Python

Mahtab Farrokh
Saman Fekri

Overview

- Python is Interpreted
- Python is Interactive
- Python is Object-Oriented
- Python is a Beginner's Language

Python features

- Easy to Read
- Python is Interactive
- Databases
- GUI Programming

Syntax

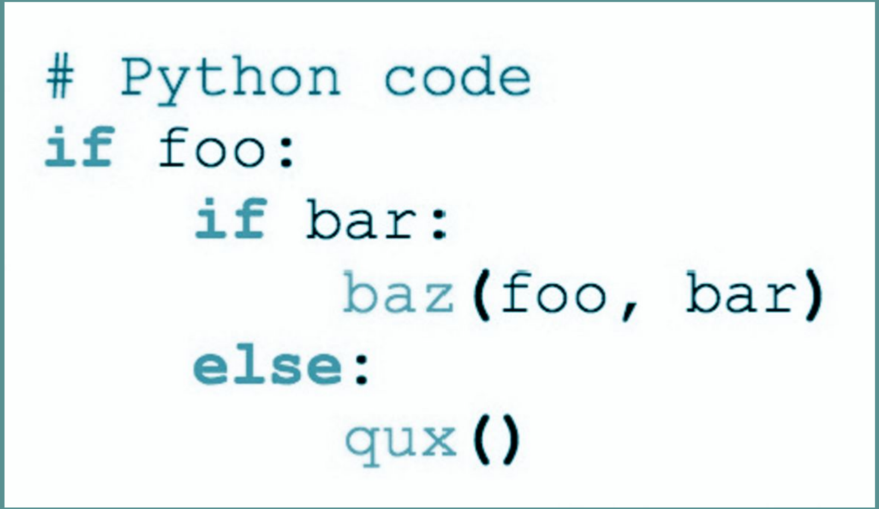
Hello World!

```
print('Hello World!')
```

Reserved words

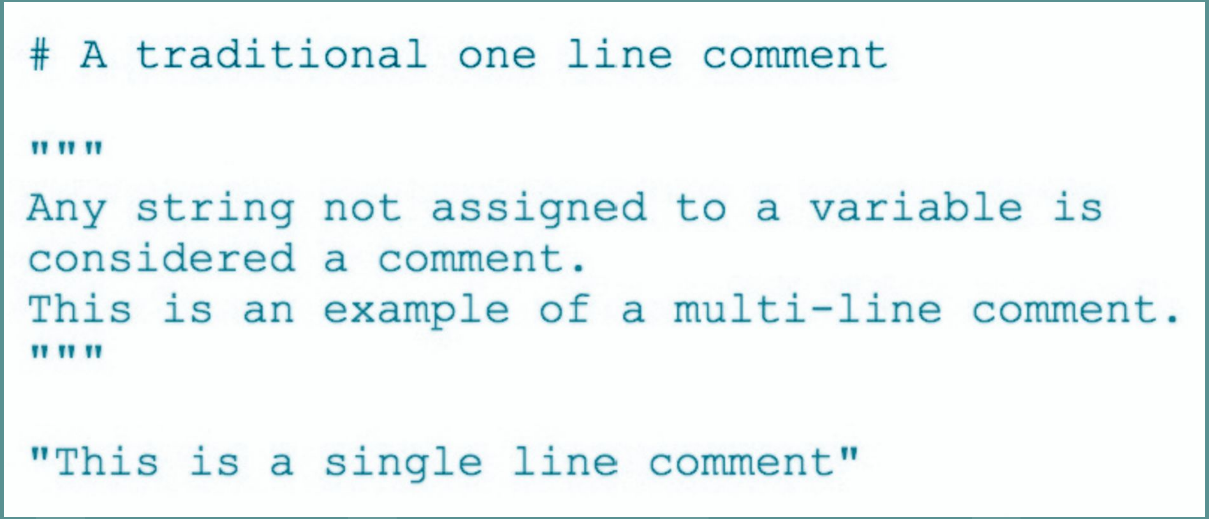
- And
- As
- Assert
- Break
- Class
- Continue
- Def
- Del
- Elif
- Else
- Except
- Exec
- Finally
- For
- From
- Global
- If
- Import
- In
- Is
- Not
- Or
- Pass
- Or
- Pass
- Print
- Return
- Try
- While
- With

Lines and Indentation



```
# Python code
if foo:
    if bar:
        baz(foo, bar)
    else:
        qux()
```

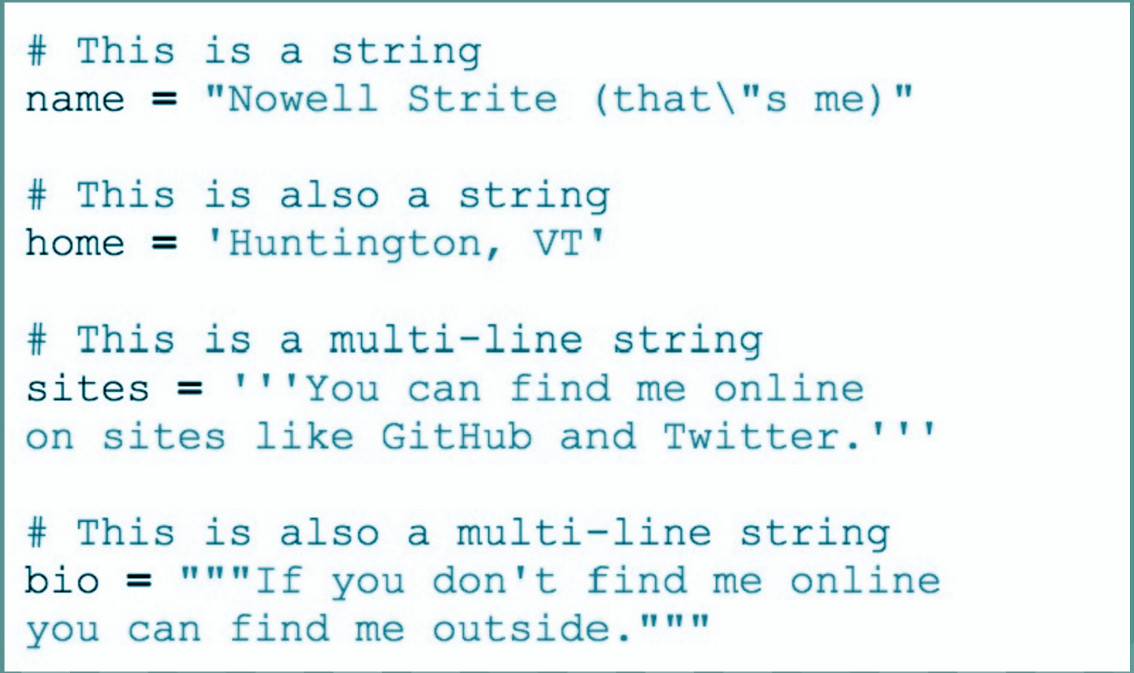
Comments



```
# A traditional one line comment

"""
Any string not assigned to a variable is
considered a comment.
This is an example of a multi-line comment.
"""

"This is a single line comment"
```

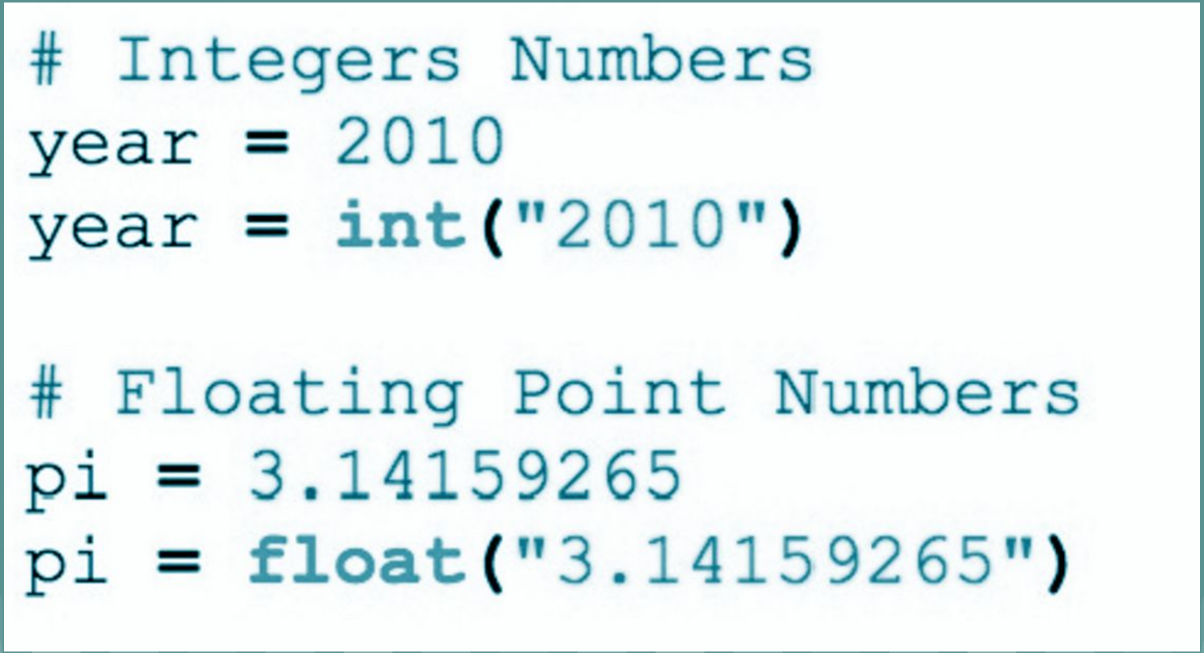
Types

Strings



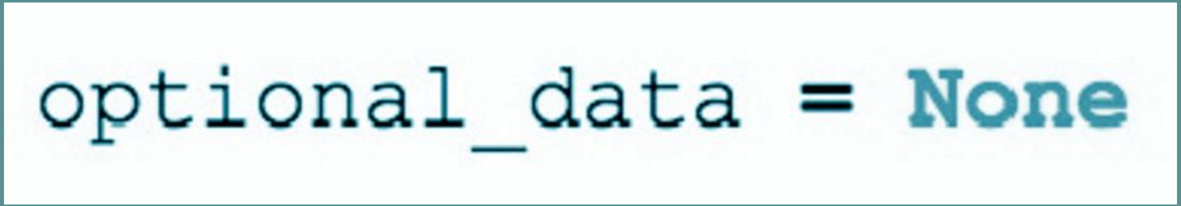
```
# This is a string
name = "Nowell Strite (that\"s me)"

# This is also a string
home = 'Huntington, VT'

# This is a multi-line string
sites = '''You can find me online
on sites like GitHub and Twitter.'''

# This is also a multi-line string
bio = """If you don't find me online
you can find me outside."""
```

Numbers



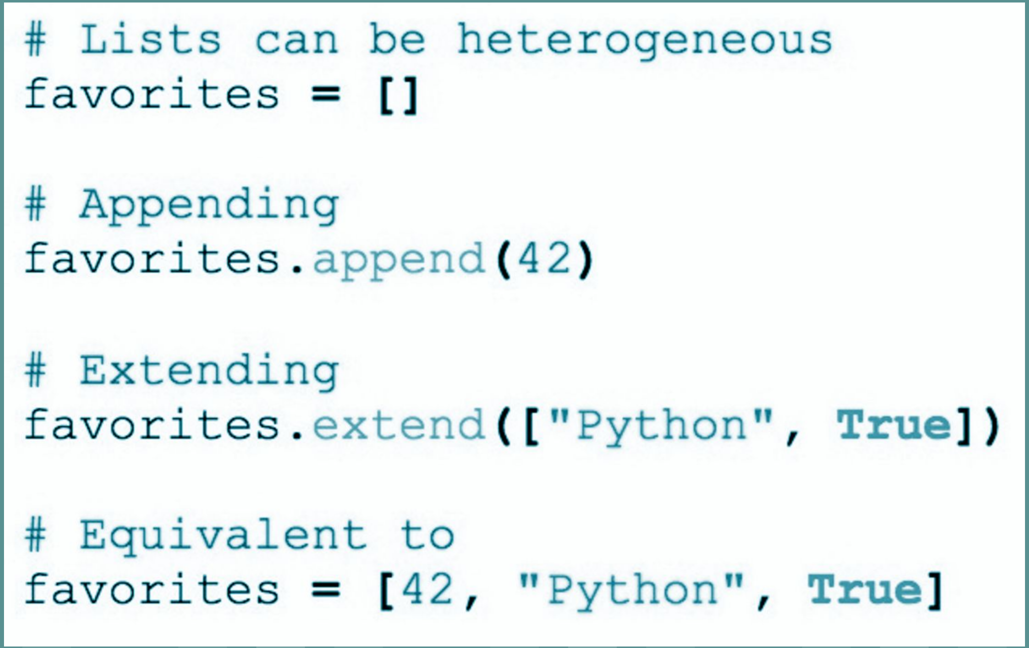
```
# Integers Numbers
year = 2010
year = int("2010")

# Floating Point Numbers
pi = 3.14159265
pi = float("3.14159265")
```

