

# Conditional and Iterative Statements

Prof. Harald Gall,  
University of Zurich, Institute of Informatics

# Outline

- comparison operators
- conditional statements
- iterative statements (loops)
  - while
  - for
- example exercises

# Comparison Operators

Operation	Result
<code>x == y</code>	returns True if x and y are equal, False otherwise
<code>x != y</code>	returns True if x and y are <b>not</b> equal, False otherwise
<code>x &gt; y</code>	returns True if x is greater than y, False otherwise
<code>x &lt; y</code>	returns True if x is less than y, False otherwise
<code>x &gt;= y</code>	returns True if x is greater or equal than y, False otherwise
<code>x &lt;= y</code>	returns True if x is less or equal than y, False otherwise

# Comparison Operators - Examples

```
>>> 5 == 5
```

```
True
```

```
>>> 5 = 5 # Error!!!
```

```
>>> a = 5
```

```
>>> 5 > 7
```

```
False
```

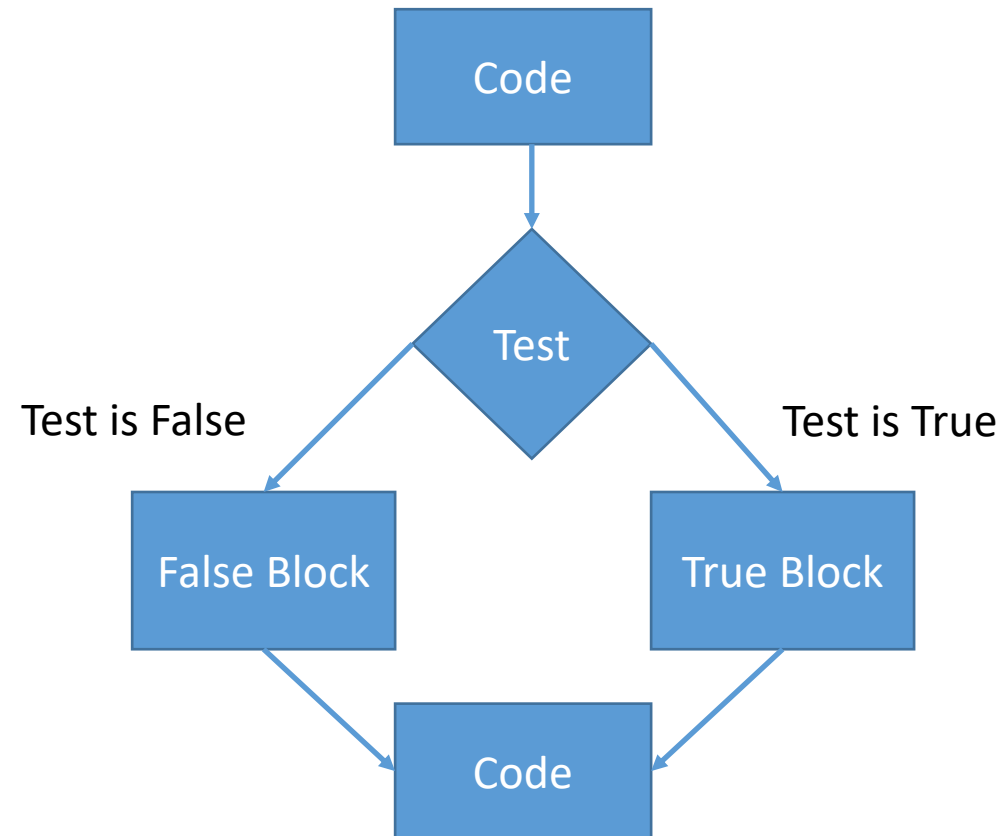
```
>>> 5 <= 10
```

```
True
```

# Boolean Expressions - Examples

```
>>> a = 5
>>> b = 6
>>> a == b and a < 10
False
>>> a == b or a < 10
True
```

# Conditional Execution



# Conditional Execution

```
if condition_1:  
    block_1  
elif condition_2:  
    block_2  
elif condition_3:  
    block_3  
else:  
    block_4
```

