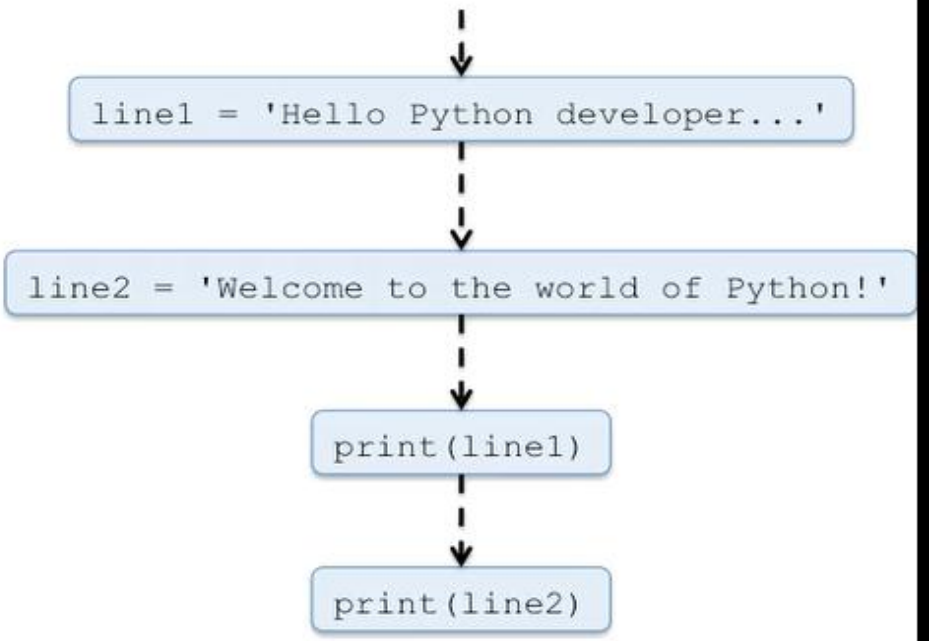
Making Choices

Python program

A Python program is a sequence of Python statements

- Stored in a text file called a Python module
- Executed using an IDE or "from the command line"




```
line1 = 'Hello Python developer...'
line2 = 'Welcome to the world of Python!'
print(line1)
print(line2)
```

```
$ python hello.py
Hello Python developer…
Welcome to the world of Python!
```

Making Choices

Making choices is fundamental concept of programming, we do this whenever we want our program to behave differently depending on the data it's working with.

- Making choice called *control flow* statements
- They control the way the computer executes programs.
- These statements involve a Python type that is used to represent truth and false.
- Unlike the integers, floating-point numbers, and strings

A Boolean Type

- In Python, there is a type called bool (without an “e”).
- Unlike int and float, which have billions of possible values,
- Bool has only two: True and False. True and False
- Bool are values, just as much as the numbers 0 and -43.7.
- They have the value 0 or 1

Boolean Operators

- There are only three basic Boolean operators:
- and,
- or, and
- not.
- not has the highest precedence, followed by and, followed by or.
- not is a unary operator: it is applied to just one value

Examples:

>>> not True

>>> not False

and **is a binary operator, the expression produces True if both left and right are True, and it produces False otherwise:**

>>> True and True

>>> False and False

>>> True and False

>>> False and True

Boolean Operators

or **is also a binary operator. It produces True if *either* operand is True, and it produces False only if both are False:**

```
>>> True or True
>>> False or False
>>> True or False
>>> False or True
```

This definition is called *inclusive or*, since it allows both possibilities as well as either.

Boolean Operators

EXAMPLES

```
>>> b1 = False
>>> b2 = False
>>> (b1 and not b2) or (b2 and not b1)
```

Result?

```
>>> b1 = False
>>> b2 = True
>>> (b1 and not b2) or (b2 and not b1)
```

Result?

```
>>> b1 = True
>>> b2 = False
>>> (b1 and not b2) or (b2 and not b1)
```

Result?

```
>>> b1 = True
>>> b2 = True
>>> (b1 and not b2) or (b2 and not b1)
```

Result?

