

Python Q&A

Baboo J. Cui, Yangang Cao

June 20, 2019

Contents

1	Python Basics	2
2	Functions	2
3	OOP-Object Oriented Programming	2
4	Metaclass	3
5	Enum	5
6	Error, Debug and Test	5
7	Others	8
	7.1 Virtual Environment	8
	7.2 GUI	9
8	PyCharm	10

1 Python Basics

1. How to use “del”?
del can delete a variable, in the other word, reference.

2 Functions

1. How to define a function?
Easy...
2. How to specify the return type of a function?
Use syntax `def FUN_NAME () -> RETURN_TYPE`:
3. Disadvantages of declaration like `def func(a=[1,2])`?
Args are mutable, may cause unpredicted errors!

3 OOP-Object Oriented Programming

1. What is OOP?
Take object as the basic unit for programming, OOP vs procedure-oriented-programming(set of instruction)
2. Why OOP?
Everything are objects(a set of objects) and execution becomes the interaction between instances
3. What does an object include?
2 parts, properties(data) and methods(functions)
4. Class vs instance?
Class is template, instance is the specified class, instance is based on class
5. 3 characteristics of OOP?
Encapsulation, inheritance, polymorphism
6. How to add properties to an instance dynamically?
Use form `instance.var_name = val` to dynamically add a property, this won't work for other instances
7. How to add method to class dynamically?
Use form `class.method_name = func_name`, this may arise warning from the IDE
8. What is `__slots__` for?
A var in class which control the possible property names
9. What is the range of `__slots__`?
Only work for current class, won't work for subclass
10. What is the type of `__slots__`?
It is a tuple

11. What if both a class and its base have `__slots__`?
The possible property will be the union of the two `__slots__`
12. What is `@property` for?
A decorator which may help us to take a method like property
13. How many ways are there to define class methods?
3 ways, regular definition(related to self), decorated by `@classmethod`(related to cls) and `@staticmethod`
14. Difference between self and cls?
Self is bound to instance of class and cls is bound to class
15. Call of a regular method in a class?
Can be called by object but not class, or by class with first arg to be an instance of that class
16. Call of a class method in a class?
Can be called by both instance and class directly
17. Call of a static method in a class?
Can be called by both instance and class directly
18. Why static method in class instead of an independent func?
To indicate that the method belongs to the class and by inheritance, code can be managed better
19. What is MRO?
Method resolve order, a mechanism for inheritance

4 Metaclass

1. Biggest difference between static and dynamic language?
Static: definition is done during compiling process. Dynamic: definition are created during runtime
2. How class can be defined?
Two ways, by general declaration and by `type()` method
3. How to define a class by `type()`?
Use form `CLASS_NAME = type('NAME', (BASE_CLASS,), dic(METHOD1=FUNC1,...))`,
dynamic way to define a class
4. What does `type()` do?
It can show the type of an object. It also can be used to define a class
5. Difference between `__new__()` and `__init__()`?
`__new__()` create the obj and `__init__()` initialize the obj. Initialization comes after creation
6. How class is created(not asking how defined)?
During running time, essentially by function `type()`, tip: not from declaration!!!

7. Difference between general class definition and `type()`?
First one is in static way, the second one can be used in dynamic process, essentially, both share the same purpose
8. Dynamically create a class by static or dynamic language?
Easier for dynamic language(itself support), static language requires constructing the source code in the beginning.
9. What is the type of a class?
All class name itself has type: "type"
10. Relationship between type and object?
Type is subclass of object, metaclass of object is type. Both are created during the execution of interpreter
11. What is metaclass?
An class that controls how another class is defined, can be considered as template for other classes
12. What is the parent class of metaclass?
Must be type, cannot be object
13. How class derived from metaclass is created?
By calling `type.__new__(mcs, name, bases, attrs)`, it is the return of `__new__()` function in the metaclass!
14. How metaclass is defined?
Name end with Metaclass by convention(not necessary), inherit from type, define `__new__()` to control how other classes are created
15. What type is name in `__new__(mcs, name, bases, attrs)`?
It is str, who has value of the name of the class that take it as template
16. What type is bases in `__new__(mcs, name, bases, attrs)`?
It is tuple, a tuple that contains the parent classes in the target class
17. What type is attrs in `__new__(mcs, name, bases, attrs)`?
It is dict, has form `var_name: value`, var could be either function or properties
18. How are args in `__new__(mcs, name, bases, attrs)` passed?
When interpreter reading a class, it will use `type()` to create a class, and args are passed in the conventional way
19. Does any class has a corresponding metaclass?
Yes, and the metaclass is usually implicitly inherited
20. What is abstract class?(Not that important in python)
Class abstraction from many classes with certain similarities, it has a higher abstraction, a template for other classes
21. How to declare an abstract class in python?
First from `abc` import `abstractmethod`, `ABCMeta`, then declare a class with arg `metaclass=ABCMeta` and decorate method with `@abstractmethod`