# Conditional Execution - Examples

```
x = int(input('x = '))  
if x < 0:  
    print('x is negative')
```



# Conditional Execution - Examples

```
x = int(input('x = '))
if x < 0:
    print('x is negative')
else:
    print('x is positive or zero')
```

# Conditional Execution - Examples

```
x = int(input('x = '))
if x < 0:
    print('x is negative')
elif x > 0:
    print('x is positive')
else:
    print('x is zero')
```

# Conditional Execution - Examples

```
x = int(input('x = '))
if x < 0:
    print('x is negative')
# if and else conditions
x = int(input('x = '))
if x == 0:
    print('x is zero')
else:
    print('x is nonzero')
```

# Indentation

- Python programs are structured through indentation
- not correctly indenting each block of code will result in a syntax error
- all statements within the same distance to the right belong to the same block of code
- makes code easier to understand and read

# Indentation Explained

**Block 1**

```
x = int(input("A number: "))
if x < 0:
    print("You have introduced a negative number")
    x = int(input("Another number: "))
    print("The new number is %d" % x)
    if x < 0:
        print("This is another negative number")
        print("Please introduce a positive one this time")
        x = int(input("A positive number: "))
        print("The new number: %d" % x)
```

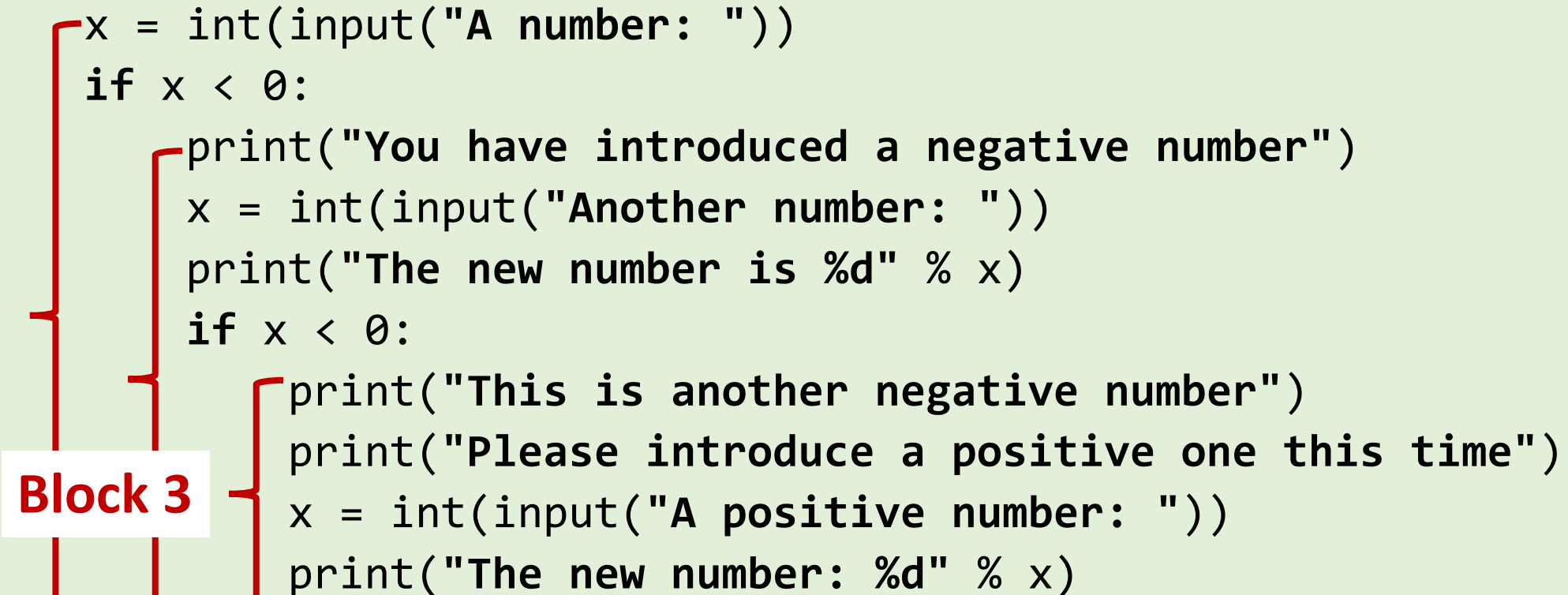
# Indentation Explained

## Block 2

```
x = int(input("A number: "))
if x < 0:
    print("You have introduced a negative number")
    x = int(input("Another number: "))
    print("The new number is %d" % x)
    if x < 0:
        print("This is another negative number")
        print("Please introduce a positive one this time")
        x = int(input("A positive number: "))
        print("The new number: %d" % x)
```

