# Assignment 1

Varun Sundar, EE16B068

January 27, 2018

*EE2703: Applied Programming*

## 0 Introduction

### Conventions

1. We are using Python 3, GCC for C
2. Underscore naming vs Camel Case
3. PEP 25 style of coding (non-tensorflow)
4. All c code is executed in seperate c files, which may be found under respective folders

## 1 Question 1

Pythonic Code (Brute, Optimised):

```
In [6]: %time
        # Brute
        from pprint import pprint
        n=1
        n_old=1
        L=[]
        L.append(1)
        L.append(2)
        for k in range(3,11):
            '''new=n+n_old
            n_old=n
            n=new'''
            n,n_old=n+n_old,n
            L.append(n)
        pprint(dict(zip(range(1,11),L[:11])))

CPU times: user 2 µs, sys: 0 ns, total: 2 µs
Wall time: 5.72 µs
{1: 1, 2: 2, 3: 2, 4: 3, 5: 5, 6: 8, 7: 13, 8: 21, 9: 34, 10: 55}
```

```
In [9]: # Optimised
        %time
        series=[]
        series.append(1)
        series.append(1)
        [series.append(series[k-1]+series[k-2]) for k in range(2,100000)]
        series=dict(zip(range(1,11),series))
        print(series)
```

```
CPU times: user 4 µs, sys: 1 µs, total: 5 µs
Wall time: 9.06 µs
{1: 1, 2: 1, 3: 2, 4: 3, 5: 5, 6: 8, 7: 13, 8: 21, 9: 34, 10: 55}
```

```
In [4]: %%bash
        # Fibonacci Series
        gcc -o c_1 c_1.c
        ./c_1
```

```
1, 1
2, 1
3, 2
4, 3
5, 5
6, 8
7, 13
8, 21
9, 34
10, 55
```

## 2 Question 2

Pythonic And C methods

```
In [8]: import math
        import math
        pi=math.pi
        modf=math.modf
        n=[0]
        n[0]=0.2
        alpha=pi
        for k in range(1,999):
            n.append(modf((n[k-1]+pi)*100)[0])

        # Lets print only around 30 values to keep our report clean
        print ("Lenght of resultant series...",len(n))
        pprint (n [:30])
```

```
Lenght of resultant series... 999
[0.2,
 0.1592653589793258,
 0.08580125691190688,
 0.7393910501700134,
 0.09837037598066445,
 0.9963029570457707,
 0.7895610635563912,
 0.11537171461844764,
 0.6964368208240899,
 0.8029474413883122,
 0.4540094978105458,
 0.5602151400339039,
 0.18077936236971937,
 0.23720159595126233,
 0.8794249541055592,
 0.10176076953524671,
 0.3353423125039967,
 0.6934966093789967,
 0.5089262968789967,
 0.05189504687899671,
 0.3487700468789967,
 0.03627004687899671,
 0.7862700468789967,
 0.7862700468789967,
 0.7862700468789967,
 0.7862700468789967,
 0.7862700468789967,
 0.7862700468789967,
 0.7862700468789967]
```

```
In [19]: %%timeit
         # Optimised
         n=[0.2]
         [n.append(modf((n[k-1]+pi)*100)[0]) for k in range(1,999)]
```

```
308 μs ± 8.56 μs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

**Now for the C implementation.**
*Note that the corresponding file (here c_2) maybe found under the same directory.*

```
In [22]: %%bash
         # Random Series
         gcc -o c_2 c_2.c
         ./c_2
```

```
0, 0.200000
1, 0.159265
2, 0.085801
3, 0.739391
4, 0.098370
5, 0.996303
6, 0.789561
7, 0.115372
8, 0.696437
9, 0.802947
10, 0.454009
11, 0.560215
12, 0.180779
13, 0.237202
14, 0.879425
15, 0.101761
16, 0.335342
17, 0.693497
18, 0.508926
19, 0.051895
20, 0.348770
21, 0.036270
22, 0.786270
23, 0.786270
24, 0.786270
25, 0.786270
26, 0.786270
27, 0.786270
28, 0.786270
29, 0.786270
30, 0.786270


c_2.c:3:9: warning: 'M_PI' macro redefined [-Wmacro-redefined]
#define M_PI acos(-1.0)
        ^
/Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX10.13.
#define M_PI        3.14159265358979323846264338327950288   /* pi             */
        ^
1 warning generated.
```

## 3 Question 3

Comes under File IO. Evaluated code is Python.

File Name: the-hound-of-the-baskervilles.txt (Same Directory)

```
In [10]: import re
```

```
        import string

        filename="the-hound-of-the-baskervilles.txt"
        with open(filename,"r") as f:
            contents=f.read()

        regex = re.compile('[%s]' % re.escape(string.punctuation))
        contents=regex.sub('', contents)

        words=contents.split()
        d={} # creates an empty dictionary

        # Update a word
        def do(w):
            global d
            if w in d: # this also works, and is quite fast
                d[w]+=1
            else:
                d[w]=1

        # Because List comprehensions!
        [do(w) for w in words]
        print ("We've got our words")
```

```
We've got our words
```

```
In [12]: # Now lets display the count
         # Again lets keep our report clean so,display only top 40 most used words
         [print(k,"...\t",d[k]) for k in sorted(d,key=d.get,reverse=True)[:40]]
         print()
```

```
the ...          3128
of ...          1642
and ...          1542
I ...           1465
to ...           1432
a ...           1254
that ...           1082
in ...           881
was ...           792
it ...           784
you ...           752
he ...           718
his ...           659
is ...           621
have ...           529
had ...           501
```

```
with ...          427
for ...          420
my ...          420
which ...           419
as ...          383
not ...          365
we ...          350
at ...          345
be ...          341
this ...          332
me ...          324
upon ...          314
him ...          313
from ...          280
but ...          262
The ...          258
Sir ...          248
said ...          241
one ...          227
on ...          222
been ...          221
so ...          216
by ...          215
all ...          210
```

End of Assignment.