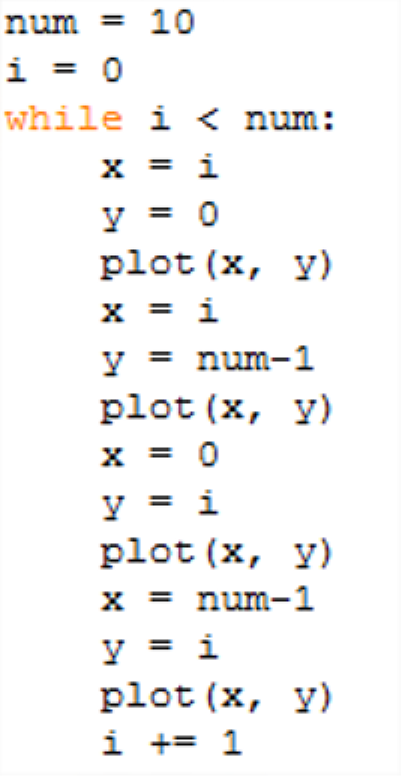
Exercise (1)

Simulate the execution of the Python program below

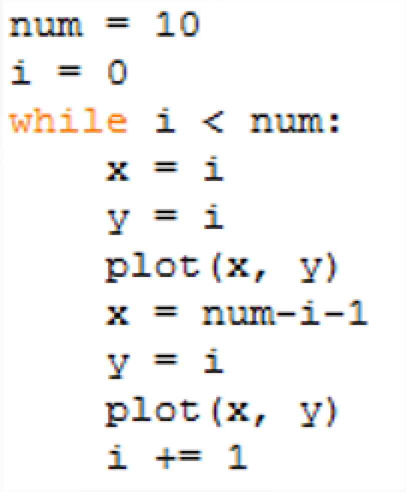


```
num = 10
i = 0
while i < num:
     x = i
     y = 0
     plot(x, y)
     x = i
     y = num-1
     plot(x, y)
     x = 0
     y = i
     plot(x, y)
     x = num-1
     y = i
     plot(x, y)
     i += 1
```

	0	1	2	3	4	5	6	7	8	9
0										
1										
2										
3										
4										
5										
6										
7										
8										
9										

Exercise (2)

Simulate the execution of the Python program below

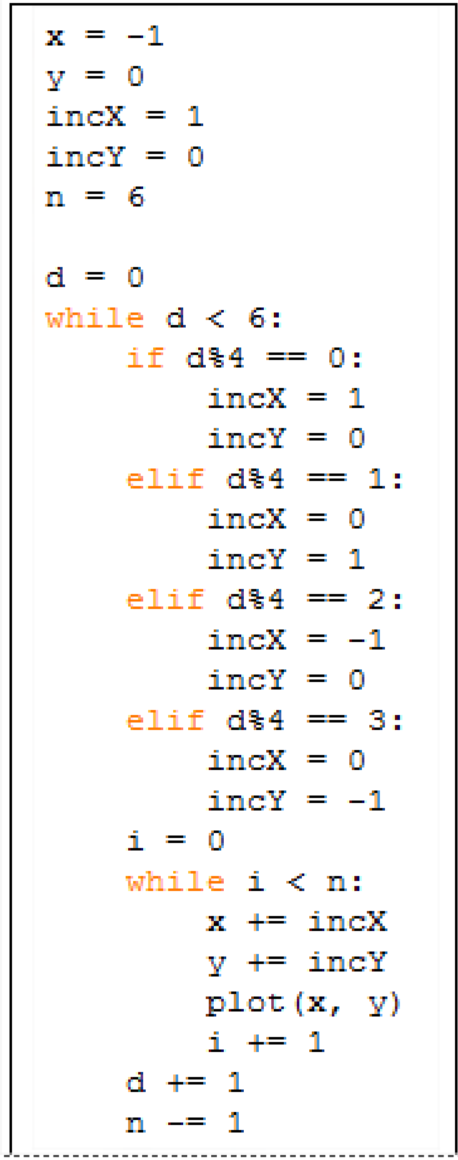


```
num = 10
i = 0
while i < num:
    x = i
    y = i
    plot(x, y)
    x = num-i-1
    y = i
    plot(x, y)
    i += 1
```

	0	1	2	3	4	5	6	7	8	9
0										
1										
2										
3										
4										
5										
6										
7										
8										
9										

Simulate the execution of the Python program below

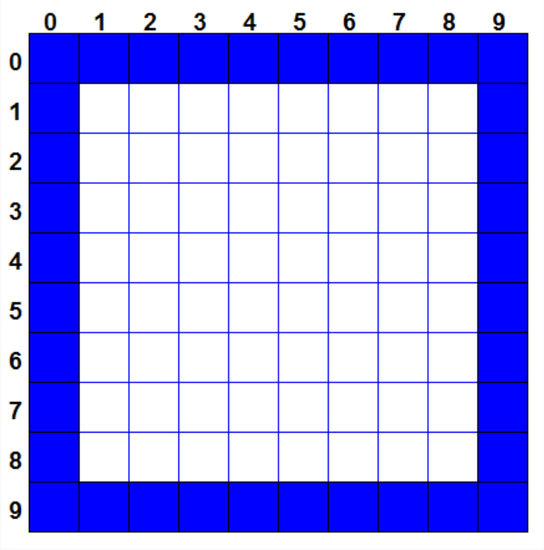
Exercise (3)