# Indentation Explained

```
x = int(input("A number: "))
if x < 0:
    print("You have introduced a negative number")
    x = int(input("Another number: "))
    print("The new number is %d" % x)
    if x < 0:
        print("This is another negative number")
        print("Please introduce a positive one this time")
        x = int(input("A positive number: "))
        print("The new number: %d" % x)
```



**Block 3**

# Conditional Expressions

```
a, b = 1, 10
if a <= b:
    minvalue = a
else:
    minvalue = b
# Alternative
minvalue = a if a <= b else b
```

- The condition of the expression is evaluated first, then the expression to the left is evaluated if the expression is True otherwise the expression after else.



# False Value Testing

The following values are considered False by Python, everything else is considered True:

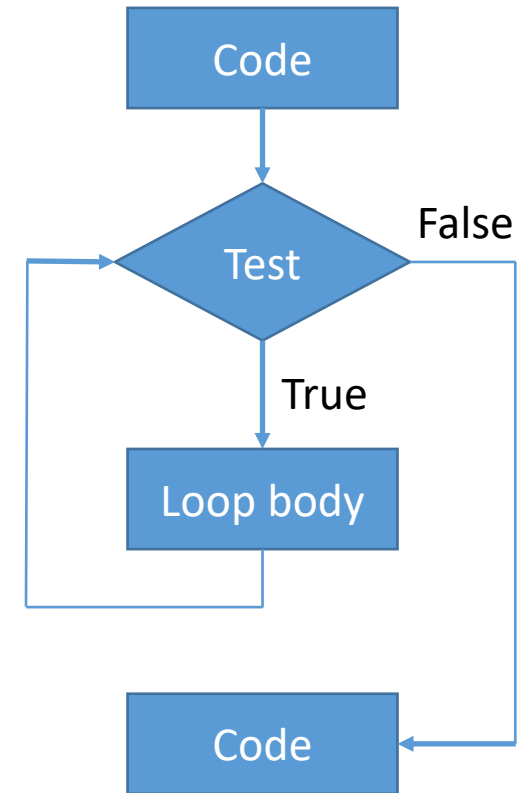
- False (a boolean expression)
- None
- 0, 0L, 0.0, 0j
- empty sequences: "", (), [] (will be covered later)
- empty mapping: {} (later...)
- instances of user-defined classes with methods `__nonzero__()` and `__len__()` that return 0 or value False (later...)

# False Value Testing - Examples

```
>>> if None: # can replace with "", 0, False, etc.
...     print("IF branch")
... else:
...     print("ELSE branch")
...
ELSE branch
>>> if not None:
...     print("IF branch")
...
IF branch
```

# Iterative Execution - Loops

- the **loop body** is executed as long as the **test** evaluates to True
- once it evaluates to False the rest of the code is executed
- if **test** never evaluates to True the loop body will never be executed
- if **test** always evaluates to True we will have an **infinite loop** (never terminates)



# While loop

```
while condition:  
    statement_1  
    statement_2  
    statement_3  
    ...
```

# While Loop - Example

```
# print the first 10 numbers  
nr = 1  
while nr <= 10:  
    print(nr)  
    nr += 1 # the same as nr = nr + 1  
print('Finished')
```

# Infinite Loops

```
nr = 1
while True:
    print(nr)
    nr += 1
# it runs forever
print('Finished')
```

# Break Statement

```
nr = 1
while True:
    print(nr)
    nr += 1
    if nr > 10:
        break
# it finishes after printing 10
print('Finished')
```

