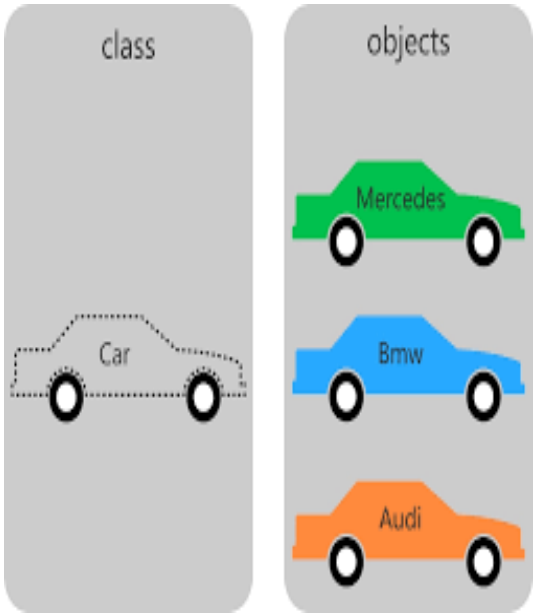
Objects

Introduction

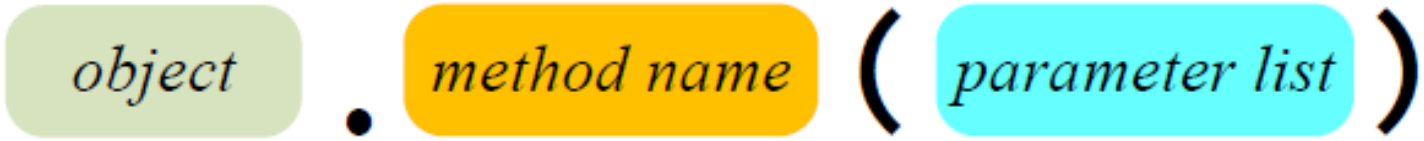
- Software development today is increasingly component based.
- A software system can be built largely by assembling pre-existing software building blocks.
- Python supports various kinds of software building blocks.
- The simplest of these is the function.
- A more powerful technique uses software objects.
- Python is object oriented(OO) programming language.
- An OO programming language allows the programmer to define, create, and use objects

Using Objects

- An object is an instance of a class.
- Integers, floating-point numbers, strings, and functions are all objects in Python.
- A typical object consists of two parts: data and methods.
- An object's data consists of its instance variables.
- Other names for instance variables include attributes and fields.
- Methods are like functions, and they are known also as operations.



Using Objects




```
word = "ABCD"
print(word.rjust(10, "*"))
print(word.rjust(3, "*"))
print(word.rjust(15, ">"))
print(word.rjust(10))
```

```
******ABCD
ABCD
>>>>>>>>>>>ABCD
      ABCD
```

String Objects

str Methods
`upper` Returns a copy of the original string with all the characters converted to uppercase
`lower` Returns a copy of the original string with all the characters converted to lower case
`rjust` Returns a string right justified within an area padded with a specified character which defaults to a space
`ljust` Returns a string left justified within an area padded with a specified character which defaults to a space
`center` Returns a copy of the string centered within an area of a given width and optional fill characters; fill characters default to spaces
`strip` Returns a copy of the given string with the leading and trailing whitespace removed; if provided an optional string, the `strip` function strips leading and trailing characters found in the parameter string

str Methods
`startswith` Determines if the string parameter is a prefix of the invoking string
`endswith` Determines if the string parameter is a suffix of the invoking string
`count` Determines the number times the string parameter is found as a substring within the invoking string; the count includes only non-overlapping occurrences
`find` Returns the lowest index where the string parameter is found as a substring of the invoking string; returns –1 if the parameter is not a substring of the invoking string
`format` Embeds formatted values in a string using {0}, {1}, etc. position parameters

String Objects

```
#  Strip leading and trailing whitespace and count substrings
s = "     ABCDEFGHBCDIJKLMNOPQRSBCDTUVWXYZ      "
print("[", s, "]", sep="")
s = s.strip()
print("[", s, "]", sep="")

#  Count occurrences of the substring "BCD"
print(s.count("BCD"))
```

```
[     ABCDEFGHBCDIJKLMNOPQRSBCDTUVWXYZ      ]
[ABCDEFGHBCDIJKLMNOPQRSBCDTUVWXYZ]
```