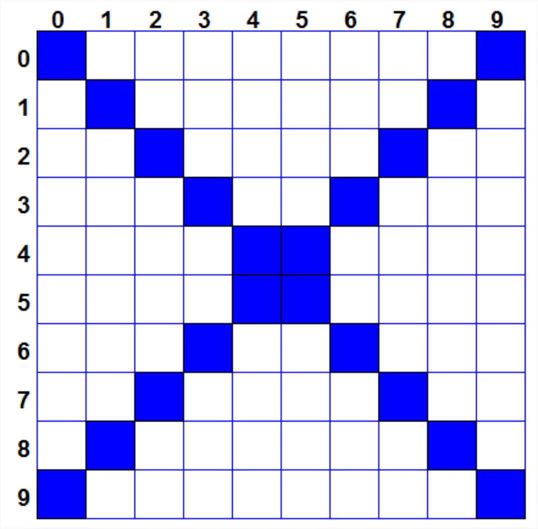
```
x = -1
y = 0
incX = 1
incY = 0
n = 6

d = 0
while d < 6:
    if d%4 == 0:
        incX = 1
        incY = 0
    elif d%4 == 1:
        incX = 0
        incY = 1
    elif d%4 == 2:
        incX = -1
        incY = 0
    elif d%4 == 3:
        incX = 0
        incY = -1
    i = 0
    while i < n:
        x += incX
        y += incY
        plot(x, y)
        i += 1
    d += 1
    n -= 1
```

	0	1	2	3	4	5	6	7	8	9
0										
1										
2										
3										
4										
5										
6										
7										
8										
9										

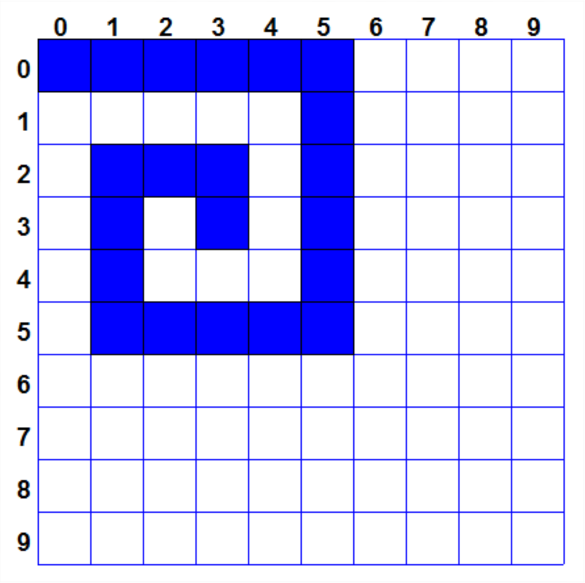
Solutions



	0	1	2	3	4	5	6	7	8	9
0	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
1	🟦									🟦
2	🟦									🟦
3	🟦									🟦
4	🟦									🟦
5	🟦									🟦
6	🟦									🟦
7	🟦									🟦
8	🟦									🟦
9	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦



	0	1	2	3	4	5	6	7	8	9
0	🟦									🟦
1		🟦							🟦	
2			🟦					🟦		
3				🟦			🟦			
4					🟦	🟦				
5					🟦	🟦				
6				🟦			🟦			
7			🟦					🟦		
8		🟦							🟦	
9	🟦									🟦