Null



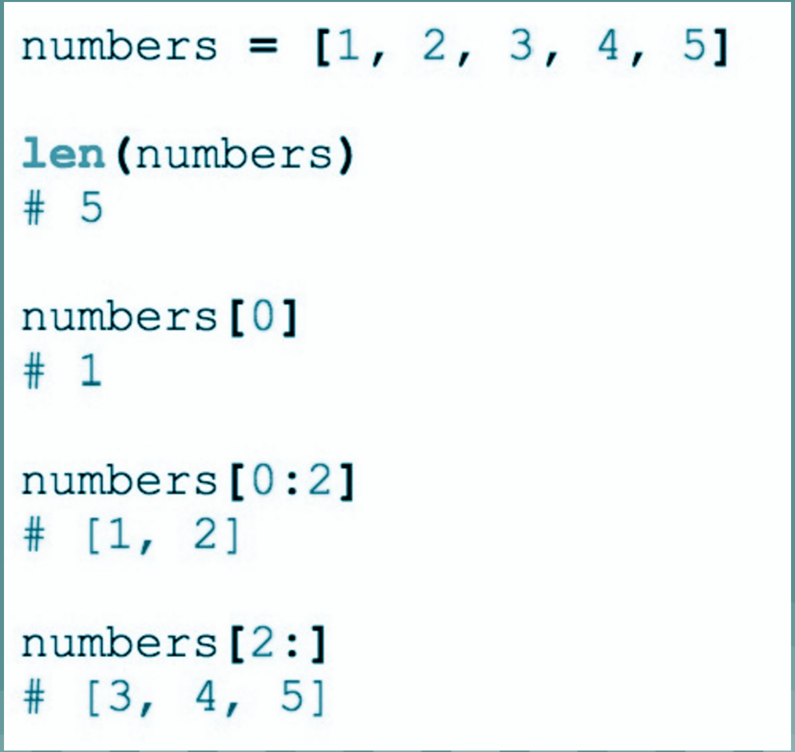
```python
optional_data = None
```

Lists



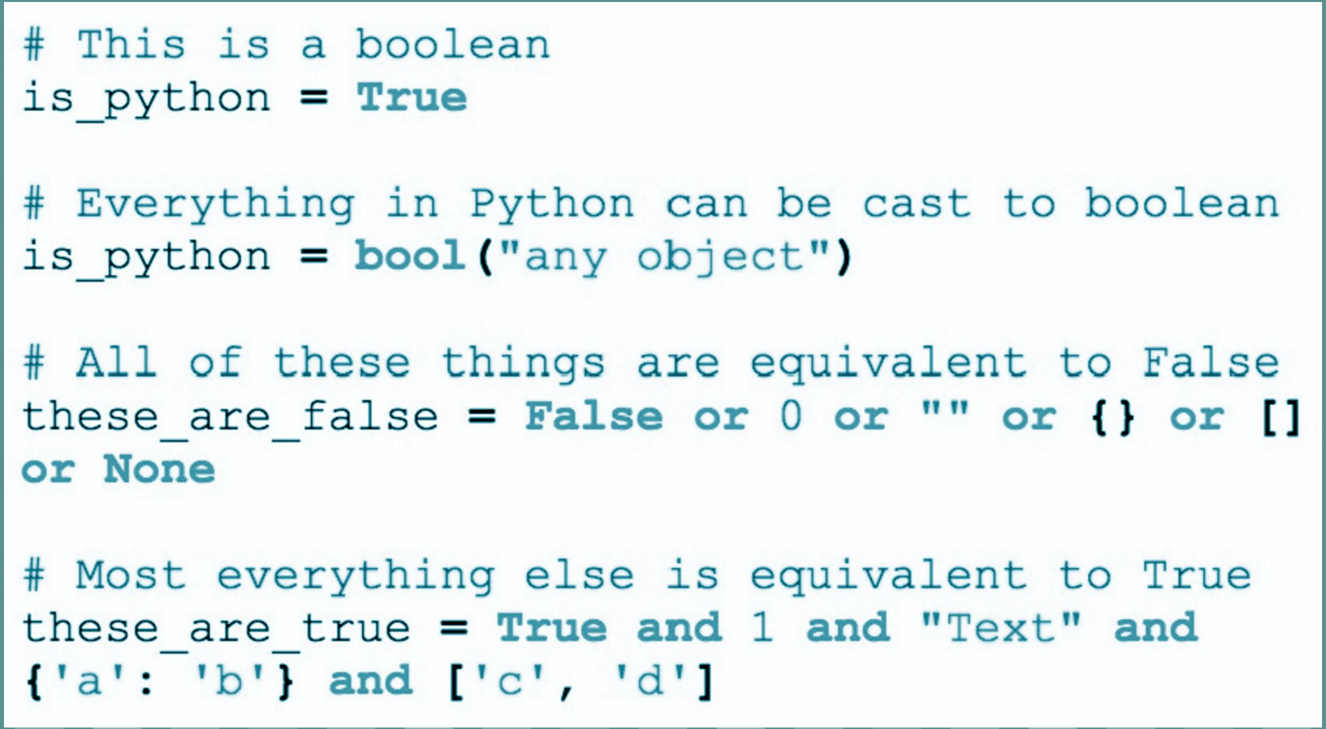
```python
# Lists can be heterogeneous
favorites = []

# Appending
favorites.append(42)

# Extending
favorites.extend(["Python", True])

# Equivalent to
favorites = [42, "Python", True]
```

Lists



```
numbers = [1, 2, 3, 4, 5]

len(numbers)
# 5

numbers[0]
# 1

numbers[0:2]
# [1, 2]

numbers[2:]
# [3, 4, 5]
```

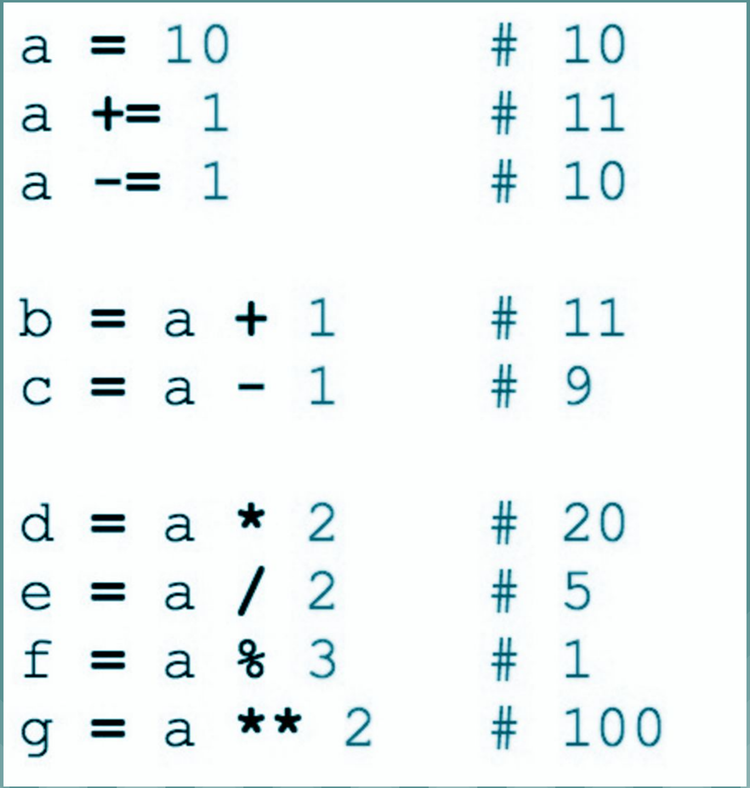
Booleans



```
# This is a boolean
is_python = True

# Everything in Python can be cast to boolean
is_python = bool("any object")

# All of these things are equivalent to False
these_are_false = False or 0 or "" or {} or []
or None

# Most everything else is equivalent to True
these_are_true = True and 1 and "Text" and
{'a': 'b'} and ['c', 'd']
```

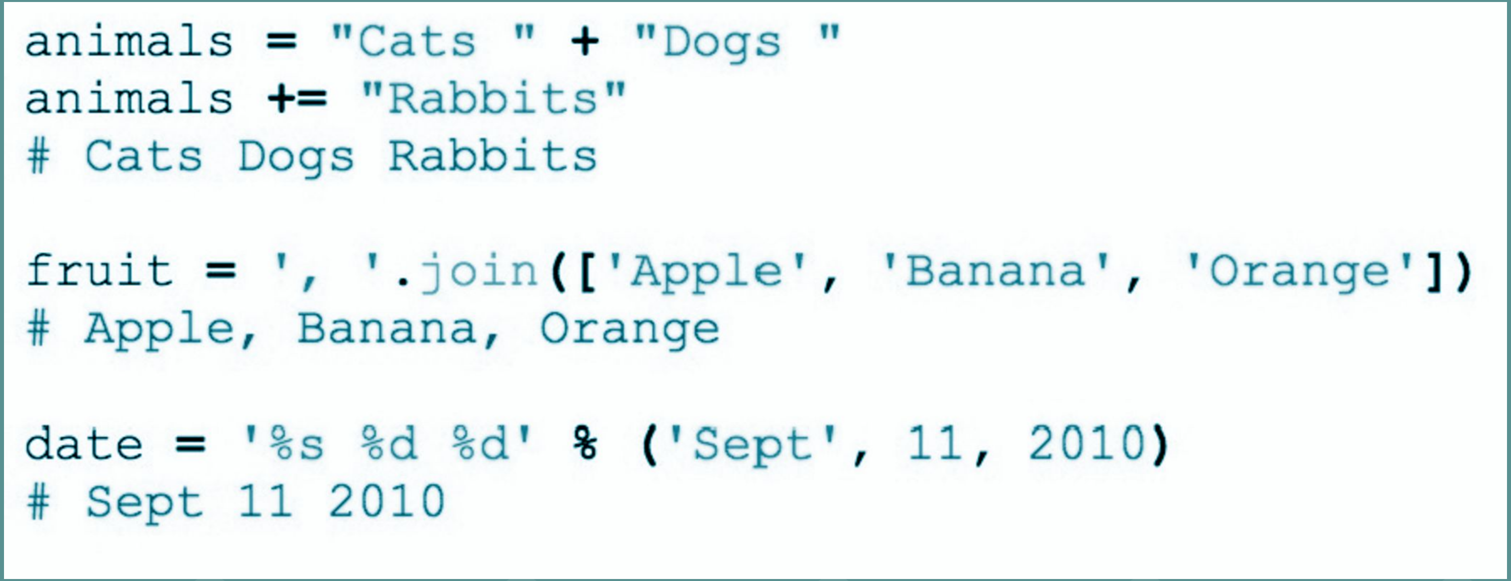
Operators

Arithmetic



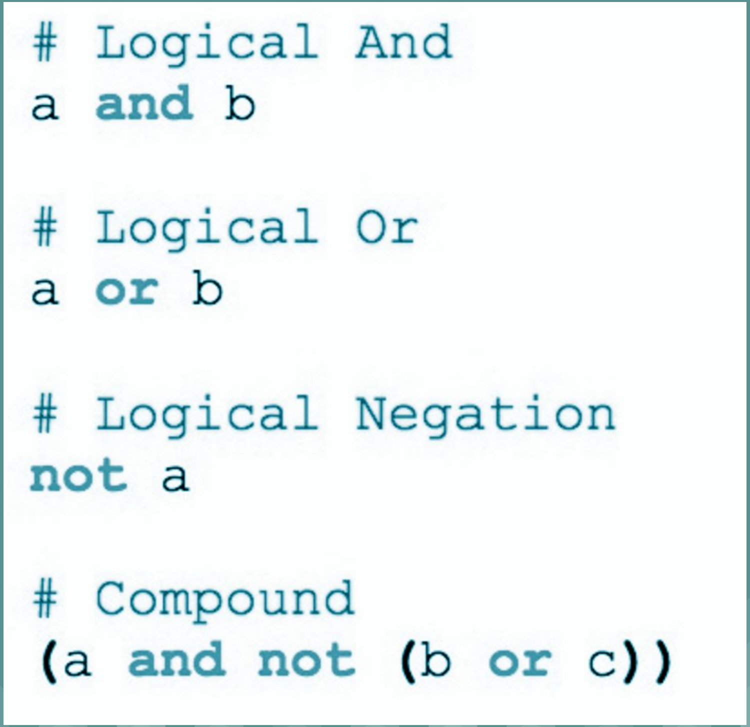
```
a = 10              # 10
a += 1              # 11
a -= 1              # 10

b = a + 1           # 11
c = a - 1           # 9

d = a * 2           # 20
e = a / 2           # 5
f = a % 3           # 1
g = a ** 2          # 100
```

