Python for Engineers Assignment 2018

Answer all questions.
すべての質問に答えなさい。

Submission deadline 07/16 (July 16).
提出締め切り07/16 (7月16日）。

Please submit your answer by email.
どうぞ、メールによってあなたの答えを提出してください。

You may use the following file formats:
あなたは以下のファイル形式を使ってもよい:

- .py
- .ipynb
- .txt

You may use multiple files.
あなたは複数のファイルを使ってもよい。

You will be awarded points for: あなたは以下のためのポイントを授与される：

- correct solution. 正しい答え。
- concise solution. 簡潔な答え。
- optimising your code by importing relevant library functions. 外部や基準のライブラリやパッケージを利用します。
- writing functions to avoid repetition. 重複を防ぐために関数（functions)を使いなさい。
- using comments to explain what your code does. あなたのコードを説明するために、コメントを用いる。

Part A : Data Analysis　データ分析

Question 1

(2 marks)
Import the data from the url to a Pandas data frame:
`Pandas`を使って、以下urlからデータを読み込めよ
https://raw.githubusercontent.com/justmarkham/DAT8/master/data/u.user
(https://raw.githubusercontent.com/justmarkham/DAT8/master/data/u.user)

In [15]:

```
import pandas as pd
import numpy as np

url = 'https://raw.githubusercontent.com/justmarkham/DAT8/master/data/u.user'

data = pd.read_csv(url, sep='|', index_col='user_id')
```

Question 2

(2 marks)
Remove all entries with occupation `other` or `retired` from the data set.
職業occupationの中にotherとretiredに関連するデータを取り除く。

In [16]:

```
not_other = data["occupation"] != 'other'
not_retired = data["occupation"] !='retired'

data = data[not_other & not_retired]

#data = data[((data["occupation"] != 'other') & (data["occupation"] !='retired
'))]

display(data.head())
```

	age	gender	occupation	zip_code
user_id				
1	24	M	technician	85711
3	23	M	writer	32067
4	24	M	technician	43537
6	42	M	executive	98101
7	57	M	administrator	91344

Question 3

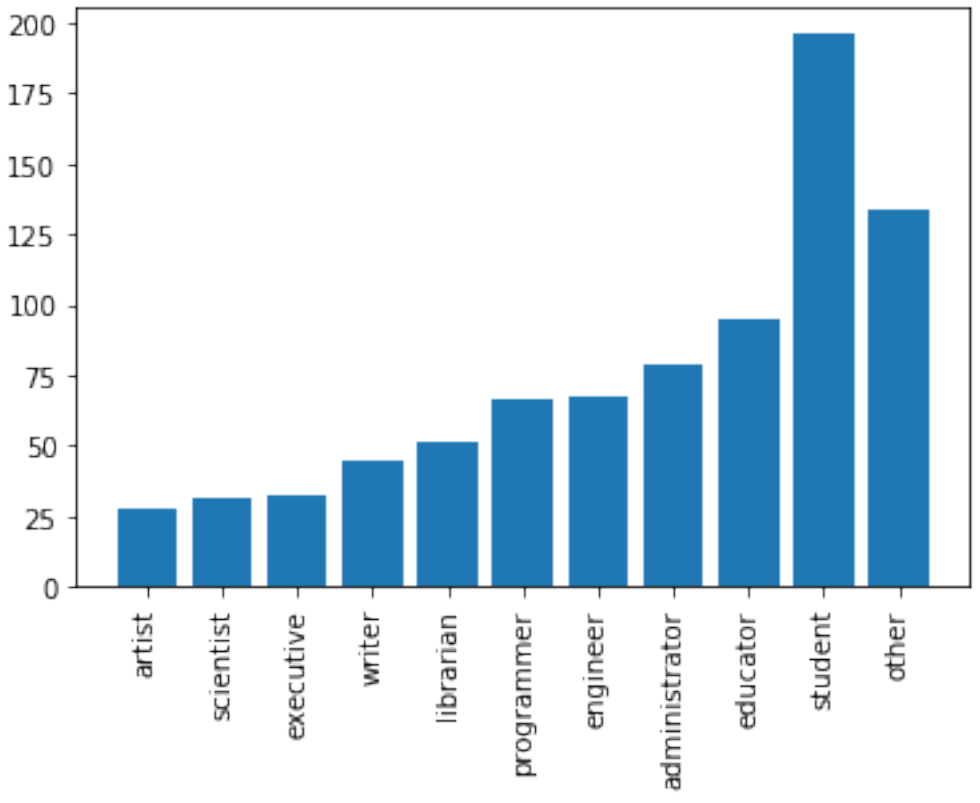
(6 marks)
Plot a bar chart showing:
棒グラフをプロットしなさい：