# Continue Statement

```
nr = 0
# print all even numbers from 0 to 100
while True:
    nr += 1
    if nr % 2 == 1:
        continue
    print(nr)
    if nr >= 100:
        break
```



# Nested Loops

```
a = 1
# print number 1 one time, number 2 two times, etc
# number 3 three times, etc. until 10
while a <= 10:
    b = 0
    while b < a:
        print(a)
        b += 1
    a += 1
```

# While Loop – Revisited

- repeating a sequence of code and **incrementing** a variable (adding 1) at each step is a common pattern in programming

```
a = 1
while a < 10:
    if a % 2 == 0:
        print("%d is even" % a)
    a += 1
```

# For Loop

- this pattern can be replaced with a **for** loop
- `range(1, 10)` generates the sequence 1, 2, 3...9 (for a full explanation check <https://docs.python.org/3/library/stdtypes.html#typeseq-range>)
- the **for** loop iterates through each element in the sequence and assigns to **a** the value of the current element at each iteration

```
for a in range(1, 10):  
    if a % 2 == 0:  
        print("%d is even" % a)
```

# For and While

```
i = 1
while i < 11:
    print(i)
    i += 1
# equivalent to
for i in range(1, 11):
    print(i)
```

# More about Strings

- String are sequences of characters: we can iterate over the characters using a **for** loop

```
a_string = input("Please enter a string: ")
string_length = 0
for ch in a_string:
    string_length += 1
print("The length of your string is: %d" % string_length)
```

# More about Strings

- an easier way to obtain the length of a string is using the built-in **len** function

```
a_string = input("Please enter a string: ")  
print("The length is: %d" % len(a_string))
```

# More about Strings

- strings are **immutable**: the content of a string cannot be modified once it was defined

```
>>> hello = "hello"
>>> hello[0] = "H"
TypeError: 'str' object does not support item assignment
>>> hello = "H" + hello[1:]
>>> hello
'Hello'
```

# Exercise 1

Compute the sum of the first  $n$  numbers, where  $n$  is a value provided by the user.



# Ask the user for the value n

```
n = int(input('n = '))
```

# Initialize the total\_sum and a counter variable

```
n = int(input('n = '))  
total_sum = 0  
count = 0
```

# Iterate over the first n values in a while loop

```
n = int(input('n = '))
total_sum = 0
count = 0
while count < n:
    count += 1
    print(count)
```

# Add value to the total\_sum

```
n = int(input('n = '))
total_sum = 0
count = 0
while count < n:
    count += 1
    total_sum += count
```

# Print the total\_sum

```
n = int(input('n = '))
total_sum = 0
count = 0
while count < n:
    count += 1
    total_sum += count
print('The sum of the first %d values is %d.' % (n,
total_sum))
```

## Print result – with For

```
n = int(input('n = '))
total_sum = 0
for count in range(1, n + 1):
    total_sum += count
print('The sum of the first %d values is %d.' % (n,
total_sum))
```

## Exercise 2

Ask the user to enter a phone number and validate it. The number should satisfy the following conditions:

- it should have length equal to 10
- only contain characters: 0, 1, 2
- start with 011

If the number does not satisfy any of the conditions than print a corresponding error message, otherwise print "Valid" at the end of the program

# Ask for the number

```
phone_number = input("Please enter a phone number: ")
```



# Check that it satisfies the length condition

```
phone_number = input("Please enter a phone number: ")  
if len(phone_number) != 10:  
    print("Your phone number should be 10 digits long!")
```

## Otherwise...

```
phone_number = input("Please enter a phone number: ")  
if len(phone_number) != 10:  
    print("Your phone number should be 10 digits long!")  
else:
```

# Check for the valid characters (only contains 0, 1, 2)

```
phone_number = input("Please enter a phone number: ")  
if len(phone_number) != 10:  
    print("Your phone number should be 10 digits long!")  
else:
```

# We need to iterate over each character

```
phone_number = input("Please enter a phone number: ")
if len(phone_number) != 10:
    print("Your phone number should be 10 digits long!")
else:
    for ch in phone_number:
```

