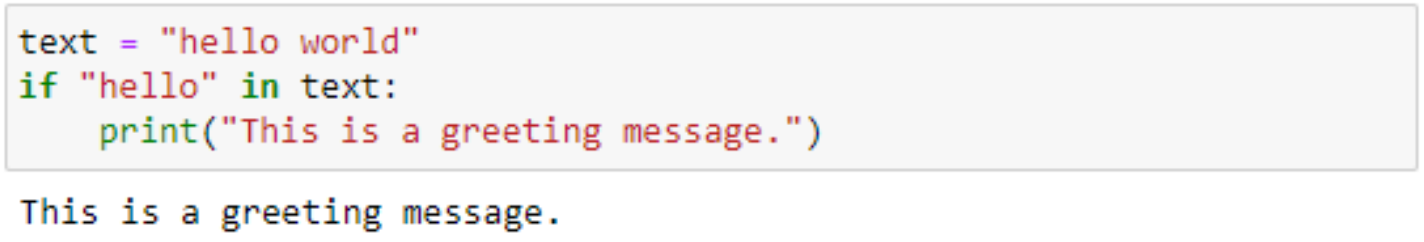
- If the condition is true (x>0), then the indented line gets executed.
- If the condition is false, the indented line is skipped.



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

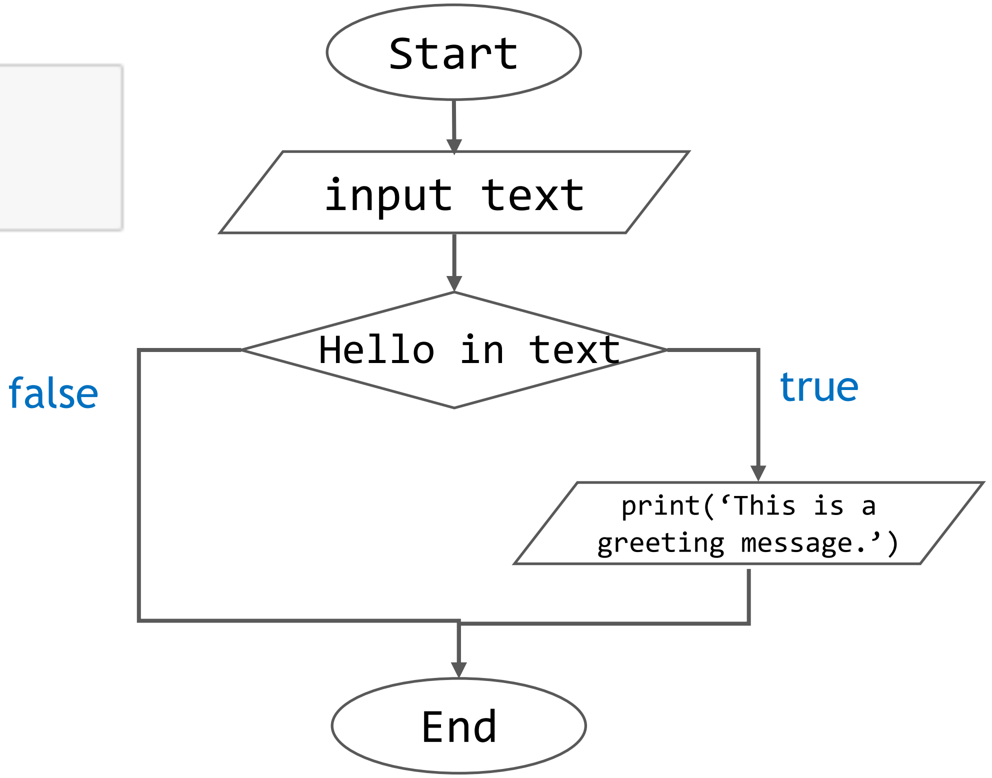
Conditional statements - if

- Example



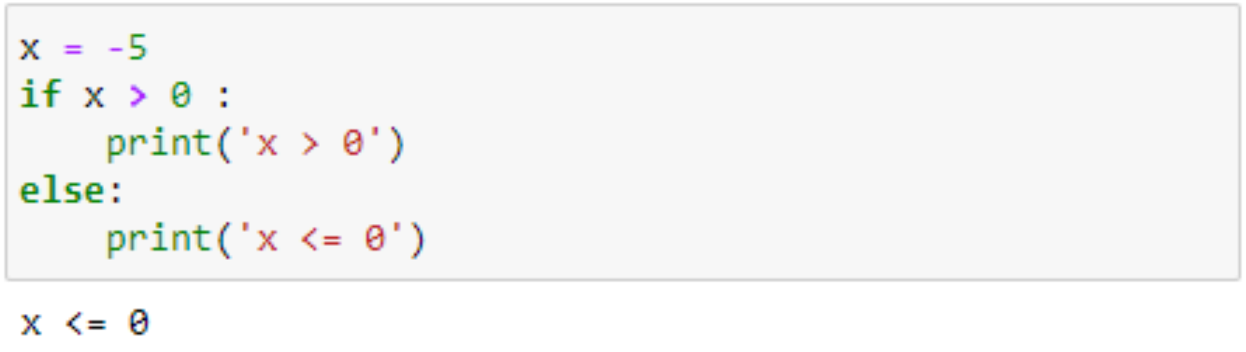
```
text = "hello world"
if "hello" in text:
    print("This is a greeting message.")
```

