Apprameyah Technologies Pvt. Ltd.

Setup

- Please make sure you have python installed
- Windows Download: https://www.python.org/ftp/python/3.4.3/python-3.4.3.amd64.msi
- Mac download: https://www.python.org/ftp/python/3.4.3/python-3.4.3-macosx10.6.pkg
- On command line "python3 -V" for mac and "python -V" for windows
- Development tool: eclipse. Needs JDK 7

Aprameyah
Technologies Pvt. Ltd.

Python

Aprameyah Technologies Pvt. Ltd.

About Me

- Maruthi R Janardhan
  - Been with IBM, ANZ, HCL-HP,
  - I run my own startup Leviossa - we are into development services and training.
  - Total 16 years programming C, C++, Python, Java, Javascript, Ruby, Perl, PHP, Grails, etc
  - I hold two pending Indian patents in telephony area

Some French

- you - vous
- a - un
- student - étudiant
- are - sont, êtes, sommes
- they - ils
- we - nous
- she - elle
- he - il
- life - vie
- of - de
- the - la

Build Some Sentences

- We are students of life
- Nous sommes des étudiants de la vie
- you are a student of life
- vous êtes un étudiant de la vie
- Learning a language without practice is like learning to speak using a dictionary

About Python

- Supports Object Oriented programming
- Supports Imperative Programming (Scripting as a sequence of commands - subroutines)
- Supports Functional Programming (Stateless behaviour of functions - functions)
- Dynamically Typed Language (Type identified only at runtime)
- Automatic Memory Management

Aprameyah
Technologies Pvt. Ltd.

About Python

- Interpreter Available for almost all platforms
- Possible to bundle a program as an executable that can run without an interpreter (pyinstaller)
- CPython is the reference implementation and most widely used
    - There are others - Ironpython (Runs on CLR), JPython (Runs on JVM)
- Guido Van Rossum was reading the scripts of the Monty Python TV show while designing this new language and he named his new language “Python” as a joke

Aprameyah Technologies Pvt. Ltd.

Check Python Installation

- Windows Download: https://www.python.org/ftp/python/3.4.3/python-3.4.3.amd64.msi
- Mac download: https://www.python.org/ftp/python/3.4.3/python-3.4.3-macosx10.6.pkg
- On command line “python3 -V” for mac and “python -V” for windows
- Development tool: eclipse. Needs JDK 7
- Setup eclipse plugin from: http://pydev.sf.net/updates/
- or google for “pydev eclipse market place” and drag drop

Aprameyah
Technologies Pvt. Ltd.

Whats New In Python3

- print is a function needing regular braces unlike it being a keyword in python 2
- All strings are now unicode and dont need “u” prefix like u’Hello’
- Byte strings need a prefix b in python3
- raw_input() is removed and input() now returns strings without evaluating anything
- Dividing integers returns a float now instead of ints

Aprameyah Technologies Pvt. Ltd.

Hello World

- Launch any text editor and create a new file “hello.py” with this content

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

- On command line run it as: python hello.py (or python3 hello.py for mac)
- Print is a Built-in function (BIF)
- Script lines are not checked till execution time

Apprameyah Technologies Pvt. Ltd.

Python in a File

- Python files have a .py extension
- Code saved in python script can be executed with the interpreter like this: python3 myfile.py
- Alternatively on unix put this at the top of the file, #!/usr/bin/python
  - Execute with ./myfile.py after granting execute permissions

Blocks (Suites)

- There are no braces or BEGIN-END kind of structures.
    - Blocks are identified by indentation and statements needing blocks end with colon
    - The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. Both blocks in this example are fine:

```
if True:
    print "True"
else:
  print "False"
```

Aprameyah Technologies Pvt. Ltd.

Console I/O

- name = input(“Enter your name: “)
  - Reads text from console.
  - Notice variables need no declaration or a type
  - Input is read as strings. To convert to int, we use int() function: x = int(y)
- Conditionals dont need brackets like functions. All function invocations like print need brackets

```
if x == 5 :
        print(“x is {}”.format(x))
elif x<0 :
        print(“x is negative”)
    print (“x is 5”)
```

Ex 1: Accept name and age from user and print one of the below statements depending on the users age (65+, 30-65, <30)

```
"Hello Krishna you are never too old to learn python at 65"
"Hello Krishna how did you get by for 30 years without python!"
"Hello krishna today is a great day to learn python"
```

Aprameyah
Technologies Pvt. Ltd.

Python Datatypes

- 5 Standard Datatypes
  - Numbers (int, long, float, complex)
    - Long literals have L suffix (in 2.x.. not needed in 3.x - everything is long)
    - Complex type example: 9.322e-36j
  - String
  - List
  - Tuple
  - Dictionary
- Python accepts single ('), double (") and triple (''' or """) quotes to denote string literals, as long as the same type of quote starts and ends the string.

Aprameyah
Technologies Pvt. Ltd.

Operators

Operator	Meaning
+ - * / %	Arithmatic operators
**	Exponent
//	Floor Division (9//2 = 4 and not 4.5)
!=, <>, ==, >, <, >=, <=	Comparison Operators
+=, -=, =, *=, /=, %=,	Assignment Operators
&, |, ~, ^, <<, >>	Bitwise operators
and, or, not	Logical Operators
in, not in	Membership operators for lists, tuples, strings
is, is not	Identity operators for reference comparison

Ex2: Using the following formula, accept user input and compute the loan eligibility of an user: P= [{(1+R)N -1}EMI]/[R*(1+R)N]

Aprameyah Technologies Pvt. Ltd.

Strings

- String is assigned and accessed as below

```
print (str)          # Prints complete string
print (str[0])       # Prints first character of the string
print (str[2:5])     # Prints characters starting from 3rd to 5th
print (str[2:])      # Prints string starting from 3rd character
```

- There are a lot of methods on strings to format/ process strings
    - There is almost a mini language to format strings
- String literals can use single or double quotes

Aprameyah Technologies Pvt. Ltd.

String Special Operators (Many work on lists too)

Operator	Meaning
+	Concatenation
*	Repetition - Creates new strings, concatenating multiple copies of the same string “Hello”*2 gives HelloHello
[n]	Slice - Gives the character from the given index
[n:m]	Range slice - gives characters between index n and m
in	Membership - Returns true if a character exists in the given string
not in	Membership - Returns true if a character does not exist in the given string
r/R	Raw string - prefixed on a given string to remove meaning from escape characters. Example r”Hello\n”
%	String format operator

Aprameyah Technologies Pvt. Ltd.

String formatting

- % Works very similar to the C printf/scanf format specifiers
  - “Hello %s you are %d years old”%(name,x)
- Another way to format is use the format method
  - “Hello {0} you are {1} years old”.format(name,x)
- Formatting with positional arguments
  - "Hello {0} you are {age} years old".format(name,age=x)

Aprameyah
Technologies Pvt. Ltd.

Format Function

- format function is defined as below:
  - format(*args, **kwargs)
- so format can be called with variable normal arguments aswell as variable set of named orguments
  - format(x,y,age=a,name=n)
- Inside the format function args is available as a list and kwargs as a dict

Aprameyah
Technologies Pvt. Ltd.