String manipulation



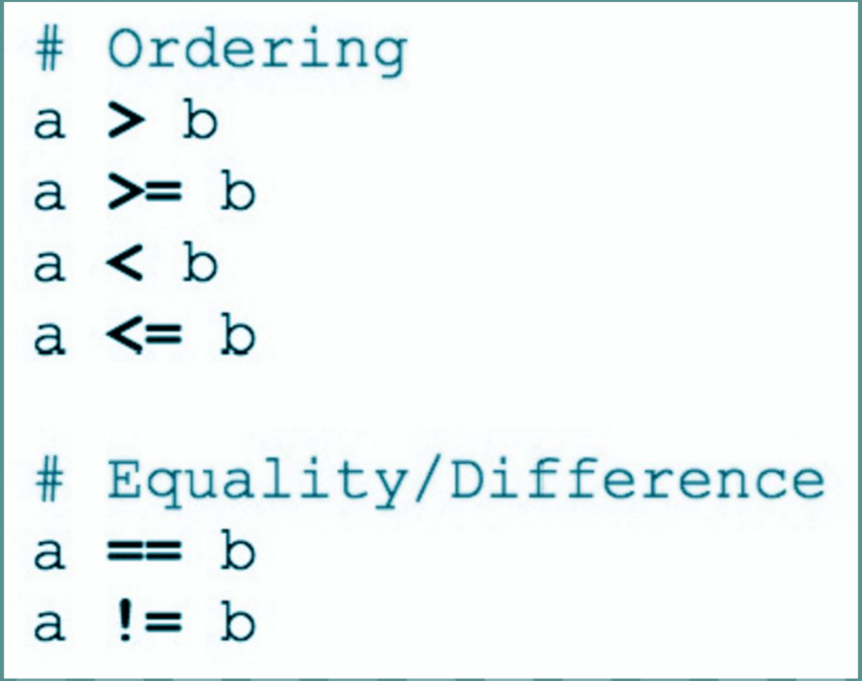
```
animals = "Cats " + "Dogs "
animals += "Rabbits"
# Cats Dogs Rabbits

fruit = ', '.join(['Apple', 'Banana', 'Orange'])
# Apple, Banana, Orange

date = '%s %d %d' % ('Sept', 11, 2010)
# Sept 11 2010
```

Logical Comparisn



```
# Logical And
a and b

# Logical Or
a or b

# Logical Negation
not a

# Compound
(a and not (b or c))
```

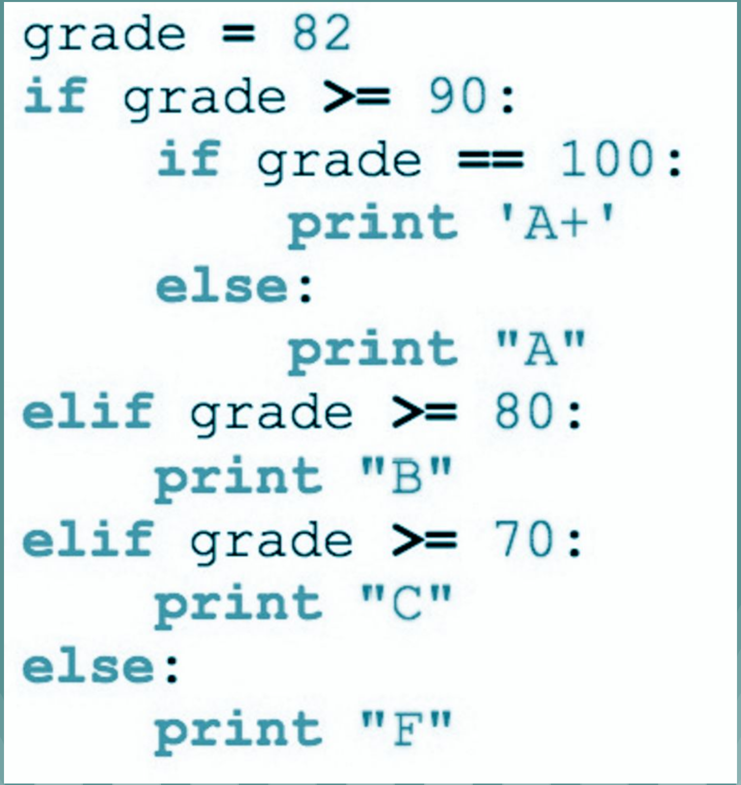
Arithmetic Comparison



```
# Ordering
a > b
a >= b
a < b
a <= b

# Equality/Difference
a == b
a != b
```

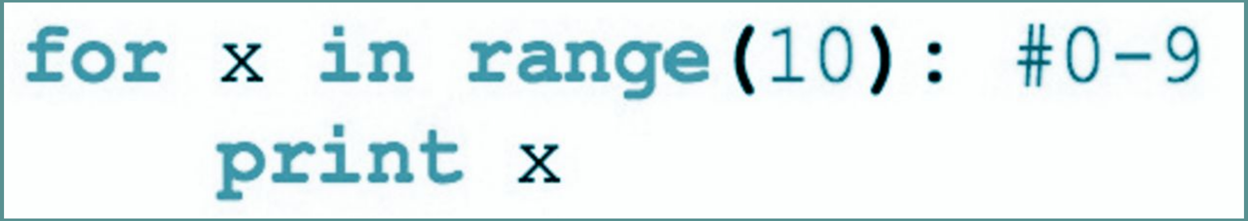
Control Flow

Conditionals

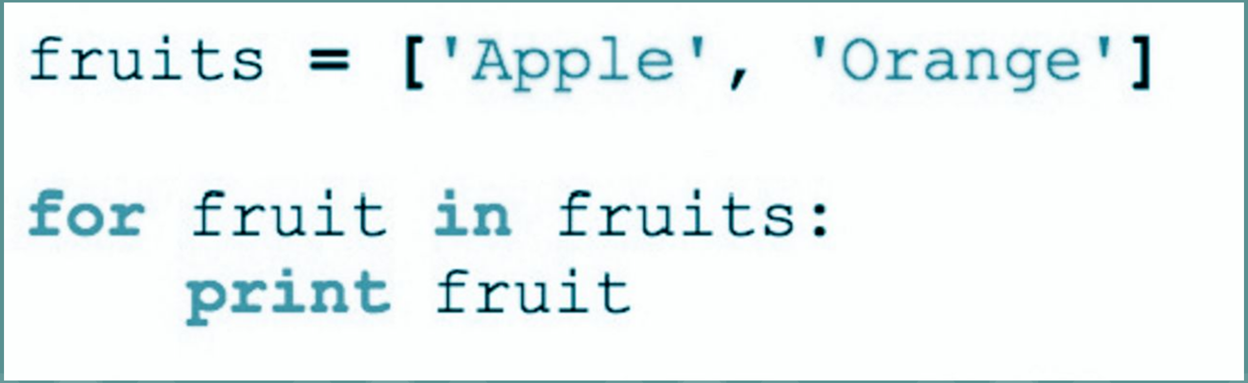


```
grade = 82
if grade >= 90:
    if grade == 100:
        print 'A+'
    else:
        print "A"
elif grade >= 80:
    print "B"
elif grade >= 70:
    print "C"
else:
    print "F"
```

For Loops

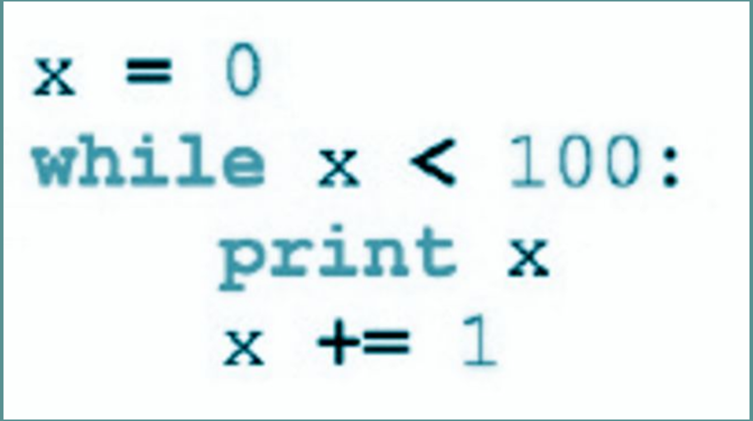


```
for x in range(10): #0-9
    print x
```



```
fruits = ['Apple', 'Orange']

for fruit in fruits:
    print fruit
```

While Loop



```
x = 0
while x < 100:
    print x
    x += 1
```

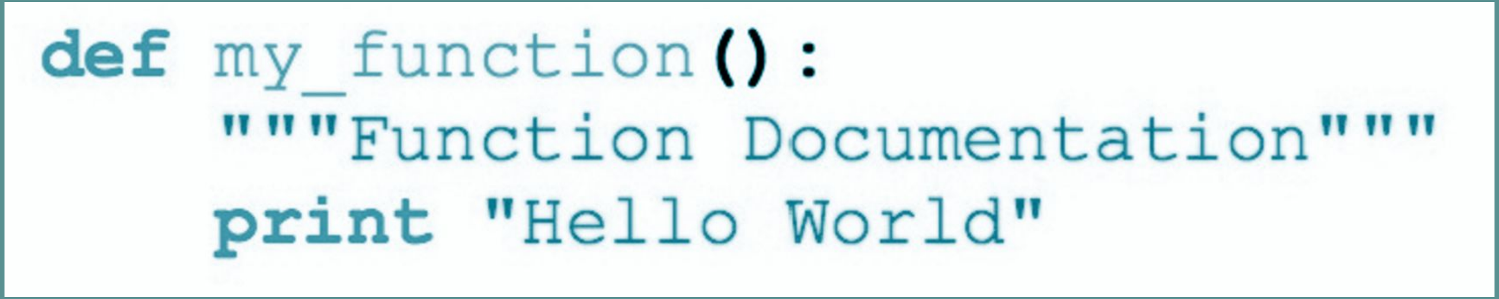
List Comprehensions

```
odds = [ x for x in range(50) if x % 2 ]
```

```
odds = []
for x in range(50):
    if x % 2:
        odds.append(x)
```