- the number of people per occupation for the 10 occupations that occur most frequently in the data set.
  取り除かない各職業の総人数を計算し、総人数最も多い10つの職業を取り出し、各職業の総人数を棒グラフで書きなさい。
- plot an additional bar, labelled other, showing the sum of all remaining entries in the data set.
  その以外の職業の人数をまとめて、`other`という名前を付け、同じ棒グラフに書きなさい。

In [17]:

```
# new data frame with only one instance of each occupation
jobs = data.drop_duplicates('occupation')

# empty list to store workers that do each job
workers = []

# loop through each occupation
for job in jobs['occupation']:
    # new data frame : only one occupation
    population = data[data['occupation']==job]
    # store number of people with that occupation
    workers.append(len(population))


# add a new column to jobs data frame
jobs['workers']=workers

# sort the data frame by number of workers
jobs_sorted = jobs.sort_values(by='workers')

# select the top 10 occupations that occur most frequently in the original dat
a set
top10 = jobs_sorted.iloc[-10:, :]

# select the remaining occupations
others = jobs_sorted.iloc[:-10, :]

# count the number of workers not in the top 10 jobs
num_others = others.workers.sum()

x_vals = list(top10['occupation'])
x_vals.append('other')
y_vals = list(top10['workers'])
y_vals.append(num_others)


# Create an array with the position of each bar along the x-axis
x_pos = np.arange(len(x_vals))

# Produce bar plot
plt.bar(x_pos, y_vals);

# Replace the x ticks with the year group name
# Rotate labels 90 degrees
plt.xticks(x_pos, x_vals, rotation=90);
```

```
/Users/hemma/anaconda3/lib/python3.6/site-packages/ipykernel_launc
her.py:16: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pan
das-docs/stable/indexing.html#indexing-view-versus-copy
  app.launch_new_instance()
```



Question 4

(2 marks)
Find the:

- mean
- standard deviaion

of the age of people with the occupation: `administrator`.

`administrator`という職業`occupation`の人達を見つけ出す。その人達の年齢の

- 平均値
- 標準偏差

を求めよ。

In [18]:

```
mean_age = data[data['occupation']=='administrator']['age'].mean()

std_age = data[data['occupation']=='administrator']['age'].std()

print(f'The mean age of an administrator is: {mean_age}')
print(f'The standard deviation is: {std_age}')
```

```
The mean age of an administrator is: 38.74683544303797
The standard deviation is: 11.123396864533204
```

Part B : Movement of a Particle 粒子の動き

A particle moves along a curve.
粒子がある軌跡を沿って移動する。

The cartesian coordinates of the position of the particle (r_x, r_y) are:
粒子の位置（r_x、r_y）は時間tの変数である。その式は以下：

$$r_x = \exp\left(-\frac{t}{T_1}\right)$$
$$r_y = 2\cos\left(\frac{t}{T_2}\right)$$

t = time (s)

Constants:　その中に、Tは定数
$T_1 = 1$ s
$T_2 = 1/3$ s.