Conditions

- Multi line if statement

```
var1 = 100
if var1:
  print "Non zero values evaluate to true"
  print var1
```

- Single line

```
if ( var  == 100 ) : print "Value of expression is 100"
```

- if else

```
var1 = 100
if var1 >= 100:
     print "More than 100"
elif var < 50 and var > 25:
     print "Expression value is less than 50"
else:
     print "False condition"
```

Aprameyah
Technologies Pvt. Ltd.

Loops

```
count = 0
while count < 5:
   print count, " is  less than 5"
   count = count + 1
else:
   print count, " is not less than 5”
```

```
flag = 1
while flag: print 'Given flag is really true!’
```

```
for letter in 'Python':
  print 'Current Letter :', letter

fruits = ['banana', 'apple',  'mango']
for fruit in fruits:
   print 'Current fruit :', fruit
```

Aprameyah
Technologies Pvt. Ltd.

Lists

- Simple arrays are augmented with a lot of functionality and hence called Lists than Arrays in python

- Can contain mixed datatypes

```python
flights = ['9W811','IA414', '9W522']
print(len(flights))
flights.append('MH213')
flights.extend('QA400') #Treats this as an array of chars
flights.extend(['SL419'])
flights.insert(1,'9w811')
flights.remove('9w811')
print(flights.pop())
```

Aprameyah Technologies Pvt. Ltd.

Lets Try It(3)

- len BIF can be used to find length of arrays and strings
- Casting looks a little different in python. We use constructor functions: int(x) converts string x to an int
- Defining arrays and adding to arrays:
  - arr = []
  - arr.append(“hello”)
- Looping thru arrays:

  for val in arr:

  print(val)

  Build a hardcoded list of phrases. Loop thru this list and present the phrases to the user asking him to count the words in them. Compare the word count and respond. (Use str.split(“ “))

More things about list

- there are methods to extend(), insert(), remove(), pop(), popLeft() - can be used as a stack or as a queue
- del operator can remove elements in a range
    - del arr[3]
    - del arr[3:5]

Aprameyah
Technologies Pvt. Ltd.

Tuples

- A tuple is a sequence of immutable Python objects.

- Tuples are sequences, just like lists. The only difference is that tuples can't be changed.

- Tuples use parentheses and lists use square brackets.

```
tup1 = ('Kelly', 'Perry', 1997, 2000)
tup2 = (1, 2, 3, 4, 5 )
tup3 = "a", "b", "c", "d"

print(tup1[2])
print(isinstance(tup3, tuple))
```

Creating Variable Groups

- Tuple syntax can be used to create a group of variables

```
(a,b,c) = ['Maruti','Fiat','Hyundai']
(x,y) = "Maruthi:swift".split(":")
```

Aprameyah Technologies Pvt. Ltd.

Sets

- A set is an unordered collection with no duplicate elements
- myset = set()
- fruits = {'apple', 'orange', 'apple', 'pear', 'orange', ‘banana'}
- some useful methods:
  - intersection, intersection_update: A n B
  - symmetric_difference, symmetric_difference_update: (A u B) - (A n B)
  - difference, difference_update: A - (A n B)
  - union: A u B
  - issubset, issuperset,

Aprameyah
Technologies Pvt. Ltd.

Lets Try It(4)

- Modify the previous exercise by counting a word just once
- Find words that are common to all the phrases presented to the user

Aprameyah Technologies Pvt. Ltd.

References

- Strings are immutable

```
s1 = "abc"
s2 = s1
s1 = s1+"xyz"
print(s1)
print(s2)
```

- Others are immutable too

```
k1 = 10
k2 = k1
k1=k1+15
print(k1)
print(k2)
```

Aprameyah
Technologies Pvt. Ltd.

Loops with Ranges and Else

```
for num in range(10,20):  #to iterate between 10 to 20
      for i in range(2,num): #to iterate on the factors of the number
         if num%i == 0:      #to determine the first factor
            j=num/i          #to calculate the second factor
            print '%d equals %d * %d' % (num,i,j)
            break #to move to the next number, the #first FOR
      else:                  # else part of the loop
         print num, 'is a prime number'
```

Aprameyah
Technologies Pvt. Ltd.

More about lists

- Iterating lists can be done in loops

```
for flight in flights:
    print("Flight = "+str(flight))
```

- Walking out of the loop throws IndexError

```
IndexError: list index out of range
```

- Lists can contain lists

```
flights.insert(2,['QA300','QA303'])
```

Aprameyah Technologies Pvt. Ltd.

BIFs

- Built-in-Functions can be displayed with dir(__builtins__)

- BIFs are references

```
x = str
print ("Sachin "+x(10)+"dulkar")
```

- built in functions and types can get accidentally overwritten

```
count =0
for str in dir(__builtins__):
    count+=1
    print(str)
print("Value of count ="+str(count))
#-------------------------------
int = 0
int+=1
print(10+int('20'))
```

Aprameyah
Technologies Pvt. Ltd.

Dictionary

- Dicts are associative arrays. Keys are unique and immutable
- tel = {}
- tel = {'jack': 4098, 'sape': 4139}
- tel['guido'] = 4127
- del tel[‘sape']
- if 'guido' in tel:

Functions

- Function definitions are like below and they can have default values to arguments (Arguments during invocation are optional)

```
def getLength(arr=[]):
    return len(arr)
```

- Variables are passed to functions by reference and kept in function scope

- Functions have to be declared before use

Positional, Named and Default Arguments

```
def myfunction( var1, var2, var3):
    print("{} {} {}".format(var1, var2, var3))

myfunction (“abc",var3="xyz",var2="10")

def myfunction( var1, var2="lmn", var3="xyz"):
    print("{} {} {}".format(var1, var2, var3))

myfunction ("abc",var3="xyz")
```

Apprameyah Technologies Pvt. Ltd.

Lets Try It(5)

- Build a phonebook maintained in a dict. Create these methods: add_phone_book_entry, lookup_phone_book_entry

- Make it menu driven

```
while True:
    print("Choose an action:")
    print("1. Add a phone book entry")
    print("2. Lookup a phone book entry")
    print("3. Exit")
```

Aprameyah Technologies Pvt. Ltd.

File Handling

- open() bif takes a file name and open mode (read by default)

```
print(os.getcwd())
file = open(os.getcwd()+"/Scratchpad.py")
while True:
    line = file.readline();
    if not line:
        break
    else:
        print(line, end='')
file.close()
```

Aprameyah Technologies Pvt. Ltd.

Modes

- Text mode opened files can be treated as an iterator of lines
- We can loop thru files reading lines from it just like a collection

```
for line in file:
        print(line)
```

- Binary mode files do not read strings but rather read bytes
- Random seeking is possible too with seek() function

File Open Modes

- r read mode
- w write mode
- w+ write and read mode - overwrites existing file
- r+ read and write mode
- a+ write in append mode and allow reading too

Aprameyah
Technologies Pvt. Ltd.

File Methods

- read(size) - when opened in binary mode (rb)
- readline, seek, close
- It is also an iteratable

```
with open(os.getcwd()+"/Scratchpad.py") as f :
    for line in f:
        print(line)
```

- Can also be used with print function

```
print(line, file=f2, end='')
```

Apprameyah Technologies Pvt. Ltd.