Boolean Operators

```
>>> cold = True
>>> windy = False
>>> (not cold) and windy
>>> not (cold and windy)
```

Exercise: complete the following table

cold	windy	cold and windy	cold or windy	(not cold) and windy	not (cold and windy)
TRUE	TRUE				
TRUE	FALSE				
FALSE	TRUE				
FALSE	FALSE				

Relational Operators

Symbol	Operation
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
==	Equal to
!=	Not equal to

Relational Operators

Examples

```
>>> 45 > 34

>>> 45 > 79

>>> 23.1 >= 23

>>> 23.1 >= 23.1

>>> 23.1 <= 23.1

>>> 67.3 == 87

>>> 67.0 == 67

>>> 67.0 != 67

>>> 67.0 != 23
```

Relational Operators

Examples : comparison using variables

```
>>> x = 5
>>> y = 10
>>> z = 20
>>> (x < y) and (y < z)
?
>>> x = 3
>>> (1 < x) and (x <= 5)
?
>>> x = 7
>>> (1 < x) and (x <= 5)
?
>>> 3 < 5 != True
?
>>> 3 < 5 != False
?
```

Relational Operators

Examples : comparison using variables

```
>>> (3 < 5) and (5 != True)
?
>>> (3 < 5) and (5 != False)
?
```

Since 5 is neither True nor False, the second half of each expression is True, so the expression as a whole is True as well.

```
>>> (3 < 5) and (False != False)
?
```

Comparing Strings

The characters in strings are represented by integers: a capital *A*, for example, is represented by 65, while a space is 32, and a lowercase *z* is 172. This encoding is called *ASCII*.

Examples : comparison of strings

```
>>> date = input('Enter a date in the format DD MTH YYYY: ')
Enter a date in the format DD MTH YYYY: 24 Feb 2013
>>> 'Jan' in date
?
>>> date = input('Enter a date in the format DD MTH YYYY: ')
Enter a date in the format DD MTH YYYY: 03 Jan 2002
>>> 'Jan' in date
?
```

Try it

Comparing Strings

The in operator produces True exactly when the first string appears in the
second string. This is case sensitive:

>>> 'a' in 'abc'

?

>>> 'A' in 'abc'

?

The empty string is always a substring of every string:

>>> ‘ ' in 'abc'

?

>>> '' in ''

?