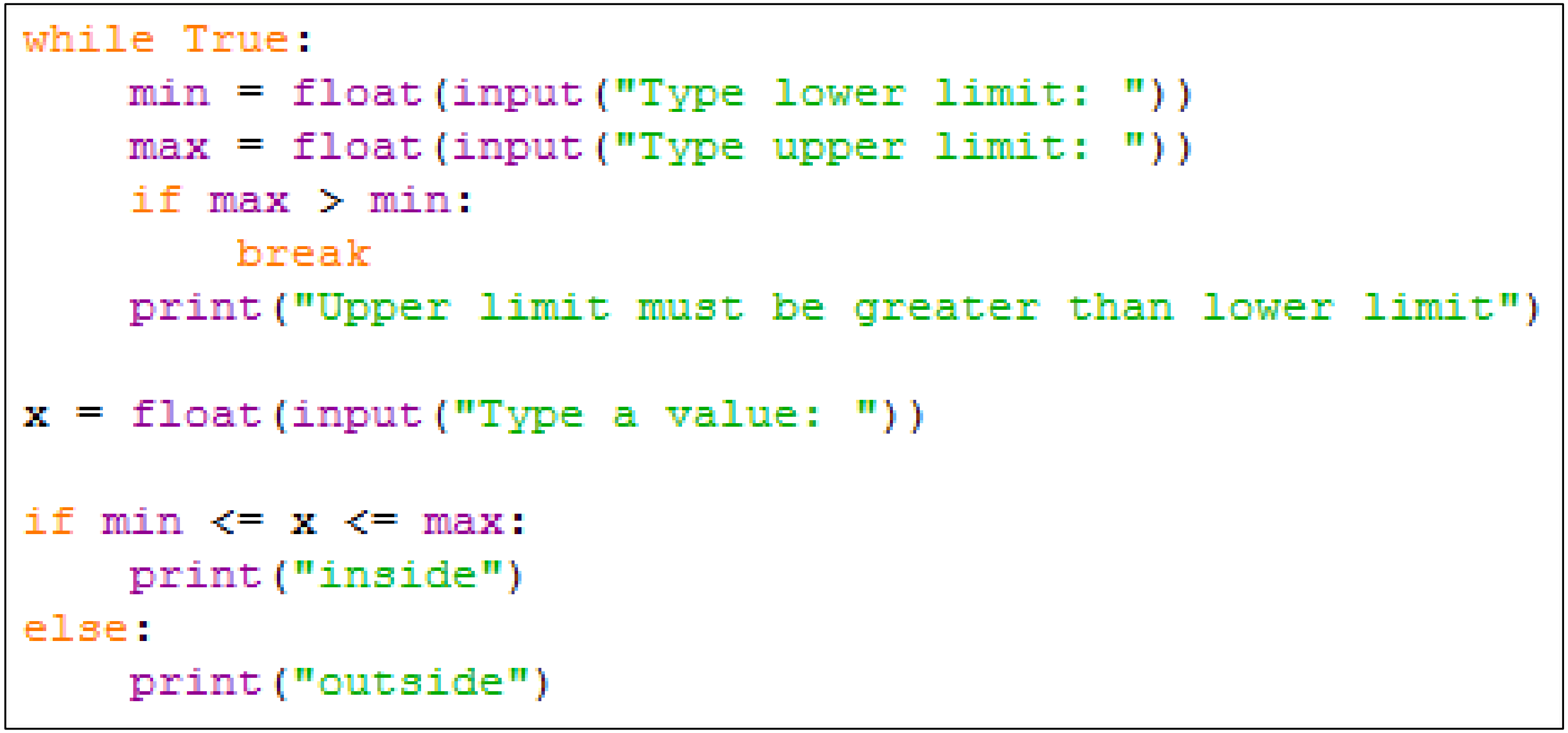


	0	1	2	3	4	5	6	7	8	9
0	🟦	🟦	🟦	🟦	🟦	🟦				
1						🟦				
2		🟦	🟦	🟦		🟦				
3		🟦		🟦		🟦				
4		🟦				🟦				
5		🟦	🟦	🟦	🟦	🟦				
6										
7										
8										
9										

Exercise - check range

- Write a program that reads 2 values min and max, defining a range [min,max].
- The program must ensure that the upper limit is greater than the lower limit of the range.
- Then the program must read another value x and indicate whether x lies inside or outside the given range.

Solution

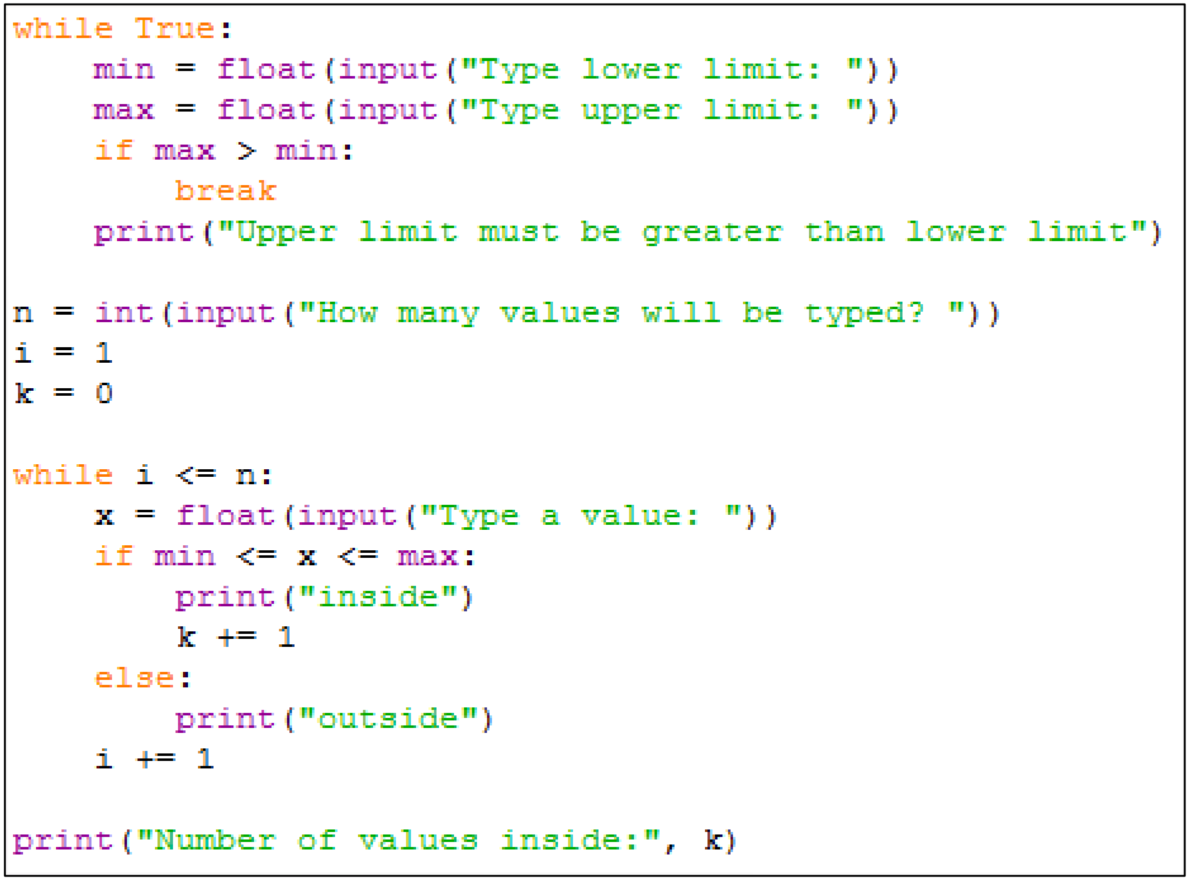


```
while True:
    min = float(input("Type lower limit: "))
    max = float(input("Type upper limit: "))
    if max > min:
        break
    print("Upper limit must be greater than lower limit")

x = float(input("Type a value: "))

if min <= x <= max:
    print("inside")
else:
    print("outside")
```

Exercise - check range (2)

- Extend the program in order to read one range and several other values, indicating whether the values lie inside or outside the given range.
- In order to read several values, use the approach #1 given in the lectures (ask the user how many values will be typed).

