**Python Conditions and If statements**

Python supports the usual logical conditions from mathematics:

Equals: a == b

Not Equals: a != b

Less than: a < b

Less than or equal to: a <= b

Greater than: a > b

Greater than or equal to: a >= b

These conditions can be used in several ways, most commonly in “if statements” and loops.

An “if statement” is written by using the if keyword.

Example

If statement:

a = 33

b = 200

if b > a:

Print(“b is greater than a”)

b is greater than a

In this example we use two variables, a and b, which are used as part of the if statement to test whether b is greater than a. As a is 33, and b is 200, we know that 200 is greater than 33, and so we print to screen that “b is greater than a”.

**Indentation**

Python relies on indentation (whitespace at the beginning of a line) to define scope in the code. Other programming languages often use curly-brackets for this purpose.

Example

If statement, without indentation (will raise an error):

A = 33

B = 200

If b > a:

Print(“b is greater than a”) # you will get an error

**Elif**

The elif keyword is Python’s way of saying “if the previous conditions were not true, then try this condition”.

Example

A = 33

B = 33

If b > a:

Print(“b is greater than a”)

Elif a == b:

Print(“a and b are equal”)

a and b are equal

In this example a is equal to b, so the first condition is not true, but the elif condition is true, so we print to screen that “a and b are equal”.

**Else**

The else keyword catches anything which isn’t caught by the preceding conditions.

Example

A = 200

B = 33

If b > a:

Print(“b is greater than a”)

Elif a == b:

Print(“a and b are equal”)

Else:

Print(“a is greater than b”)

a is greater than b

In this example a is greater than b, so the first condition is not true, also the elif condition is not true, so we go to the else condition and print to screen that “a is greater than b”.

You can also have an else without the elif:

Example

A = 200

B = 33

If b > a:

Print(“b is greater than a”)

Else:

Print(“b is not greater than a”)

b is not greater than a

**And**

The and keyword is a logical operator, and is used to combine conditional statements:

Example

Test if a is greater than b, AND if c is greater than a:

A = 200

B = 33

C = 500

If a > b and c > a:

Print(“Both conditions are True”)

Both conditions are True

**Or**

The or keyword is a logical operator, and is used to combine conditional statements:

Example

Test if a is greater than b, OR if a is greater than c:

A = 200

B = 33

C = 500

If a > b or a > c:

Print(“At least one of the conditions is True”)

At least one of the conditions is True

**Not**

The not keyword is a logical operator, and is used to reverse the result of the conditional statement:

Example

Test if a is NOT greater than b:

A = 33

B = 200

If not a > b:

Print(“a is NOT greater than b”)

a is NOT greater than b

**Nested If**

You can have if statements inside if statements, this is called nested if statements.

Example

X = 41

If x > 10:

Print(“Above ten,”)

If x > 20:

Print(“and also above 20!”)

Else:

Print(“but not above 20.”)

Above ten,

And also above 20!

**Python For Loops**

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

This is less like the for keyword in other programming languages, and works more like an iterator method as found in other object-orientated programming languages.

With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc.

Example

Print each fruit in a fruit list:

Fruits = [“apple”, “banana”, “cherry”]

For x in fruits:

Print(x)

apple

Banana

Cherry

The for loop does not require an indexing variable to set beforehand.

**Looping Through a String**

Even strings are iterable objects, they contain a sequence of characters:

Example

Loop through the letters in the word “banana”:

For x in “banana”:

Print(x)

B

A

N

A

N

A

**The break Statement**

With the break statement we can stop the loop before it has looped through all the items:

Example

Exit the loop when x is “banana”:

Fruits = [“apple”, “banana”, “cherry”]

For x in fruits:

Print(x)

If x == “banana”:

Break

apple

Banana

Example

Exit the loop when x is “banana”, but this time the break comes before the print:

Fruits = [“apple”, “banana”, “cherry”]

For x in fruits:

If x == “banana”:

Break

Print(x)

Apple

**The continue Statement**

With the continue statement we can stop the current iteration of the loop, and continue with the next:

Example

Do not print banana:

Fruits = [“apple”, “banana”, “cherry”]

For x in fruits:

If x == “banana”:

Continue

Print(x)

apple,cherry

**The range() Function**

To loop through a set of code a specified number of times, we can use the range() function,

The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

Example

Using the range() function:

For x in range(6):

Print(x)

0

1

2

3

4

5

6

Note that range(6) is not the values of 0 to 6, but the values 0 to 5.