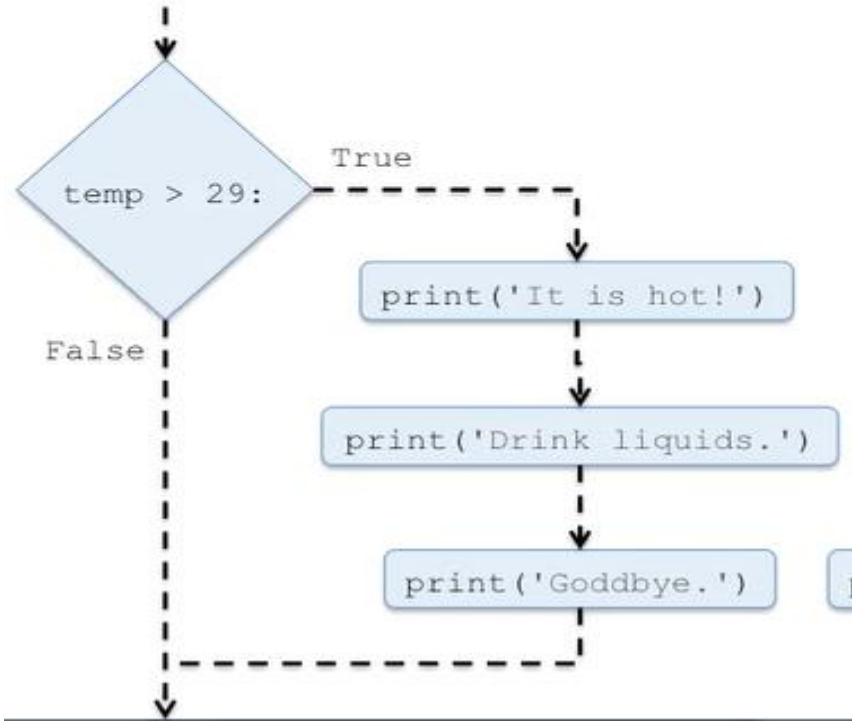
Try it

Choosing Which Statements to Execute

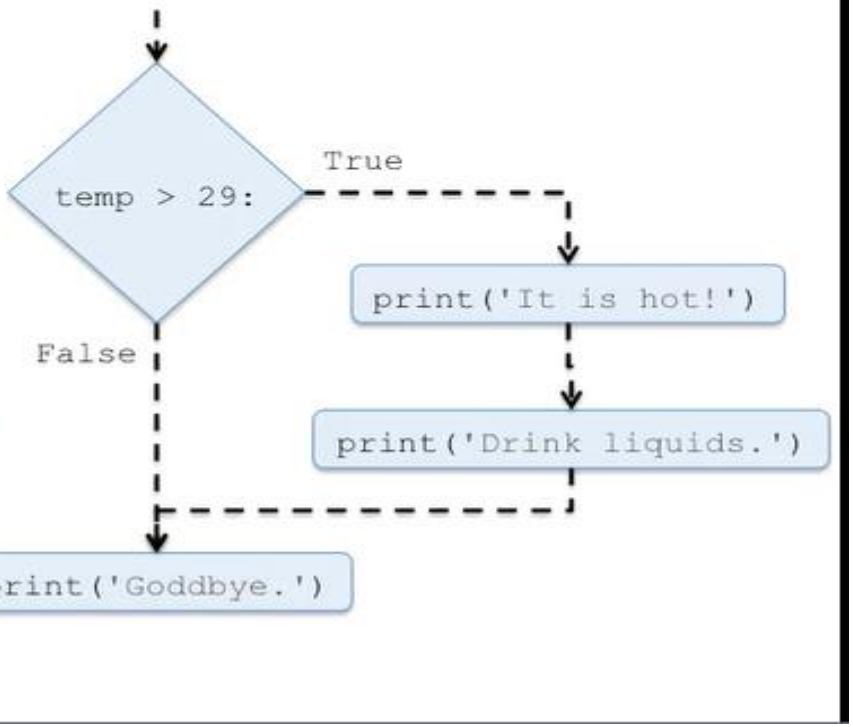
An if statement lets you change how your program behaves based on a condition.

Indentation is critical

```
if temp > 29:
    print('It is hot!')
    print('Drink liquids.')
    print('Goodbye.')
```




```
if temp > 29:
    print('It is hot!')
    print('Drink liquids.')
print('Goodbye.')
```



Example1

```
def test():
    ph = float(input('Enter the pH level: '))
    if ph > 50:
        print("There is a $25 charge for luggage that heavy.")
        print("Thank you for your business.")
```

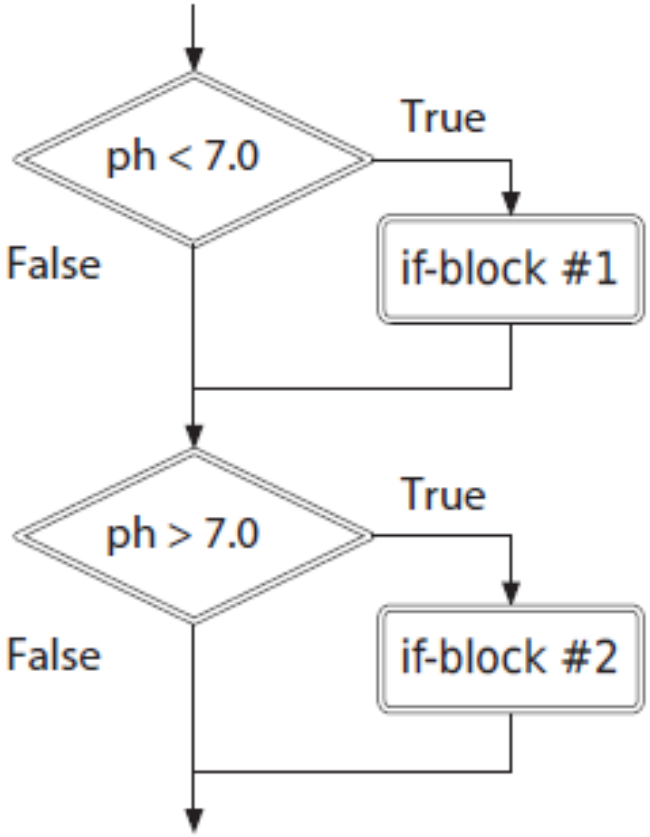
Example2

```
def test():
    ph = float(input('Enter the pH level: '))

    if ph > 50:
        print("There is a $25 charge for luggage that heavy.")
        print("Thank you for your business.")
    print(" Goodbye ")
```