Lets Try It - Persist the Phonebook(6)

- Open a phonebook file in a+ mode and persist phonebook entries using a method called update_phone_book in a CSV format

- Seek to the start of the file, clear it with truncate() method and write the dict to the phone book

  - dicts can be looped over using for key in dict: syntax

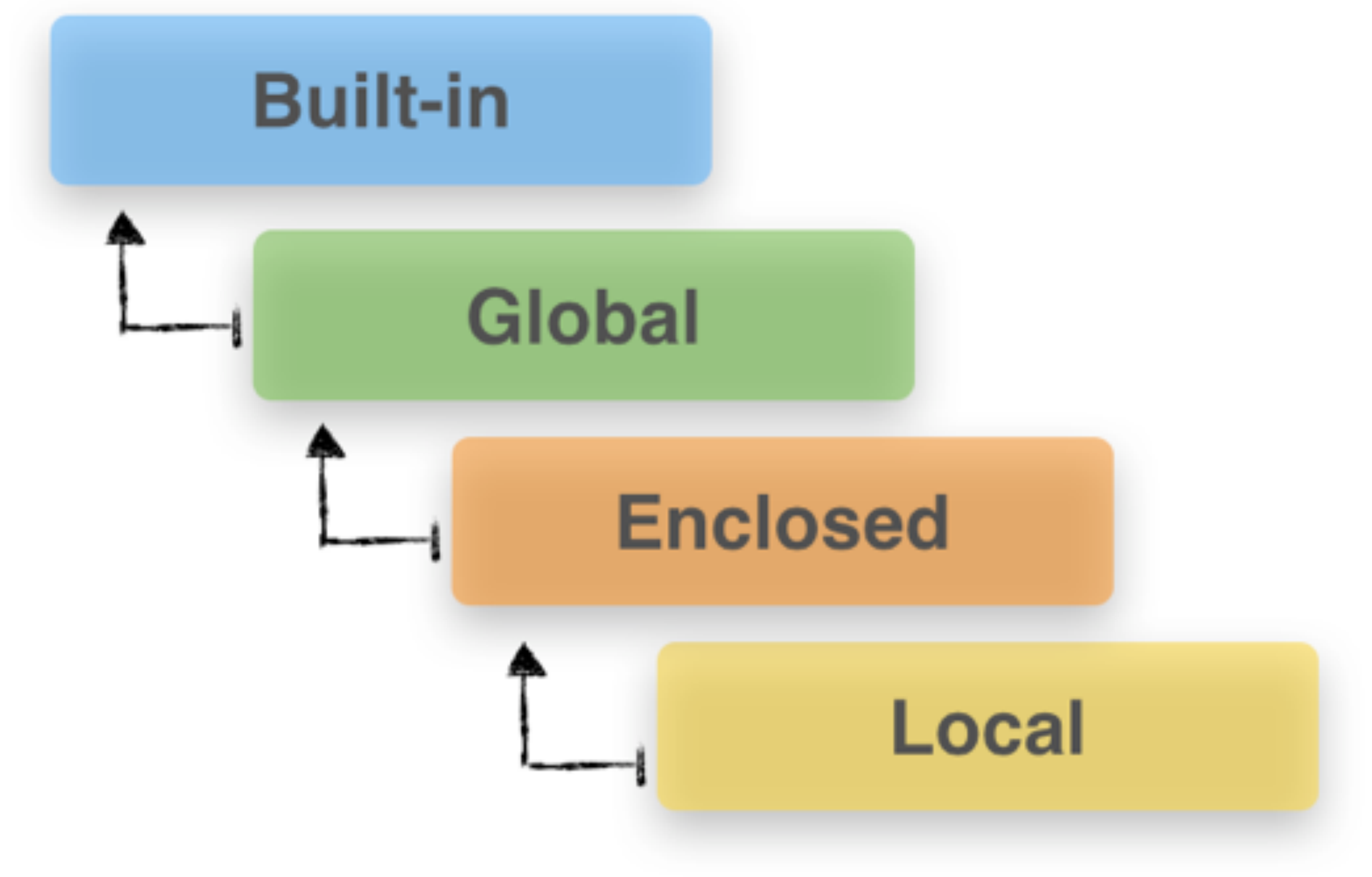
- Create another method called load_phone_book_from_file which is called at the start of the app to load the phonebook dict

Aprameyah Technologies Pvt. Ltd.

More on Dicts

- Dicts can contain other lists as values
    - mydict[‘key’]=[1,2,3] #List
    - mydict[‘key’] = (3,4,5) #tuples
    - mydict[‘key’] = {‘abc’,’xyz’} #sets
- Ex 7: Change the phonebook app to also accept address. Store the number and address as a list in the phonebook dict. Suitably change persistance logic to handle this

Scopes



- No block (or suite) scope

```
for flight in flights:
    k=10
    print("Flight = "+str(flight))
print("value of k = "+str(k))
```

Aprameyah Technologies Pvt. Ltd.

Try This to Understand “global” keyword

```
x = 5
for i in range(10):
    x+=1
print(x)

def updateX():
    x=10
updateX()
print(x)

def myfunc():
    for i in range(10):
        x+=1
myfunc()
print(x)
```

Aprameyah Technologies Pvt. Ltd.

Classes

- A class represents a classification or a template for object creation

```
class Employee:
    'Carries employee details'
    emp_count = 0  # this is like static variable

    def __init__(self, name, number):
        self.name = name #instance variables created javascript style
        self.number = number
        Employee.emp_count += 1

    def displayCount(self):
        print ("Total Employee %d" % Employee.emp_count)

    def displayEmployee(self):
        print ("Name : ", self.name, ", Number: ", self.number)

emp1 = Employee('Manish','9845343334')
emp2 = Employee('Jack','9845393334')
```

Aprameyah Technologies Pvt. Ltd.

Accessing Attributes

```
emp1.displayEmployee()
emp2.displayEmployee()
print ("Total Employee %d" % Employee.emp_count)

emp1.age = 7   # Add an 'age' attribute.
emp1.age = 8   # Modify 'age' attribute.
del emp1.age   # Delete 'age' attribute.

hasattr(emp1, 'age')    # Returns true if 'age' attribute exists
getattr(emp1, 'age')    # Returns value of 'age' attribute
setattr(emp1, 'age', 8) # Set attribute 'age' at 8
delattr(emp1, 'age')    # Delete attribute 'age'
```

Builtin Class Attributes

- __dict__ : Dictionary containing the class's namespace.
- __doc__ : Class documentation string or None if undefined.
- __name__: Class name.
- __module__: Module name in which the class is defined. This attribute is "__main__" in interactive mode.
- __bases__ : A possibly empty tuple containing the base classes, in the order of their occurrence in the base class list.

Apprameyah Technologies Pvt. Ltd.

Lets Try It (8)

- Create a class with fields name, number and address to hold a phone record. Make the dict value a PhoneRecord class.
- Alter all logic to deal with this class instead of array as a value

Aprameyah Technologies Pvt. Ltd.

Pickling

- Picking is one way of persisting the class state for posterity
- We have to use a module to achieve this since its not part of the standard BIFs

```
import pickle
```

- Pickling stores the class contents in a binary python proprietary format.
- Saving and loading an object from pickle file

```
pickle.dump(mgr,employees_file)
mgr = pickle.load(employees_file)
```

Consider this code

```
openAFile();
readFromFile();
openNetworkConnection();
sendDataOverNetwork();
```

Same code with C style error handling

```
if(openAFile()==1){
    if(readFromFile()==1){
        if(openNetworkConnection()==1){
            if(sendDataOverNetwork()==1){

            }else{
                print("Data could not be sent");
            }
        }else{
            print("Network connection cant be opened");
        }
    }else{
        print("Cant read from file");
    }
}else{
    print("Cant open file");
}
```

Aprameyah
Technologies Pvt. Ltd.

