Built-in function `input()`

Function `input()` requests and reads input from the user interactively

- It's (optional) input argument is the request message
- Typically used on the right side of an assignment statement

When executed:

1. The input request message is printed
2. The user enters the input
3. The *string* typed by the user is assigned to the variable on the left side of the assignment statement

```
>>> name = input('Enter your name: ')
Enter your name: Michael
>>> name
'Michael'
>>> ========= RESTART ============
>>>
Enter your first name: Michael
Enter your last name: Jordan
Hello Michael Jordan...
Welcome to the world of Python!
```

```
first = input('Enter your first name: ')
last = input('Enter your last name: ')
line1 = 'Hello' + first + '' + last + '…'
print(line1)
print('Welcome to the world of Python!')
```

input.py

Exercise

Write a program that:

1. Requests the user's name
2. Requests the user's age
3. Computes the user's age one year from now and prints the message shown

```
>>>
Enter your name: Marie
Enter your age: 17
Marie, you will be 18 next year!
```

```
name = input('Enter your name: ')
age = int(input('Enter your age: '))
line = name + ', you will be ' + str(age+1) + ' next year!'
print(line)
```

Exercise

Write a program that:

1. Requests the user's name
2. Requests the user's age
3. Prints a message saying whether the user is eligible to vote or not

Need a way to execute a Python statement if a condition is true

Comments and docstrings

Python programs should be documented

- So the developer who writes/maintains the code understands it
- So the user knows what the program does

Comments

```
def f(x):
    res = x**2 + 10    # compute result
    return res         # and return it
```

Docstring

```
def f(x):
    ''' (number)->number
    returns x**2 + 10 '''
    res = x**2 + 10    # compute result
    return res         # and return it
```

```
>>> help(f)
Help on function f in module
__main__:

f(x)

>>> def f(x):
        'returns x**2 + 10'
        res = x**2 + 10
        return res

>>> help(f)
Help on function f in module
__main__:

f(x)
    (number)->number
    returns x**2 + 10

>>>
```

Boolean expressions

In addition to algebraic expressions, Python can evaluate Boolean expressions

- Boolean expressions evaluate to `True` or `False`
- Boolean expressions often involve comparison operators
  <, >, ==, !=, <=, and >=

```
>>> 2 < 3
True
>>> 2 > 3
False
>>> 2 == 3
False
>>> 2 != 3
True
>>> 2 <= 3
True
>>> 2 >= 3
False
>>> 2+4 == 2*(9/3)
True
```

In a an expression containing algebraic and comparison operators:

- Algebraic operators are evaluated first
- Comparison operators are evaluated next

Boolean operators

In addition to algebraic expressions, Python can evaluate Boolean expressions

- Boolean expressions evaluate to `True` or `False`
- Boolean expressions may include Boolean operators `and`, `or`, and `not`

```
>>> 2<3 and 3<4
True
>>> 4==5 and 3<4
False
>>> False and True
False
>>> True and True
True
>>> 4==5 or 3<4
True
>>> False or True
True
>>> False or False
False
>>> not(3<4)
False
>>> not(True)
False
>>> not(False)
True
>>> 4+1==5 or 4-1<4
```

In a an expression containing algebraic, comparison, and Boolean operators:

- Algebraic operators are evaluated first
- Comparison operators are evaluated next
- Boolean operators are evaluated last

Compound Boolean expression

Suppose variable age refers to a an age of a person.

How would you write a Boolean expression evaluates to True if age between 18 and 64?

```
age >= 18 and age < 65
```

The following is an equivalent expression that works in Python but not in most other languages:

```
18 <= age <= 65
```

Compound Boolean expression

How would you write a compound Boolean expression evaluates to True if age is less than 18 or at least 65, and otherwise evaluates to False.

```
age < 18 or age >= 65
```

This is an equivalent expression:

```
not( age>=18 and age< 65)
```

This is an equivalent expression:

```
not(18 <= age <65)
```

Boolean Expressions

Boolean expressions evaluate to `True` or `False`

Logical/Boolean operators in Math and Python:

Math	Python
AND	and
OR	or
NOT	not

Here is how your compare two variables a and b in Math and Python

Math	Python
a = b	a == b
a ≤ b	a <= b
a ≥ b	a >= b
a ≠ b	a != b

Truth table

A TRUTH TABLE for a compound Boolean expression shows the results for all possible combinations of the simple expressions:

x	y		x and y	x or y	not x
False	False		False	False	True
True	False		False	True	False
False	True		False	True	
True	True		True	True	