SORAY CENGİZ
161024077

HW02:

10.1.1 fft-coefficients-usage-python

10.1.1.1 os_fft_operations.py

NOT: “python2” ve “python3”deki bazı farklardan dolayı bazı sorulara python3 base alınarak cevap verilmiştir ve bu cevaplar python2’ye göre yanlıştır, bu hatalar malesef zaman eksikliğinden dolayı düzeltilememiştir. (Bu durum biraz geç fark edilmiştir python2 base alınarak soruların cevaplanması gerekirdi.)

Örneğin;

type(range(10)) -> python2 buna list

type(range(10)) -> python3 buna range return edecektir.

Dolayısı ile python2’de range(10) + range(10) hata çıkartmaz fakat python3’de bu hata veriyor çünkü range tipi için “+” overload edilmemiş.

In Python:

```
19  import matplotlib
20  #matplotlib.use("gtk")
21  matplotlib.use("QT4Agg")
22  """
23  Q:
24    The plotting engine I declare to use above may not apply
25      for your platform. How do you get a list available
26      plotting engines for yourself?
27  """
28
29  """
```

23.satırın cevabı:

Backend	Description
GTKAgg	Agg rendering to a *GTK* 2.x canvas (requires PyGTK)
GTK3Agg	Agg rendering to a *GTK* 3.x canvas (requires PyGObject)
GTK	GDK rendering to a *GTK* 2.x canvas (not recommended) (requires PyGTK)
GTKCairo	Cairo rendering to a *GTK* 2.x canvas (requires PyGTK and pycairo)
GTK3Cairo	Cairo rendering to a *GTK* 3.x canvas (requires PyGObject and pycairo)
WXAgg	Agg rendering to to a *wxWidgets* canvas (requires wxPython)
WX	Native *wxWidgets* drawing to a *wxWidgets* Canvas (not recommended) (requires wxPython)
TkAgg	Agg rendering to a *Tk* canvas (requires TkInter)
Qt4Agg	Agg rendering to a *Qt4* canvas (requires PyQt4)
macosx	Cocoa rendering in OSX windows (presently lacks blocking show() behavior when matplotlib is in non-interactive mode)

```
N:
  Import aliases below make life easier.
"""
import copy as cp
import numpy as np
import matplotlib.pyplot as plt

class fft_coefs_manipulator(object):
  """
  fft_coefs_manipulator(object)
    Computes fft coefficients of input signal and keeps the
    discrete time and frequency representations updated even if
    allowed operations are carried out on the signal.
  """

  def __init__(self, x_t, flag_real=True, w_o=1):
    """
    __init__
      ctor of class. Carries out the necessary initializations
      and has the fft coefficients of the input signal computed.
      By default the input signal is assumed to be real.
    """
    None

    """
    Q:
      Why not only assignments in the following two lines?
        Why do we need to copy and then store as member
        variables?
    """
    self.x_t_original = cp.copy(x_t)
    self.x_t = cp.copy(x_t)
```

55.satırın cevabı:

"x_t" class içerisinde birkaç farklı yerde daha kullanılabilir, Pthon'da herşey call by reference olduğu için buradaki kopyalama sorun çıkarmaz.

```
    Q:
      Do we particularly need to make use of the None type in here?
        Look up a situation where it would be meaningful to have
        the None type utilized.
    """
    self.x_f = None
```

63.satırın cevabı:

Bir fonksiyon yazalım ismi no_return olsun ve içerisinde pass denilip çıkılsın kod içerisinde bu fonksiyon çağırılıp yazdırılmaya çalışılırsa bize "None" çıktısını verecektir, None genellikle kayıp veya default parametreler için bir sinyal olarak kullanılır. Boş bir değişkeni veya bir nesneyi tanımlamak için kullanılır, None bir "0" veya "false"'a karşılık gelmez, class içinde self.x_f = None yazarak bu class'dan türetilecek tüm instance'lerin bu member variable'a sahip olacağı kesindir.

```
    Q:
      If self.x_t is of the numpy array type, then the following
        assignment should suggest to you some pieces of information
        about the dimensions of self.x_t. What are they?
      What does the "shape" call return in terms of type and
        nature of data?
    """
    self.N   = self.x_t.shape[0]
```

