

Conditionals and Iterations

Conditional Execution

- To write useful programs we need the ability to check **conditions** and change the behaviour of the program accordingly.
- Different conditional statements in python are:
 - IF
 - IF ---Else (Alternative Execution)
 - IF--- ELIF----- ELSE (Chained Conditionals)
 - Nested Conditionals

If Condition

- If $x > 0$:

```
print( "x is positive")
```

- The Boolean expression after the if statement is called the condition. If it is true, then the indented statement gets executed. If not, nothing happens.
- Structure of If
 - HEADER:
FIRST STATEMENT
...
LAST STATEMENT

If Condition

- There is no limit on the number of statements that can appear in the body of an if statement, but there has to be at least one.
- Occasionally, it is useful to have a body with no statements (usually as a place keeper for code you haven't written yet). In that case, you can use the **pass** statement, which does nothing.

Alternative Execution

- A second form of the if statement is alternative execution, in which there are two possibilities and the condition determines which one gets executed.
- Eg:

```
if x%2 == 0:  
    print( x, "is even")  
else:  
    print( x, "is odd")
```
- The alternatives are called branches.

Continue....

- Since the condition must be true or false, exactly one of the alternatives will be executed. The alternatives are called **branches**, because they are branches in the flow of execution.

Chained Conditionals

- Sometimes there are more than two possibilities and we need more than two branches.

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")
```

NOTE: There is no limit of the number of elif statements, but the last branch has to be an else statement

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)`

if $0 < x$ and $x < 10$:

```
print ("x is a positive single digit.")
```

- Python provides an alternative syntax that is similar to mathematical notation:

if $0 < x < 10$:

```
print ("x is a positive single digit.")
```

Shortcuts for Conditions

- Numeric value 0 is treated as False
- Empty sequence "", [] is treated as False
- Everything else is True

if $m \% n$:

$(m, n) = (n, m \% n)$

else:

$\text{gcd} = n$

Avoid Nested If

- **For example,** We can rewrite the following code using a single conditional:

```
if 0 < x:
```

```
    if x < 10:
```

```
        print ("x is a positive single digit.")
```

Better way:

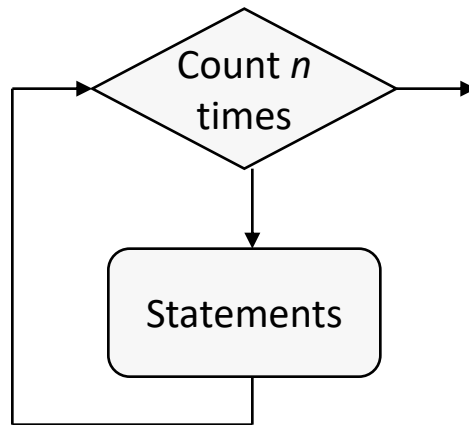
```
if 0 < x and x < 10:
```

```
    print ("x is a positive single digit.")
```

ITERATION

Doing the Same Thing Many Times

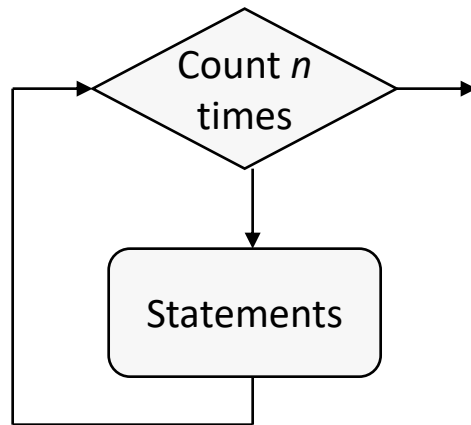
- It's possible to do something repeatedly by just writing it all out
- Print 'hello' 5 times



```
>>> print('Hello!')
Hello
>>> print('Hello!')
Hello
>>> print('Hello!')
Hello
>>> print('Hello!')
Hello
>>> print('Hello!')
Hello
```

Iteration and Loops

- A *loop* repeats a sequence of statements
- A *definite loop* repeats a sequence of statements a predictable number of times



```
>>> for x in range(5): print('Hello!')  
...  
Hello  
Hello  
Hello  
Hello  
Hello
```

The for Loop

- Python's **for** loop can be used to iterate a definite number of times

`for <variable> in range(<number of times>): <statement>`

- Use this syntax when you have only one statement to repeat