Error Handling In Python

```
try:
    openAFile()
    readFromFile()
    openNetworkConnection()
    sendDataOverNetwork()
except Exception as e:
    print(e)
```

Aprameyah Technologies Pvt. Ltd.

Error Handling

- try/except mechanism works very similar to exception handling in most other languages

```
try:
  for line in f:
      print(line)
except:
  pass
```

- pass is a no op statement - an equivalent for empty block in java/C++ { }

Aprameyah
Technologies Pvt. Ltd.

Error Handling

```
try:
   You do your operations here;
   ......................
except ExceptionI:
   If there is ExceptionI, then execute this block.
except ExceptionII:
   If there is ExceptionII, then execute this block.
   ......................
else:
   If there is no exception then execute this block.
-----------------------------------------------------------------------------
try:
   fh = open("testfile", "w")
   fh.write(“File open succeeded!!")
except IOError:
   print "Error: can't open file"
else:
   print “File write succeeded"
   fh.close()
```

Finally Suite

- The finally block is a place to put any code that must execute, whether the try-block raised an exception or not.

```
try:
    fh.write("This is my test file for exception handling!!")
finally:
    print "Going to close the file"
    fh.close()
```

Raising Exceptions

- An exception can be a string, a class or an object. Most of the exceptions that the Python core raises are classes, with an argument:
  - raise "Invalid arguments!", args
- Exceptions can have arguments

```
try:
    if(not validate_host(host_name))
       raise Networkerror("Bad hostname")
except Networkerror as e:
    print e.arg
```

Aprameyah
Technologies Pvt. Ltd.

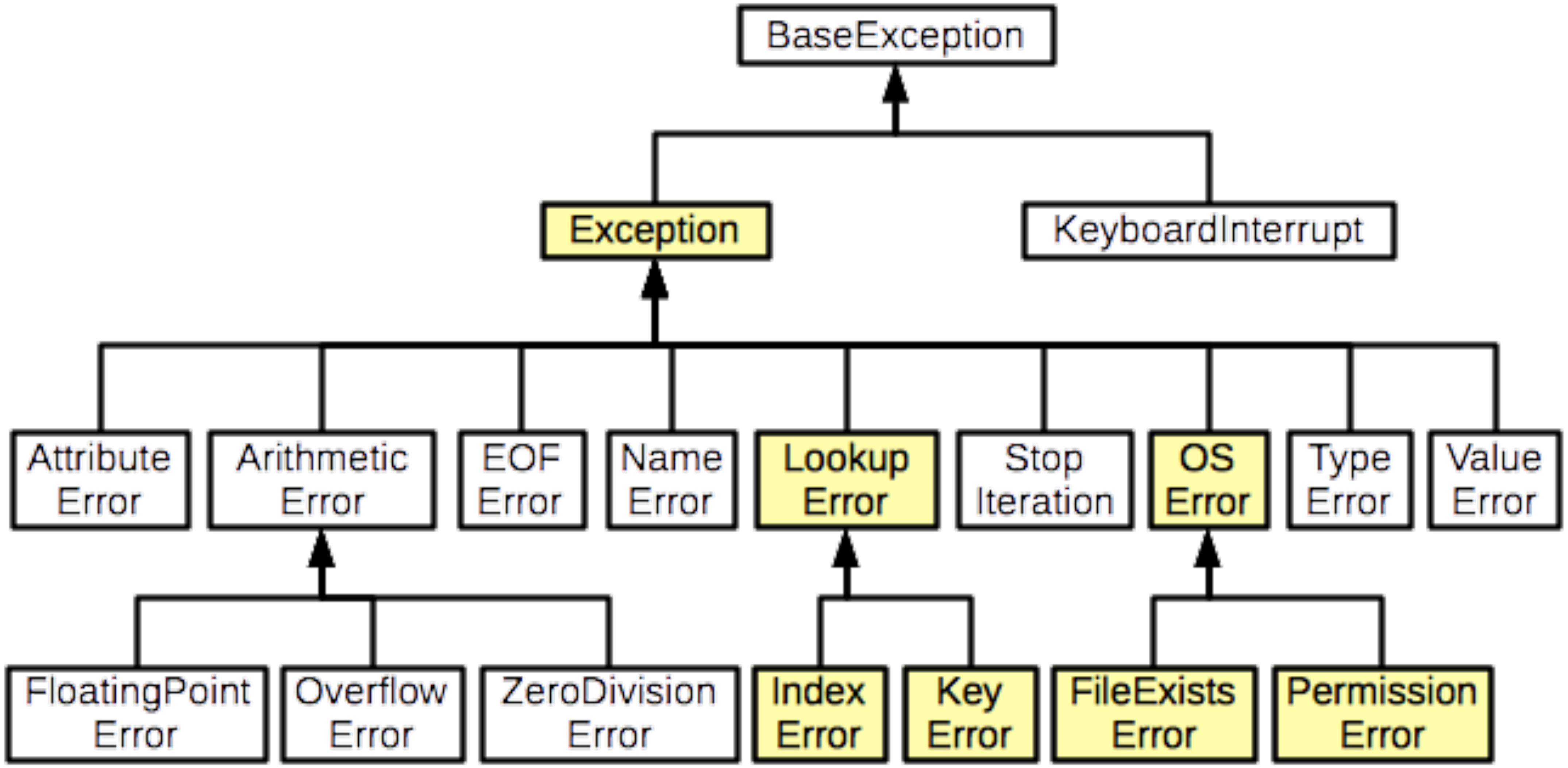
Defining Exception Classes

- Create a class that extends one of the Errors

```
class Networkerror(RuntimeError):
    def __init__(self, arg):
        self.arg = arg
```

Aprameyah Technologies Pvt. Ltd.

Exception Hierarchy



Apprameyah Technologies Pvt. Ltd.

Using With

- File operations are typically wrapped into a with statement to ensure file closure even in case of errors like below:

```
with open(os.getcwd()+"/Scratchpad.py") as f:
    while True:
        line = f.readline();
        if not line:
            break
        else:
            print(line, end='')
```

Using With

- The with statement is used to wrap the execution of a block with methods defined by a context manager.

- This allows common try...except...finally usage patterns to be encapsulated for convenient reuse.

Aprameyah Technologies Pvt. Ltd.

Using With

- The context expression (the expression given in the with_item) is evaluated to obtain a context manager.

- The context manager's __enter__() method is invoked and its result assigned to provided target .

- The with statement guarantees that if the __enter__() method returns without an error, then __exit__() will always be called.

- The suite is executed.

- The context manager's __exit__() method is invoked. If an exception caused the suite to be exited, its type, value, and traceback are passed as arguments to __exit__(). Otherwise, three None arguments are supplied.

Aprameyah
Technologies Pvt. Ltd.

Lets Try It (9)

- Use pickling to save the phone records.
  - Modify load_phone_book_from_file to open file in binary mode (rb).
    - call pickle.load(file) in a loop
    - Loop exit condition will be an error pickle.UnpicklingError. Trap this exception and break loop when this happens
  - Modify update_phone_book to open file in (wb+) mode and call pickle.dump() for each phone record

Lets Try It (9 contd..)

