

**KIX 3004: PYTHON PROGRAMMING**

**Semester 1, Session 2018/2019**

**Assignment 1**

Name : Koay Hong Vin

Matric No. : KIE160111

Lecturer : Mr. Mahazani Bin Mohamad

Date of Submission : 26 December 2018

**Q1. Comparing with C programming language, what are the advantages and disadvantages of Python.**

Answer:

Even though C programming languages dominate the embedded systems programming while Python, which is a high-level programming language, contains a lot of strengths that are able to be used in embedded system.

**The advantages of Python over C programming language:**

* Python has object-oriented capabilities which C does not. This means that Python will probably win in larger projects.
* Python has a lot of extensive support libraries, including web services, protocols, mathematical tools and Internet.
* Python has integration feature that allow different languages to be called, for example Jython for Java and Cython for C.
* Python is widely used in prototyping, from machine learning to computer vision, libraries are available for build and testing before releasing to the public.
* Python has greater scalability when building multi-protocol applications.

**The disadvantages of Python over C programming language:**

* Python is an interpreted language while C is a compiled language. Interpreted language needs bulkier software on system to run the code, which is vital when the memory is limited.
* Python is slower compared to C since Python executes with the help of interpreter instead of compiler.
* Python is not preferred in memory-intensive tasks and CPU-intensive tasks while C is favourable in memory and CPU intensive tasks.

To prove the computing speed, a small test is carried out. A simple snippet of code is used to simulate. The chosen function is factorial using recursion.

|  |
| --- |
| **Python** |
| import timeit  def factorial(n):  if n == 1:  return n  else:  return n\*factorial(n-1)  print(timeit.timeit('factorial(10)',globals=globals())) |

The computational time is 12.226305947 seconds.

|  |
| --- |
| **C++** |
| #include <iostream>  #include <chrono>  using namespace std;  using namespace std::chrono;  long factorial(int n)  {  if(n==0)  return 1;  else  return (n\*factorial(n-1));  }  int main()  {  auto start = high\_resolution\_clock::now();  factorial(10);  auto stop = high\_resolution\_clock::now();  duration<double> elapsed = stop - start;  cout<<elapsed.count();  } |

The computational time is 1.92e-07 seconds.

The two code is run in the same configuration of computer, and is tested several times and found to be consistent. This simple code can be shown that Python do take a longer time to compute a simple recursion function.

**Q2. Despite being relatively slow (for eg: as compared to C), Python is gaining popularity, especially in the scientific community, and as a matter of fact, many well-known universities include Python as part of their syllabus. Explain why is this so?**

Answer:

Python is all about end-user experience. As we all know that Python is a higher-level language, which means it abstracts the details of computer from you, allows you to write programs in a way closer to humans think.

Despite being slower in executing, the Python community is big. A lot of extensions have included Python library allowing the programmer to easily prototype their application. From scientific research to day-to-day uses, Python is able to cover all aspects.

Even if using more memory-effective language, such as C or Java, but badly designing would still be slow. Gone are the days where server components or memory are expensive, with a lot of new services with lower costs, having an easily understand language to prototype and build an application is more efficient.

All in all, it is still back to the development time. I believe that Python is more productive compared to other language. Python helps to abstract a lot of things, helping you focus on what you are trying to code, preventing you deciding a vector or an array!

**Q3. One of the attractions of Python, is having many useful and easy to use libraries or modules (eg: NumPy, SciPy, etc). Pick up 5 popular modules, describe briefly the modules, how they are used (eg: give example codes) and if there is connection/relation between the modules, state it.**

Below are the modules that is commonly used in data wrangling and visualization.

1. **Numpy**

Numpy allows Python to support a larger, multi-dimensional arrays and matrices. It also comes with a large library of high-level mathematical functions to operate on these arrays.

|  |
| --- |
| import numpy as np  a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])  b = a[:2, 1:3]  print(a[0, 1])  b[0, 0] = 77  print(a[0, 1]) |

Output:

|  |
| --- |
| 2  77 |

The code above illustrates how easy it is to alter an array. Variable a is used to create a rank 2 array with 3 rows ad 4 column. Variable b uses slicing to pull out the subarray consisting the first 2 rows and columns 1 and 2, which makes b a 2 by 2 array. A slice of an array is a view into the same data, where modifying it will also modify the original array. b[0, 0] is the same piece data of a[0, 1].