Solution

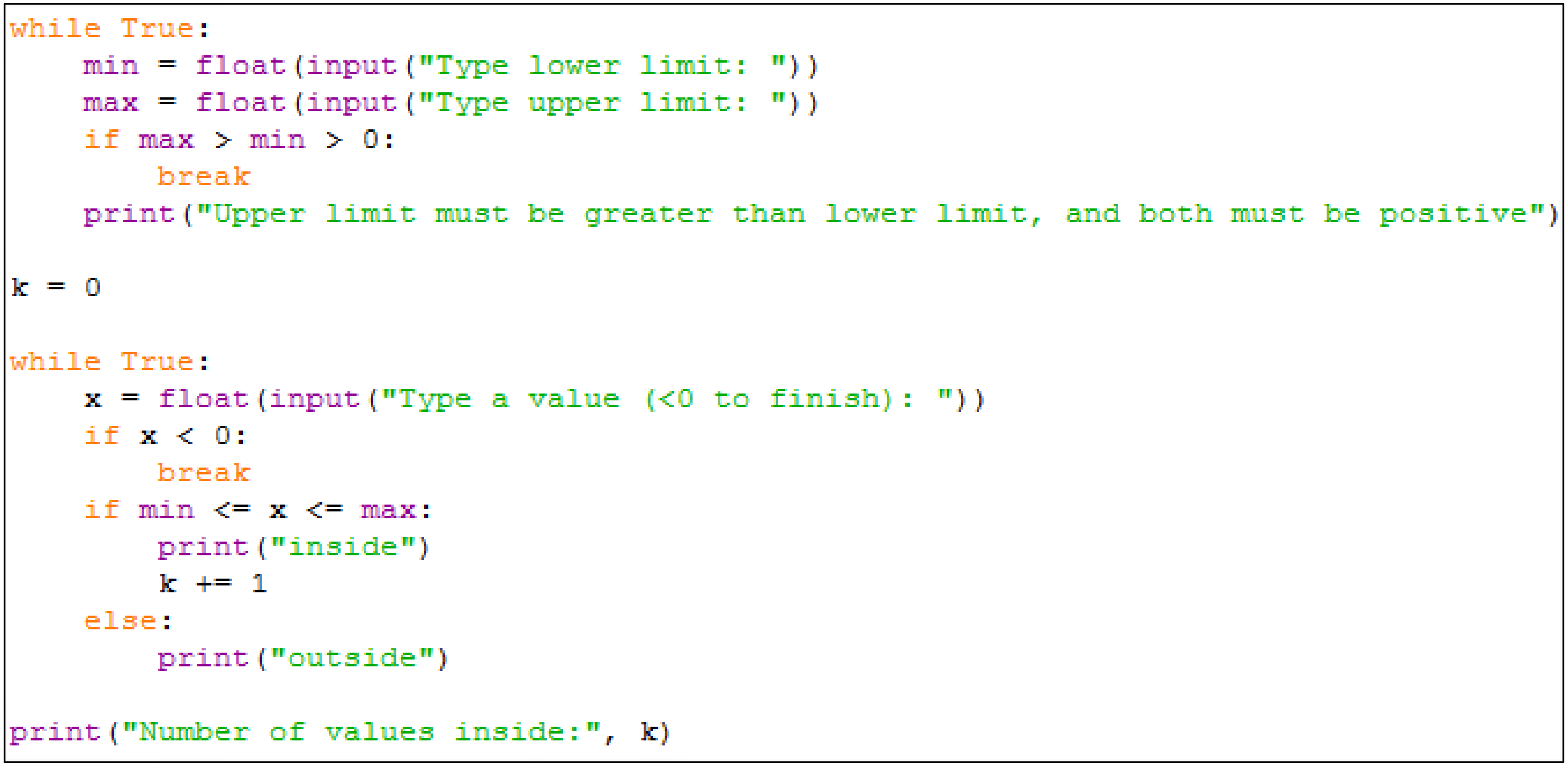


```
while True:
    min = float(input("Type lower limit: "))
    max = float(input("Type upper limit: "))
    if max > min:
        break
    print("Upper limit must be greater than lower limit")

n = int(input("How many values will be typed? "))
i = 1
k = 0

while i <= n:
    x = float(input("Type a value: "))
    if min <= x <= max:
        print("inside")
        k += 1
    else:
        print("outside")
    i += 1

print("Number of values inside:", k)
```

Exercise - check range (3)

- Modify the program in order to ensure also that the values typed by the user for the lower and upper limits of the range will be positive.
- Then read several other values, indicating whether they lie inside or outside the given range. Now use approach #2 given in the lectures (a special value denoting end of input). In order to finish the input, use a negative number.