- Use with suites to perform pickled read and write operations

- Make the methods that read from pickled store static methods of PhoneRecord class by putting in the decorator @staticmethod (More on decorators later)

Aprameyah Technologies Pvt. Ltd.

Shelves

- The shelve module implements persistent storage for arbitrary Python objects which can be pickled, using a dictionary-like API.
- The shelve module can be used as a simple persistent storage option for Python objects when a relational database is overkill.
- The shelf is accessed by keys, just as with a dictionary.
- The values are pickled and written to a database created and managed by anydbm.

Aprameyah
Technologies Pvt. Ltd.

Reading & Writing Shelves

```
import shelve

s = shelve.open('test_shelf.db','c')
try:
    s['key1'] = { 'int': 10, 'float':9.5, 'string':'Sample data' }
    print s['key1']
finally:
    s.close()
```

Open mode meaning

Value	Meaning
'r'	Open existing shelf for reading only (default)
'w'	Open existing shelf for reading and writing
'c'	Open shelf for reading and writing, creating it if it doesn't exist
'n'	Always create a new, empty shelf, open for reading and writing

ex 9a: With shelves, we can remove the explicit loading and persisting lists as we can deal with the database as a dict directly. Modify the phonebook to use shelves to persist

Aprameyah Technologies Pvt. Ltd.

JSON Persistance

- json module will encode Python objects as JSON strings, and decode JSON strings into Python objects.

- The json module provides an API similar to pickle for converting in-memory Python objects to a serialized representation known as JavaScript Object Notation (JSON).

- JSON in a non proprietary text format, making it suitable for inter-application communication.

- JSON is most widely used for communicating between the web server and client in an AJAX application, but is not limited to that problem domain.

Aprameyah
Technologies Pvt. Ltd.

Encoding & Decoding

- The encoder understands Python's native types by default (string, unicode, int, float, list, tuple, dict).

- When decoded, it might not yield the same datatype

```
import json

data = [ { 'a':'A', 'b':(2, 4), 'c':3.0 } ]
data_string = json.dumps(data)
print ('ENCODED:', data_string)

decoded = json.loads(data_string)
print ('DECODED:', decoded)

print ('ORIGINAL:', type(data[0]['b']))
print ('DECODED :', type(decoded[0]['b']))
```

- Dumps can be customizable: json.dumps(data, sort_keys=True, indent=2)

Aprameyah Technologies Pvt. Ltd.

Encoding Custom Types

- Custom types need a converter.

```
@staticmethod
def to_json(phone_record):
        return {'__class__': 'PhoneRecord',
                    '__value__': phone_record.__dict__}

json.dump(phone_record, phone_book_file, default=PhoneRecord.to_json)

____________________________________________

@staticmethod
def from_json(json_object):
        if '__class__' in json_object:
            if json_object['__class__'] == 'PhoneRecord':
                pr = PhoneRecord()
                pr.__dict__ = json_object['__value__']
                return pr;
        else:
            return json_object

phone_record = json.loads(line, object_hook=PhoneRecord.from_json)
```

Ex 9b: Convert the phonebook to use json encoding using code snippets above

String Representations

- If we need a string representation of an object we can use either the __repr__() or __str__() function
- if we try to print an object, it first looks for an __str__() function and then a __repr__ function to get a string representation
- __str__() is to be used for a friendly readable text value
- __repr__() for a unique technically correct value

Iterators

- Python iterator object must implement two special methods, __iter__() and __next__(), collectively called the iterator protocol.

- Any object implementing this protocol can be used in the for loop structures

Aprameyah Technologies Pvt. Ltd.

Lets Try It (10)

- Define a new class called PhoneRecordIterator that takes the file name as a constructor argument
- define __iter__() function that just opens the file and returns self
- define a __next()__ function that reads a record on the file and returns a phone record.
- After last record is reached, raise StopIteration error
- implement __str__() function in PhoneRecord and create an option to print all phone records using the above iterator

Aprameyah Technologies Pvt. Ltd.

Generators

- Python generators are a simple way of creating iterators
- Simply speaking, a generator is a function that returns an object (iterator) which we can iterate over (one value at a time).
- Yield is a keyword that is used like return, except the function will return a generator.

Aprameyah
Technologies Pvt. Ltd.

Generators

- Generator function is not called till we start iterating through it
- Every subsequent call to the function continues from the last yield point with all state information intact

```
def createGenerator():
    mylist = range(5)
    for i in mylist:
        yield i*i

mygenerator = createGenerator() # create a generator
print(mygenerator) # mygenerator is an object!
for i in mygenerator:
    print(i)
```

Aprameyah
Technologies Pvt. Ltd.

Use Generators

- Replace the PhoneRecordIterator with a generator

Aprameyah
Technologies Pvt. Ltd.

RegEx

- The module re provides full support for Perl-like regular expressions in Python.
- The re module raises the exception re.error if an error occurs while compiling or using a regular expression.
- Following functions return a match object on success, None on failure. We would use group(num) or groups() function of match object to get matched expression.
    - re.match(pattern, string, flags=0)
    - re.search(pattern, string, flags=0)
- Replace function replaces repl string in place of matches
    - re.sub(pattern, repl, string, max=0)

Aprameyah
Technologies Pvt. Ltd.

RegEx Usage

```
import re

def validate_date(date_str):
    pattern = r"(\d{4})-(\d{2})-(\d{2})"
    match = re.match(pattern, date_str)
    if not match:
        raise Exception("Invalid date format")
    year = int(match.group(1))
    month = int(match.group(2))
    date = int(match.group(3))
    if month < 1 or month > 12 or date < 1 or date>31 or year<1:
        raise Exception("Invalid values")
    return True

validate_date("2008-12-10")
```

Aprameyah Technologies Pvt. Ltd.

Function References

- Function names are references and can be passed around like function pointers

```
def myfunction(data):
    print(data)

printer = myfunction
printer("Hello there")
```

- Switch case statement can be simulated using function references and a dict
  - options = {1:func1, 2:func2}
- **Ex 12: Add a new option to the phone book app to search based on part of a name and print the matching records (using __str__)**
- **replace the if-else condition with a dict of function references**

Aprameyah
Technologies Pvt. Ltd.

Error Handling

- Trap the exception or let the caller handle the error?
- Higher layer code cannot just throw a low-level exception
- It has to trap it in an except block and rethrow it as a higher level exception
    - Ex: raise PhonebookException("could not read phonebook") from e
- Ex-13: trap file not found exceptions while reading phonebook and rethrow it as above and test and note the traceback

Aprameyah
Technologies Pvt. Ltd.

File Copy using With

```
with open(os.getcwd()+"/Scratchpad.py") as f, open(os.getcwd()+"/
ScratchpadCopy.py", "w") as f2 :
    while True:
        line = f.readline();
        if not line:
            break
        else:
            print(line, end='')
            f2.writelines(line)
```

Ex 14: Create another option to backup the phone book. Accept the new filename from user, open the two files in a single “with” block and copy them. Since the files are picked, we would need to open in binary and copy bytes

Classes (Refresher)

- A class represents a classification or a template for object creation

```
class Employee:
    'Carries employee details'
    emp_count = 0  # this is like static variable

    def __init__(self, name, number):
        self.name = name #instance variables created javascript style
        self.number = number
        Employee.emp_count += 1

    def displayCount(self):
        print ("Total Employee %d" % Employee.emp_count)

    def displayEmployee(self):
        print ("Name : ", self.name, ", Number: ", self.number)

emp1 = Employee('Manish','9845343334')
emp2 = Employee('Jack','9845393334')
```

Aprameyah
Technologies Pvt. Ltd.