## And check if this is invalid

```
phone_number = input("Please enter a phone number: ")
if len(phone_number) != 10:
    print("Your phone number should be 10 digits long!")
else:
    for ch in phone_number:
        # check if ch is different from 0, 1 and 2
        if ch != "0" and ch != "1" and ch != "2":
```

## If we found an invalid character, we can exit the for loop

```
phone_number = input("Please enter a phone number: ")
if len(phone_number) != 10:
    print("Your phone number should be 10 digits long!")
else:
    for ch in phone_number:
        # check if ch is different from 0, 1 and 2
        if ch != "0" and ch != "1" and ch != "2":
            break
```

# But we also need to remember that we found an invalid character

```
phone_number = input("Please enter a phone number: ")
if len(phone_number) != 10:
    print("Your phone number should be 10 digits long!")
else:
    for ch in phone_number:
        # check if ch is different from 0, 1 and 2
        if ch != "0" and ch != "1" and ch != "2":
            break
```

# For that we use a flag variable invalid\_character

```
phone_number = input("Please enter a phone number: ")
if len(phone_number) != 10:
    print("Your phone number should be 10 digits long!")
else:
    invalid_character = False
    for ch in phone_number:
        # check if ch is different from 0, 1 and 2
        if ch != "0" and ch != "1" and ch != "2":
            break
```



## And set it to True inside the if statement

```
phone_number = input("Please enter a phone number: ")
if len(phone_number) != 10:
    print("Your phone number should be 10 digits long!")
else:
    invalid_character = False
    for ch in phone_number:
        # check if ch is different from 0, 1 and 2
        if ch != "0" and ch != "1" and ch != "2":
            invalid_character = True
            break
```

It is important to set it before the break,  
otherwise it is not executed

```
phone_number = input("Please enter a phone number: ")
if len(phone_number) != 10:
    print("Your phone number should be 10 digits long!")
else:
    invalid_character = False
    for ch in phone_number:
        # check if ch is different from 0, 1 and 2
        if ch != "0" and ch != "1" and ch != "2":
            break
    invalid_character = True
```

## If we found an invalid character

```
phone_number = input("Please enter a phone number: ")
if len(phone_number) != 10:
    print("Your phone number should be 10 digits long!")
else:
    invalid_character = False
    for ch in phone_number:
        # check if ch is different from 0, 1 and 2
        if ch != "0" and ch != "1" and ch != "2":
            invalid_character = True
            break
    if invalid_character:
```

# Print an error message

```
phone_number = input("Please enter a phone number: ")
if len(phone_number) != 10:
    print("Your phone number should be 10 digits long!")
else:
    invalid_character = False
    for ch in phone_number:
        # check if ch is different from 0, 1 and 2
        if ch != "0" and ch != "1" and ch != "2":
            invalid_character = True
            break
    if invalid_character:
        print("Found an invalid character")
```

## Otherwise...

```
phone_number = input("Please enter a phone number: ")
if len(phone_number) != 10:
    print("Your phone number should be 10 digits long!")
else:
    invalid_character = False
    for ch in phone_number:
        # check if ch is different from 0, 1 and 2
        if ch != "0" and ch != "1" and ch != "2":
            invalid_character = True
            break
    if invalid_character:
        print("Found an invalid character")
    else:
```

## Check for the last condition (it starts with 011)

```
...
for ch in phone_number:
    # check if ch is different from 0, 1 and 2
    if ch != "0" and ch != "1" and ch != "2":
        invalid_character = True
        break
if invalid_character:
    print("Found an invalid character")
else:
    if phone_number[0] == "0" and phone_number[1] == "1"
       and phone_number[2] == "1":
        print("Valid")
```

## Otherwise print an error message

```
...
    if ch != "0" and ch != "1" and ch != "2":
        invalid_character = True
        break
if invalid_character:
    print("Found an invalid character")
else:
    if phone_number[0] == "0" \
        and phone_number[1] == "1" \
        and phone_number[2] == "1":
        print("Valid")
    else:
        print("Does not start with 011")
```

