**Interactive Programs**

- we know how to output something on the screen:
  ```python
  print('Hello world.')
  ```
- input:
  ```python
  input(<prompt>)
  ```
  returns the input from the keyboard

**Example**

```python
name = input('type your name: ')
```

---

**If Statement**

Let's assume you want to write a program that: 1) asks a user to type his/her name, 2) checks if it is a known user, and 3) prints a welcome statement.

- we know how to do the first part:
  ```python
  known_users = ['Sandra', 'Markus']
  name = input('type your name: ')
  ```
- We can check whether a person is in the list of known users:
  ```python
  name in known_users
  ```
- But how do we tell python to print a welcome message if the name is known?

**Blocks and Indenting**

**Definition**

In python, blocks are created by the use of a colon, followed by an indented section of text.

```python
if <test>:
  do something
  do another thing
  a final thing
  do this regardless
```

**Truth Values**

- a test (in the if statement) corresponds to a yes/no question and can be either true or false
- the following values count as false:
  ```python
  False
  None
  [] (empty list)
  {} (empty dict)
  '' (empty string)
  () (empty tuple)
  ```
- everything else counts as true!
**Else Statements**

- In case the program needs to do something when the test is false, use the `else:` statement
- E.g. if a user is not known, add him/her to the list

**Example**

```python
def user_filter(users, name):
    if name in users:
        print('Hello, ' + name + '!
    else:
        known_users.append(name)
        print('Hello, ' + name + '!
```

**Nested Blocks**

- Example

```python
known_users = ['Sandra', 'Markus']
name = input('Type your name: ')  # input statement
if name in known_users:
    print('Hello, ' + name + '!
    if name.startswith('Dr. '):
        print('Taking yourself seriously, huh?'
    else:
        print('You’re my buddy.'
else:
    known_users.append(name)
    print('Hello, ' + name + '!
```

**Equality vs. Identity**

```python
>>> x = y = [1, 2, 3]
>>> x == y
True
>>> x == z
True
>>> x is y
False
>>> x is z
False
```

**Booleans**

**Definition**

You can combine conditions with `and` and `or`, and negate with `not`

**Example**

```python
if 5 < x < 10 and x not in y:
    print('x is between 5 and 10')
    print('and is not in the list y')
```
Short-circuit logic

Python evaluates the first part of an and/or condition and can short circuit
- If x in x or y is True, no need to evaluate both
- If x in x and y is False, no need to evaluate both

This means you can do things like:
```python
if line and line.startswith('%'):
```

You can also do things like:
```python
name = input('Please enter your name: ') or '<unknown>'
```