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



Conditional statements – if, else

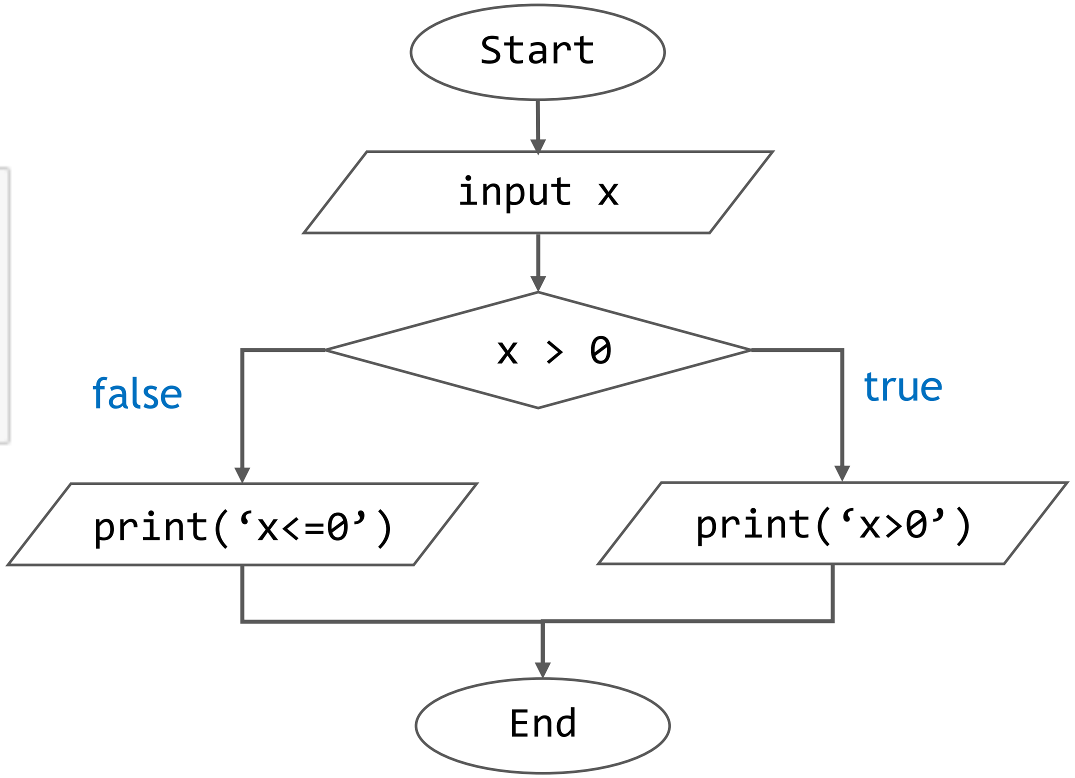
- It is also possible to execute a statement when the condition is false.



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

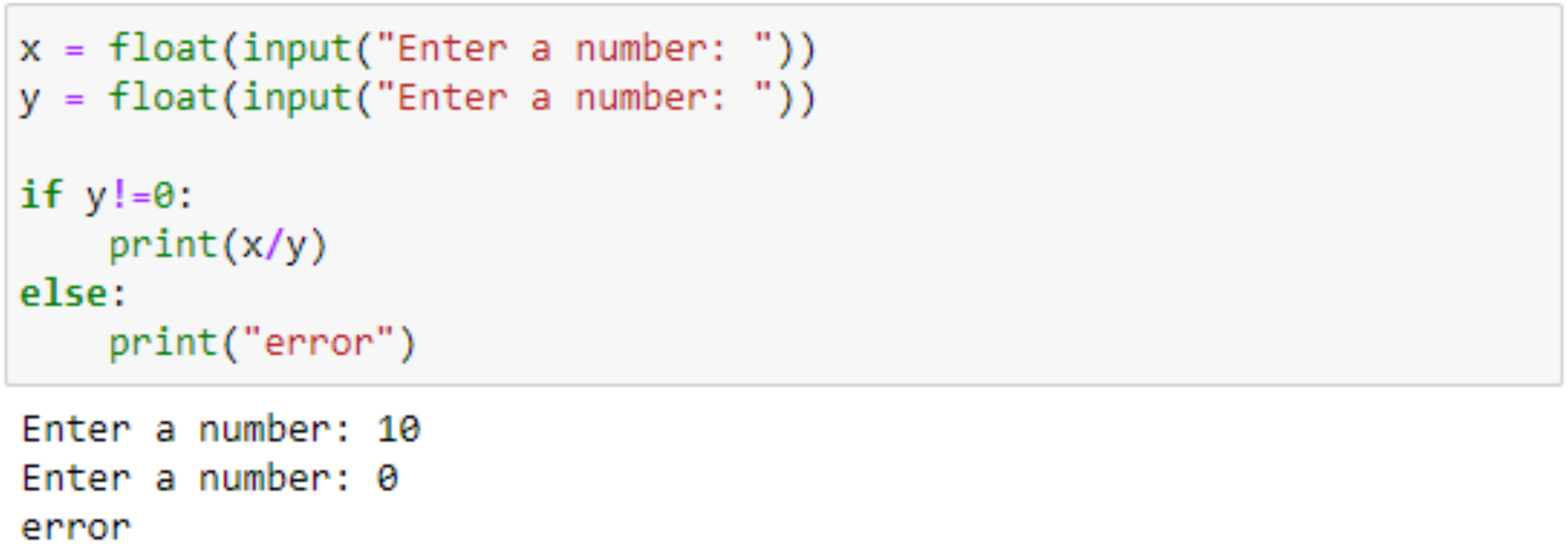
```
x <= 0
```

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



Conditional statements – if, else

- 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 to check if the score is greater than 5. If yes, print `Pass`; otherwise, print `Fail`.

(A.2) Write a program that asks users to enter a value. The program should check if the entered value is an integer. If yes, print "integer", otherwise, print "not an integer".
Hint: Use string method `isdigit()` to check if the entered value consists only of digits.