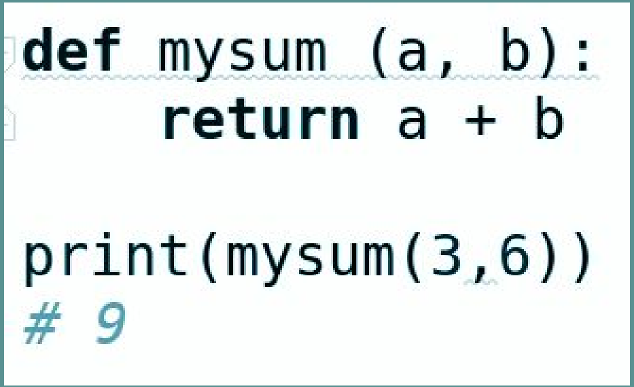
Functions

Basic Functions



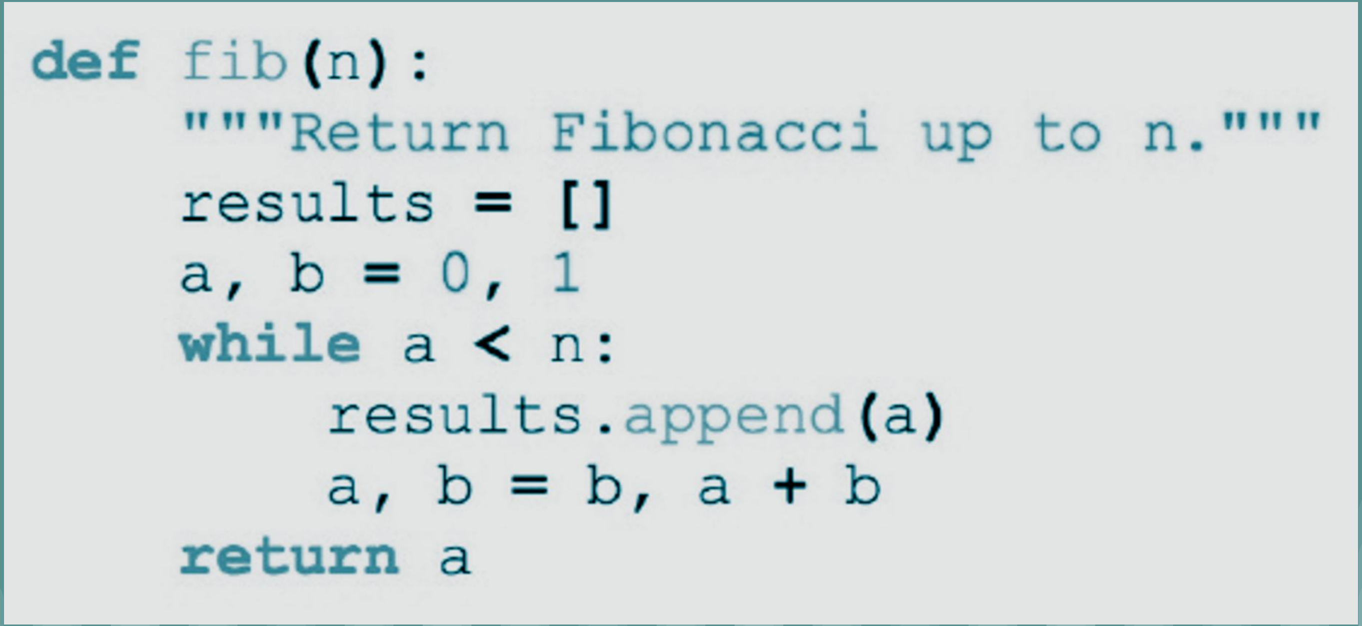
```
def my_function():
    """Function Documentation"""
    print "Hello World"
```

Sum Function



```
def mysum (a, b):
    return a + b

print(mysum(3,6))
# 9
```

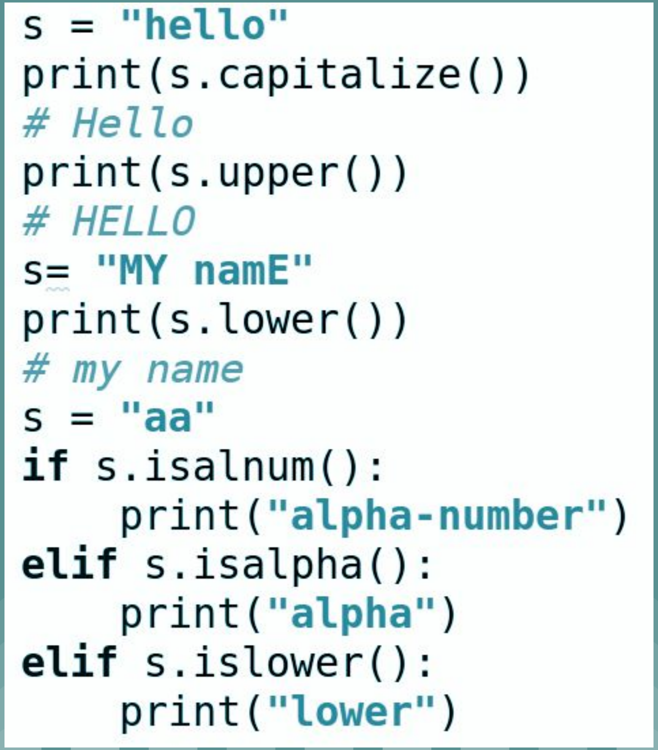
Fibonachi



```python
def fib(n):
    """Return Fibonacci up to n."""
    results = []
    a, b = 0, 1
    while a < n:
        results.append(a)
        a, b = b, a + b
    return a
```

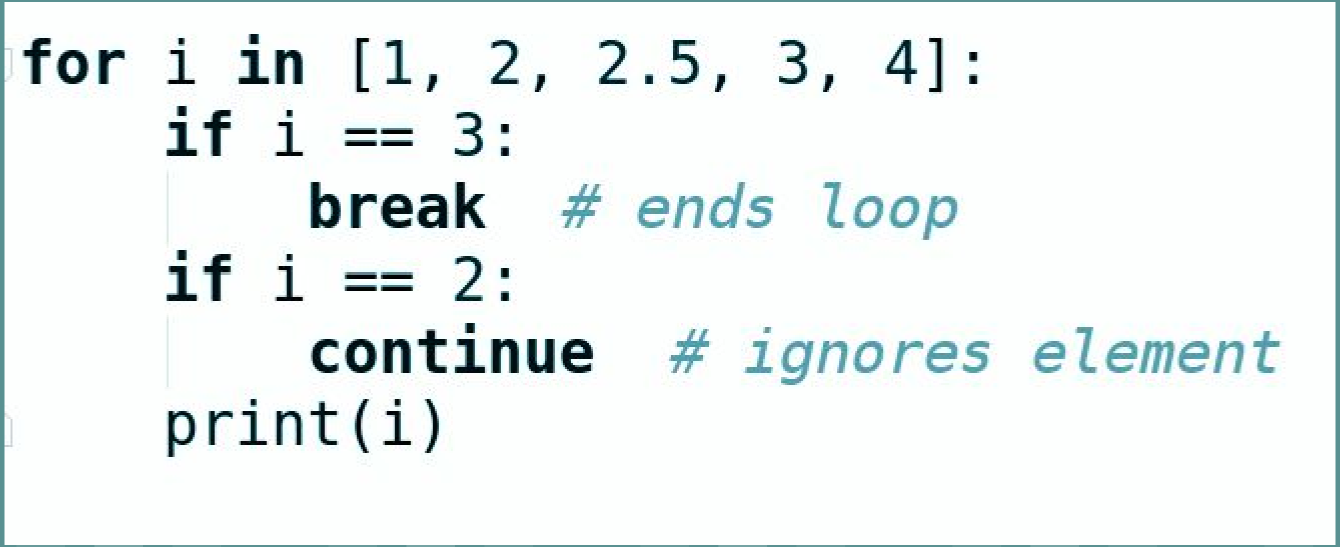
Session 2

More About String



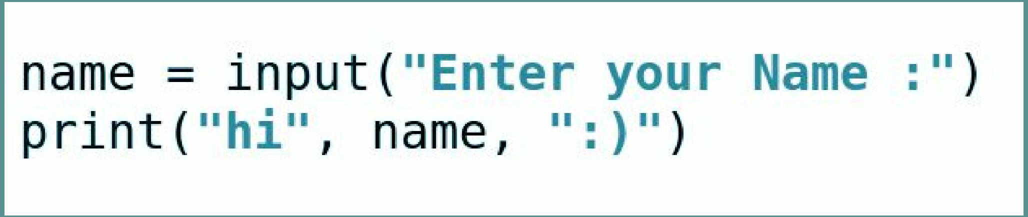
```python
s = "hello"
print(s.capitalize())
# Hello
print(s.upper())
# HELLO
s= "MY namE"
print(s.lower())
# my name
s = "aa"
if s.isalnum():
    print("alpha-number")
elif s.isalpha():
    print("alpha")
elif s.islower():
    print("lower")
```

Break - continue



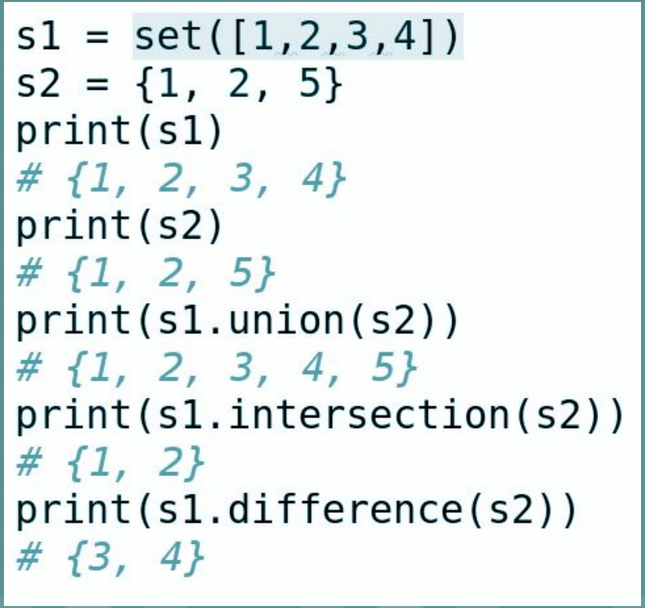
```python
for i in [1, 2, 2.5, 3, 4]:
    if i == 3:
        break  # ends loop
    if i == 2:
        continue  # ignores element
    print(i)
```

Input



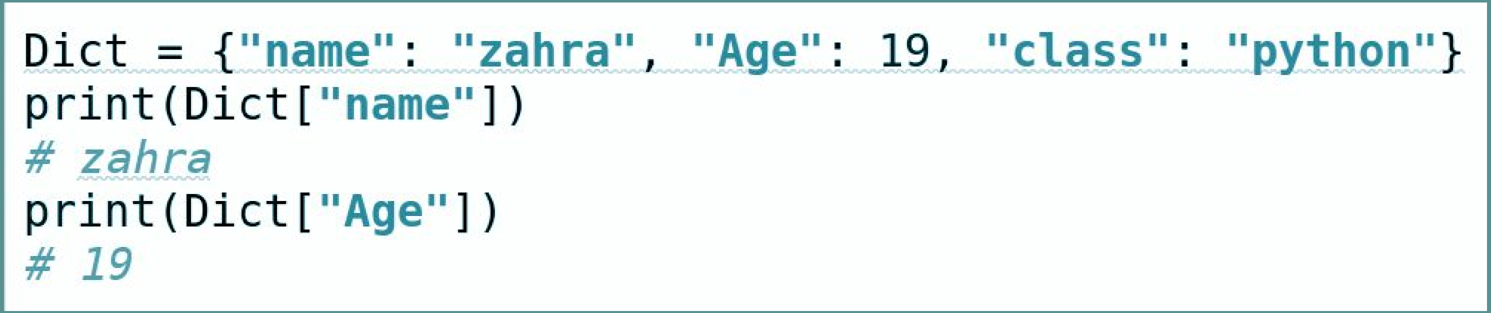
```
name = input("Enter your Name :")
print("hi", name, ":)")
```