The range() function defaults to 0 as a starting value, however it is possible to specify the starting value by adding a parameter: range(2, 6), which means values from 2 to 6 (but not including 6):

Example

Using the start parameter:

For x in range(2, 6):

Print(x)

2

3

4

5

The range() function defaults to increment the sequence by 1, however it is possible to specify the increment value by adding a third parameter: range(2, 30, 3):

Example

Increment the sequence with 3 (default is 1):

For x in range(2, 30, 3):

Print(x)

2

5

8

11

14

17

20

23

26

29

**Else in For Loop**

The else keyword in a for loop specifies a block of code to be executed when the loop is finished:

Example

Print all numbers from 0 to 5, and print a message when the loop has ended:

For x in range(6):

Print(x)

Else:

Print(“Finally finished!”)

Note: The else block will NOT be executed if the loop is stopped by a break statement.

Example

Break the loop when x is 3, and see what happens with the else block:

For x in range(6):

If x == 3: break

Print(x)

Else:

Print(“Finally finished!”)

0

1

2

**Nested Loops**

A nested loop is a loop inside a loop.

The “inner loop” will be executed one time for each iteration of the “outer loop”:

Example

Print each adjective for every fruit:

Adj = [“red”, “big”, “tasty”]

Fruits = [“apple”, “banana”, “cherry”]

For x in adj:

For y in fruits:

Print(x, y)

red apple

Red banana

Red cherry

Big apple

Big banana

Big cherry

Tasty apple

Tasty banana

Tasty cherry

**Print the first 10 natural numbers using for loop.**

# between 0 to 10

# there are 11 numbers

# therefore, we set the value

# of n to 11

N = 11

# since for loop starts with

# the zero indexes we need to skip it and

# start the loop from the first index

For I in range(1,n):

Print(i)

Copy code

Output

1

2

3

4

5

6

7

8

9

10

**Python program to calculate the sum of all numbers from 1 to a given number.**

# if the given number is 10

Given\_number = 10

# set up a variable to store the sum

# with initial value of 0

Sum = 0

# since we want to include the number 10 in the sum

# increment given number by 1 in the for loop

For I in range(1,given\_number+1):

Sum+=i

# print the total sum at the end

Print(sum)

Copy code

Output

55

**Python program to check if the given string is a palindrome.**

# given string

Given\_string = “madam”

# an empty string variable to store

# the given string in reverse

Reverse\_string = “”

# iterate through the given string

# and append each element of the given string

# to the reverse\_string variable

For I in given\_string:

Reverse\_string = I + reverse\_string

# if given\_string matches the reverse\_srting exactly

# the given string is a palindrome

If(given\_string == reverse\_string):

Print(“The string”, given\_string,”is a Palindrome.”)

# else the given string is not a palindrome

Else:

Print(“The string”,given\_string,”is NOT a Palindrome.”)

Copy code

Output

The string madam is a Palindrome String

**Python program to get the Fibonacci series between 0 to 50.**

# given upper bound

Num = 50

# initial values in the series

First\_value,second\_value = 0, 1

# iterate in the given range

# of numbers

For n in range(0, num):

# if no. is less than 1

# move to next number

If(n <= 1):

Next = n

# if number is within range

# execute the below code block

If nextnum:

Break

# print each element that

# satisfies all the above conditions

Print(next)

Copy code

Output

1

2

3

5

8

13

21

34

**Python program to find the factorial of a given number.**

# given number

Given\_number= 5

# since 1 is a factor

# of all number

# set the factorial to 1

Factorial = 1

# iterate till the given number

For I in range(1, given\_number + 1):

Factorial = factorial \* i

Print(“The factorial of “, given\_number, “ is “, factorial)

Copy code

Output

The factorial of 5 is 120

**Swap two variables**

X = 5

Y = 10

Temp = x

X = y

Y = temp

Print (“Value of x after swapping :”)

Print (“Value of y after swapping :”)

Output :

Value of x after swapping : 10

Value of y after swapping : 5

**Check Armstrong or not**

Num = int(input(“Enter a number :”))

Sum = 0

Temp = num

While temp > 0 :

Digit = temp%10

Sum += digit \*\*3

Temp // = 10

If num == sum :

Print (num, “ is an armstrong number”)

Else :

Print (num, “is not an armstrong number”)