Example-1:
Enter an integer: `abc123`
`not an integer.`

Example-2:
Enter an integer: `123`
`integer.`

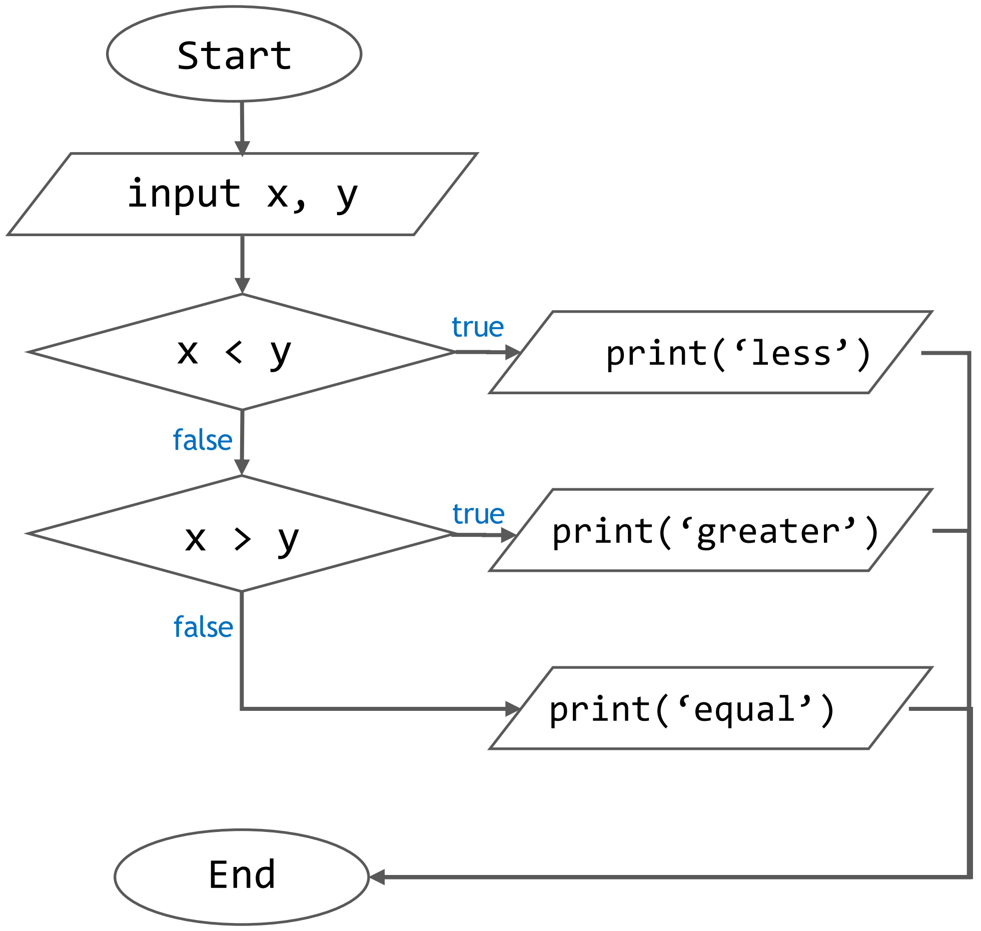
Chained conditionals

- 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
```



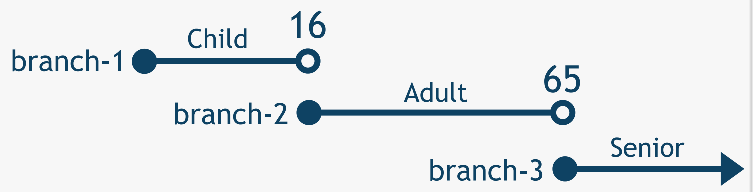
Chained conditionals

Age	Print out age group
< 16	Child
16~64	Adult
> = 65	Senior

○ Exclusive
● Inclusive

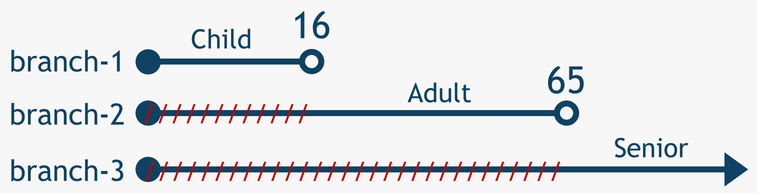
Solution-1

```python
if age < 16:
    print("Child")
elif 16 <= age < 65:
    print("Adult")
else:
    print("Senior")
```



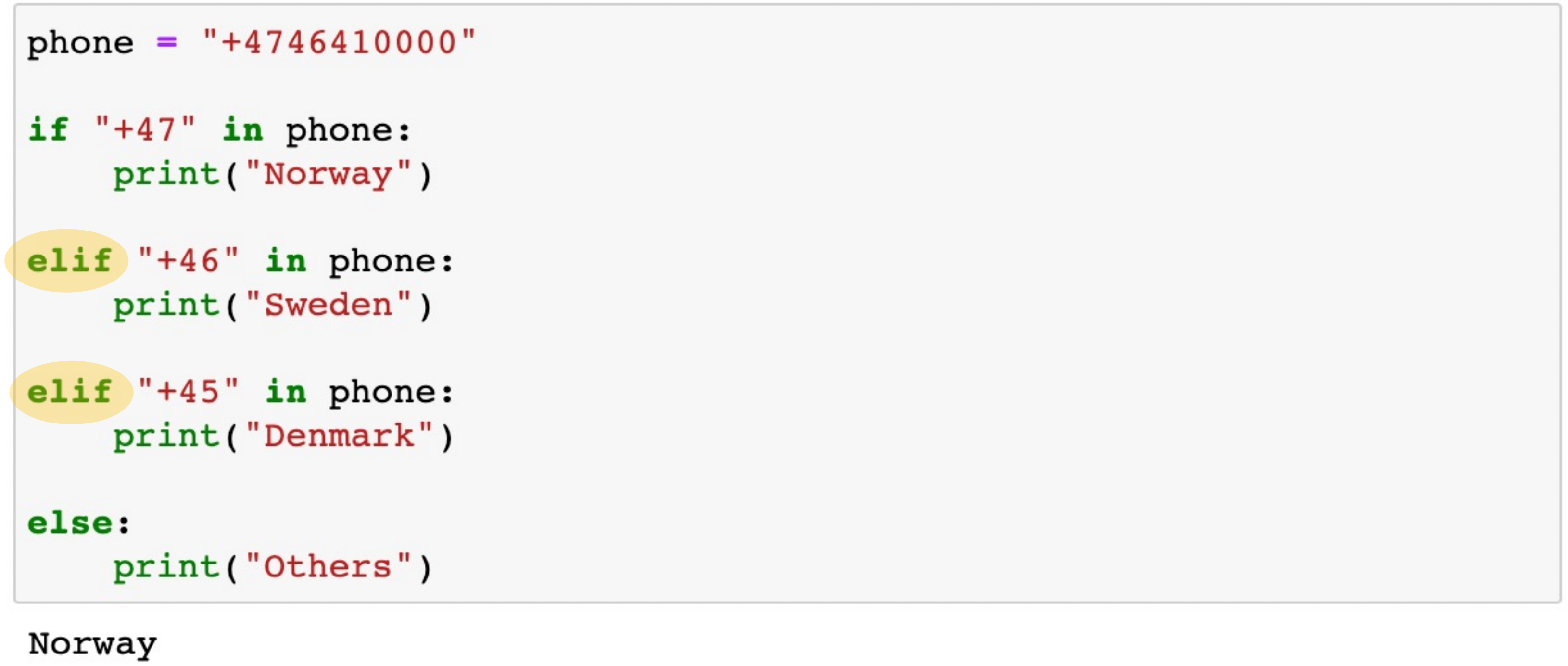
Solution-2