Set



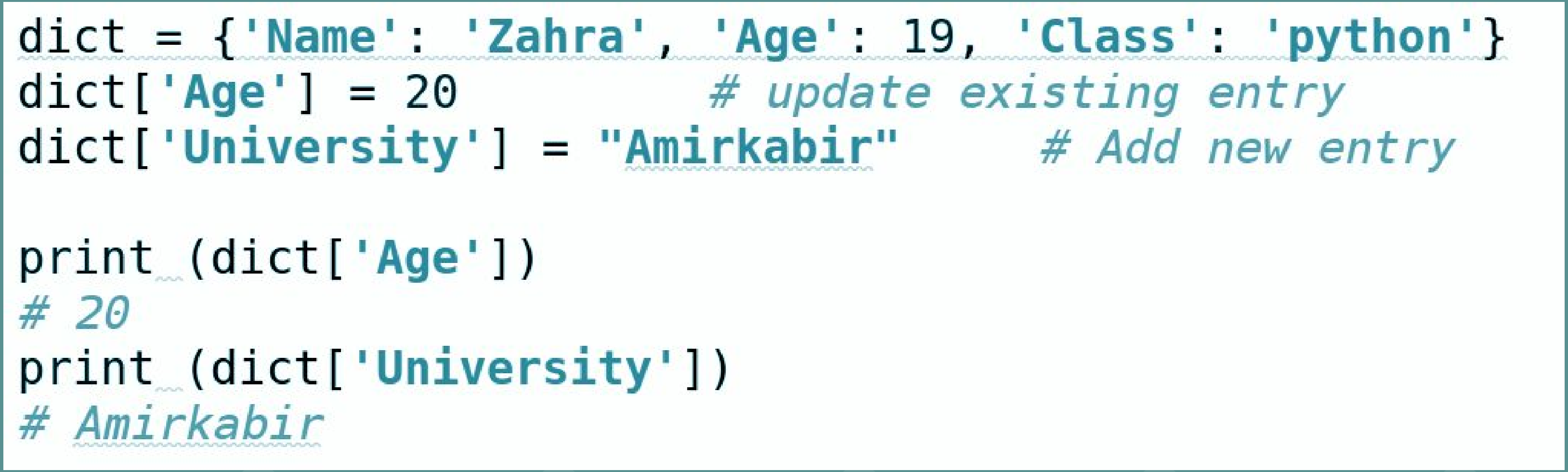
```
s1 = set([1,2,3,4])
s2 = {1, 2, 5}
print(s1)
# {1, 2, 3, 4}
print(s2)
# {1, 2, 5}
print(s1.union(s2))
# {1, 2, 3, 4, 5}
print(s1.intersection(s2))
# {1, 2}
print(s1.difference(s2))
# {3, 4}
```

Dictionaries



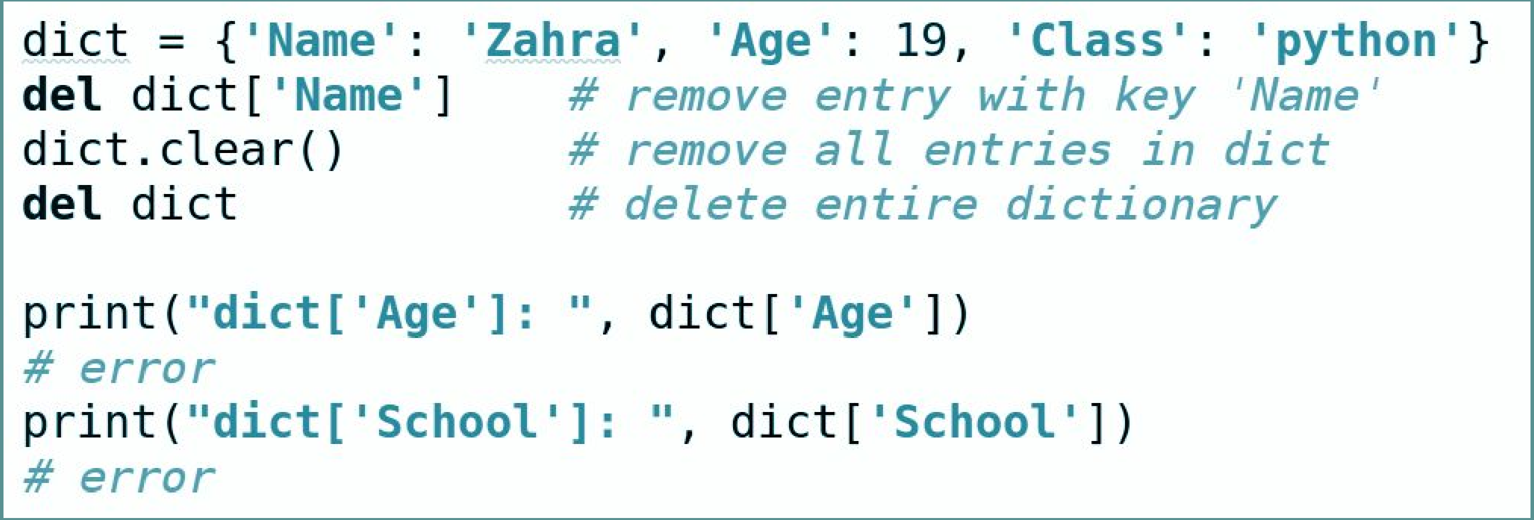
```
Dict = {"name": "zahra", "Age": 19, "class": "python"}
print(Dict["name"])
# zahra
print(Dict["Age"])
# 19
```

Updating Dictionaries



```
dict = {'Name': 'Zahra', 'Age': 19, 'Class': 'python'}
dict['Age'] = 20         # update existing entry
dict['University'] = "Amirkabir"     # Add new entry

print (dict['Age'])
# 20
print (dict['University'])
# Amirkabir
```

Delete from Dictionaries



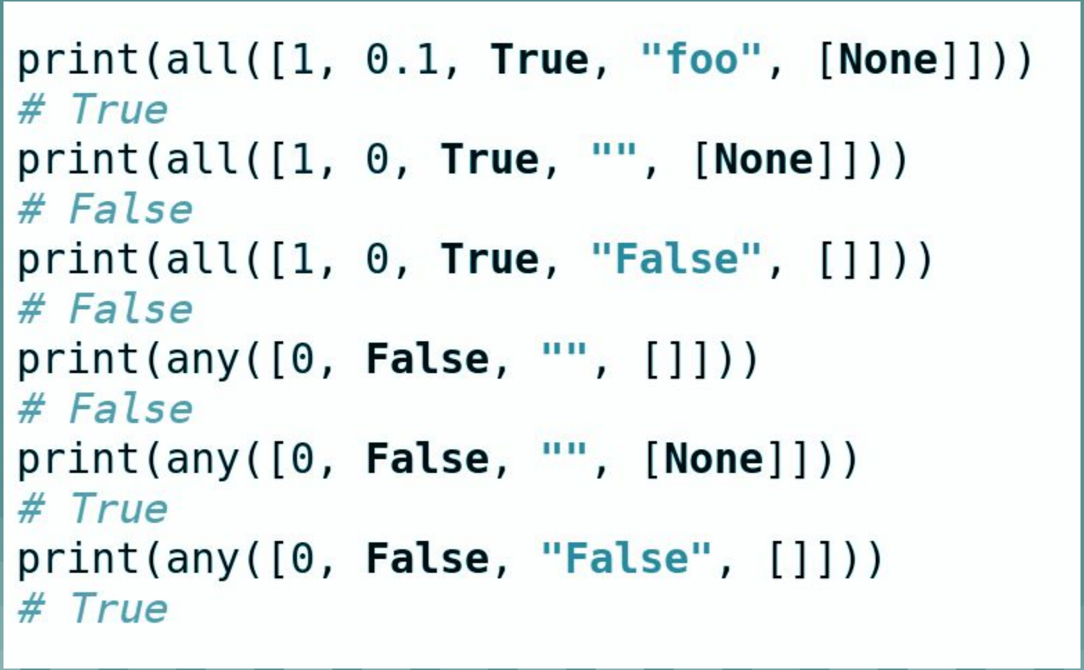
```
dict = {'Name': 'Zahra', 'Age': 19, 'Class': 'python'}
del dict['Name']     # remove entry with key 'Name'
dict.clear()         # remove all entries in dict
del dict             # delete entire dictionary

print("dict['Age']: ", dict['Age'])
# error
print("dict['School']: ", dict['School'])
# error
```

in , is

```python
l = [1, 2, 3, 4, None]
print(l is None)
# False
print(l[0] is None)
# False
print(l[4] is None)
# True
print(1 in l)
# True
print(None in l)
# True
```

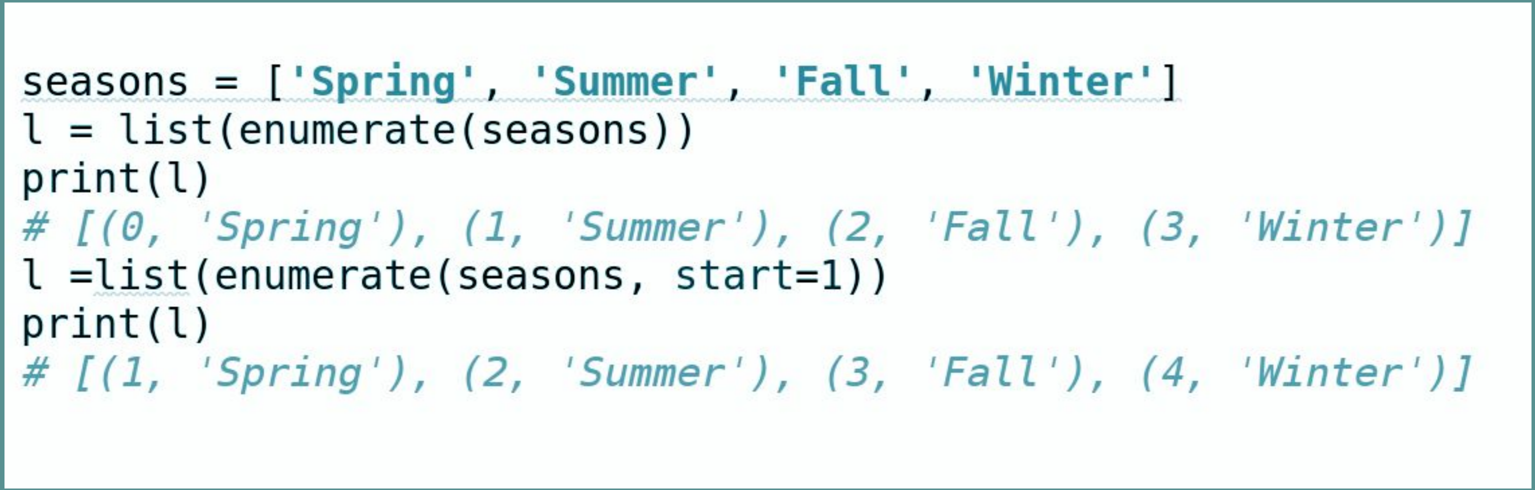
```python
l1 = None
l2 = []
l3 = [None]
print(l1 is None)
# True
print(l2 is None)
# False
print(l3 is None)
# False
print(l3[0] is None)
# True
```