Question 1

(8 marks)

Plot the trajectory of the particle in the xy-plane over the time interval $t = 0$ to $t = 2\pi$ s.
時間$t = 0$から$t = 2\pi$までの粒子の軌跡を書きなさい。軌跡を表す点の数は任意。

On the graph, show the particle's locations at $t = 0$, $\pi/2$ and 2π using:
同じ図に、以下のものを使って、$t = 0$、$t = \pi$ / 2と$t = 2\pi$の粒子の位置を強調せよ：

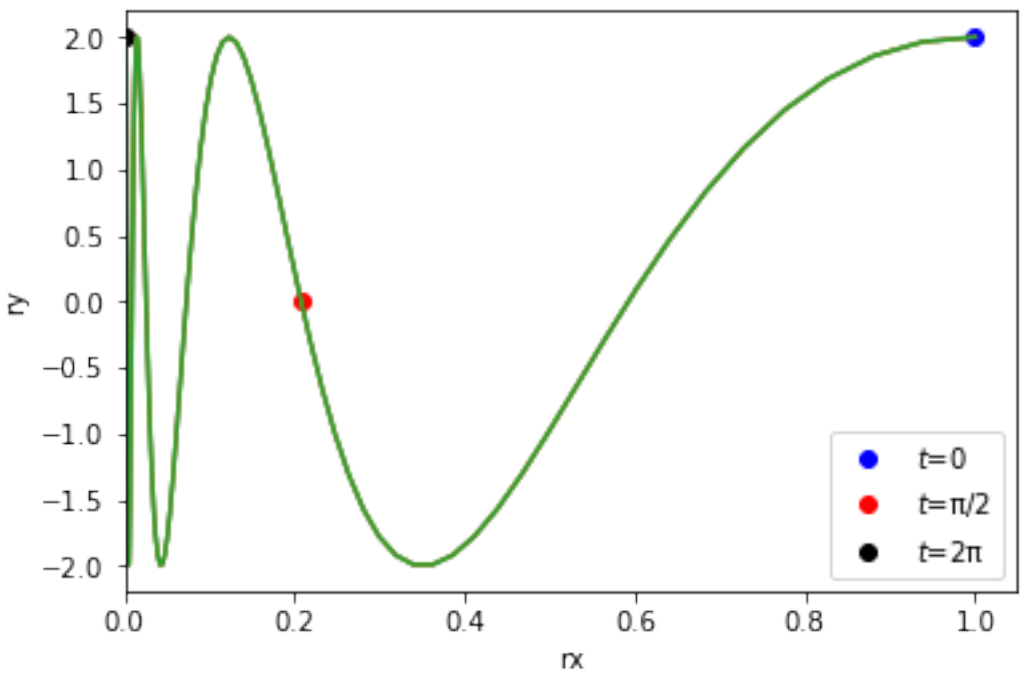
- a marker
- a label

In [49]:

```
import numpy as np
import matplotlib.pyplot as plt

T1, T2 = 1, 1/3

time = np.linspace(0, 2*np.pi, 100)

def rx(time):
    return np.exp(-time/T1)

def ry(time):
    return 2 * np.cos(time/T2)


plt.plot(rx(time), ry(time))
plt.plot(rx(0), ry(0), 'bo', label='$t$=0')
plt.plot(rx(np.pi/2), ry(np.pi/2), 'ro', label='$t$=π/2')
plt.plot(rx(2*np.pi), ry(2*np.pi), 'ko', label='$t$=2π')
plt.plot(rx(time), ry(time))
plt.plot(rx(time), ry(time))
plt.xlabel('rx')
plt.ylabel('ry')
plt.legend()
plt.xlim(xmin=0)
```

Out[49]:

```
(0, 1.0499066278634146)
```



Question 2

The x and y components of the particle's acceleration (a_x, a_y):
粒子の加速度（a_x、a_y）はの粒子の位置（r_x、r_y）の二階微分である：

$$a_x = v_x' = r_x''$$
$$a_y = v_y' = r_y''$$

velocity (v_x, v_y)
速度(v_x, v_y)

(a) Express the particles acceleration $(a_x、a_y)$ symbolically, i.e. show the equation.
粒子加速度（a_x、a_y）を計算せよ。方程式を示す。
(5 marks)

In [50]:

```
from sympy import diff, symbols, Symbol, exp, cos
import numpy as np

t = Symbol('t')

rx_sym = exp(-t/T1)
ry_sym = 2 * cos(t/T2)

ax_sym = diff(rx_sym, t, 2)
ay_sym = diff(ry_sym, t, 2)

print(ax)
print(ay)
```

```
exp(-t)
-18.0*cos(3.0*t)
```