70.satırın cevabı:

"self.x_t" bir array type olsaydı 77.satır bir tuple return edecektir sahape[0] yazıldığı için tuple'ın il elemanı N'e atanacaktır örneğin elimizde a = np.array([1,1,1]) & print(a.shape[0]) bize 3 return edecektir. Bu 3'ün tipi nedir? Cevap integer.

```
    """
    Q:
      If integers and floats are clearly distinct from each other,
        What should I assume about self.N, regarding the next
        line of assignment?
    """
    self.K   = (self.N - 1) / 2
```

79.satırın cevabı:

84.satırda self.N'in hangi tipte olduğunu bilmeyelim (type - int)/int bize bir floating point number'da return edebilir, int değer de return edebilir. Type float ise float bir sonuç elde edeceğiz, type int ise integer değer elde edeceğiz örneğin sonuç 3.5 ise 3 elde edeceğiz. Fakat 77.satırın sonucunda self.N = int return eder demiştik dolayısıyla 84.satırda 'nın tipi de integer olacaktır.

```
86      """
87      N:
88        np.arange returns ndarray type.
89      Q:
90        What is the difference between python and Octave when
91          determining ranges of indices, with the syntax for
92          np.arange given below as examples?
93      """
94      self.indices_t = np.arange(0, self.N)
95      self.indices_f = np.arange(-self.K, 1+self.K)
96
```

89.satırın cevabı:

Python'da index'ler 0'dan başlar Matlab'da ise 1'den başlar, np.arange(0, N) diyerek [0,N) gibi düşünebiliriz, 0 dahil N dahil değil, toplam N tane sayı üretilecek bu sayıların ilk değeri 0, son değeri N-1 olacaktır.

95.satır bu bağlamda (-K, K+1)'e kadar yazılmış "+1" koymanın sebebi son elemanımızın K olmasını istememizdir.

```
98      Q:
99        How are the default values for the member variables below
100         defined?
101     """
102     self.flag_real \
103       = flag_real
104     self.w_o \
105       = w_o
```

98.satırın cevabı:

"flag_real" için default değer True, "w_o" için ise default "1" belirlenmiştir.

```
108     Q:
109       For the two member variables below, do you think the values
110         should definitely be determined in the ctor of the class
111         and never modified afterwards?
112     """
113     self.flag_plot_maximize = True
114     self.plot_type          = 'stem'
115
116     """
117     N:
118       fft coefficients of the input signal computed below.
119     """
120     self.to_freq()
121
122   def set_w_o(self, w_o):
123     self.w_o = w_o
```

108.satırın cevabı:

"plot_type" olarak 'stem' belirlenmiş ayrık sinyalleri çizdirmek "plt.stem" fonksiyonu kullanılmaktadır, bu değer stem olarak belirlenip daha sonra değiştirilmemelidir (ki tek amacımız ayrık zamanlı sinyallerin fft'si alınması ise).

```
124     """
125     Q:
126       Why does not it always make sense to be able to modify
127         the fundamental frequency of the input signal in methods
128         that are not the ctor?
129     """
130
131   def revert_to_original(self):
132     self.x_t = cp.copy(self.x_t_original)
133     self.to_freq()
```

```
    """
    N:
      I prefer to keep the original form of the signal to be
        able to revert to it in case the necessity arises. May
        not always make sense.
    Q:
      Why do I have to call the "to_freq" member function above?
    """

  def compute_derivative_in_time(self):
    self.x_f = \
      ( 1j * self.w_o * np.arange(-self.K, 1+self.K) ) \
      * self.x_f
    self.to_time()
    """
    N:
      This is one of the operations that can be carried out
        in the discrete frequency domain and will affect
        the representation in the discrete time domain so that
        effectively the time derivative is computed for the
        signal in hand.
    Q:
      Review the rationale behind the computation carried out
        above.
      Why am I calling the "to_time" member function?
      An octave translation for the piece of code above would
        possibly make use of the elementwise multiplication
        operator ".*"; however, above all I have is the plain
        "*" operator. How can the statements above accomplish
        what I would like them to, in this sense? As a hint,
        look up how numpy.array behaves with the "*" operator.
        Crucial question.
    """
```

```
    N:
      The two member functions below definitely make sense; I
        should be able to modify my related preferences even
        after the construction of the class instance.
    Q:
      Why do you think it is so?
    """
    def change_plot_type(self, type_str):
      self.plot_type = type_str

    def set_flag_plot_maximize(self, lval):
      self.flag_plot_maximize = lval
```