```python
if age < 16:
    print("Child")
elif age < 65:
    print("Adult")
else:
    print("Senior")
```



Chained conditionals

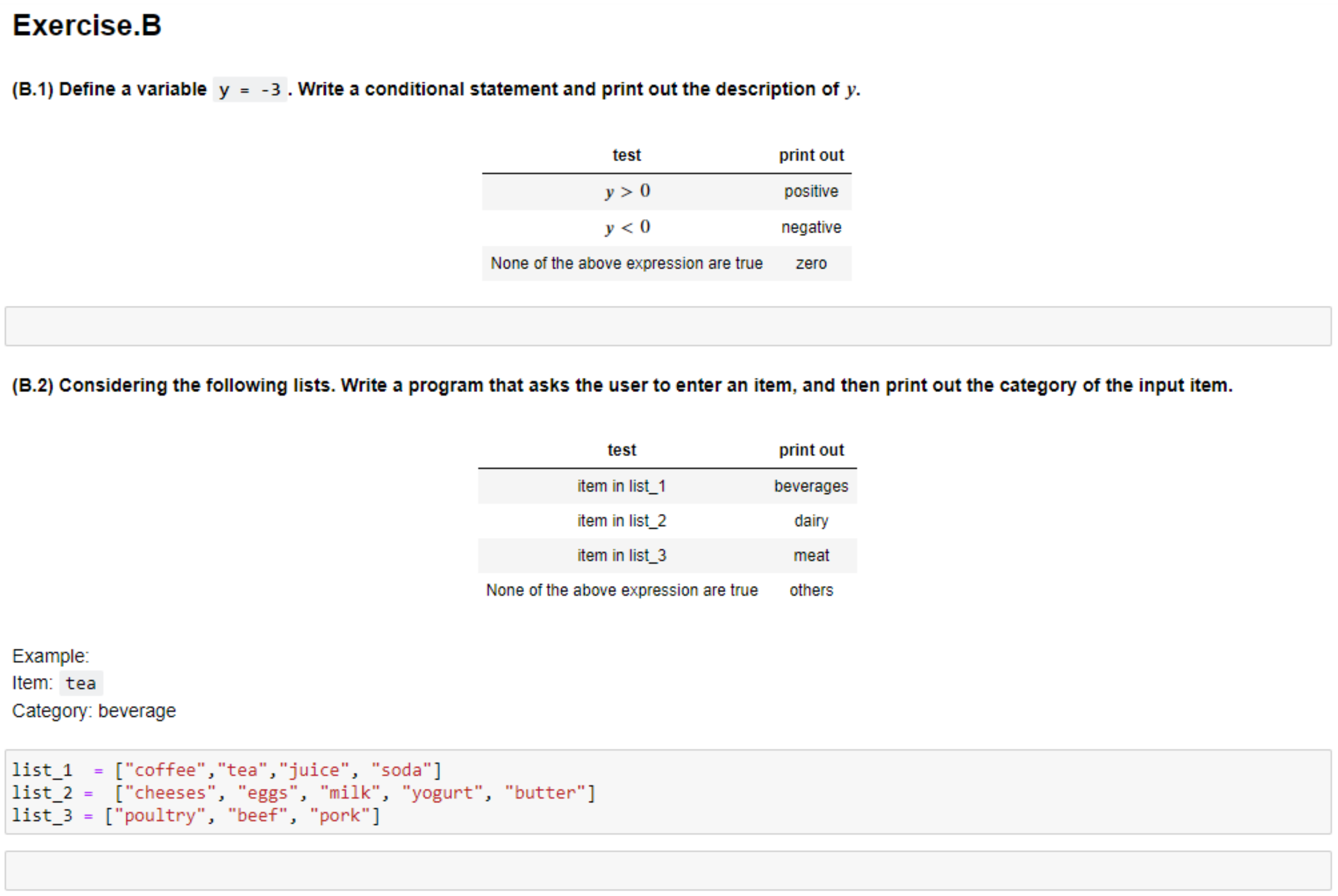
- More branches



```
phone = "+4746410000"

if "+47" in phone:
    print("Norway")

elif "+46" in phone:
    print("Sweden")

elif "+45" in phone:
    print("Denmark")

else:
    print("Others")
```

```
Norway
```

Exercise



Exercise.B

(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. Write a program that asks the user to enter an item, and then print out the category of the input item.

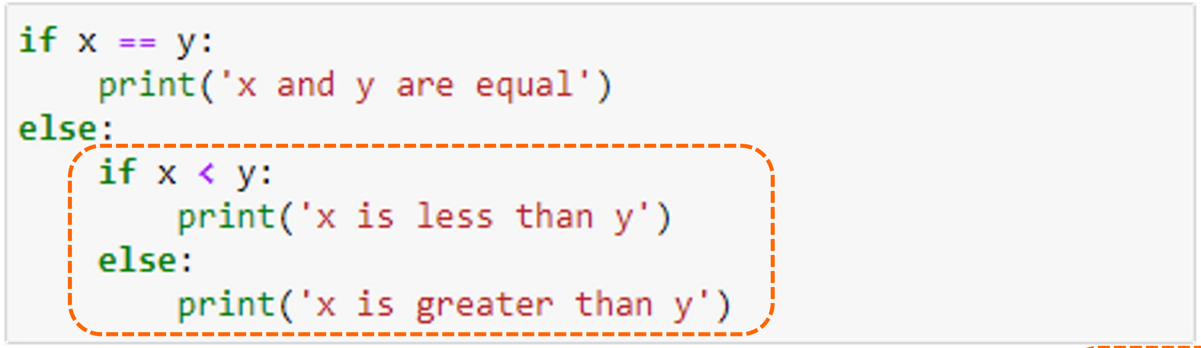
test	print out
item in list_1	beverages
item in list_2	dairy
item in list_3	meat
None of the above expression are true	others

Example:
Item: `tea`
Category: beverage