22. Characteristics of abstract class?
Methods only have declaration, no implementation. Cannot be instantiated.
Must have abstract method and must be overridden latter.
23. What is interface class?
Like header, cant be instantiated, only contains method declaration, contains methods, properties, event etc. . . Doesnt contain constants etc. . .

5 Enum

1. What is enum class?
A enumeration, just list everything, like a key value system linked by equality, usually for constants
2. How to create a enum class?
First from enum import Enum, and then create a class inherit from Enum
3. How to get key of enum?
Directly use dot “.” or by enum_name[‘KEY_VAL’]
4. How to get key value in a enum?
By enum.KEY_VAL.value
5. What is @unique for?
Make sure that both key and value won’t repeat! (bi-jective)
6. How to import unique?
Use statement from enum import unique
7. What is enum generally for?
For finding key by value

6 Error, Debug and Test

1. What is bug?
Any unexpected thing, bug must be repaired
2. What causes bug?
Programming error, wrong input, unexpected condition(disk is full. . .)
3. What is python pdb?
A way of debug, python debug
4. What is error code(value)?
When something go wrong, there will be a specific return value like return -1
5. Disadvantages of error code?
Mix the error code and general return value together
6. Try, except, else, finally?
Try to execute whats in try, if any error, jump to except and else part will be execute if no error in try, then goes to finally(optional)

7. What if there might be more than a type of error?
One try can contain more than one except block
8. How to write except part?
Write form except: or except ERROR_TYPE as e:
9. Except ZeroDivisionError as e:, what type are they?
ZeroDivisionError has type: type since it is a class, e has type:<class 'ZeroDivisionError'> since it is an instance of the error class
10. What is the base class of all error classes?
All are inherited from BaseException
11. Range of try...except works?
Function contains it and any outer part that contains them, catch the error at nice position will be okay, don't need to put it everywhere
12. What if error doesn't caught by any except?
It will be thrown upper until caught by Python interpreter to print error and stop the program
13. How to read Traceback (most recent call last):
It shows that there are error and goes from top to bottom, the last line show the real reason
14. How to record error?
Import logging and add logging.exception(e) to output error and finish the program
15. What essentially an error is?
It is an instance of a class
16. How to throw an instance of error?
Use raise, need to raise an instance of class that is well designed that inherit from some error class
17. What could raise do if put into the block of except?
Convert one error to other type, should be logically reasonable
18. Easiest way of debugging?
Use print(), have to delete it after debugging...sad
19. Syntax of assert?
Have form assert CONDITION, ERROR_MESSAGE, condition should be true, otherwise will be error
20. How to stop assert statement during execution?
Execute .py file with form \$ python -O xxx.py
21. How to log info?
Use form logging.info(String)
22. How to config the level of logging?
Add code logging.basicConfig(level=logging.INFO) after import logging

23. Level of logging?
There are 4: debug, info, warning, error, the higher level you set, the lower cases will be ignored
24. How to start a program by pdb?
Use form `$ python -m pdb xxx.py`
25. How to see code in pdb?
Use command: `"l"`
26. How to execute code one line by another?
Use command: `"n"`
27. How to get value of a var in pdb?
Use command: `"p VAR_NAME"`
28. How to quit pdb?
Use command: `"q"`
29. How to use `pdb.set_trace()`?
Put it at position where might have error, it will pause the program there , use `"p Var"` to debug and press `"c"` to continue
30. What is the best way of debugging?
Ultimately... logging
31. What is TDD?
Short for Test-Driven-Development
32. What is unit test?
Check if a module, function or class work correctly, put all test conditions in a module, after revision, check if all conditions could pass the test
33. Advantage of unit test?
It can almost guarantee that the behavior of code is correct
34. What does a test unit class inherit from?
From `unittest.TestCase`
35. Purpose of methods in unit test?
By convention, methods start with `"test"` are test methods, otherwise not test methods which won't be executed during test
36. What does `assertEqual()` do?
Has form `self.assertEqual(abs(-1), 1)`, check if expected output equals to target output
37. What does `assertRaises()` do? Has form with `self.assertRaises(ERROR_TYPE):` BLOCK, if do anything in BLOCK, there will be `ERROR_TYPE` will be thrown
38. How to run unit test by coding?
Directly use statement: `unittest.main()`
39. How to run unit test in console?
By command: `python -m unittest xxx.py`