All - Any



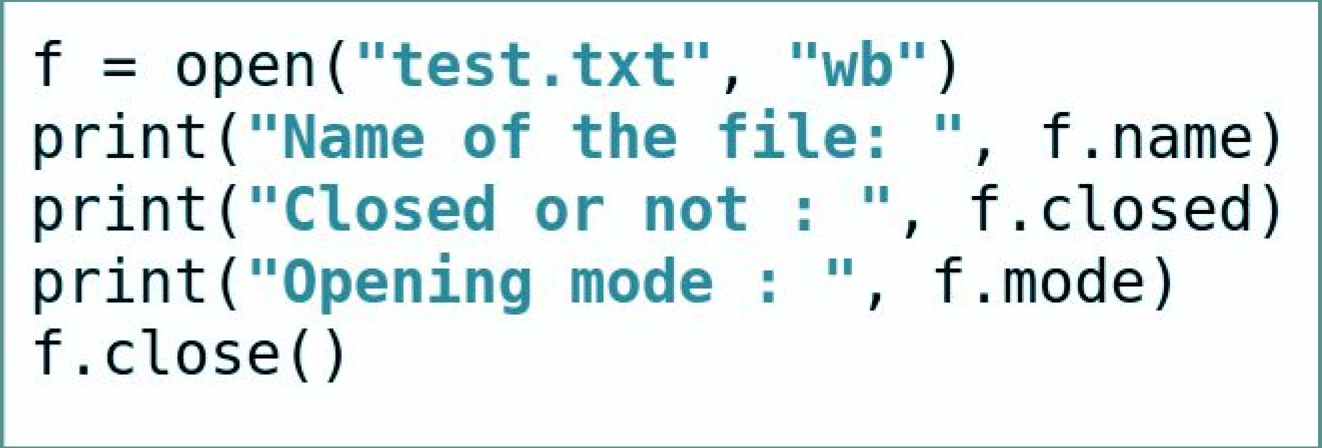
```python
print(all([1, 0.1, True, "foo", [None]]))
# True
print(all([1, 0, True, "", [None]]))
# False
print(all([1, 0, True, "False", []]))
# False
print(any([0, False, "", []]))
# False
print(any([0, False, "", [None]]))
# True
print(any([0, False, "False", []]))
# True
```

Enumerate



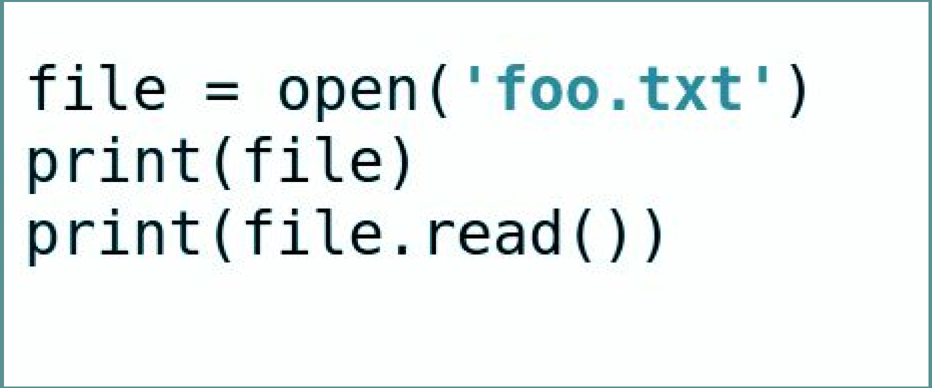
```python
seasons = ['Spring', 'Summer', 'Fall', 'Winter']
l = list(enumerate(seasons))
print(l)
# [(0, 'Spring'), (1, 'Summer'), (2, 'Fall'), (3, 'Winter')]
l =list(enumerate(seasons, start=1))
print(l)
# [(1, 'Spring'), (2, 'Summer'), (3, 'Fall'), (4, 'Winter')]
```

Files



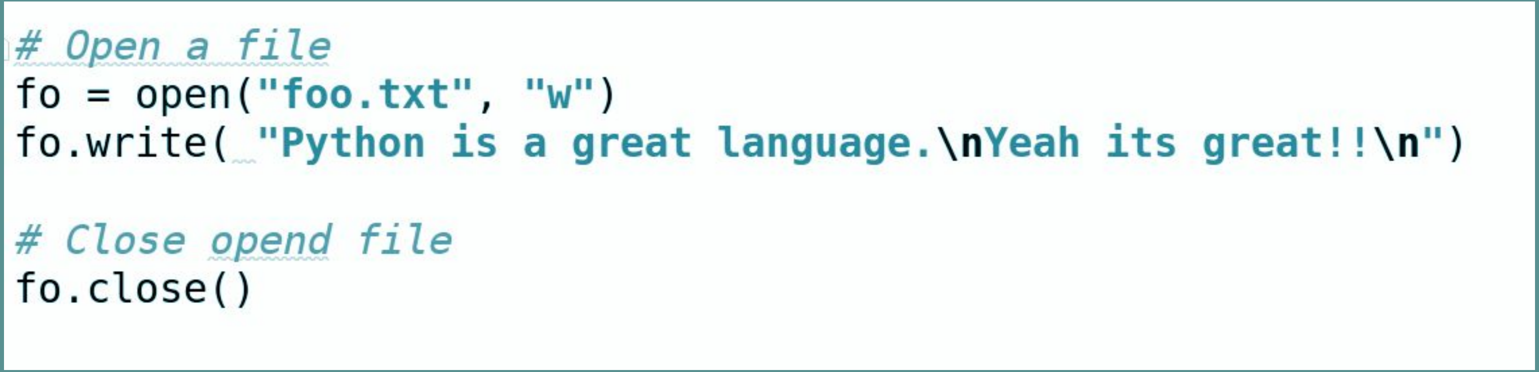
```
f = open("test.txt", "wb")
print("Name of the file: ", f.name)
print("Closed or not : ", f.closed)
print("Opening mode : ", f.mode)
f.close()
```

Reading Files



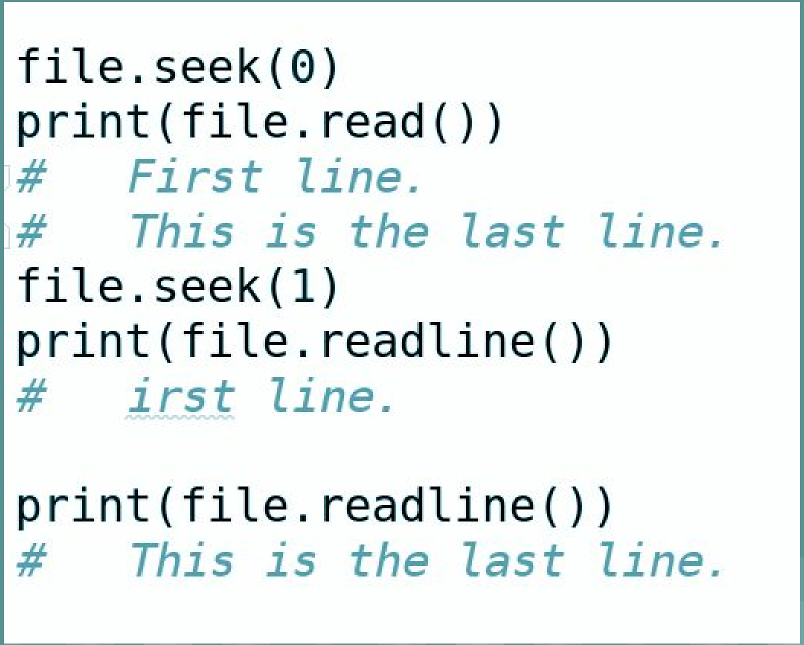
```
file = open('foo.txt')
print(file)
print(file.read())
```

Writing Files



```
# Open a file
fo = open("foo.txt", "w")
fo.write( "Python is a great language.\nYeah its great!!\n")

# Close opend file
fo.close()
```

File seek



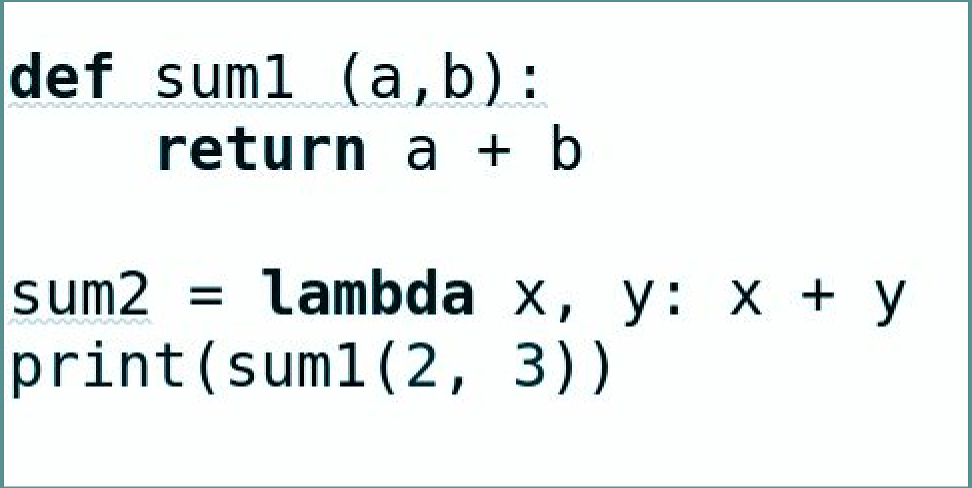
```
file.seek(0)
print(file.read())
#   First line.
#   This is the last line.
file.seek(1)
print(file.readline())
#   irst line.

print(file.readline())
#   This is the last line.
```

Complex

```
a = complex(1, 2)
b = complex(1, 2) + complex(3, 4)
c = complex(1, 2) + 2
print(a)
#   (1+2j)
print(b)
#   (4+6j)
print(c)
#   (3+2j)
```

Session 3

lambda



```
def sum1 (a,b):
    return a + b

sum2 = lambda x, y: x + y
print(sum1(2, 3))
```

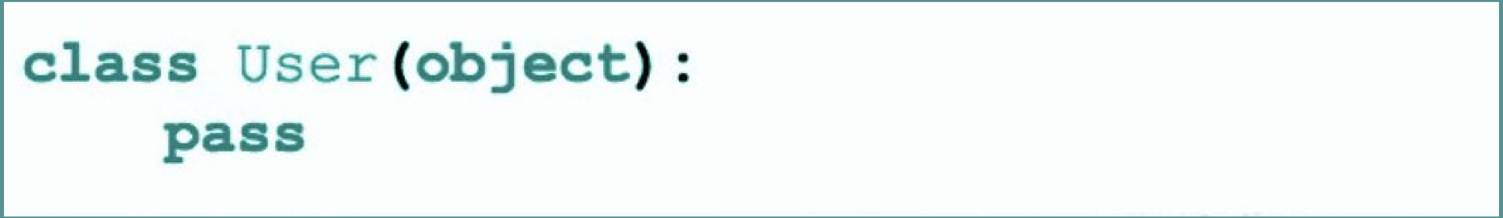
Object oriented

- Class
- Class variable
- Instance
- Instance variable
- Data member

Object oriented

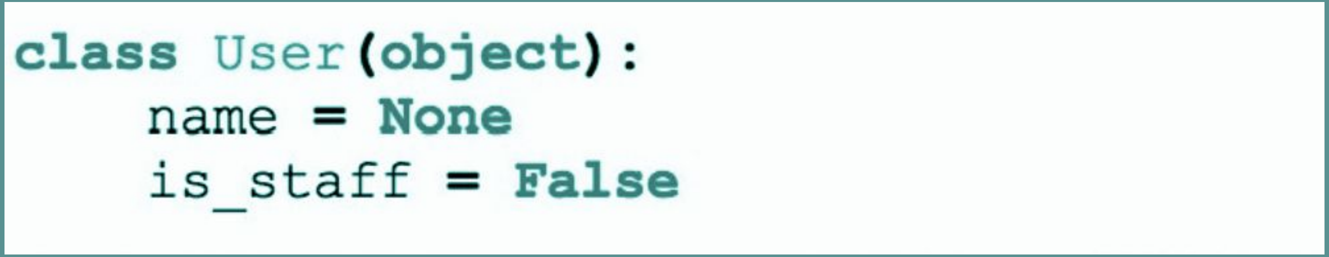
- Method
- Instantiation
- Function overloading
- Operator overloading
- Object