```
list_1  = ["coffee","tea","juice", "soda"]
list_2 =  ["cheeses", "eggs", "milk", "yogurt", "butter"]
list_3 = ["poultry", "beef", "pork"]
```

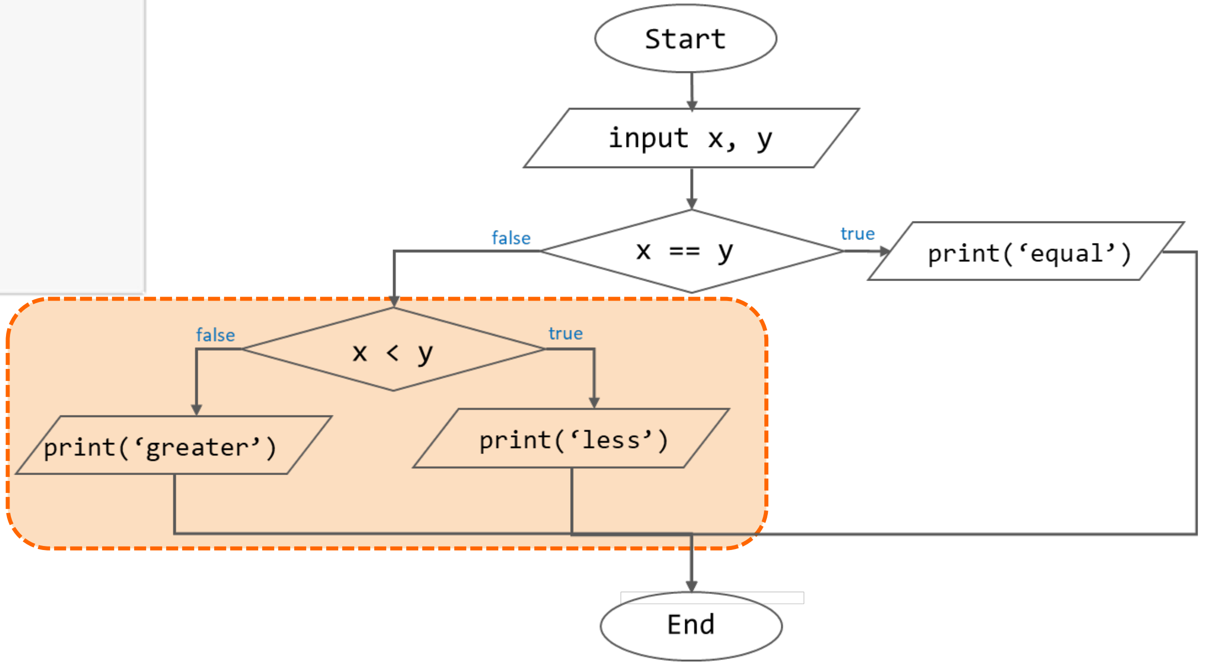
Nested conditionals

- 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')
```

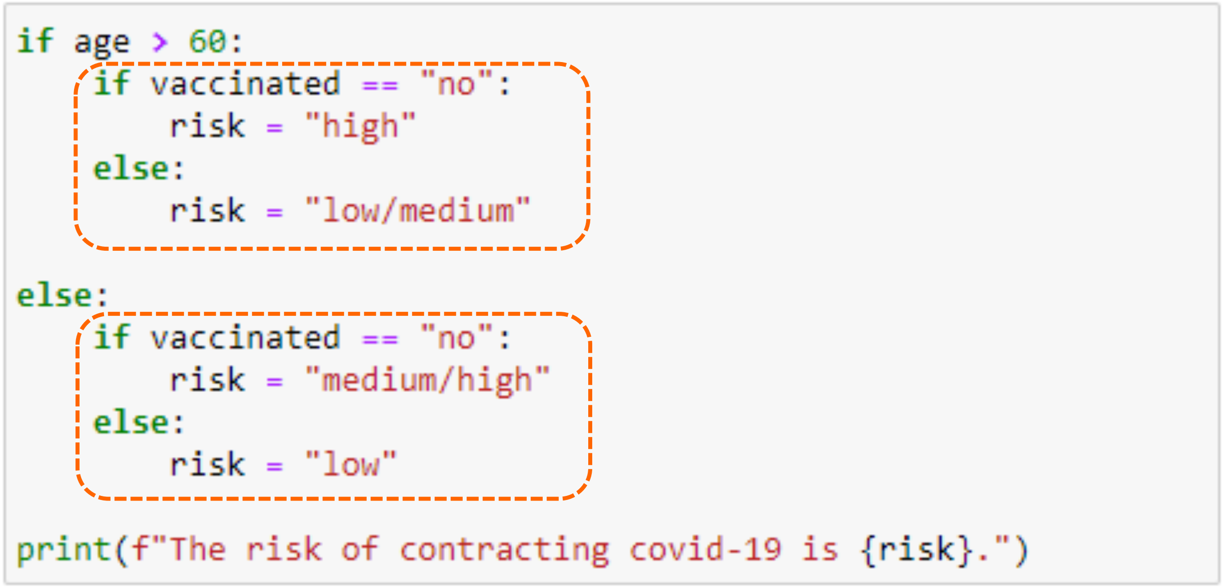
```
x is greater than y
```



Nested conditionals

- Example: Risk of contracting Covid-19

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



```
if age > 60:
    if vaccinated == "no":
        risk = "high"
    else:
        risk = "low/medium"

else:
    if vaccinated == "no":
        risk = "medium/high"
    else:
        risk = "low"

print(f"The risk of contracting covid-19 is {risk}.")
```

Exercise

Exercise.C

(C.1) Write a program that asks the user to enter an amount and whether they are a member. Print shipping fee according to the table below.

- Free shipping if the user is a member.
- If the user is not a member, the shipping fee will depend on the amount entered.

Shipping fee	Amount < 500	Amount >= 500
Non-members	69kr	49kr
Member	0	0

Example:
Amount: `350`
Are you a member (yes/no)? `no`
The shipping fee is 69 kr.