Fraction Objects

```
from fractions import Fraction

f1 = Fraction(3, 4)      # Make the fraction 3/4
print(f1)                # Print it
print(f1.numerator)      # Print numerator
print(f1.denominator)    # Print denominator
print(float(f1))         # Floating-point equivalent
f2 = Fraction(1, 8)      # Make another fraction, 1/8
print(f2)                # Print the second fraction
f3 = f1 + f2             # Add the two fractions
print(f3)                # 3/4 + 1/8 = 6/8 + 1/8 = 7/8
```

```
3/4
3
4
0.75
1/8
7/8
```

File Objects

```
f =  open('myfile.txt', 'r')
```

```
f = open('myfile.txt', 'w')
```

```
f = open('myfile.txt', 'a')
```

- `'r'` opens the file for reading
- `'w'` opens the file for writing; creates a new file
- `'a'` opens the file to append data to it

File Objects

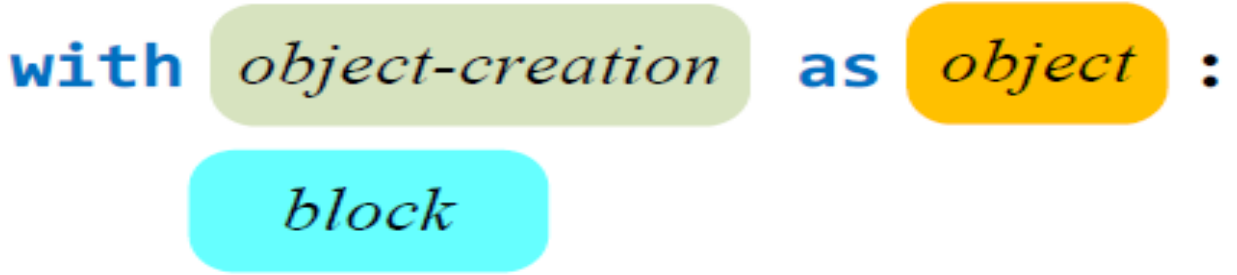
```
f = open('myfile.txt', 'w')

f.write('data')
f.write('compute')
f.write('process')
```

```
f.write('data\n')
f.write('compute\n')
f.write('process\n')
```

```
f = open('data.dat')        # f is a file object
for line in f:              # Read each line as text
    print(line.strip())     # Remove trailing newline character
f.close()                   # Close the file
```

with/as statement

The general form of the `with/as` statement is




```
with open('data.dat') as f:    # f is a file object
    for line in f:             # Read each line as text
        print(line.strip())    # Remove trailing newline character
    # No need to close the file
```

Turtle Graphics Objects

```
""" Draws in the window a spiral surrounded with an octogon """

from turtle import *

def octogon(t, x, y, color):
    """ Draws with turtle t an octogon centered at (x, y)
     with the specified color """
t.pencolor(color)   # Set pen color
t.penup()           # Lift pen to move it
t.setposition(x, y) # Move the pen to coordinates (x, y)
t.pendown()         # Place pen to begin drawing
for i in range(8):  # Draw the eight sides
    t.forward(80)
    t.right(45)
```

```
def spiral(t, x, y, color):
    """ Draws with turtle t a spiral centered at (x, y)
        with the specified color """
    distance = 0.2
    angle = 40
    t.pencolor(color)   # Set pen color
    t.penup()           # Left pen to move it
    t.setposition(x, y) # Position the pen at coordinates (x, y)
    t.pendown()         # Set pen down to begin drawing
    for i in range(100):
        t.forward(distance)
        t.left(angle)
        distance += 0.5
```

```
t = Turtle()        # Create a turtle object named t
octogon(t, -45, 100, 'red')
spiral(t, 0, 0, 'blue')
t.hideturtle()      # Make turtle t invisible
done()
```

Object Mutability and Aliasing

```
from fractions import Fraction

#  Assign some Fraction variables
f1 = Fraction(1, 2)
f2 = Fraction(1, 2)
f3 = f1

#  Examine the objects involved
print('f1 =', f1)
print('f2 =', f2)
print('f3 =', f3)

#  Examine the numerators and denominators separately
print('f1 numerator, denominator:', f1.numerator, f1.denominator)
print('f2 numerator, denominator:', f2.numerator, f2.denominator)
print('f3 numerator, denominator:', f3.numerator, f3.denominator)

#  Compare the fractions
print('f1 == f2?', f1 == f2)
print('f1 == f3?', f1 == f3)
print('f1 is f2?', f1 is f2)
print('f1 is f3?', f1 is f3)
```

```
f1 = 1/2
f2 = 1/2
f3 = 1/2
f1 numerator, denominator: 1 2
f2 numerator, denominator: 1 2
f3 numerator, denominator: 1 2
f1 == f2? True
f1 == f3? True
f1 is f2? False
f1 is f3? True
```

Object Mutability and Aliasing