Python Tutorial for ELEG4701

Term 2023-2024

\* This tutorial consists of basic Python guidelines, introduction to basic programming paradigm, and Advanced programming skills

Recommended Reading:

[Docs of Python (English)](https://docs.python.org/3/tutorial/)

[Liao's Python Tutorial (Example codes)](https://github.com/michaelliao/learn-python3/tree/master/samples)

Part 0. Guideline of self-evaluation

If you are familiar with Python, just jump to part 2

Environment and install

Normally we use IDE (Integrated Development Environment) rather than simple text editor to write and excute code. One popular IDE for Python is Microsoft VS Code.

To install VS Code in Ubuntu,

1. go to its official website: https://code.visualstudio.com/download
2. download .deb file
3. open the folder where the .deb package is, open a terminal there, execute

sudo dpkg -i code\_1.76.1-1678294265\_amd64.deb

1. input your system password and press enter to install VS Code.

Open VS Code, go to Extensions, and install the Python extension.

Q0. If you know the purpose of the following code, jump to part 2

c = "#" + "".join([random.choice('0123456789ABCDEF') for x in range(6)]) print(c)

You should understand the codes below

Q1. Variable, Type, and Equation

a = 1

b = '1'

print(a == b, type(a), type(b), type(a == b))

# the result is "False, <class 'int'>, <class 'str'>, <class 'bool'>"

Q2. Logic and Functions

def fun1():

print("the first funcntion")

for i in range(3): print('iteration', i) if i == 1:

print("print extra stuffs") if i == 3:

print("it will never happens because range stops in 3")

def fun2(sth):

print('the second funcion, input args is', sth)

def fun3(\*args, \*\*kargs):

# you don't need to write all args

# you can use \*args and \*\*kargs to hide them print(args)

if 'kw' in kargs:

print("has key arg kw:",kargs['kw']) else:

print("other kw:", kargs)

def fun4():

return 4

def fun5(functor): print(functor())

def fun():

fun1() fun2('abcd')

fun3(1, 2, 3, 4, 5, kw=6)

#in this case, the input is a number, because the fun4 is called before being a argument

fun3(fun4())

# in this case, the input is a function fun5(fun4)

fun() # call function here # the result is following:

'''

the first funcntion iteration 0

iteration 1

print extra stuffs iteration 2

the second funcion, input args is abcd (1, 2, 3, 4, 5)

has key arg kw: 6 (4,)

other kw: {} 4

'''

Q3. Inheritance and composition

class BaseClass():

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

        self.basic\_sth = 0

self.basic\_sth = 0

def print\_self(self):

print(type(self) ,'Functions will be inherited by default')

def lalala(self):

print(type(self), 'it could not be inherited because you write in the functin name so it is a private method')

@staticmethod def sfunc():

print('this function do not needs self, because it belongs to Class, not

object')

print('call it by BasicClass.sfunc()')

class A(BaseClass):

def init (self):

super(). init () # called init of super class self.a\_sth = 1

class B(BaseClass):

def init (self):

# if you do not write super(). init () here, you can not use basic\_sth from the baseclass

self.b\_sth = 2

c = BaseClass()

a = A()

b = B()

c.print\_self() a.print\_self() b.print\_self()

BaseClass.sfunc() # or you can use a.sfunc()

b.sfunc()

c.sfunc()

print(c.basic\_sth) print(a.basic\_sth, a.a\_sth) print(b.b\_sth)

# try to imagine the output

Part 1. Basic Programming Paradigm

As a python beginner, you only need to understand process-oriented and object- oriented. I will show how to work with different paradigms using the example of 'how to put elephants in the fridge.'

# A. Process oriented programming

def main():

# the flow chart is always:

# do [action] to [sth] open\_fridge() let\_in(elephant\_data) close\_fridge()

# B. Object oriented programming class BaseFridge:

def init (): self.container = []

def open(self) pass

def close(self) pass

def put(self, target): self.open() self.container.append(target) self.close()

class Siemens(BaseFridge): pass

class Midea(BaseFridge): pass

class Animal: pass

class Lion(Animal): pass

class Elephant(Animal): pass

def main():

# in this case,

# The fridge can hide thier open, and close fucntion

# Besdies, you can made many different fridge, and all of them can do the same things.

e = Siemens() #or Midea() e.put(Elephant())

Part 2. Advanced programming skills (Briefly)

# Ways to accelerate calculation

\*\* Numpy->Cupy

\*\* Numpy->dll

\*\* Numpy->Cuda

\*\* Numpy->pytorch-gpu

\*\* Cython

(CuPy provides a simple way to accelerate NumPy code on NVIDIA GPUs. By making just a few modifications to swap out NumPy for CuPy, you can experience order-of-magnitude speedups on array computations. This performance boost allows you to work with much larger datasets and models, enabling more advanced machine learning and scientific computing.)

# Using pyinstaller to package your program

\*\* pyinstaller -D -w --key [your key] [xxxx.py]

\*\* pyinstxtractor

\*\* Compile your python with cython

# Using PySide2(pyqt) to make a user interface

<img src="imgs/pyqt\_example.png" width = "300" height = "300" alt="Pyqt\_Eaxmple" align=center />

This is an pyqt demo for 1-D signal annotation. The codes writing for user interfaces is usually object-oriented. Because this coding style can maintain a user interface easily.

At the same time, because Python has 'duck type', it is easier to write an interface than other languages like Cpp or Java.

Here is only a brief introduction. If you are interested in making UI, you can also read how to use languages designed for making UI like html, qml, and other designed (they can communicate with Python easily).

# Try some interesting works

I collected some example scripts of the following works, wish it can recall your interests.

* + [Taichi, a programming language designed for high-performance computer graphics.](https://github.com/taichi-dev/taichi)

A black and white image of a three ovals

Description automatically generated

* + [A visualization script of Rapdily-exploration Random Tree](https://github.com/howardjchen/RRT)

A screenshot of a computer

Description automatically generated

* + [A script draws Doraemon](https://github.com/PerpetualSmile/Python-Painting-Doraemon)

A screen shot of a computer

Description automatically generated