Solution



```
while True:
    min = float(input("Type lower limit: "))
    max = float(input("Type upper limit: "))
    if max > min > 0:
        break
    print("Upper limit must be greater than lower limit, and both must be positive")

k = 0

while True:
    x = float(input("Type a value (<0 to finish): "))
    if x < 0:
        break
    if min <= x <= max:
        print("inside")
        k += 1
    else:
        print("outside")

print("Number of values inside:", k)
```

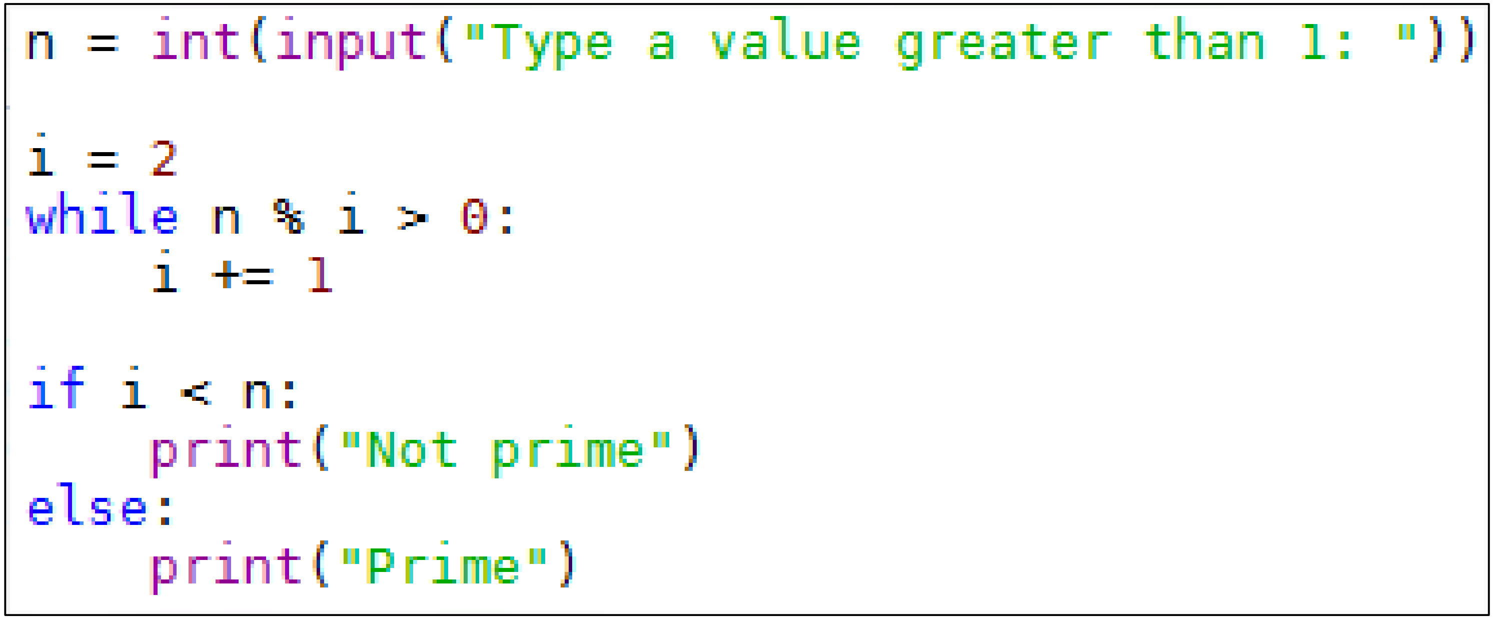
Problem: decide if a number is prime

- An integer number greater than 1 is **prime** if it has no positive divisors other than 1 and itself.
- Problem: given an integer number, decide whether it is a prime number or not.
- Exercise:
  - Write an algorithm to solve the problem.
  - Translate it to a complete Python program.

Algorithm

```
read n
i ← 2
while n is not divisible by i
  increases i by 1 unit
if i < n then
  print "Not prime"
else
  print "Prime"
```

Solution in Python



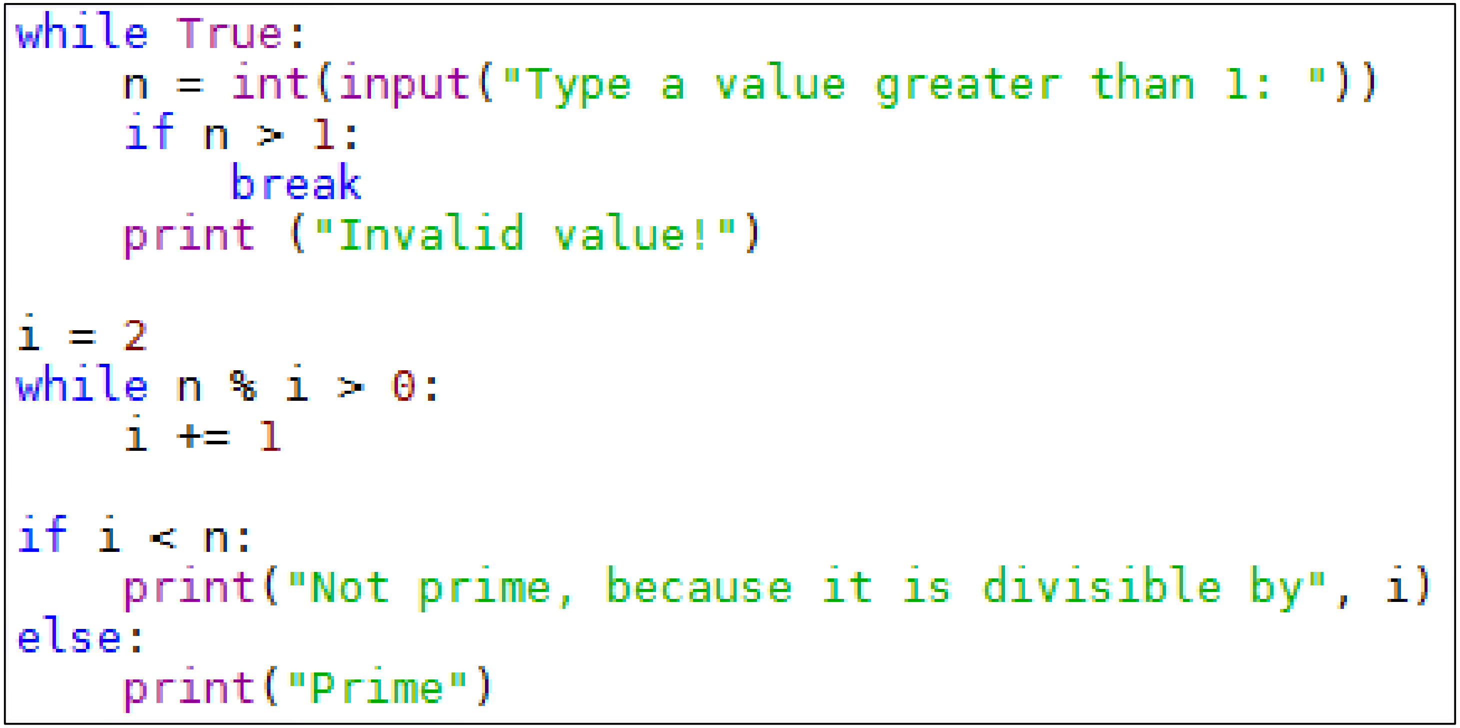
```python
n = int(input("Type a value greater than 1: "))

i = 2
while n % i > 0:
    i += 1

if i < n:
    print("Not prime")
else:
    print("Prime")
```