Class declaration



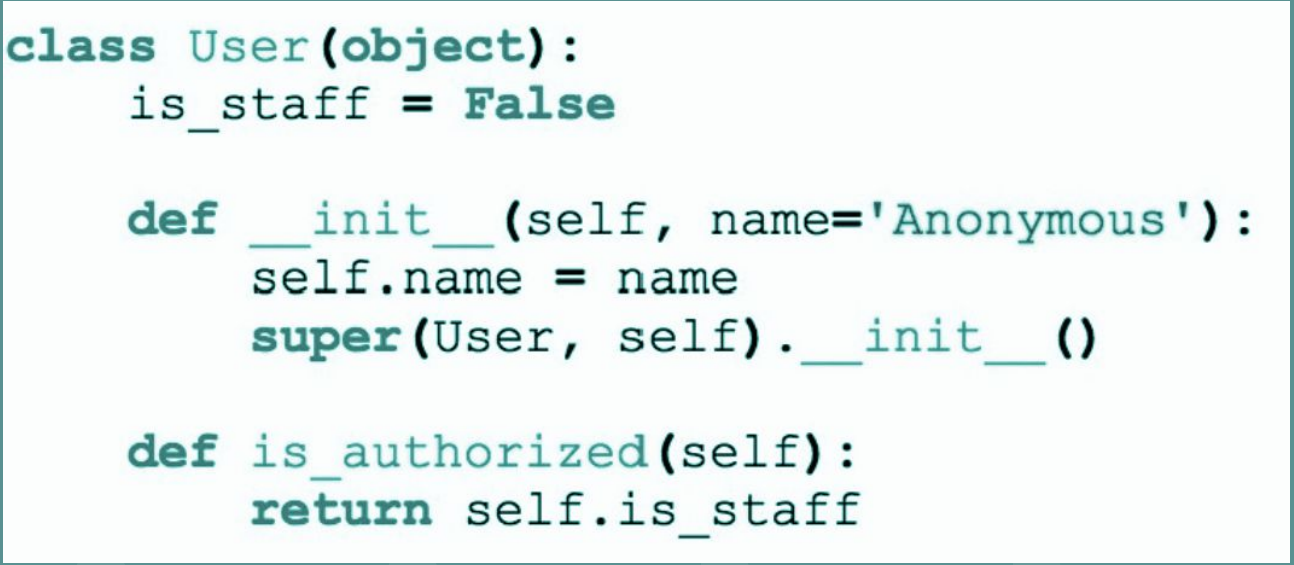
```
class User(object):
    pass
```

Class attributes



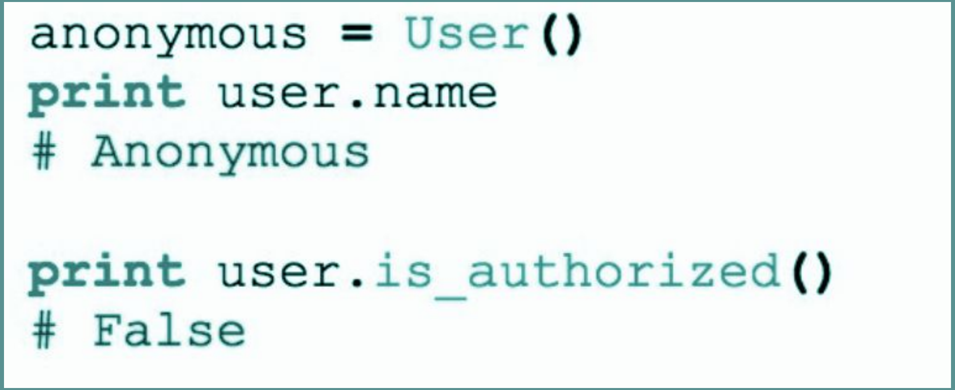
```
class User(object):
    name = None
    is_staff = False
```

Class Methods



```
class User(object):
    is_staff = False

    def __init__(self, name='Anonymous'):
        self.name = name
        super(User, self).__init__()

    def is_authorized(self):
        return self.is_staff
```

Class Instantiation & Attribute Access



```
anonymous = User()
print user.name
# Anonymous

print user.is_authorized()
# False
```

Class Inheritance

```
class SuperUser(User):
    is_staff = True
```

```
nowell = SuperUser('Nowell Strite')
print user.name
# Nowell Strite
print user.is_authorized()
# True
```

Class Inheritance

```
class SuperUser(User):
    is_staff = True
```

```
nowell = SuperUser('Nowell Strite')
print user.name
# Nowell Strite
print user.is_authorized()
# True
```

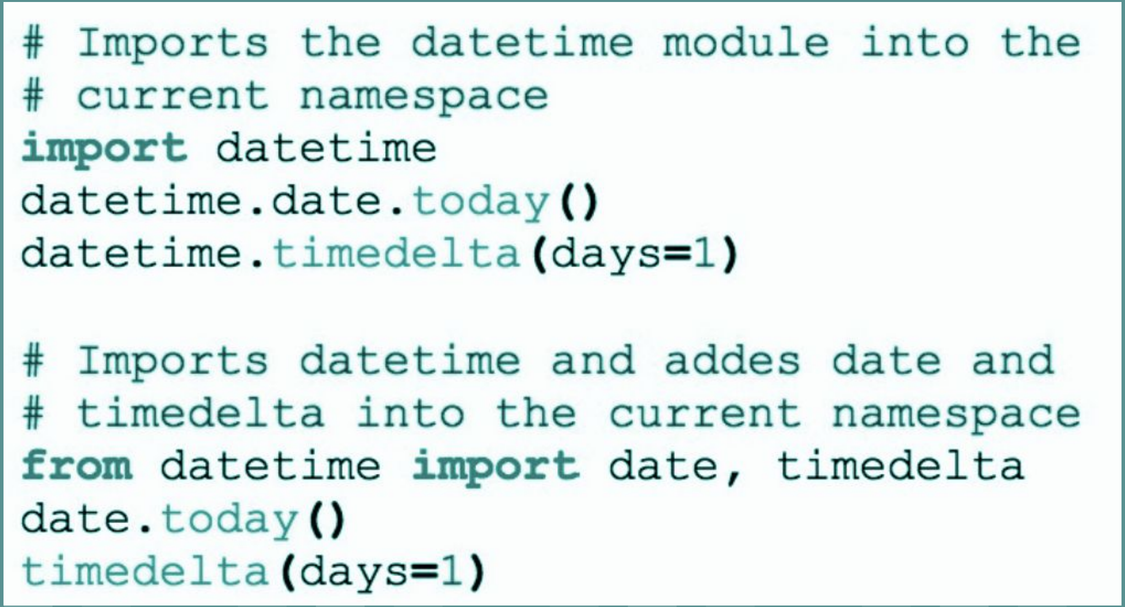
hints

- No interface
- No real private attributes/functions
- Private methods start (but not end) with double underscore
- Special method start and end with double underscore
  - __init__, __doc__, __str__, __cmp__

imports

- Code isolation
- Code reuse

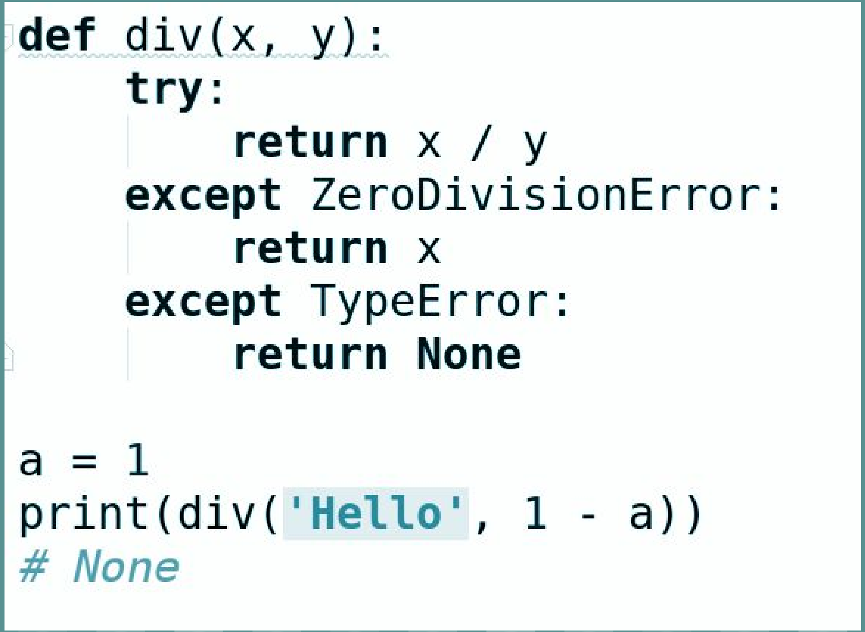
imports



```
# Imports the datetime module into the
# current namespace
import datetime
datetime.date.today()
datetime.timedelta(days=1)

# Imports datetime and addes date and
# timedelta into the current namespace
from datetime import date, timedelta
date.today()
timedelta(days=1)
```

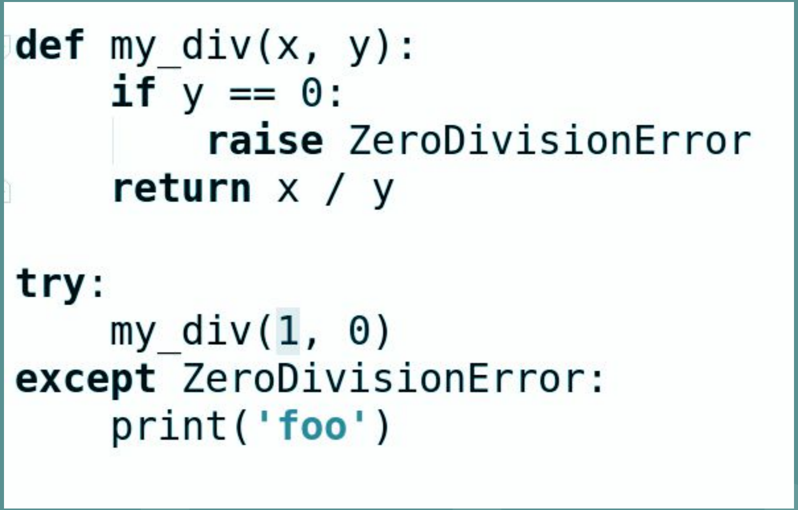
Session 4

Errors and Exceptions



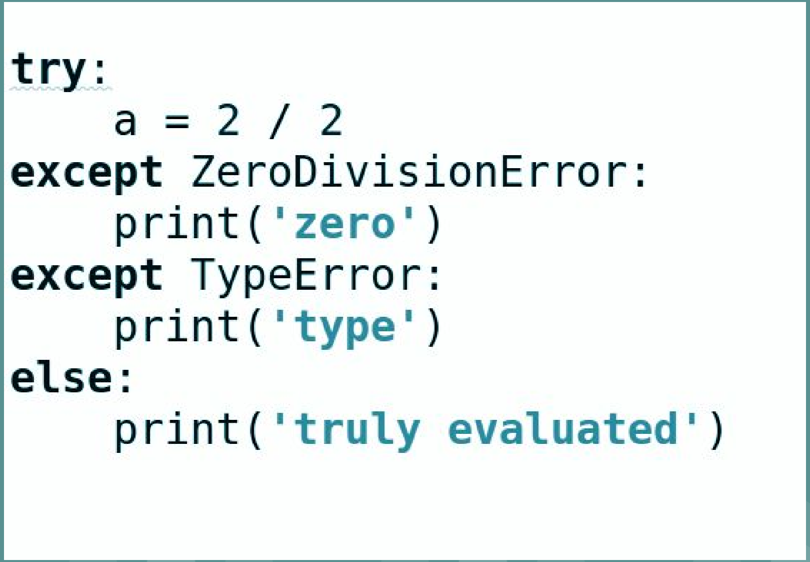
```python
def div(x, y):
    try:
        return x / y
    except ZeroDivisionError:
        return x
    except TypeError:
        return None

a = 1
print(div('Hello', 1 - a))
# None
```