173.satırın cevabı:

Plot type ctor'da 'stem' olarak default olarak belirlenmişti, sürekli zamanda bir sinyal çizdirmek ister isek bunu değiştirmemiz gerekecektir, burada plot_type için bir setter() yazılmıştır.

Aynı şekilde plot_maximize'de true olarak bir default değer alıyordu bu değeri değiştirmek istersek false yapmak istersek bunun setter'ını çağırmamız gerekecek setter üzerinden ilgili instance'nin ilgili default member variable'ı değiştirilecektir.

```
    """
    Q:
      What is a decorator in python?
      Why is a static member function (method) necessary? Does
        a static member function make use of the "self" keyword?
        Does a static member function require a class instance
        to work on?
      How do we define a static member function in python?
    """
```

173.satırın cevabı:

Decorator sayesinde mevcut nesneye bazı yeni işlevler kazandırabiliriz, mevcut nesnenin işleyişini değiştirmez onu sadece genişletir. Function and Class Decorators olmak üzere iki farklı yapıya da bu işlem uygulanabilir. Statik yöntemler o sınıfa ilişkin instance'ler yerinebir sınıfa bağlıdır. İlgili sınıf için bir nesne olmadan çağırılabilir.

```
191   @staticmethod
192   def compute_plot_min_max(
193     x, y, x_tol, y_tol):
194
195     y_val_abs = np.abs( y.max() - y.min() )
196     y_val_low  = - y_tol * y_val_abs + y.min()
197     y_val_high = + y_tol * y_val_abs + y.max()
198
199     x_val_abs = np.abs( x.max() - x.min() )
200     x_val_low  = - x_tol * x_val_abs + x.min()
201     x_val_high = + x_tol * x_val_abs + x.max()
202
203     return x_val_low, x_val_high, y_val_low, y_val_high
204     """
205     N:
206       This function is necessary to compute the boundaries
207         of the window illustrated on a plot, the boundaries
208         being dependent on the maximum and minimum levels of
209         the data to be depicted.
210     Q:
211       Why do I make use of tolerances above?
212       Explain how the high and low boundaries are computed
213         above?
214       If we are required to plot multiple sets of data on a
215         single plot, would the syntax above suffice? How
216         would you modify it to capture the necessity just
217         stated?
218       Is it possible to return multiple variables in a python
219         function?
220     """
221
222   def plot_freq(self, title_str='Signal in Freq. Domain'):
223     label_fontsize = 24
224     ticks_labelsize= 24
225     axis_border_lwidth = 3
226     axis_plot_lwidth   = 4
227     """
228     N:
229       Several plot related values to be utilized below defined
230         as variables.
231     """
232
```

210.satırın cevabı:

