Interactive Programs

Programs generally need input on which to operate
The `input` function provides a convenient way to read input values as strings
Various type conversion functions, such as `float` and `int`, can convert strings to other types
Comparing input values and responding conditionally to them is how to achieve interactivity

Conditional Statements

A conditional statement lets us choose which statement will be executed next
Therefore they are sometimes called selection statements
Conditional statements give us the power to make basic decisions
The Python conditional statements are the:

- `if` statement
- `if-else` statement
- `if-elif-else` statement

The if Statement

The `if` statement has the following syntax:

```
if condition:
  statement block
```

The condition must be a boolean expression. It must evaluate to either true or false.

If the condition is true, the statement is executed.
If it is false, the statement is skipped.

Logic of an if statement

```
condition evaluated
  true
  false
  statement
```
The if Statement

An example of an `if` statement:
```python
if sum > MAX:
    delta = sum - MAX
    print('The sum is', sum)
```

- First the **condition** is evaluated -- the value of `sum` is either greater than the value of `MAX`, or it is not
- If the condition is true, the assignment statement is executed -- if it isn't, it is skipped.
- Either way, the call to `print` is executed next

The if-else Statement

An **else clause** can be added to an `if` statement to make an `if-else` statement
```python
if condition:
    statement1
else:
    statement2
```

- If the **condition** is true, `statement1` is executed; if the condition is false, `statement2` is executed
- One or the other will be executed, but not both

Logic of an if-else statement

```
condition evaluated
```
```
true-----------false
```
```
statement1
```
```
statement2
```

The if-elif-else Statement

One or more **elif clauses** can be added to an `if-else` statement to make an `if-elif-else` statement
```python
if condition:
    statement1
elif condition:
    statement2
else:
    statement3
```

- If the 1st **condition** is true, `statement1` is executed; if the 2nd condition is true, `statement2` is executed; otherwise, `statement3` is executed
- Only one of the statements will be executed

Boolean Expressions

A condition often uses one of Python's **equality operators** or **relational operators**, which all return boolean (`True` or `False`) results:

- `==` equal to
- `!=` not equal to
- `<` less than
- `>` greater than
- `<=` less than or equal to
- `>=` greater than or equal to

Don't confuse using the **equality operator** (`==`) and the **assignment operator** (`=`). Relational operators have lower precedence than numeric operators.

Boolean Expressions

Boolean expressions are common in `if` statements:
```python
if value >= MAX:
    print('The value is too large')
else:
    print('The value is just fine')
```

Interval comparisons work like this:
```python
if MIN < value < MAX:
    print('The value is just right')
else:
    print('The value is all wrong')
```

Comparing floating point numbers is not reliable, so it is recommended to use `math.isClose` instead.
Boolean Operators

An expression is a combination of one or more operators and operands.

Comparisons in an expression can be combined using Boolean operators:

- **not**: true if X is false, false if X is true
- **and**: true only if both X and Y are true
- **or**: true if either X or Y is true, or both

```python
if X is False:
    # do something
else:
    # do something else
```

```python
X and Y
```

```python
X or Y
```

```python
if not X:
    print('X is false')
else:
    print('X is true')
```

Comparing Strings

String comparison can be done using the equality (==) operator and the not equal (!=) operator.

```python
if magician == 'Harry Houdini':
    print('World’s greatest magician!')
else:
    print('Nope. Not this time.')
```

```python
if printer_status != 'ready':
    print('printer needs attention')
else:
    print('all set')
```

Relational operators can compare the alphabetical order of strings.

```python
if word1 < word2:
    print(word1 + ' comes before ' + word2)
else:
    print(word1 + ' is not before ' + word2)
```

The input Function

The input function reads input from the keyboard and returns it as a string.

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

An optional string parameter will display a prompt.

```python
something = input('Enter something: ') # the default prompt
```

To read an integer value, you must convert input to the desired type using the int type conversion function.

```python
count = int(input('Enter the number of items: '))
```

To read a floating point value, use the float function.

```python
item_cost = float(input('Enter the item cost: '))
```

Total cost: count * item_cost

Program Style

Programming conventions describe good ways to design code. Indentation is used for groups or blocks of code:

```python
if fruit == 'apple':
    print('Mmm... I love apples!')
else:
    print('Maybe you prefer bananas?')
```

It is conventional to indent by 4 spaces and be consistent!

```python
if fruit == 'apple':
    print('Mmm... I love apples!')
else:
    print('Maybe you prefer bananas?')
```

Names in Python should follow naming conventions. Use all lowercase for variable and function names, with underscores between words.

```python
name = 'John Cleese'
given_name = 'John'
family_name = 'Cleese'
```

Constants should use all UPPERCASE to make them stand out.

```python
MAX_SIZE = 1000
LITERS_PER_GALLON = 3.78541
```
Program Style

Quotes in strings should try to avoid escape characters

```
my_string = "She said, "I have nothing to say.""
```

Using a backslash in front of a double-quote will escape it.

```
my_string = "She said, \"I have nothing to say.\""
```

It’s better to use single quotes around a string containing double quotes, and vice versa.

Google for the online Python Style Guide for more!

Example: The High-Low Game

```
import random

# Choose a random number for the user to guess. Accepts guesses repeatedly, informing the user
# whether the guess was high or low.

target = random.randint(1, 100)

guess = 0  # Initial value out of range

while guess is not target:
    guess = int(input("Guess what it is: "))
    if guess < target:
        print("Too low!")
    elif guess > target:
        print("Too high!")
else:
    print("That's it! You got it in", count, "guesses.")

print("Thanks for playing.")
```