(b) Plot the *magnitude* of the acceleration of the particle against time over the time interval $t = 0$ to $t = 2\pi$ s.
時間$t = 0$から$t = 2\pi$までの粒子の加速度の絶対値（マグニチュード加速度）を計算し、時間tに対する図を書きなさい。つまり、X軸は時間t、Y軸は加速度の絶対値。絶対値の式は以下
(5 marks)

The magnitude of acceleration, A:

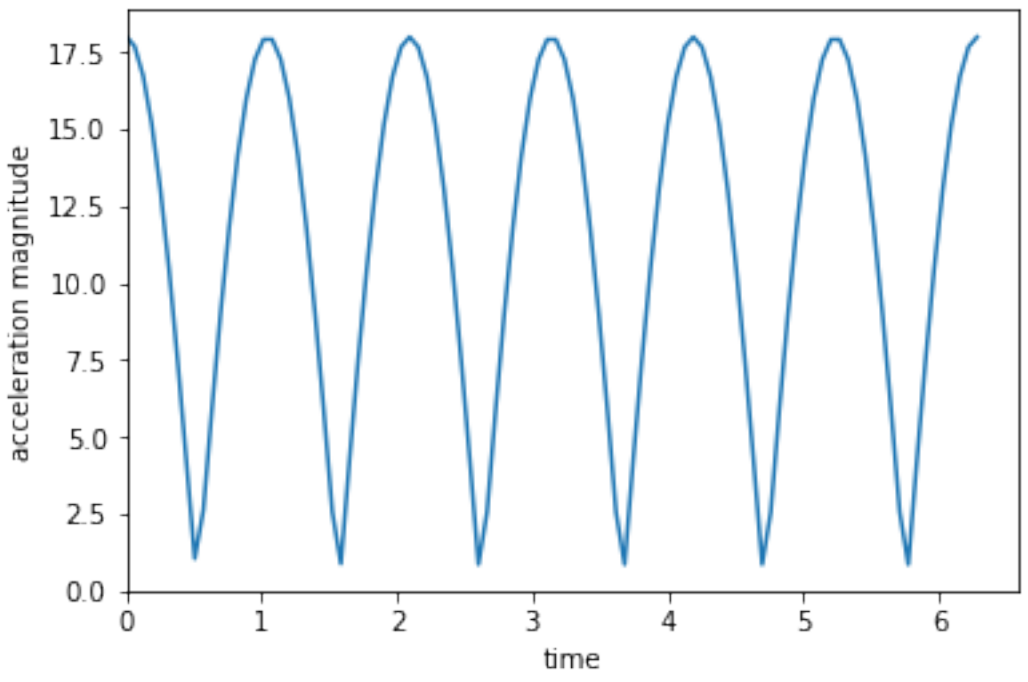
$$A = \sqrt{a_x^2 + a_y^2}$$

In [51]:

```
import numpy as np

#Method 1 : for loop
a_mag = []

for ts in time:
    a = np.linalg.norm([float(ax_sym.subs([(t, ts)])),
                        float(ay_sym.subs([(t, ts)]))])
    a_mag.append(a)



# Method 2 : list comprehension
#a_mag = [np.linalg.norm([float(ax.subs([(t, ts)])), float(ay.subs([(t, ts)]))
]) for ts in time]



plt.plot(time, a_mag)
plt.xlabel('time')
plt.ylabel('acceleration magnitude')
plt.xlim(xmin=0)
```

Out[51]:

```
(0, 6.5973445725385655)
```



Question 3

(a) Find the magnitude velocity of the particle at time $t = 1$ s.
$t = 1$ 時刻の粒子の速度の絶対値（マグニチュード速度）を見つけ出す。
(5 marks)

In [52]:

```
vx_sym = diff(rx_sym, t)
vy_sym = diff(ry_sym, t)

t1 = 1

vx1 = vx_sym.subs([(t, t1)])
vy1 = vy_sym.subs([(t, t1)])

v_mag = np.linalg.norm([float(vx1), float(vy1)])
print(v_mag)

# a_check = np.linalg.norm([float(ax.subs([(t, time)])), float(ay.subs([(t, ti
me)])])])
# print(a_check)
```