Birden fazla grafiği tek bir plot üzerinde gösterebiliriz, 203.satırda 4 tane değişken return edilmiş ki bu python'da mümkündür. Her bir değişken x'e göre çizdirilmek istenirse bu 4 değişken de main içerisinde fonksiyon çağırımı sonrasında 4 ayrı değişkene atanmalıdır yani a,b,c,d = compute_plot_min_max(..) şeklinde çağırılması gerekecek.

```
232
233     y_tol = 0.1
234     x_tol = 0.005
235     """
236     N:
237       Tolerances used to have the plots look nicer, otherwise we
238         would have left it to the plotting engine to decide which
239         window of the data is to be depicted, would in some
240         cases cause lots of unused plotting area to be illustrated.
241     """
242
243     fig = plt.figure()
244     rect = fig.patch
245     rect.set_facecolor('white')
246     """
247     Q:
248       Examining above, how do we set the background color for the
249         space surrounding the actual plot in a figure?
250     """
251
252     x  = self.indices_f
253     y  = self.x_f
254     """
255     Q:
256       Do the assignments above produce copies or aliases (references)
257         only? Which one would you prefer in this context?
258     """
259
```

255.satırın cevabı:

Bir kopyalama gerçekleşmeyecek sadece x de, indices_f'in gösterdiği item'ı işaret edecektir. Eğer indices_f'e işaret etmeyen her hangi bir değişken kodda kalmamışsa o item carbage collection'a gönderilecektir.

```
    """
    N:
      Below we are going to have two subplots, one each for
        plotting the real and imaginary parts of the FFT
        coefficients of the current signal.
    Q:
      Why is it advantageous to make use of the list of lists
        defined in the control flow statement below (range based
        for loop)?
      Would an implementation in Octave be so compact? Would it
        be possible to code in Octave?
    """
```

```
    for splt in \
      [
        [ 211 , np.real(y) , 'Real Parts' , 'blue' ] ,
        [ 212 , np.imag(y) , 'Imag Parts' , 'red'  ]
        ]:
      """
      N:
        Below is the compact unpacking syntax, saves much
          effort.
      """
      splt_id, splt_signal, splt_ylabel, splt_plot_color \
        = splt
```

265.satırın cevabı:

282.satırda splt'değişkenine bir liste atanmaktadır ve bu listede 4 tane eleman vardır, bu 4 eleman ayrı ayrı 4 farklı değişkene atanmıştır range-based loop için "in" kullanarak listenin içine git ve listeyi çek split'e ata demişiz.

```
      """
      N:
        Below, according to a choice stored in "self.plot_type"
          for the instance, we command the plotting engine
          to depict either a stem or a continuous plot.
        Below is an example for the case where I regret not
          having the facility to make use of switch-case control
          flow statement available in other programming languages.
      Q:
        Why do I regret the absence of the stated feature in here,
          whereas we must have touched upon the possible
          alternatives for switch-case?
      """
      ax = fig.add_subplot(splt_id)
      if   self.plot_type == 'stem':
        """
```

293.satırın cevabı:

Python'da switch case yapısı yoktur, bunun alternatifi olarak "if" yapısı kullanılmıştır ve benzer şeyler yapılabilmektedir.

```
        N:
          The plotting features and management of attributes
            constitute a particularly powerful aspect of python.
        Q:
          Look up the other available methods for the instances
            of plotting ingredients, with examples as given below.
        """
        markerline, stemlines, baseline = \
          ax.stem( x, splt_signal, linefmt='grey',
                   markerfmt='o', bottom=0.,
                   use_line_collection=True )
        markerline.set_markerfacecolor('red')
        markerline.set_markersize(20)
        markerline.set_mec('black')
```

```
315         markerline.set_mew('2')
316         baseline.set_c ('blue')
317         baseline.set_lw(5.)
318         """
319         Q:
320           Why do we need the for loop just below, while
321             it was not necessary for the settings above?
322         """
323         for item in stemlines:
324           item.set_lw(5.)
325       elif self.plot_type == 'plot':
326         ax.plot( x , splt_signal ,
327                  color = splt_plot_color ,
328                  linewidth = axis_plot_lwidth )
329       else:
330         pass
331
332       """
333       Q:
334         Explain how we call a static member function
335           with the necessary arguments and unpack the
336           returned multiple data into separate variables,
337           in just a single line of code as below.
338       """
339       x_val_low, x_val_high, y_val_low, y_val_high \
340         = fft_coefs_manipulator.compute_plot_min_max(
341             x, splt_signal, x_tol, y_tol)
342
343       """
344       Q:
345         How do we set the limits/boundaries of the subplot?
346       """
347       ax.set_ylim( [ y_val_low , y_val_high ] )
348       ax.set_xlim( [ x_val_low , x_val_high ] )
349
350       """
351       Q:
352         How do we set the line widths for the bounding box
353           of the frame encapsulating the actual plot?
354       """
```

