python notes

python env configuration

. python projects are usually built on the top of thirdparty libraries. the

dependencies in java are managed by maven. the 3rd libraries are stored in

maven central repository. the similar concept is used in python. the central

lib repository of python is called pypi. the libraries are downloaded by tool

pip. therefore,

Read it-->ref: https://packaging.python.org/installing/

. pip must be installed before we start

wget https://bootstrap.pypa.io/get-pip.py

python get-pip.py #install pip

pip install -U pip #linux env to upgrade pip

pip install -U pip setuptools #install setuptools for creating package

. creation of virtualenv. there are many python tools like pyvenv, virtualenv able to create virtual env.

virtualevn is most popular choice.

pyenv comes with all version of python.

pyvenv comes with python standard distribution from version3.4. but virtualenv we have to download from pypi by pip.

$pip install virtualenv

#create a virtual environment for a project

$mkdir myProjectFolder

$cd myProjectFolder

$virtualenv venvName #(venvName can be any name of the virtual env)

$source venvName/bin/activate

$deactivate #deactivate virtualenv

$cd..

$rm -rf venvName #remove the virtualenv folder

Or we can create python virtual environment directly from python command by

‘python -m venv virtualPyDirectory’

‘cd virtualPyDirectory’

‘scripts\activate’

Like mentioned above, pip is used to manage the 3rd party libs for python. It has many command pip search/list/install/freeze. Here is a real scenario to see how pip works for manage the libraries.

Let’s say we have a PROD env ready and like to install the same env on different machine. The common solution is to find what 3rd party libs have been installed on prod env and reinstall the new env.

Python provides the quick way to reinstall everything by pip.

. pip freeze > requirements.txt #find all installed libs including the version and stored into a file

. pip install –r requirements.txt #install all required libs by reading a file

. project folder structure

like jEE project, incorrect folder structure won't be recognized by jEE

application server. the same point in python, the project must be declared

as certain folder structure. this is the example of mini python project.

http://python-packaging.readthedocs.io/en/latest/minimal.html

from Feb. 2017, i started the march of learning Python. the motivation is from bigdata solution. after many of research,

python, scala and java are the major 3 language in bigdata area. most of bigdata analysis functions are coded by python.

scala is the nature language for apache spark. all of areas have java apis. the most important reason is cloudera certificate

must write either python or scala code to approve some concepts. so it is valuable to learn python.

Basic syntax

. indentation to define the block instead of {} in java

. dynamic type boundle, so no need to define the type when declare a variable

.

type

. numbers, string, list, tuple, dictionary

numbers includes int, long, float, complex

OOP in python

python is OO language. everything in python is managed as object. int, float, function, object etc are objects.

python is more flexible than java. python supports multiple inheritance.

class Subclass(Superclass1,[Superclass2,...]):

# initializer

# methods

override operators

\_\_add\_\_ --> +

\_\_min

class property and instance property are presented differently in python from java.

in java, the class level property must be defined as static which is enabled when jvm

load the class. instance property is available when the class be instanced. in java,

the instance properties must be declared in the class without static modifier.

in python, the class level property is defined in the class. the instance property

is defined in constructor. instance property is not explicitly decalared in class.

eg.

class Employee:

'Common base class for all employees'

empCount = 0

def \_\_init\_\_(self, name, salary):

self.name = name

self.salary = salary

Employee.empCount += 1

example above empCount is class property. name,salary is instance properties.

like java, every class has built-in toString() method inherited from Object class.

in Python, each class has built-in \_\_init\_\_, \_\_del\_\_,\_\_str\_\_,\_\_repr\_\_,\_\_comp\_\_ methods.

they can be overridden. the operators (+,-,\*,/)of the classes can also be overridden

by the counterpart of methods \_\_add\_\_,\_\_min\_\_....

private class property can be distinct by variable name which starts from double underscore.

July 23, 2017

number,string type in python is immutable like string in java. when it is assigned to another value, a new instance

will be created.

Module/package

python provides the pre-compilation function, but it doesn't improve any running speed. it only improves the initial

module loading speed.

module

python module is just a file containing definitions and statements. the file name is the module name. a module can

contain function class etc. it also allows to have executable statements. these statements are intended to

initialize the module. they are executed only the first time the module name is encountered in an import statement.