Inheritance and Overriding

```
class Manager(Employee):
    def __init__(self,name,number,branch):
        super(Manager, self).__init__(name,number)
        self.branch = branch

    def displayEmployee(self):
        Employee.displayEmployee(self)
        print("Manager at %s branch" % self.branch)

mgr = Manager('Kelly','792382932','Lawrencevile')
mgr.displayEmployee()
```

Aprameyah Technologies Pvt. Ltd.

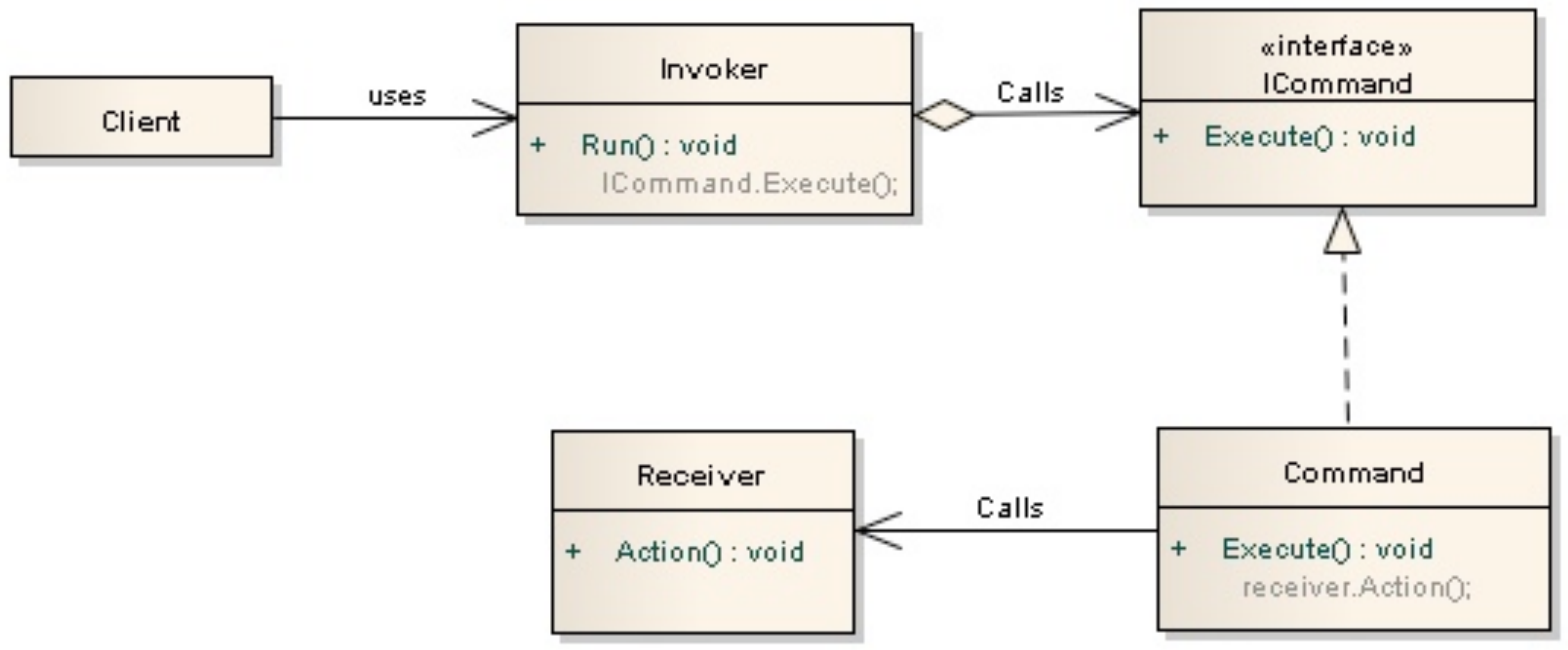
Runtime type checking

- isinstance()
  - Used for class inheritance hierarchies - subtype is a type of base type
- type()
  - used for exact type matching

```
print(isinstance(flights, list))
print(type(flights) == list)
print(flights.__class__ == list)
```

Apprameyah Technologies Pvt. Ltd.

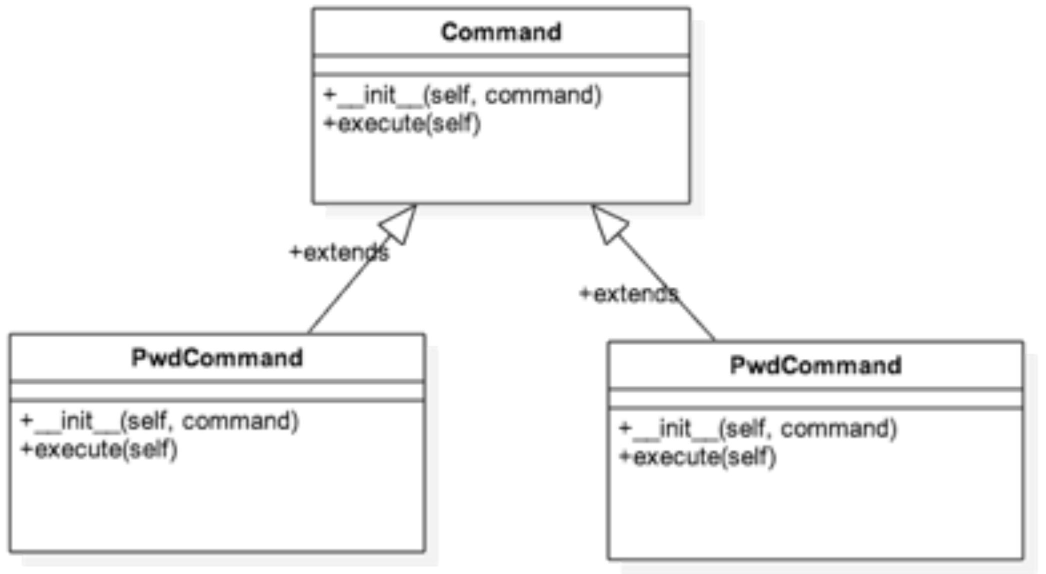
Command Pattern



Aprameyah Technologies Pvt. Ltd.

Command Processor (15)

- Lets design a command processor that acts very close to the unix shell with just a few commands



- execute methods implement ls, pwd and chdir commands using the os module

```
os.listdir(folder_path)
os.getcwd()
os.chdir(folder_path)
```

Aprameyah
Technologies Pvt. Ltd.

Command Processor (15)

- Create a CommandFactory
  - __init__ method holds reference to each of the command implementation class constructors in a dict
  - get_command(command) returns the command implementation for the required command
- The main command loop looks like this:

```
while True:
    command = input("> ")
    cf = CommandFactory()
    command = cf.get_command(command)
    command.execute()
```

Aprameyah
Technologies Pvt. Ltd.