1. **SciPy**

Scipy is a collection of mathematical algorithms and convenience functions built on the Numpy extension of Python. It adds significant power to the interactive Python session by providing the user with high-level commands and classes for manipulating and visualizing data.

|  |
| --- |
| import scipy.integrate as integrate  result=integrate.quad(lambda x: 3\*x\*\*2+3,0,2)  print(result) |

Output:

|  |
| --- |
| (14.000000000000002, 1.5543122344752194e-13) |

The code above is used to calculate the function, . The function quad is used to integrate the unction of one variable between the intervals. The result is a tuple, with the first element holding the estimated value of the integral and the second element holding the upper bound on the error.

1. **Pandas**

Pandas is a library written for data manipulation and analysis in Python. It offers data structures and operations for manipulating numerical tables and time series.

|  |
| --- |
| import pandas as pd  from numpy.random import \*  d = pd.date\_range('20181212', periods=6)  df = pd.DataFrame(randn(6,4), index=d, columns=list('ABCD'))  print(df) |

Output:

|  |
| --- |
| A B C D  2018-12-12 0.459859 0.286056 -0.459610 -0.901669  2018-12-13 -1.974383 -0.196662 -0.816543 1.433066  2018-12-14 -1.890922 0.384868 0.809074 0.815495  2018-12-15 -0.455030 0.536437 0.888036 0.645764  2018-12-16 -1.015884 -1.301203 -1.444979 1.707913  2018-12-17 0.817272 -0.562533 0.570537 -0.758527 |

The code above creates a Data Frame by passing a Numpy random generated array with date being the index and label the columns as A, B, C and D.

1. **Matplotlib**

Matplotlib is a Python 2D plotting library. It is able to generate plots, histograms, power spectra, bar charts and many more with few lines of code.

|  |
| --- |
| import matplotlib  import matplotlib.pyplot as plt  import numpy as np  t = np.arange(0.0, 2.0, 0.01)  s = 1 + np.sin(2\*np.pi\*t)  fig, ax = plt.subplots()  ax.plot(t, s)  ax.set(xlabel='Time', ylabel='Voltage',title='Sine Wave')  ax.grid()  plt.show() |

Output:

|  |
| --- |
|  |

The code simply plot the graph of from 0 to 2 with step interval of 0.01.

1. **Bokeh**

Bokeh is an interactive visualization library that targets modern web browsers for presentation. Its goal is to provide elegant, concise construction of versatile graphics, and to extend this capability with high-performance interactivity over very large or streaming datasets.

|  |
| --- |
| import numpy as np  import scipy.special  from bokeh.layouts import gridplot  from bokeh.plotting import figure, show, output\_file  mu, sigma=0,0.5  measured=np.random.normal(mu,sigma,1000)  hist, edges = np.histogram(measured, density=True, bins=50)  x = np.linspace(-2, 2, 1000)  pdf = 1/(sigma \* np.sqrt(2\*np.pi)) \* np.exp(-(x-mu)\*\*2 / (2\*sigma\*\*2))  cdf = (1+scipy.special.erf((x-mu)/np.sqrt(2\*sigma\*\*2)))/2  p=figure(title='Normal Dist',background\_fill\_color='#fafafa')  p.quad(top=hist,bottom=0,left=edges[:-1], right=edges[1:],  fill\_color="navy", line\_color="white", alpha=0.5)  p.line(x, pdf, line\_color="#ff8888", line\_width=4, alpha=0.7,  legend="PDF")  p.line(x, cdf, line\_color="orange", line\_width=2, alpha=0.7,  legend="CDF")  p.y\_range.start = 0  p.legend.location = "center\_right"  p.legend.background\_fill\_color = "#fefefe"  p.xaxis.axis\_label = 'x'  p.yaxis.axis\_label = 'Pr(x)'  p.grid.grid\_line\_color="white"  show(p) |

Output:

|  |
| --- |
|  |

The code illustrate the normal distribution histogram plot with and .