(program **prime.py**)

INF 100 – Introdução à Programação

Improving the solution

Ensure that value typed is greater than 1:

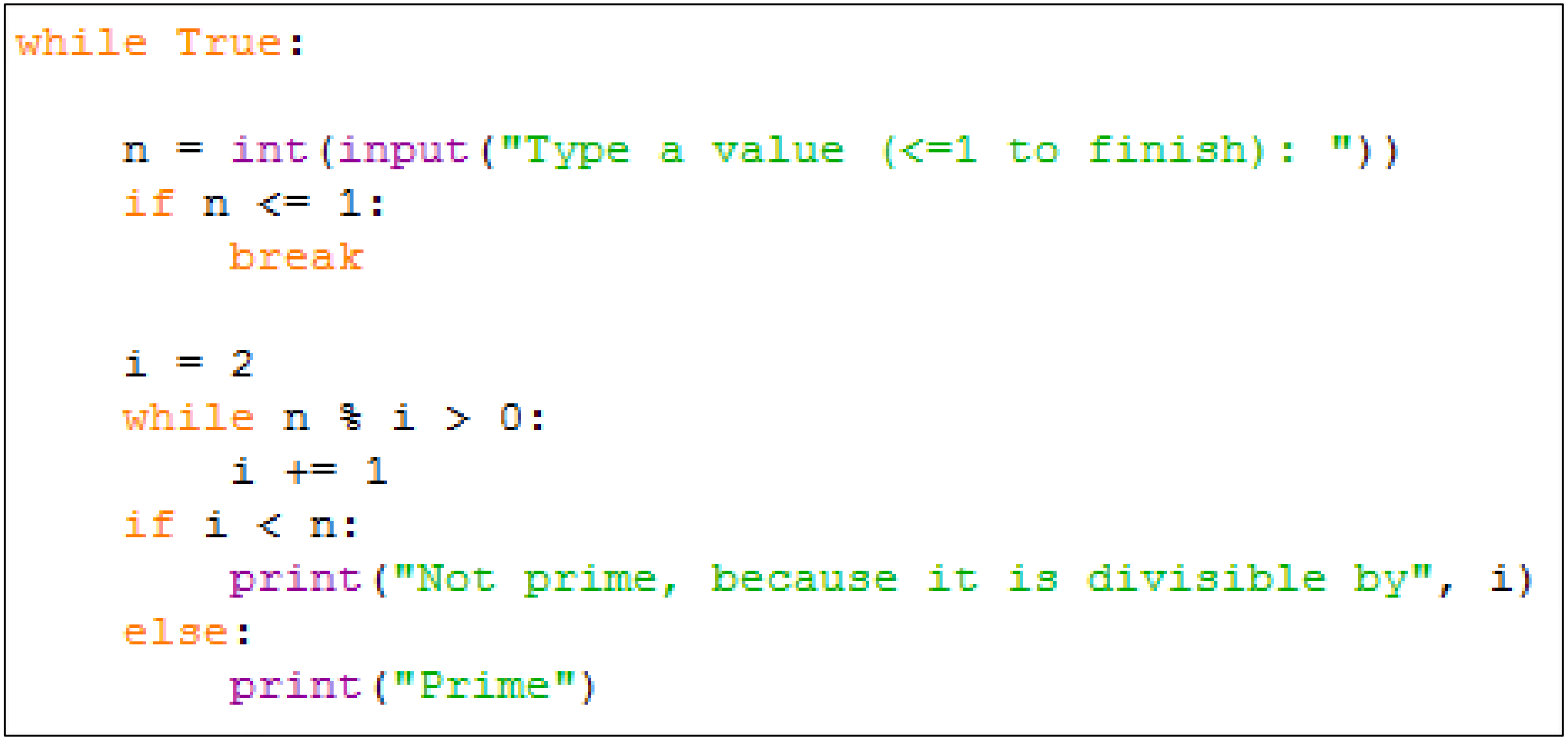


```
while True:
    n = int(input("Type a value greater than 1: "))
    if n > 1:
        break
    print ("Invalid value!")

i = 2
while n % i > 0:
    i += 1

if i < n:
    print("Not prime, because it is divisible by", i)
else:
    print("Prime")
```