# Full solution

```
phone_number = input("Please enter a phone number: ")
if len(phone_number) != 10:
    print("Your phone number should be 10 digits long!")
else:
    invalid_character = False
    for ch in phone_number:
        # check if ch is different from 0, 1 and 2
        if ch != "0" and ch != "1" and ch != "2":
            invalid_character = True
            break
    if invalid_character:
        print("Found an invalid character")
    else:
        if phone_number[0] == "0" and phone_number[1] == "1" and phone_number[2]
        == "1":
            print("Valid")
        else:
            print("Does not start with 011")
```



## Exercise 3

Assign a variable to a random value and ask the user to guess the value. If the value is less than the chosen value print 'less', if it is greater print 'greater' otherwise print 'correct' and terminate the program.

<https://docs.python.org/2/library/random.html>

# First import the random module

```
import random
```

# Generate a random value

```
import random  
# Returns a random integer N such that  $0 \leq N \leq 100$   
rand_val = random.randint(0, 100)  
print(rand_val)
```

# Ask the user to guess the value

```
import random
rand_val = random.randint(0, 100)
print(rand_val)
while True:
    guess_val = int(input('guess: '))
```

# Check if the user guessed correctly

```
import random
rand_val = random.randint(0, 100)
print(rand_val)
while True:
    guess_val = int(input('guess: '))
    if rand_val == guess_val:
        break
```

# Otherwise show a hint

```
import random
rand_val = random.randint(0, 100)
print(rand_val)
while True:
    guess_val = int(input('guess: '))
    if rand_val < guess_val:
        print('less')
    elif rand_val > guess_val:
        print('larger')
    else:
        break
```

# Full solution

```
import random
rand_val = random.randint(0, 100)
print(rand_val)
while True:
    guess_val = int(input('guess: '))
    if rand_val < guess_val:
        print('less')
    elif rand_val > guess_val:
        print('larger')
    else:
        print('correct!')
        break
```

## Exercise 4

Ask the user to provide a list of grades that ends with -1 and print out the average of the grades. Afterwards ask the user if he wants to provide another list of grades, if he types 'Yes' then repeat the first action, otherwise terminate the program.



# We need a while loop to ask for the grades

```
while True:  
    # input returns a string, but the grade  
    # should be a float  
    grade = float(input('grade = '))  
    print(grade)
```

# If the input is -1, exit the while loop

```
while True:  
    grade = float(input('grade = '))  
    if grade == -1:  
        break
```

# We need to store the sum of the grades so far

```
grades_sum = 0
while True:
    grade = float(input('grade = '))
    if grade == -1:
        break
    grades_sum += grade
```

## ... and how many grades we have

```
grades_sum = 0
num_grades = 0
while True:
    grade = float(input('grade = '))
    if grade == -1:
        break
    grades_sum += grade
    num_grades += 1
```

# Print the average before exiting the loop

```
grades_sum = 0
num_grades = 0
while True:
    grade = float(input('grade = '))
    if grade == -1:
        avg = grades_sum / num_grades
        print('Average grade = %.2f' % avg)
        break
    grades_sum += grade
    num_grades += 1
```

## Now we need to repeat the whole loop

```
while True:
    grades_sum = 0
    num_grades = 0
    while True:
        grade = float(input('grade = '))
        if grade == -1:
            avg = grades_sum / num_grades
            print('Average grade = %.2f' % avg)
            break
        grades_sum += grade
        num_grades += 1
```

... ask the user if he/she wants to continue

```
while True:
    grades_sum = 0
    num_grades = 0
    while True:
        grade = float(input('grade = '))
        if grade == -1:
            avg = grades_sum / num_grades
            print('Average grade = %.2f' % avg)
            break
        grades_sum += grade
        num_grades += 1
    answer = input('continue? ')
    if answer != "Yes":
```

# If not exit the loop

```
while True:
    grades_sum = 0
    num_grades = 0
    while True:
        grade = float(input('grade = '))
        if grade == -1:
            avg = grades_sum / num_grades
            print('Average grade = %.2f' % avg)
            break
        grades_sum += grade
        num_grades += 1
    answer = input('continue? ')
    if answer != "Yes":
        break
```