Boolean expression – logical operators

- Logical operators are used to combine different conditions.

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

$$\underbrace{\texttt{x == 2}}_{\text{Condition A}}\ \texttt{and}\ \underbrace{\texttt{y == 5}}_{\text{Condition B}}$$

$$\underbrace{\texttt{x == 2}}_{\text{Condition A}}\ \texttt{or}\ \underbrace{\texttt{y == 5}}_{\text{Condition B}}$$

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

```
x == 2
```

```
True
```

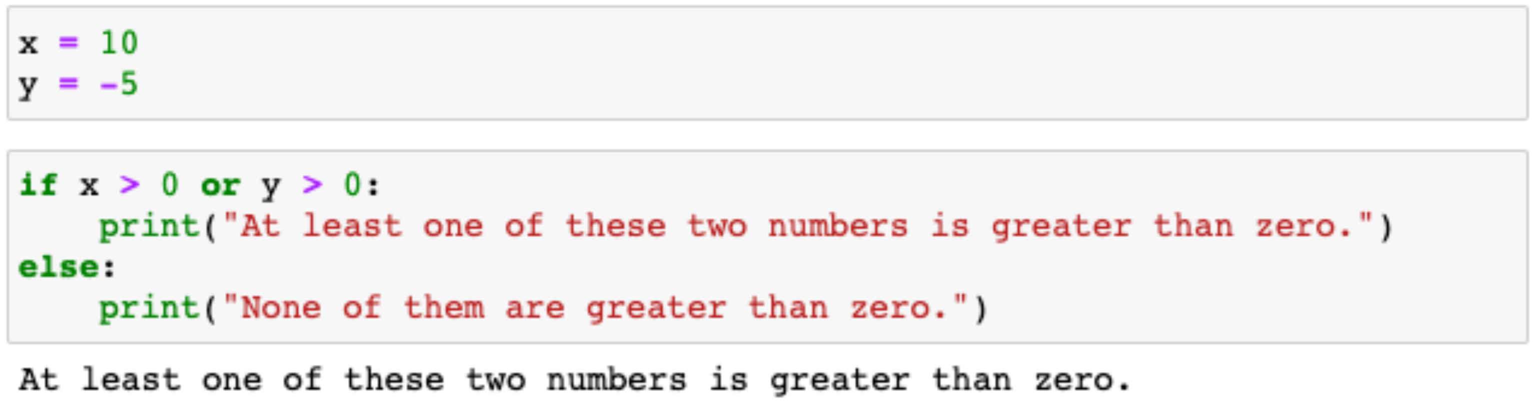
```
not (x==2)
```

```
False
```

A	not A
True	False
False	True

Conditional statements with logical operators

- Example:



```
x = 10
y = -5
```

```
if x > 0 or y > 0:
    print("At least one of these two numbers is greater than zero.")
else:
    print("None of them are greater than zero.")
```

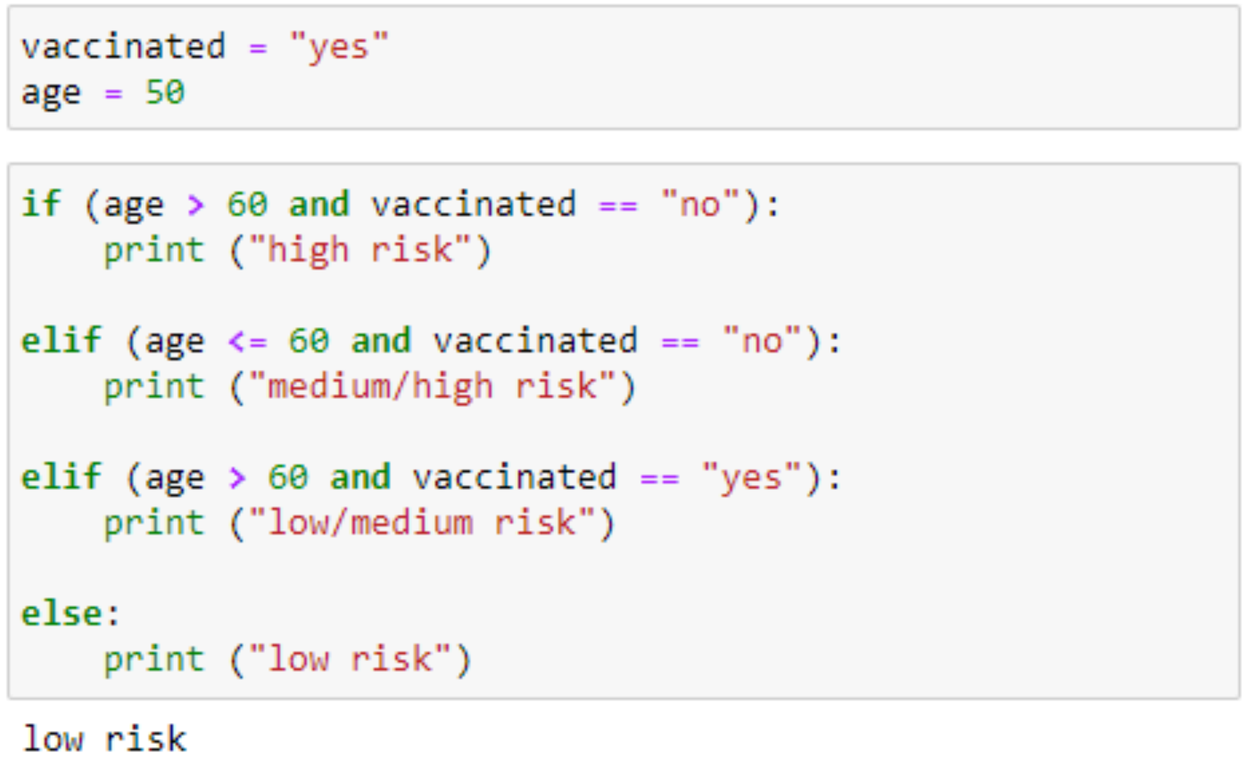
```
At least one of these two numbers is greater than zero.
```

x>0	y>0	x>0 or y>0
True	True	True
True	False	True
False	True	True
False	False	False