(program **prime2.py**)

Extending the program

Check several values, until user types value not greater than 1:



```
while True:

    n = int(input("Type a value (<=1 to finish): "))
    if n <= 1:
        break

    i = 2
    while n % i > 0:
        i += 1
    if i < n:
        print("Not prime, because it is divisible by", i)
    else:
        print("Prime")
```

(program **prime3.py**)

Calculating square root

Write a program that reads a value x and calculates its square root using the method developed by Heron of Alexandria:

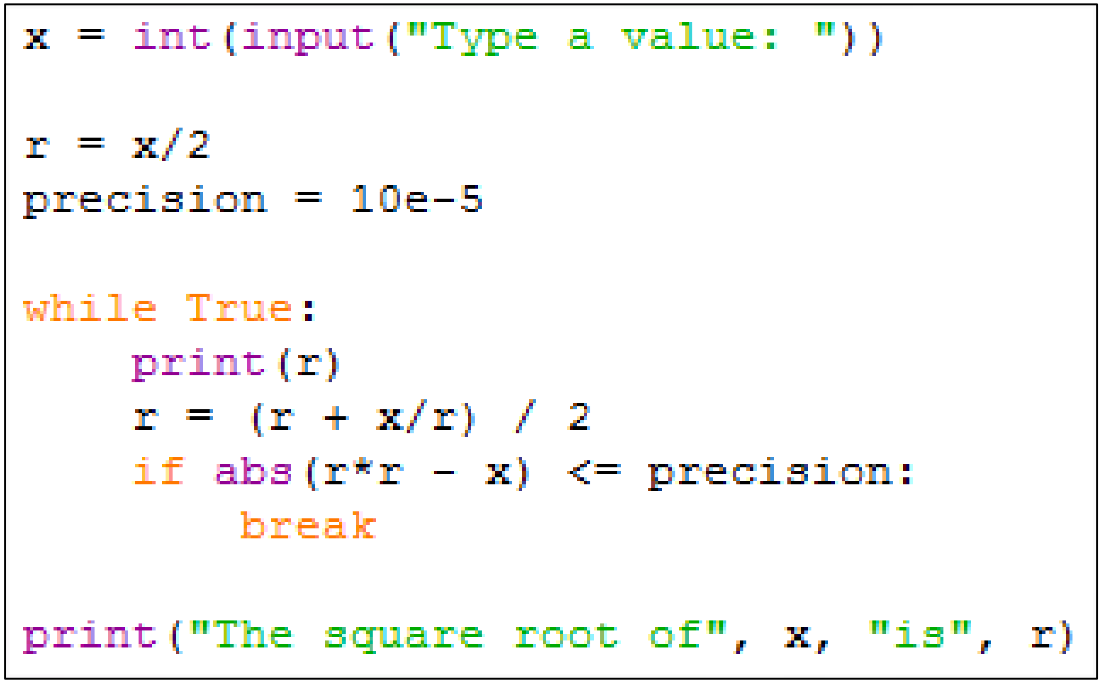
1. Guess a initial value r for the square root
   (for instance,, $r = x/2$).

2. Make $r = (r + x/r) / 2$.

3. If $|r^2 - x| > \varepsilon$, return to step 2.
   Obs.: choose ε as the precision you want for the square root, for instance, 10^{-5}.

4. Print r, the square root calculated.

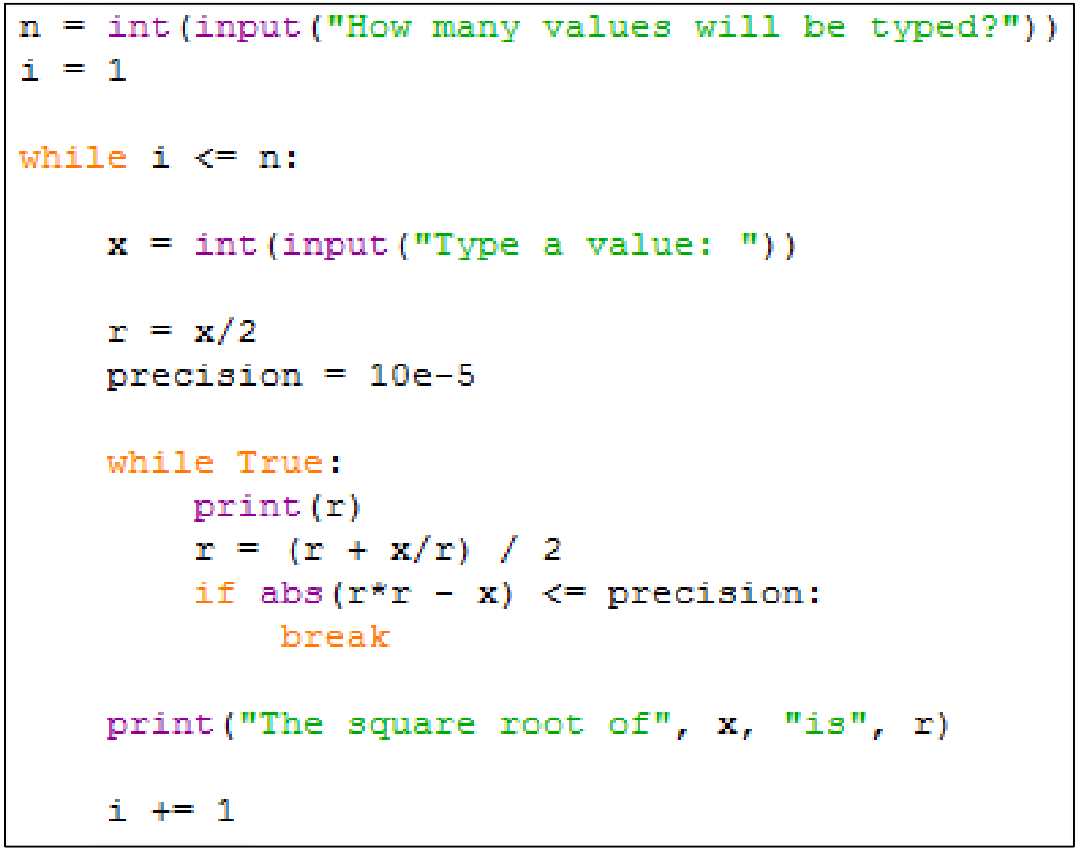
Solution



```
x = int(input("Type a value: "))

r = x/2
precision = 10e-5

while True:
    print(r)
    r = (r + x/r) / 2
    if abs(r*r - x) <= precision:
        break

print("The square root of", x, "is", r)
```