UNDERSTAND: the difference of 'from package import item' and 'import package.subpackage.item'

in first case, item can be anything like subpackage, module, class, function, variable

in the second case, item must be either sub-package or module. it cannot be function, class or varible.

from package import \*

it is rarely used. it only imports the names (module function class or variable are defined in \_\_all\_\_ list in \_\_init\_\_.py file).

if \_\_all\_\_ not defined, it doesn't import all submodules. it just ensures the initialization script invoked and the package imported

and whatever names are defined in the \_\_init\_\_.py file. It also includes any submodules of the package that were explicitly loaded

by previous import statements. Consider this code:

import sound.effects.echo

import sound.effects.surround

from sound.effects import \*

In this example, the echo and surround modules are imported in the current namespace because they are defined in the sound.effects

package when the from...import statement is executed. (This also works when \_\_all\_\_ is defined.)

July 30, 2017

start a small project by using python

Aug 10, 2017

built-in functions

.bytearray

python string type is similar to java String. it is immutable. it brings very bad performance if many modifications happen on a string

data. in java, StringBuffer is the solution to make the string mutable. the same idea is used in python the counterpart of java

StringBuffer.append is bytearray.extend

. classmethod

python supports both static method and class methods. it likes the method having static modifier in java class. they are similar.

but they have some difference.

see example below. the classmethod factory method can only create instanceOf baseClass. but the factory method from classMethod

allows derived classes to have their own class type.

from datetime import date

# random Person

class Person:

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

@classmethod

def fromBirthYear(classname, name, birthYear):

return classname(name, date.today().year - birthYear)

def display(self):

print(self.name + "'s age is: " + str(self.age))

person = Person('Adam', 19)

person.display()

person1 = Person.fromBirthYear('John', 1985)

person1.display()

Nov 30, 2017

======================================

module

package

distribution package -- bundled up package which has source format, built distribution format(bdist\_egg, bdist\_wheel)

sdist -- tar.gz

bdist -- b stands for built not binary distribution

wheel distribution

universal -- one package can be used anywhere

pure python -- for different versions of python

platform --

Pypi -- python package index --> repository

not say pai pai since there is another concept called pypy which is just in time in python

component of package

setup.py -- all information of package which is executeable

setup.cfg -- configuration around wheel

manifest.in -- define resource

README.rst --

DESCRIPTION.rst -- readme in pypi

requirements:

pip install wheel

pip install twine -- upload to pypi in security

pip install tox -- test runner

pip install cookiecutter -- generate all package related files

make virtualenv

http://scikit-learn.org/stable/

========================================

March 25, 2018

quickest way to learn python

. learn basic python syntax

. learn data structure type -- sequences -- list, tuple, set, dictionary

when parameters are stored in a sequence variable like params=range(3). if we wanna pass (1,2) as parameter of a function.

we need to unpack them as position arguments. for listvariable unpack is \*listvariable

\*\*dictionaryvariable

tuple

immutable

tuplevar=() #create empty tuple

tuplevar=1, #assign 1 value tuple, tuplevar=(1) is assigned integer number 1 to variable which is not tuple

tuple=1,2,3 #assign more than 1 value tuple

list

mutable

listvar=[] #create empty list

listvar=oldListVar[:] #create the new instance of list from old list (clone)

set

no order no duplicates

setVariable={1,2}

setVar=set() #create empty set

when comparing two sequence, they must have the exact same type. otherwise exception

. learn module package management

UNDERSTAND: the difference of 'from package import item' and 'import package.subpackage.item'

in first case, item can be anything like subpackage, module, class, function, variable

in the second case, item must be either sub-package or module. it cannot be function, class or varible.

from package import \*

it is rarely used. it only imports the names (module function class or variable are defined in \_\_all\_\_ list in \_\_init\_\_.py file).

if \_\_all\_\_ not defined, it doesn't import all submodules. it just ensures the initialization script invoked and the package imported

and whatever names are defined in the \_\_init\_\_.py file. It also includes any submodules of the package that were explicitly loaded

by previous import statements. Consider this code:

import sound.effects.echo

import sound.effects.surround

from sound.effects import \*

In this example, the echo and surround modules are imported in the current namespace because they are defined in the sound.effects

package when the from...import statement is executed. (This also works when \_\_all\_\_ is defined.)