40. What are setUp() and tearDown() functionality?
They will be executed before and after a test
41. Which module is used for doc test?
Use import doctest
42. How to run doc test?
Use statement doctest.testmod()
43. Where the doc test code should be?
Within the triple comments: ““xxx”””
44. How to write test doc?
All statement start with >>> STATEMENT(could be more than one) and then the next line follow the output

7 Others

7.1 Virtual Environment

1. What is virtual environment for?
To build isolated environment for different programs. Different programs may depend on different python version and packages
2. What command is used to build virtual env?
Use virtualenv
3. How to install virtualenv?
Use command \$ pip3 install virtualenv by default, or \$ conda install virtualenv(not recommend for conda, use create)
4. How to build up a virtual env?
Use command \$ virtualenv no-site-packages ENV_NAME, where the ENV_NAME will generate a new folder that contains everything
5. What is no-site-packages for?
Avoid the copy of the third party package, make the env very clean
6. How to activate virenv?
Use command \$ source ENV_PATH/bin/activate to activate
7. Where the packages will be installed in virenv?
At path: ./lib/pythonx.x/site-packages/
8. How to deactivate a virtual env?
Use command \$ deactivate
9. How to show all installed packages?
Use command \$ pip list
10. How to check packages installed under conda?
Use command \$ conda list
11. How to check existing env?
Use command \$ conda env list or \$ conda info -e

12. How to check info of conda?
Use command `$ conda info`
13. How to create virtual env by conda?
Use command `$ conda create -n ENV_NAME python=3.6`, python version must be added!
14. How to activate virenv by conda?
Use command `$ conda activate ENV_NAME`
15. How to quit virenv by conda?
Use command `$ conda deactivate`
16. How to delete virenv by conda?
Use command `$ conda remove -n ENV_NAME all` (2 - before all)
17. How to install a package?
Use command `$ conda install PACK_NAME`
18. How to delete a package?
Use command `$ conda remove PACK_NAME`, this is intent to delete package under that environment
19. How to update conda?
Use command `$ conda update -n base -c defaults conda`
20. How to install requirement.txt?
Use command `$ pip install -r requirements.txt` or by `$ conda install -file requirements.txt`
21. How to generate requirement.txt for virenv?
Use command `$ pip freeze > requirement.txt`
22. How to add Tsinghua source?
Use command `$ conda config - -add channels`
`https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/`
`$ conda config - -add channels`
`https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main/`
`$ conda config - -set show_channel_urls yes`
23. How to reset source?
Use command `$ conda config - -remove-key channels`

7.2 GUI

1. Any packages for GUI?
Tk, wxWidgets, QT, GTK...
2. Advantage of TK?
It can be used directly
3. Which language is TK based on?
Based on TCL

4. How to import TK?
Add statement from tkinter import *, MUST be *
5. Which class is Application inherit from?
Class Frame
6. What args should `__init__()` of Application contains?
Two args: self and master=None
7. What is done in `__init__()`?
3 things: `Frame.__init__(self, master)`, `self.pack()`, `self.createWidgets()`,
(declaration and pack)
8. What is widget?
Any GUI object in TK is known as widget
9. What is `pack()` for?
Add widget into GUI container and achieve layout, after declare a component,
`pack()` function is mandatory
10. Functions for layout?
2 functions, `pack()` for easy layout and `grid()` for more complex ones
11. What is label API?
Use statement `Label(self, text="TEXT")`, and then `pack()`
12. What is button API?
Use statement `Button(self, text="TEXT", command=self.COMMAND)`, and
then `pack()`
13. What is entry API?
Use statement `Entry(self)`, and then `pack()`
14. How to use messagebox?
First import `tkinter.messagebox` as `messagebox`, and follow `messagebox.showinfo('TITLE', 'MESS_DISP')`
15. How to instantiate an application?
3 steps, create instance, set `instance.master.title("TEXT")` and start `instance.mainloop()`

8 PyCharm

1. How to search?
Use shortcut `cmd+O`, then easy
2. Meaning of set a folder as resources root?
For searching file, will add the path to search path