Extending the program

Calculate square root of several values:



```
n = int(input("How many values will be typed?"))
i = 1

while i <= n:

    x = int(input("Type a value: "))

    r = x/2
    precision = 10e-5

    while True:
        print(r)
        r = (r + x/r) / 2
        if abs(r*r - x) <= precision:
            break

    print("The square root of", x, "is", r)

    i += 1
```

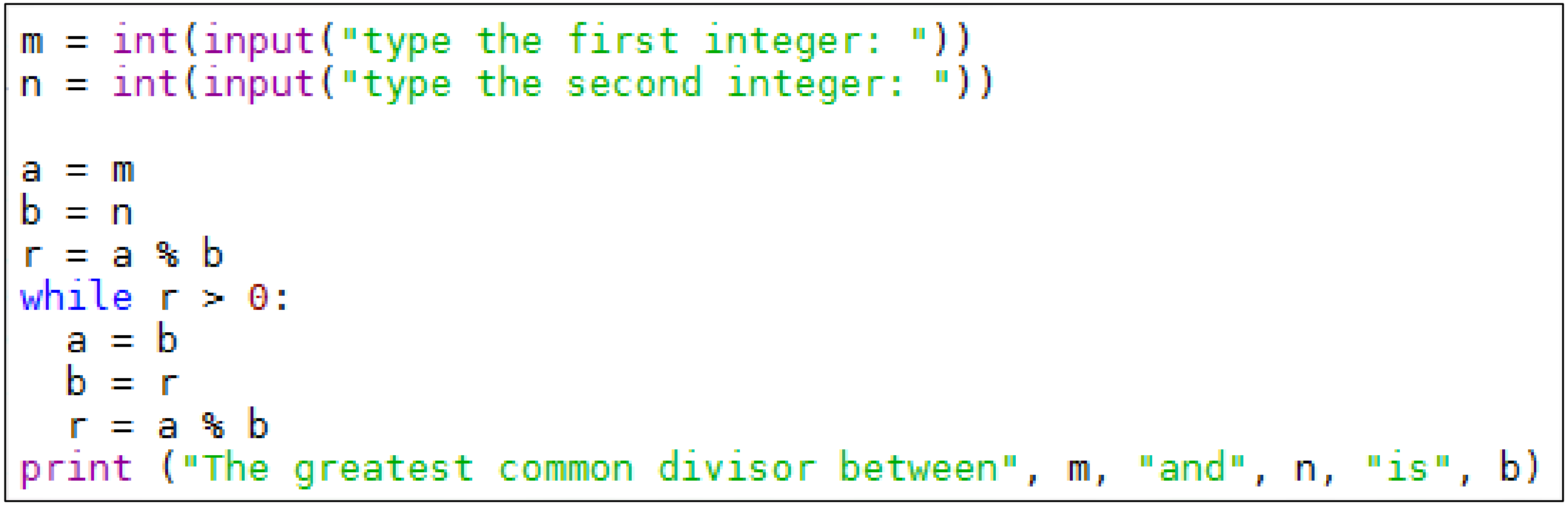
GCD: Greatest Common Divisor

- The GCD between 2 integer numbers is the largest positive integer that divides the numbers.
- The GCD may be found by the Euclidean algorithm.
- An example: to compute gcd(48,18), divide 48 by 18 to get a quotient of 2 and a remainder of 12.
  Then divide 18 by 12 to get a quotient of 1 and a remainder of 6.
  Then divide 12 by 6 to get a remainder of 0, which means that 6 is the gcd.

Algorithm

```
read m and n
a ← m
b ← n
r ← remainder of the division of a by b
while r is greater than 0
  a ← b
  b ← r
  r ← remainder of the division of a by b
print "The greatest common divisor between"
    m "and" n "is" b
```

Solution in Python



```python
m = int(input("type the first integer: "))
n = int(input("type the second integer: "))

a = m
b = n
r = a % b
while r > 0:
  a = b
  b = r
  r = a % b
print ("The greatest common divisor between", m, "and", n, "is", b)
```