. learn basic libraries

1. output format, read/write file,

2. Exception

if an exception occurs which doesn't match the exception named in the exception clause, it is passed on to outer try statements;

if no handler is found, it is a unhandled exception and execution stops with a message. when having exception fired, it will be caught

by exception clauses. from the first to the last. we can also raise an Exception or instance of Exception. eg. raise Exception --

it will create the instance of Exception. raise Exception('message','message detail') -- manually attached some information into

exception. the information is stored in exceptionInstance.args. see the excerpt below:

try:

... raise Exception('spam', 'eggs')

... except Exception as inst: #inst is the reference of exception instance

... print(type(inst)) # the exception instance

... print(inst.args) # arguments stored in .args

... print(inst) # \_\_str\_\_ allows args to be printed directly,

... # but may be overridden in exception subclasses

... x, y = inst.args # unpack args

... print('x =', x)

... print('y =', y)

else:

#do something when no exception thrown

all customized exception should be directly or indirectly from Exception class.

3. namespace and scope

namespace and scope are fundation of all languages. namespace is like package in java. but snice python is interpret language,

it doesn't work really like java package. it more likes package styled class declaration in javascript. the purpose is to

avoid the duplicates of names (including variable name, module name and class name etc. so in the document it is called names in general.)

scope is the lifecycle of a variable. the current available name in currect scope can be listed in dir()

if we like to use global variable in a function rather than creating a new local variable, we must explicitly add global key word front of variable.

see example below:

a\_num = 10

b\_num = 11

def outer\_func():

global a\_num

a\_num = 15 #this variable is global variable which declare above

b\_num = 16 #variable b\_num is local variable whose scope is only in this function

def inner\_func():

global a\_num

a\_num = 20

b\_num = 21

print('a\_num inside inner\_func :', a\_num)

print('b\_num inside inner\_func :', b\_num)

inner\_func()

print('a\_num inside outer\_func :', a\_num)

print('b\_num inside outer\_func :', b\_num)

outer\_func()

print('a\_num outside all functions :', a\_num)

print('b\_num outside all functions :', b\_num)

# a\_num inside inner\_func : 20

# b\_num inside inner\_func : 21

# a\_num inside outer\_func : 20

# b\_num inside outer\_func : 16

# a\_num outside all functions : 20

# b\_num outside all functions : 11

import module is tightly related to namespace.

import package.sub-package.modulename #it is good practise because all names in the module are under the namespace of the modulename

from package.sub-package[.module] import \* #bad practise since all names under the package or module will add to the current namespace without prefix.

#it hides lots of issues. if having identical name imported into the same namespace, the later name will override

#the prior imported name (here, name could be name of function or variable or module or package etc.)

. predefined clean-up actions

in python, some built-in features are very useful to save some codes which like auto-close in java. see example below:

for line in open("myfile.txt"):

print(line, end="")

with open("myfile.txt") as f:

for line in f:

print(line, end="")

the first piece of code doesn't have explicit f.close(). so it will leave the file open for a while.

the second code, will automatically close the file since it has with(built-in feature)

. learn class

**class concept in python is closer to javascript instead of java because both are interpretive languages**.

python allows us to assign instance variable(python called data attribute) and method (assign function to a instance variable) to object after created,

which is not allowed in java. because of this reason, data attribute/instance variable no need to declare in python. it is automatically declared when the first time assignment.

in python, it supports class level variables. class variables share to all class instances. instance varible/data attribute is only for the instance itself.

we should pay more attention when we assign List, dictionary(mutable object) as class variable. the change will impact all instance.

object type can be determined from obj.\_\_class\_\_ attribute or issubclass(obj,classname)/isinstance(obj,classname) functions.

**some basic rules** which are completely different from java patterns. java is the strong compiled language. patterns are directly related to our code

quality and maintenance. in most case of java class, the properties are defined as private to avoid directly access. Encapsulation makes sure what can be exposed to

outside what should be private for properties and methods. **but in python, productivity is its most important factor to consider. it rathers to define the properties**

**as public to simplify the code complexity**.

see example below:

class Dog:

kind = 'canine' # class variable shared by all instances

def \_\_init\_\_(self, name):

self.name = name # instance variable unique to each instance

>>> d = Dog('Fido')

>>> e = Dog('Buddy')

>>> d.kind # shared by all dogs

'canine'

>>> e.kind # shared by all dogs

'canine'

>>> d.name # unique to d

'Fido'

>>> e.name # unique to e

'Buddy'

. practise