Constant Pattern (15)

- Python does not have a constant type built in but there is a way we can build constants this way:

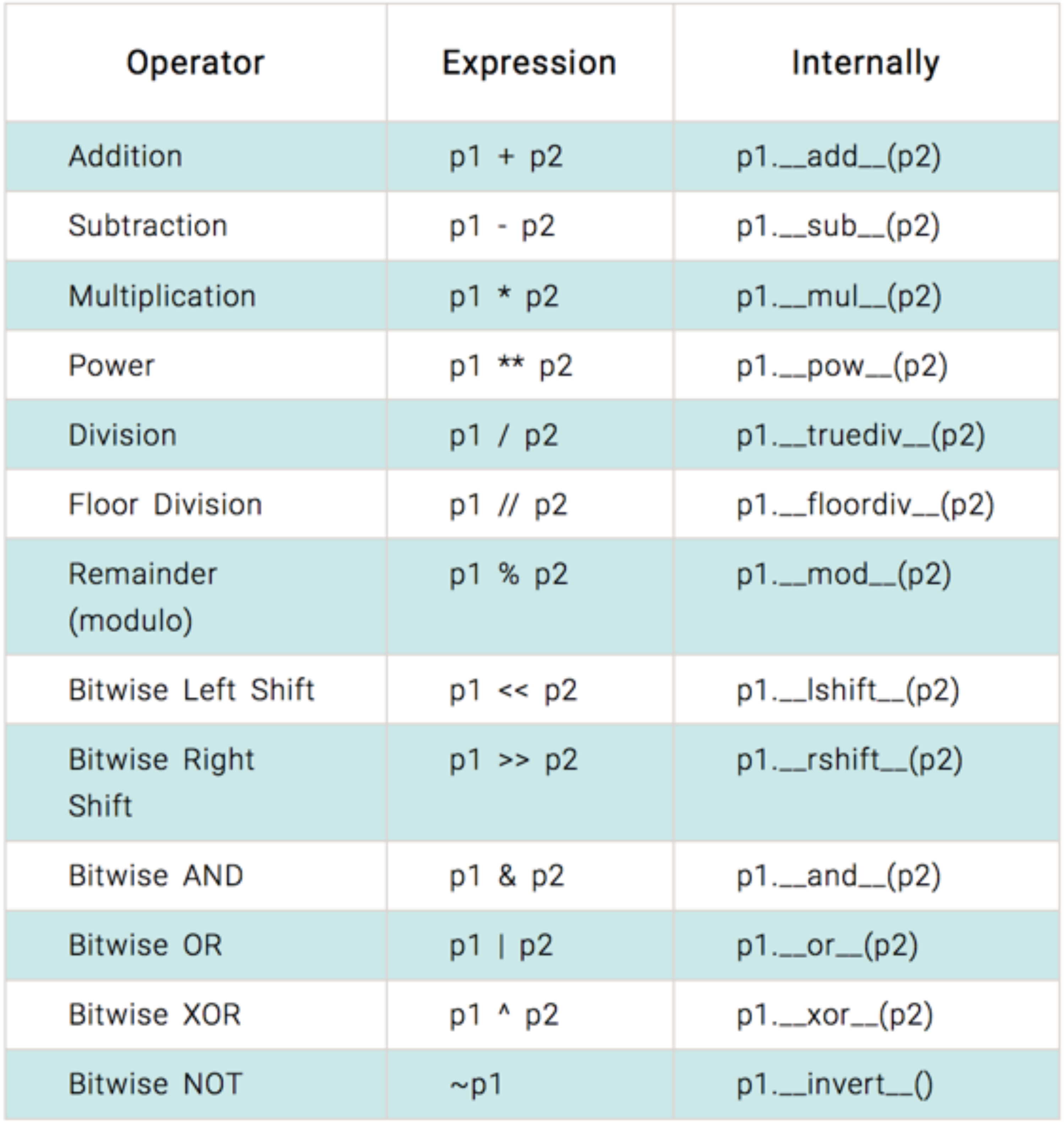
```
class Const:
    class ConstError(TypeError):
        pass

    def __setattr__(self,name,value):
        if name in self.__dict__:
            raise const.ConstError("Can't rebind const {}".format(name))
        self.__dict__[name]=value

const = Const()
const.unknown_command = "unknown_command"
```

- The __setattr__ is a specialized magic method called by python whenever we set a value into an object

- Use this constant to define all the commands in the command factory

Aprameyah
Technologies Pvt. Ltd.