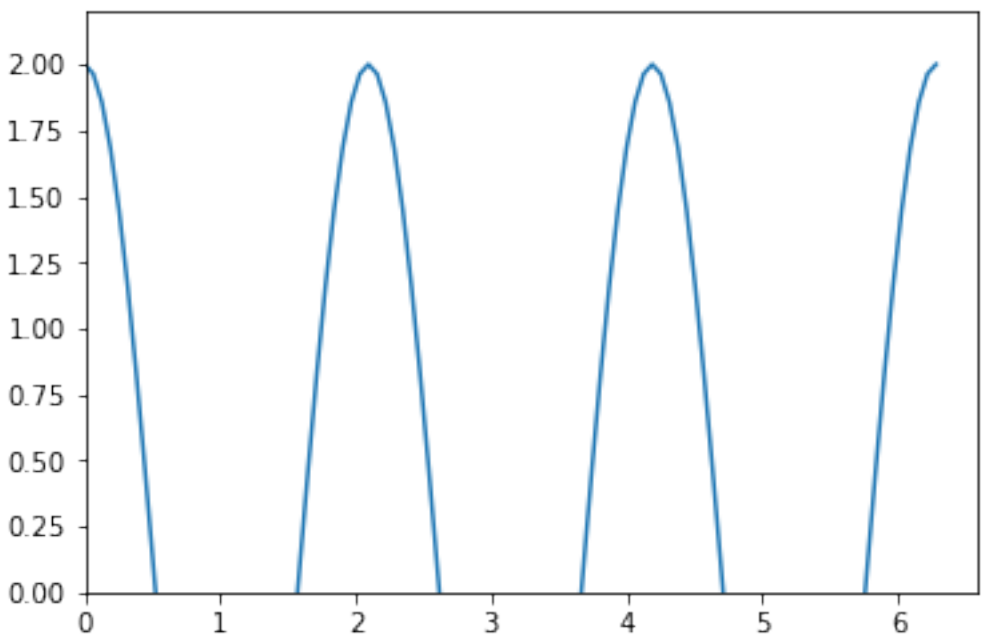
```
0.923184772150204
```

(b) Estimate the time/times when $r_y = 0$
$r_y = 0$を満足する時間（時間達）を逆計算せよ。
(5 marks)

In [53]:

```
from scipy.optimize import fsolve

time = np.linspace(0, 2*np.pi, 100)
plt.plot(time, ry(time))
plt.xlim(xmin=0)
plt.ylim(ymin=0)

guesses = [0.5, 2, 2.5, 3.5, 4.5, 5.7]

for guess in guesses:
    tsol, = fsolve(lambda ts: 2 * np.cos(ts/T2), guess)
    print(tsol,)
```

```
0.5235987755982989
1.5707963267948966
2.6179938779914944
3.6651914291880923
4.71238898038469
5.759586531581288
```



Question 4

(10 marks)
Animate the trajectory of the particle and save your work as a .mp4 file.
粒子の移動軌跡をアニメーション化し、.mp4ファイルとして保存しなさい。

In [61]:

```
# Example Solution
from matplotlib import animation, rc
from IPython.display import HTML

# Creates a figure window.
fig = plt.figure()

# Creates axes within the window
ax = plt.axes(xlim=(0, 1), ylim=(-2, 2))

# Object to animate
point, = ax.plot([1], [1], marker='o', ms=10)  # for points

T1, T2 = 1, 1/3

# Position of mass as function of time
def fun(time):
    x = np.exp(-time/T1)
    y = 2 * np.cos(time/T2)
    return x, y

def animate(i):
    x, y = fun(i/10)
    point.set_data(x, y)
    return (point,)

# Animates the data; 50 frames, 50ms delay between frames, blit=True : only re
-draw the parts that have changed.
anim = animation.FuncAnimation(fig, animate, frames=50, interval=50, blit=True
)

rc('animation', html='html5')

anim

writer = animation.writers['ffmpeg'](fps=15, bitrate=1800)
anim.save('movie.mp4', writer=writer)
```