Conditional statements with logical operators

- Example: Risk of contracting Covid-19

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

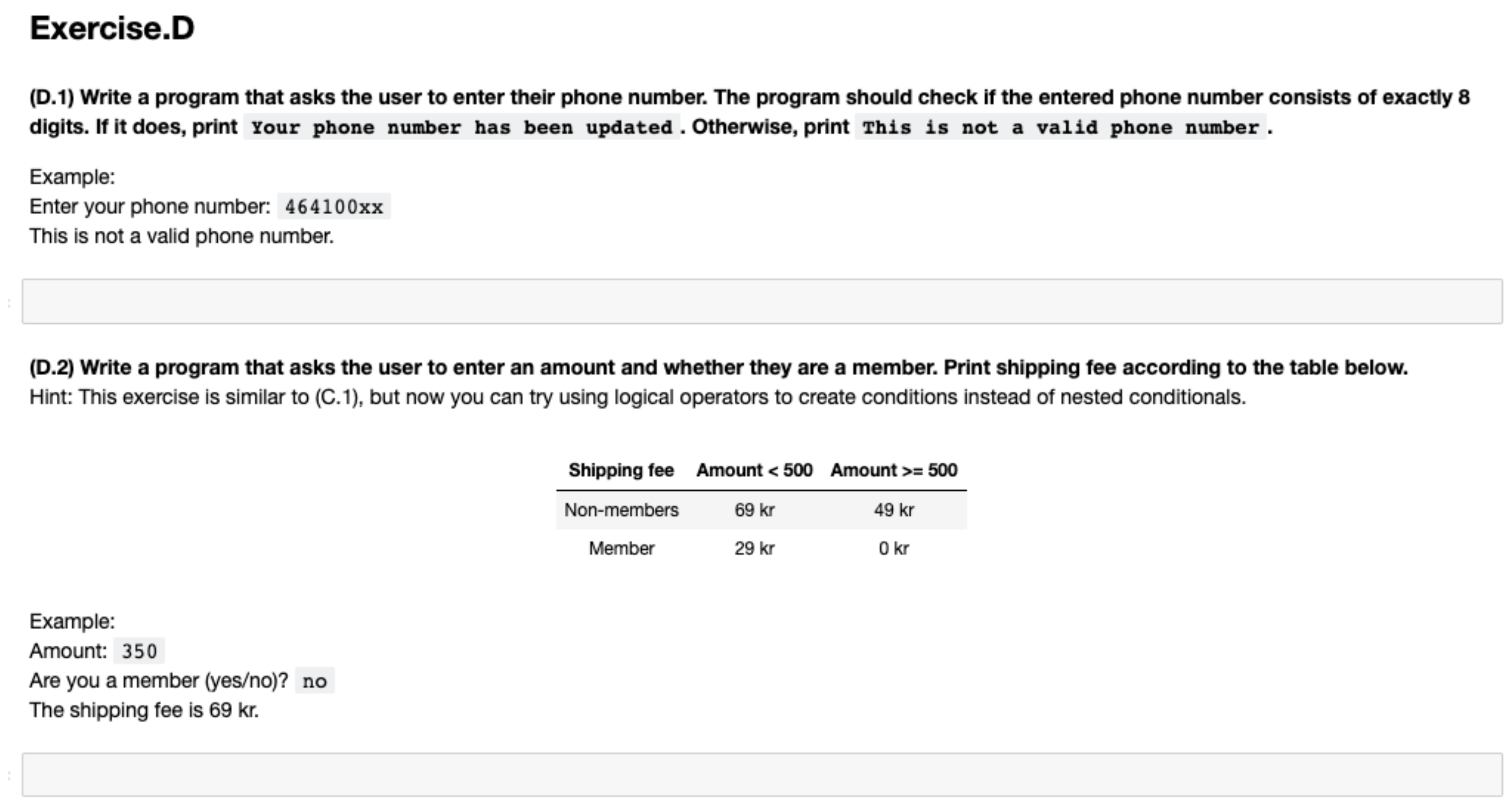


```
vaccinated = "yes"
age = 50
```

```
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



Exercise.D

(D.1) Write a program that asks the user to enter their phone number. The program should check if the entered phone number consists of exactly 8 digits. If it does, print `Your phone number has been updated`. Otherwise, print `This is not a valid phone number`.

Example:
Enter your phone number: `464100xx`
This is not a valid phone number.

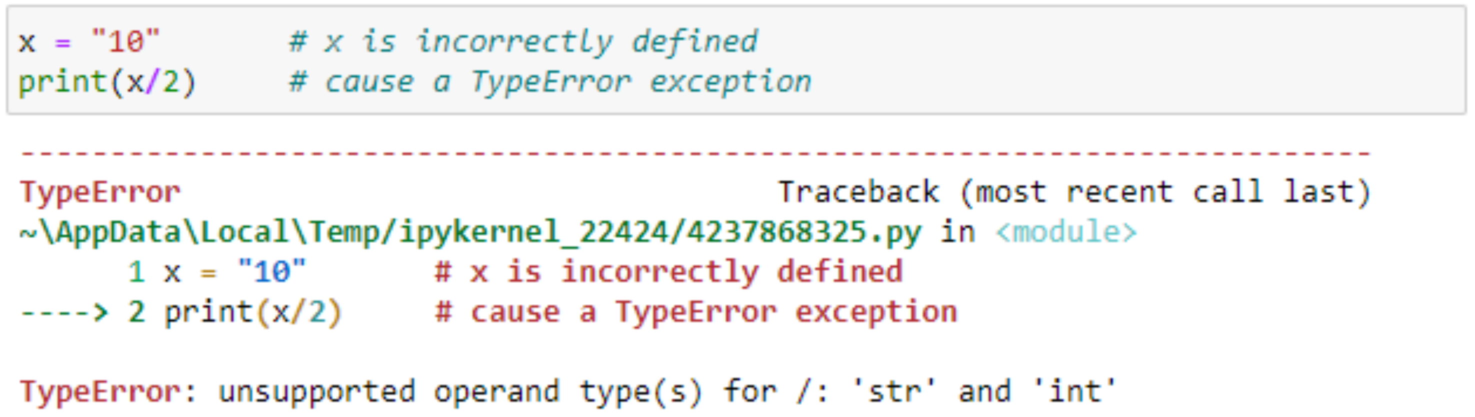
(D.2) Write a program that asks the user to enter an amount and whether they are a member. Print shipping fee according to the table below.
Hint: This exercise is similar to (C.1), but now you can try using logical operators to create conditions instead of nested conditionals.

Shipping fee	Amount < 500	Amount >= 500
Non-members	69 kr	49 kr
Member	29 kr	0 kr

Example:
Amount: `350`
Are you a member (yes/no)? `no`
The shipping fee is 69 kr.

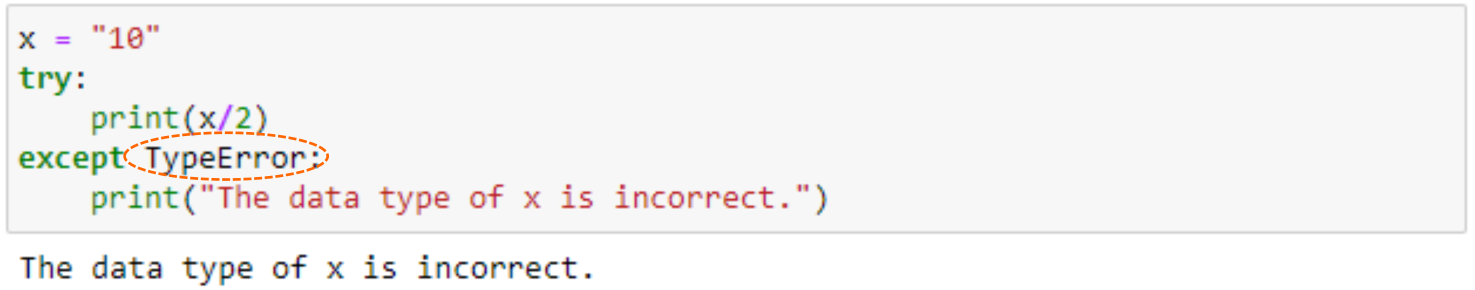
Exception handling – Try Except

- Use `try` and `except` to respond to the occurrence of an exception.