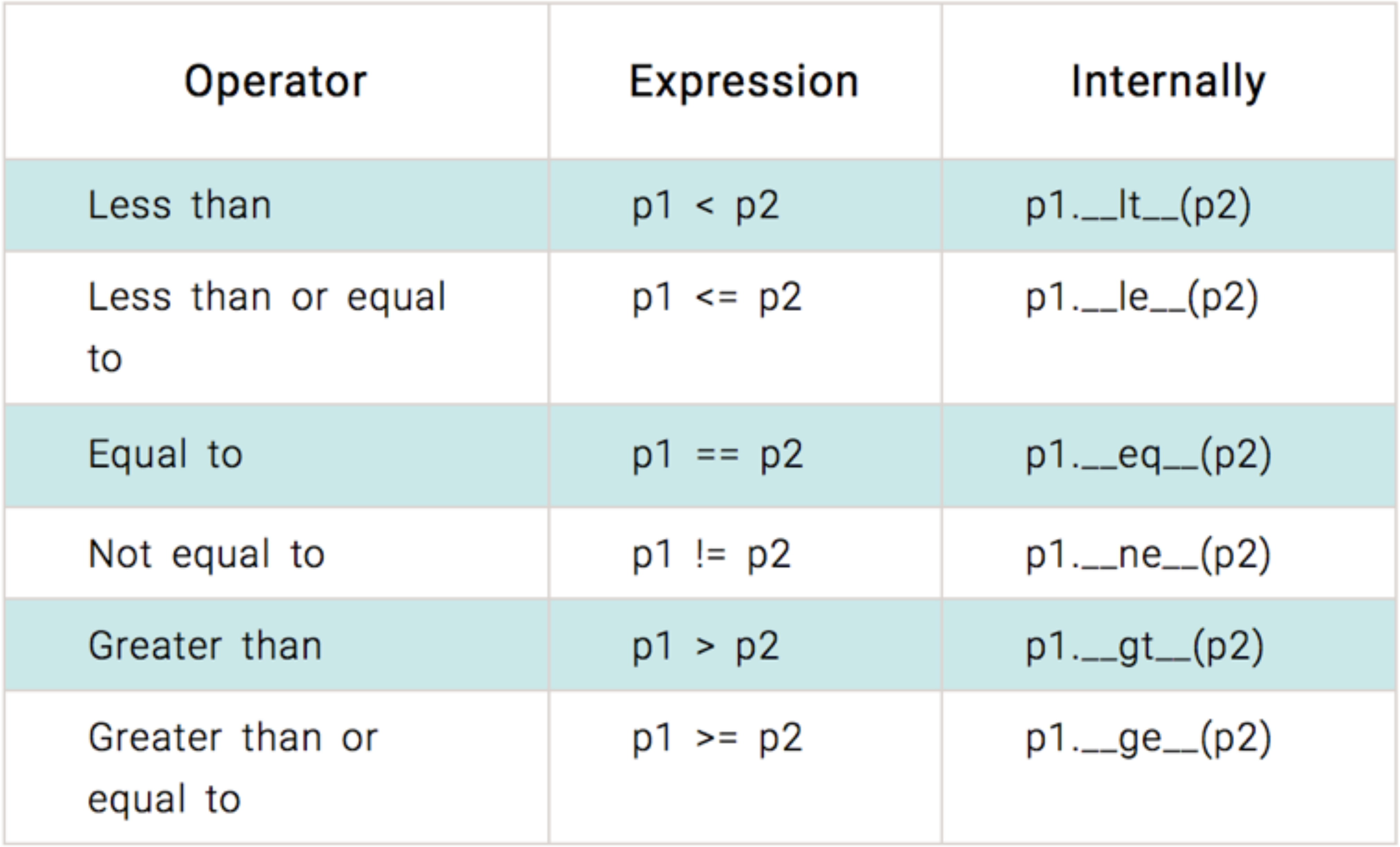
Operator Overloading



Operator	Expression	Internally
Addition	p1 + p2	p1.__add__(p2)
Subtraction	p1 - p2	p1.__sub__(p2)
Multiplication	p1 * p2	p1.__mul__(p2)
Power	p1 ** p2	p1.__pow__(p2)
Division	p1 / p2	p1.__truediv__(p2)
Floor Division	p1 // p2	p1.__floordiv__(p2)
Remainder (modulo)	p1 % p2	p1.__mod__(p2)
Bitwise Left Shift	p1 << p2	p1.__lshift__(p2)
Bitwise Right Shift	p1 >> p2	p1.__rshift__(p2)
Bitwise AND	p1 & p2	p1.__and__(p2)
Bitwise OR	p1 | p2	p1.__or__(p2)
Bitwise XOR	p1 ^ p2	p1.__xor__(p2)
Bitwise NOT	~p1	p1.__invert__()

Aprameyah
Technologies Pvt. Ltd.

Operator Overloading



Operator	Expression	Internally
Less than	p1 < p2	p1.__lt__(p2)
Less than or equal to	p1 <= p2	p1.__le__(p2)
Equal to	p1 == p2	p1.__eq__(p2)
Not equal to	p1 != p2	p1.__ne__(p2)
Greater than	p1 > p2	p1.__gt__(p2)
Greater than or equal to	p1 >= p2	p1.__ge__(p2)

Aprameyah
Technologies Pvt. Ltd.

Data Hiding

- you can name attributes with a double underscore prefix, and those attributes will not be directly visible to outsiders.

```
class JustCounter:
   __secretCount = 0

   def count(self):
      self.__secretCount += 1
      print (self.__secretCount)

counter = JustCounter()
counter.count()
counter.count()
print (counter.__secretCount) # ERROR
print(counter._JustCounter__secretCount)
```

Copy Command (16)

- Implement a copy command in the command processor. Example command: cp .* temp/
    - First param is a regex. Just enough to process current directory
- Get list of files in dir: os.listdir()
- Find if a file is a directory: os.path.isdir(file_)
- Use binary copy logic from phonebook backup sample

Closures

- Closure is a way of implementing private class state in javascript by playing on variable scope

```
def add():
    counter=0
    counter += 1
    print(counter)

add()
add()
----------------------------

def get_add_fn():
    counter = 0
    def add():
        nonlocal counter
        counter += 1
        print(counter)
        return counter
    return add

add = get_add_fn()
add()
add()
```

Closures As Function Factories

```
def notifyUser(email) :
    def inner_notify(text) :
        print("Sending '"+text+"' to "+email)
    return inner_notify

notifyWill = notifyUser('wberger@lo.com')
notifyKelly = notifyUser('kperry@lo.com')

notifyWill("Hello, you have a new join request")
notifyKelly("Hello you are late to work!")
```

Apprameyah Technologies Pvt. Ltd.

Closures

- Characteristics
  - We must have a nested function (function inside a function).
  - The nested function must refer to a value defined in the enclosing function.
  - The enclosing function must return the nested function

When to Use Closures

- Closures can avoid the use of global values and provides some form of data hiding.
- When there are few methods (one method in most cases) to be implemented in a class, closures can provide an alternate and more elegant solutions. But when the number of attributes and methods get larger, better implement a class.

Aprameyah
Technologies Pvt. Ltd.

Nested Directory Listing (17)

- Implement a command called “lsr” for recursively listing directory. Just do one level nesting (no need for recursion)
    - lsr command needs access to the ls command to call on each subdirectory. Passing this extra parameter will break the command pattern we have now.
    - Make the command factory a closure to give access to commands dict to all commands without passing them while creating the command to every command
    - Command factory boils down to one function (Implement the two closures):

```
def get_command(command):
    commands = {"ls" : ListCommandClosure, "lsr" : ListRecursiveCommandClosure}
    command_name = command.split(sep=" ")[0]
    if command_name in commands :
         return commands[command_name](command,commands)
```

Aprameyah Technologies Pvt. Ltd.

Decorators

- Decorators add functionality to an existing code.

- This is also called metaprogramming as a part of the program tries to modify another part of the program at compile time.

- Functions and methods are called callable as they can be called. In fact, any object which implements the special method __call__() is termed callable

  - Basically, a decorator takes in a function, adds some functionality and returns it.

Aprameyah Technologies Pvt. Ltd.

Decorators

```python
def make_pretty(func):
    def inner():
        print("I got decorated")
        func()
    return inner

def ordinary():
    print("I am ordinary")

pretty = make_pretty(ordinary)
pretty()
```

Aprameyah Technologies Pvt. Ltd.

Decorator Syntax

- Decorator pattern of the following is common:

```
ordinary = make_pretty(ordinary)
ordinary()
```

- Special syntax exists for just this

```
@make_pretty
def ordinary():
    print("I am ordinary")
```

Aprameyah Technologies Pvt. Ltd.

Decorating Functions With Params

```
def smart_divide(func):
    def inner(a,b):
        print("I am going to divide",a,"and",b)
        if b == 0:
            print("Whoops! cannot divide")
            return

        return func(a,b)
    return inner

@smart_divide
def divide(a,b):
    return a/b

divide(10, 0)
```

Aprameyah Technologies Pvt. Ltd.

Writing Generic Decorators

```
def star(func):
    def inner(*args, **kwargs):
        print("*" * 80)
        func(*args, **kwargs)
        print("*" * 80)
    return inner

def percent(func):
    def inner(*args, **kwargs):
        print("%" * 80)
        func(*args, **kwargs)
        print("%" * 80)
    return inner

@star
@percent
def printer(msg):
    print(msg)

printer("Decorated message")
```

Aprameyah Technologies Pvt. Ltd.

Lets Try It (18)

- Create a function to measure time of any method being decorated. Use it to measure time of the copy method written earlier

- Use these methods

```
start = datetime.datetime.now()
...
end = datetime.datetime.now()
duration = end - start
print("Method call {} microseconds".format(duration.microseconds))
```

Writing Decorators With Parameters

- Decorator syntaxes can use functions. One way of holding parameters with it is to make it a class and use the constructor function

- The class instance will then be made callable by implementing __call__

Aprameyah
Technologies Pvt. Ltd.

Decorator With Parameter

```
class ParamDecorator(object):
    def __init__(self,param):
        self.param = param

    #Presense of the __call__ method makes this object a callable. Means an instance of the object x can be used as x(original_func)
    def __call__(self, original_func):
        decorator_self = self
        def decorator_function( *args, **kwargs):
            print(decorator_self.param)
            original_func(*args,**kwargs)
        return decorator_function
```

- Ex 18a: On these lines write a decorator that sets the current working directory for a command as “/Users”. Will be setup as @ContextFolder(“/Users”)