Two way If statement with else

Here's a flow chart that shows how Python executes the if statements. The diamonds are conditions, and the arrows indicate what path to take depending on the results of evaluating those conditions:



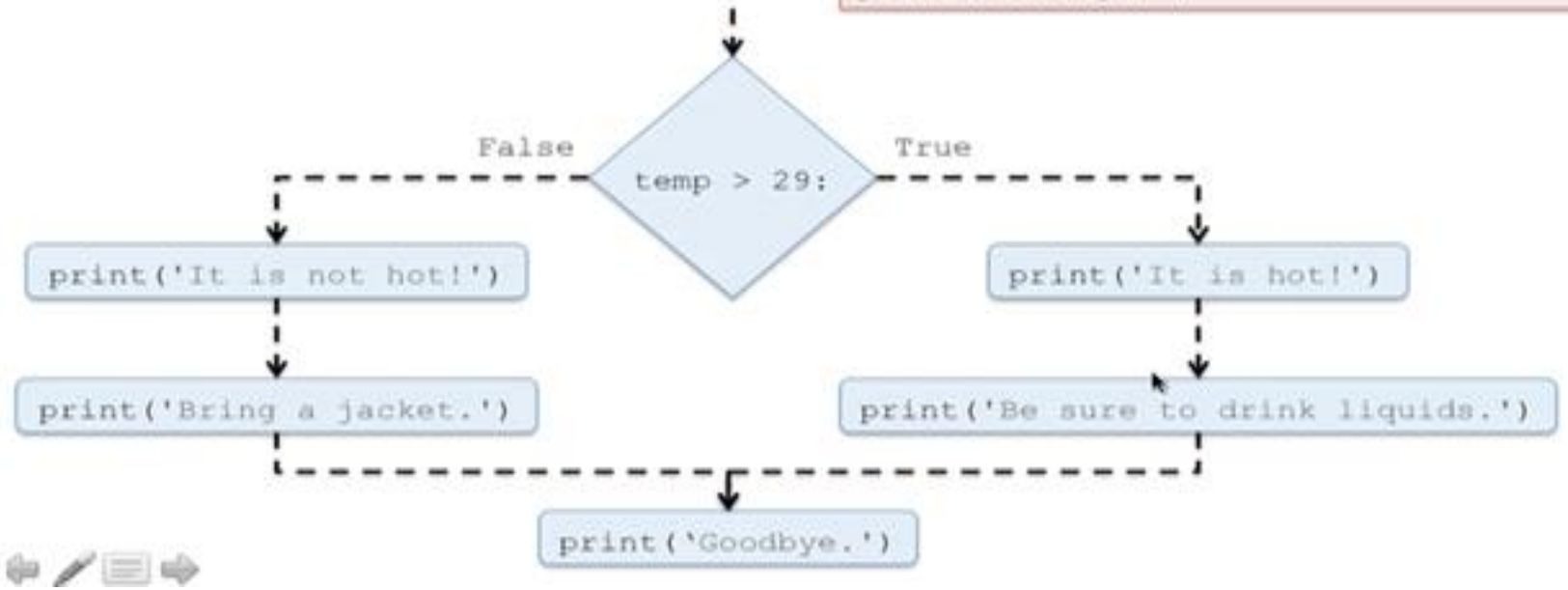
Where will be the control if the value = 7

Two way If statement with else

Here's a flow chart that shows how Python executes the if - else statements. The diamonds are conditions, and the arrows indicate what path to take depending on the results of evaluating those conditions:

```
if <condition>:
    <indented code block 1>
else:
    <indented code block 2>
<non-indented statement>
```

```
if temp > 29:
    print('It is hot!')
    print('Be sure to drink liquids.')
else:
    print('It is not hot.')
    print('Bring a jacket.')
print('Goodbye.')
```