```
for <variable> in range(<number of times>):  
    <statement-1>  
    <statement-2>  
    ...  
    <statement-n>
```

- Use *indentation* to format two or more statements below the *loop header*

```
>>> for x in range(3):  
...     print('Hello!')  
...     print('goodbye')  
...  
Hello!  
goodbye  
Hello!  
goodbye  
Hello!  
goodbye
```

Using the Loop Variable

- The *loop variable* picks up the next value in a *sequence* on each pass through the loop
- The expression **range(n)** generates a sequence of **ints** from 0 through **n - 1**

loop variable

```
>>> for x in range(5): print(x)
...
0
1
2
3
4
>>> list(range(5)) # Show as a list
[0, 1, 2, 3, 4]
```


Counting from 1 through n

- The expression **range(low, high)** generates a sequence of **ints** from **low** through **high - 1**

```
>>> for x in range(1, 6): print(x)
...
1
2
3
4
5
```

Counting from n through 1

- The expression **range(high, low, step)** generates a sequence of **ints** from **high** through **low+1**.

```
>>> for x in range(6, 1, -1): print(x)
...
6
5
4
3
2
```

Skipping Steps in a Sequence

- The expression **range(low, high, step)** generates a sequence of **ints** starting with **low** and counting by **step** until **high - 1** is reached or exceeded

```
>>> for x in range(1, 6, 2): print(x)
...
1
3
5
>>> list(range(1, 6, 2)) # Show as a list
[1, 3, 5]
```

Using a Loop in a Real Problem

- An investor deposits \$10,000 with the Get-Rich-Quick agency and receives a statement predicting the earnings on an annual percentage rate (APR) of 6% for a period of 5 years. Write a program that prints the beginning principal and the interest earned for each year of the period. The program also prints the total amount earned and the final principal.

- **Pseudocode:**

```
principal = 10000
rate = .06
term = 5
totalinterest = 0
for each year in term
    print principal
    interest = principal * rate
    print interest
    principal = principal + interest
    totalinterest = totalinterest + interest
print totalinterest
print principal
```

While Loop

A for loop is used when a program knows it needs to repeat a block of code for a certain number of times.

A while loop is used when a program needs to loop until a particular condition occurs.

```
i = 0
while i < 10:
    print(i)
    i += 1
```

Flow of Execution for WHILE Statement

1. Evaluate the condition, yielding 0 or 1.
2. If the condition is false (0), exit the `while` statement and continue execution at the next statement.
3. If the condition is true (1), execute each of the statements in the body and then go back to step 1.

Looping Until User Wants To Quit

```
quit = "n"  
while quit == "n":  
    quit = input("Do you want to quit? ")
```

Range Function

- Range returns an immutable sequence objects of integers between the given start integer to the stop integer.
- range() constructor has two forms of definition:
 - range(stop)
 - range(start, stop, step)
 - **start** - integer starting from which the sequence of integers is to be returned
 - integer before which the sequence of integers is to be returned.
The range of integers end at **stop - 1**.
 - **step (Optional)** - integer value which determines the increment between each integer in the sequence

Exercise

1. Write a password guessing program to keep track of how many times the user has entered the password wrong. If it is more than 3 times, print "You have been denied access." and terminate the program. If the password is correct, print "You have successfully logged in." and terminate the program.
2. Write a program that asks for two numbers. If the sum of the numbers is greater than 100, print "That is a big number" and terminate the program.
3. Write a Python program that accepts a number from the user and reverse it.
4. Write a python program to find those numbers which are divisible by 7 and multiples of 5, between 1500 and 2700.

5. Write a Python program to get the Fibonacci series between 0 to 50.
6. Write a Python program which iterates the integers from 1 to 50. For multiples of three print "Fizz" instead of the number and for the multiples of five print "Buzz". For numbers which are multiples of both three and five print "FizzBuzz"
7. Write a Python program to print alphabet pattern 'A'.
8. Write a Python program to print alphabet pattern 'D'.
9. Write a Python program to check whether an alphabet is a vowel or consonant.
10. Write a Python program to check a triangle is equilateral, isosceles or scalene.
Note :An equilateral triangle is a triangle in which all three sides are equal.
A scalene triangle is a triangle that has three unequal sides.
An isosceles triangle is a triangle with (at least) two equal sides.