```
x = "10"         # x is incorrectly defined
print(x/2)       # cause a TypeError exception
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_22424/4237868325.py in <module>
      1 x = "10"         # x is incorrectly defined
----> 2 print(x/2)       # cause a TypeError exception

TypeError: unsupported operand type(s) for /: 'str' and 'int'
```

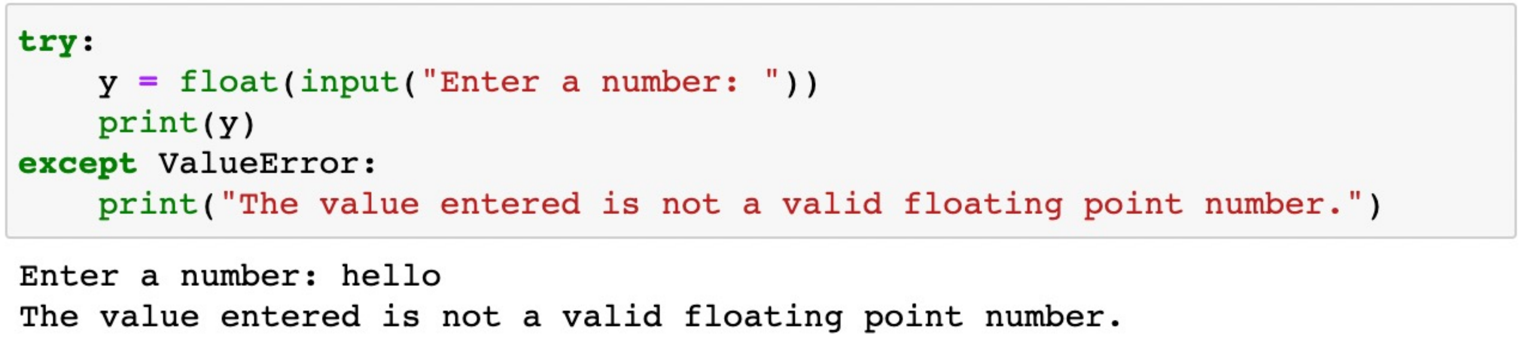


```
x = "10"
try:
    print(x/2)
except TypeError:
    print("The data type of x is incorrect.")
```

```
The data type of x is incorrect.
```

Exception handling – Try Except

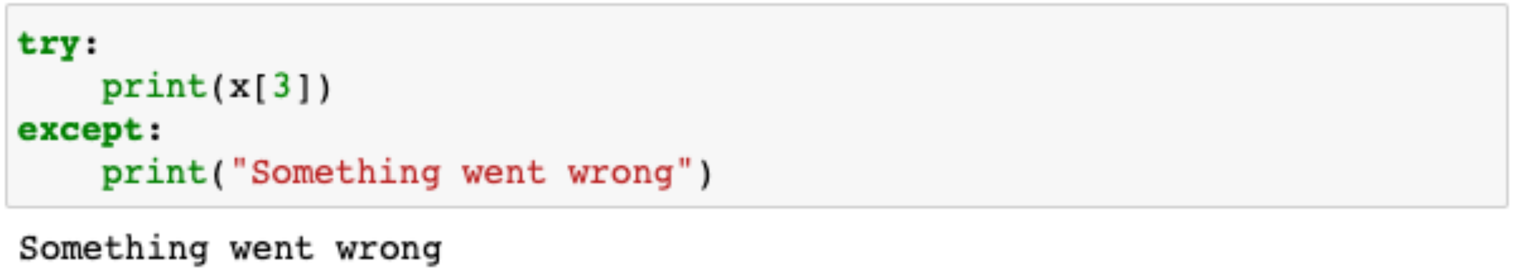
- Catch ValueError



```python
try:
    y = float(input("Enter a number: "))
    print(y)
except ValueError:
    print("The value entered is not a valid floating point number.")
```

```
Enter a number: hello
The value entered is not a valid floating point number.
```

- Catch all unknown errors



```python
try:
    print(x[3])
except:
    print("Something went wrong")
```

```
Something went wrong
```