Where will be the control if the value = 29

Example3

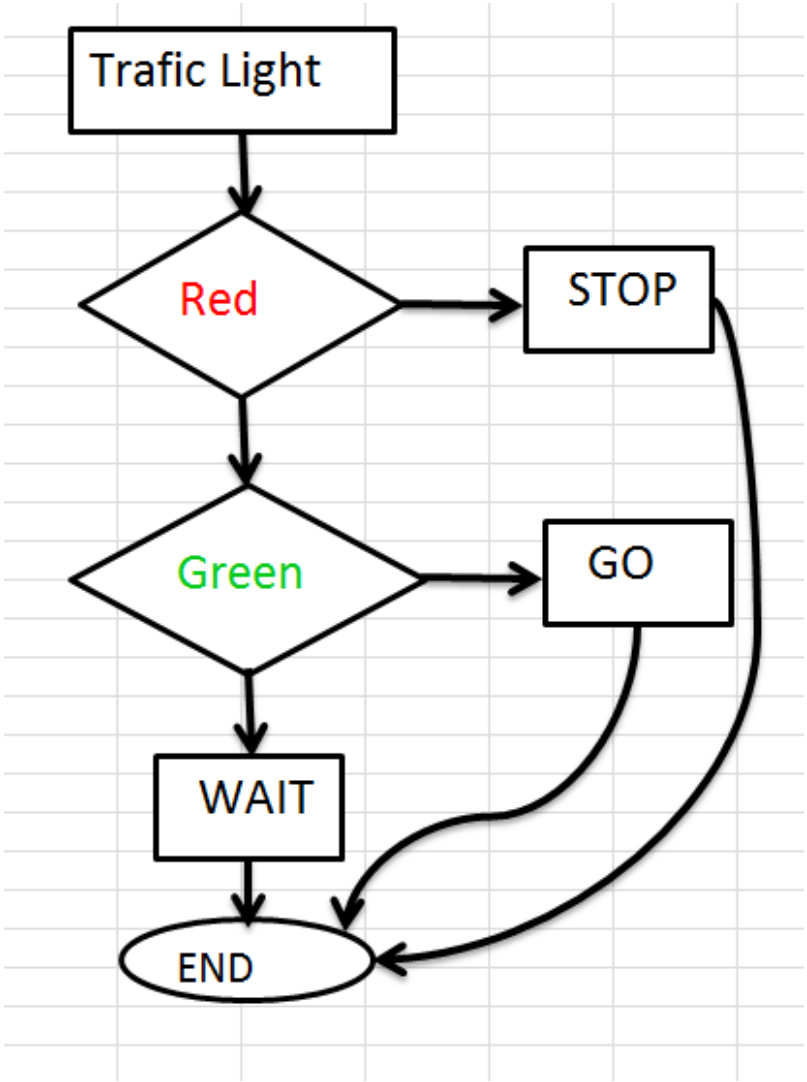
```
def test3():
    # Indentation is very importatnt
    temp = float(input('Enter the temprature today: '))

    if temp > 29:
        print("It is hot today.")
        print("be sure to drink liquid")
    else:
        print("Today is nice weather .")
        print("Enjoy your time")

    print("Thank you for your business - Goodbye ")
```

Two way If statement with elif

The elif is checked only when the first if condition is evaluated to False. Here's a flow chart for this code:



Example

```
def test4():
    # Indentation is very important
    light = (input('Enter the trafic light color: '))

    if  light  == 'RED':
        print("You must STOP.")
        print("be sure not to pass the red")
    elif light == 'GREEN':
        print("It is green,ylu can  go")
        print("Enjoy drive")
    else:
        print("WAIT PLEASE ")
    print("Thank you for your business - Goodbye ")
```

You may avoid capital and small letter compare by converting input to capital:

```
light  = light.upper()
```

Exercise

Convert the grade to it's corresponding letter grade using logical operator and

Criteria:

If grade more than 90 inclusive then LG is A+
Between 80 – 89 inclusive then LG is A
Between 70 – 79 inclusive then LG is B
Between 60 – 69 inclusive then LG is C
Between 0 – 59 inclusive then LG is F
Grade out of the range 0-100 bring error message

Use elif