Errors and Exceptions



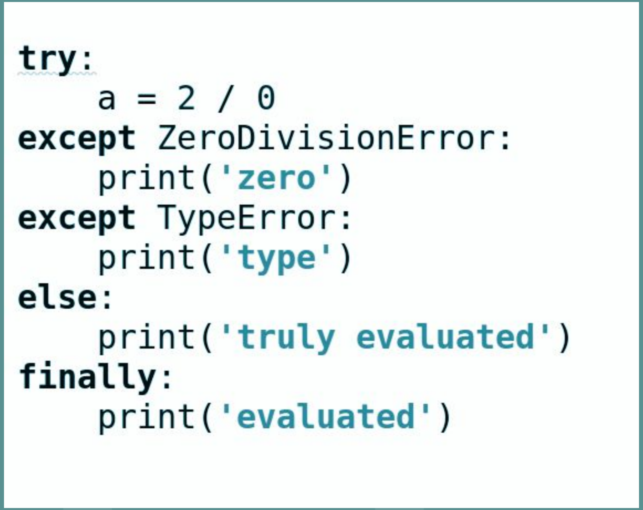
```python
def my_div(x, y):
    if y == 0:
        raise ZeroDivisionError
    return x / y

try:
    my_div(1, 0)
except ZeroDivisionError:
    print('foo')
```

Errors and Exceptions



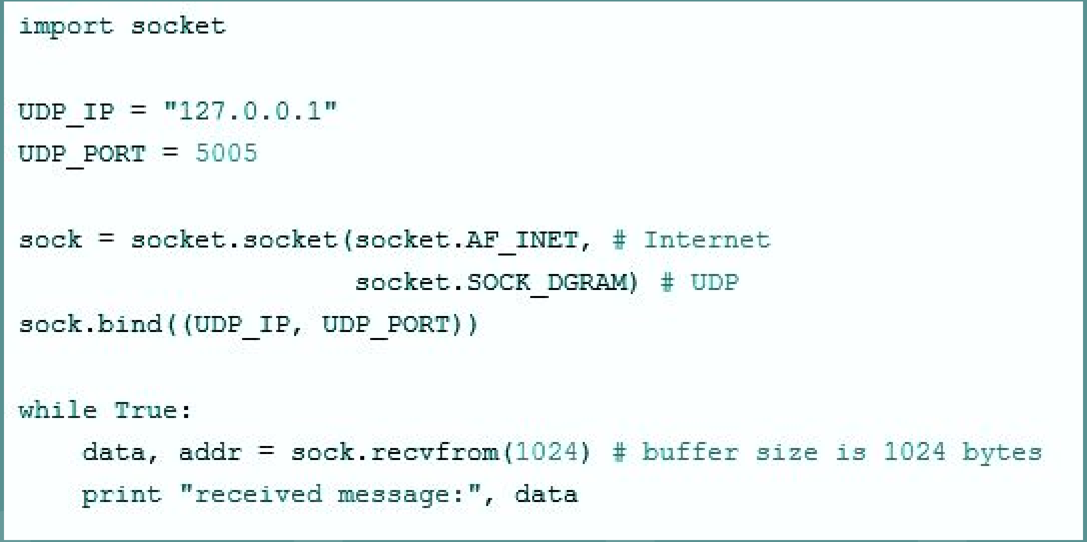
```python
try:
    a = 2 / 2
except ZeroDivisionError:
    print('zero')
except TypeError:
    print('type')
else:
    print('truly evaluated')
```

Errors and Exceptions



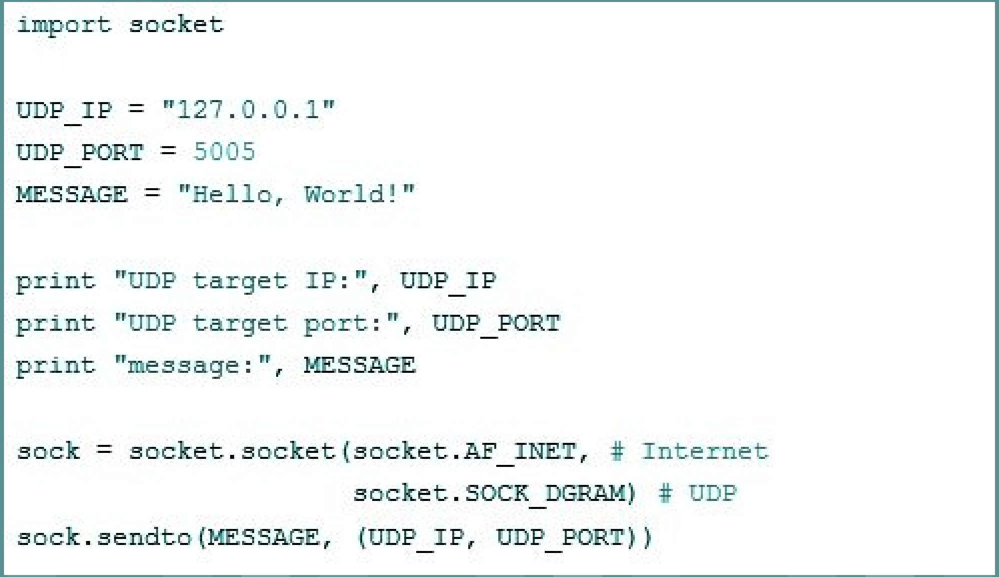
```
try:
    a = 2 / 0
except ZeroDivisionError:
    print('zero')
except TypeError:
    print('type')
else:
    print('truly evaluated')
finally:
    print('evaluated')
```

UDP Server



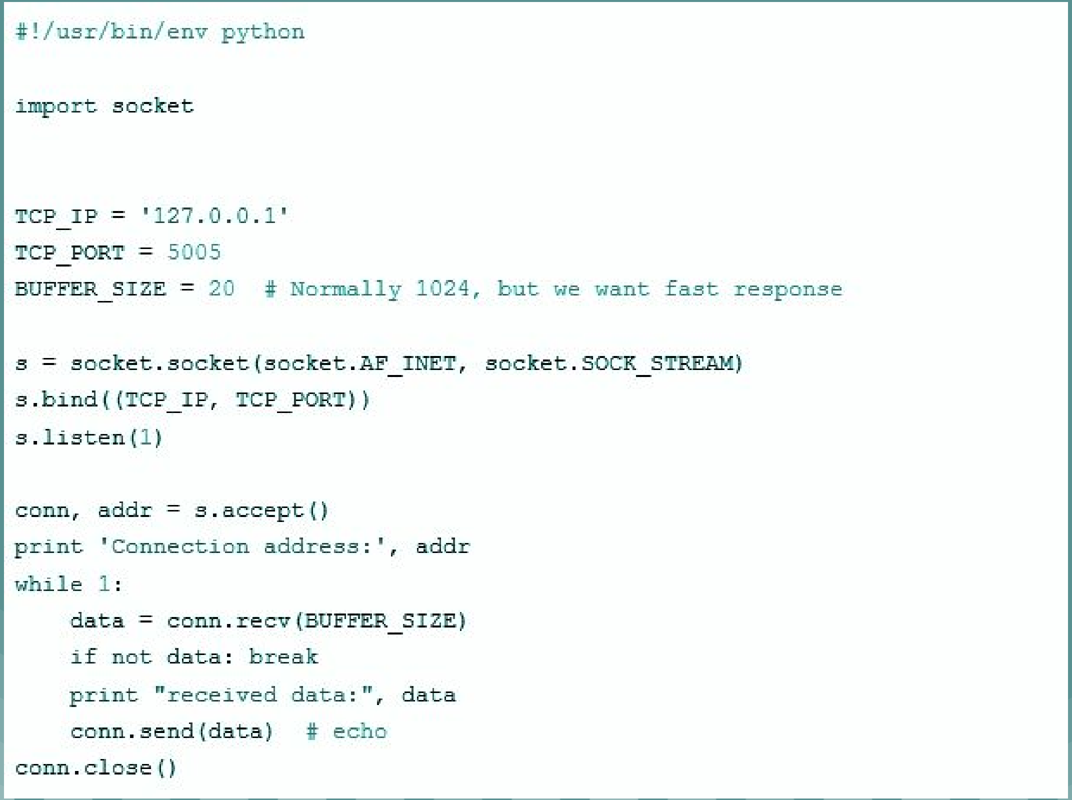
```
import socket

UDP_IP = "127.0.0.1"
UDP_PORT = 5005

sock = socket.socket(socket.AF_INET, # Internet
                     socket.SOCK_DGRAM) # UDP
sock.bind((UDP_IP, UDP_PORT))

while True:
    data, addr = sock.recvfrom(1024) # buffer size is 1024 bytes
    print "received message:", data
```

UDP client



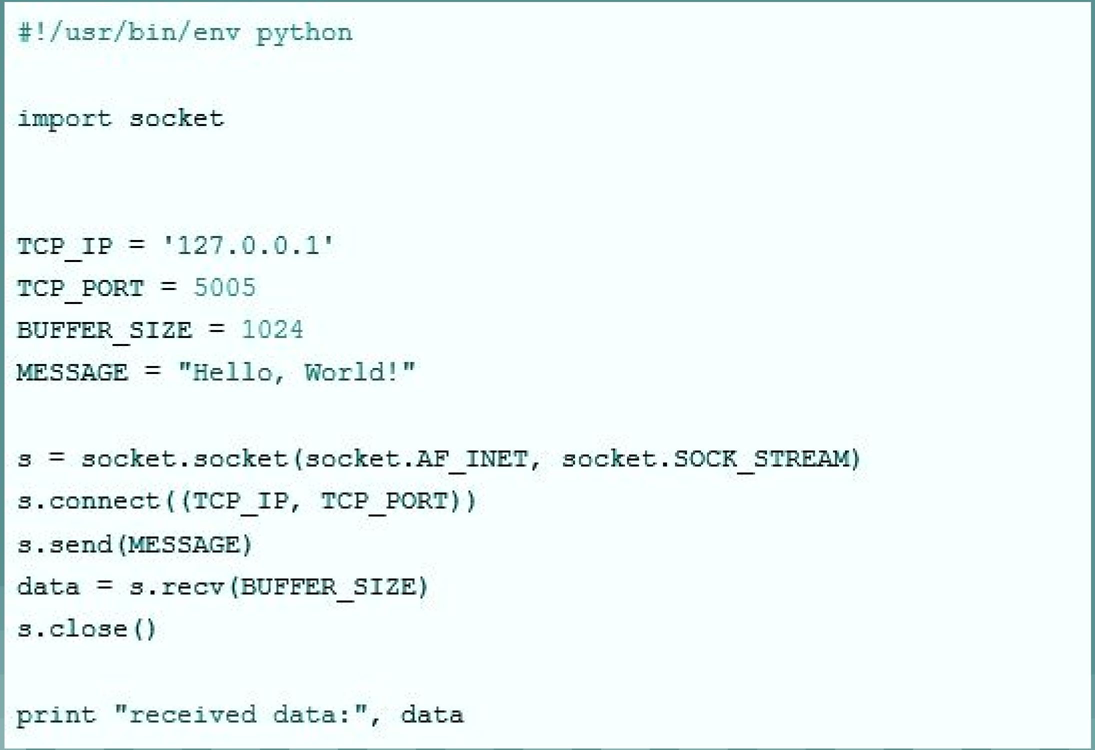
```
import socket

UDP_IP = "127.0.0.1"
UDP_PORT = 5005
MESSAGE = "Hello, World!"

print "UDP target IP:", UDP_IP
print "UDP target port:", UDP_PORT
print "message:", MESSAGE

sock = socket.socket(socket.AF_INET, # Internet
                     socket.SOCK_DGRAM) # UDP
sock.sendto(MESSAGE, (UDP_IP, UDP_PORT))
```

TCP server



```python
#!/usr/bin/env python

import socket


TCP_IP = '127.0.0.1'
TCP_PORT = 5005
BUFFER_SIZE = 20  # Normally 1024, but we want fast response

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((TCP_IP, TCP_PORT))
s.listen(1)

conn, addr = s.accept()
print 'Connection address:', addr
while 1:
    data = conn.recv(BUFFER_SIZE)
    if not data: break
    print "received data:", data
    conn.send(data)  # echo
conn.close()
```

TCP client



```python
#!/usr/bin/env python

import socket



TCP_IP = '127.0.0.1'
TCP_PORT = 5005
BUFFER_SIZE = 1024
MESSAGE = "Hello, World!"

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((TCP_IP, TCP_PORT))
s.send(MESSAGE)
data = s.recv(BUFFER_SIZE)
s.close()

print "received data:", data
```