344.satırın cevabı:

“fig, ax = plt.subplot(2)” yazarak ax kullanılarak ax[0].plot() ve ax[1].plot() çizimi mümkün olacaktır. “fig, ax = plt.subplot(2,2)” de olabilirdi.

352.satırın cevabı:

347. ve 348.satırların görevi çizilecek plot’un x ve y eksenlerinin uzunluklarını ayarlamaktır, 347.satır için y ekseni ilgili 2 değer aralığında oluşacaktır bu 2 değer minumum ve maximum değerlerdir. Y ekseninin uzunluğu = high – low olacaktır.

```
355       for loc in ['top', 'right', 'bottom', 'left']:
356         ax.spines[loc].set_linewidth(axis_border_lwidth)
357
358       """
359       Q:
360         If for some reason I were not satisfied with
361           the ticks and labels in a plot, would I be
362           able to manage them myself? Does python provide
363           me with the necessary features to accomplish this
364           one?
365       """
366       ax.tick_params(labelsize=ticks_labelsize)
367
368       ax.xaxis.grid()
369       ax.yaxis.grid()
370
```

359.satırın cevabı:

Cevap evet plot’un başlığını xlabel, ylabel isimlendirmelerini kendimiz yapabiliriz, bunun dışında ek bilgilerin gösterileceği sağ yukarı bir bilgi kutucuğu da ekleyebiliriz bu kutucukta hangi renkteki grafik neyi temsil etmektedir tarzı bilgiler verilebilir, bunların yanı sıra plot’un arka planı küçük ve eşit uzunluklu karelerden’de oluşabilir (grid).

```
    Q:
      Play around with the "y = ..." keyword argument below
        and see what happens.
    """
    ax.set_title ( '%s (%s)' % (title_str, splt_ylabel) ,
                   fontsize = label_fontsize , y = 1.01 )
    ax.set_ylabel( splt_ylabel ,
                   fontsize = label_fontsize )

    """
    Q:
      If I chose to, would I also be able to store the handles
        of all of the axes generated in the individual iterations
        of this for loop as well?
      Why am I assigning "ax_last" in each iteration? Is there
        no other way of doing what I am trying to do (whatever
        it is)?
    """
    ax_last = ax
    del splt_id, splt_signal, splt_ylabel
```

372.satırın cevabı:

Başlık plot'un ne kadar üstünde olsun arttırdıkça başlık daha yukarıya konumlanıyor(pad = 1.01).

382.satırın cevabı:

"ax_last = ax" diyerek 390.satırda ilgili figüre için yapılan tüm işlemler bitmiştir ve o figür ax_last' atanmıştır.

```
    N:
      We are out of the for loop responsible for generating
        the indicated subplots.
    Q:
      Tell me why I have the following statement making use of
        the "ax_last" reference.
    """
    ax_last.set_xlabel( 'Discrete Freq. Indices' ,
                        fontsize = label_fontsize )
    """
    Q:
      How do we optionally maximize a figure for it to assume
       or exploit the whole screen size in python?
    """
    if self.flag_plot_maximize:
      figManager = plt.get_current_fig_manager()
      figManager.window.showMaximized()

  def plot_time(self, title_str='Signal in Time Domain'):
    label_fontsize = 24
    ticks_labelsize= 24
    axis_border_lwidth = 3
    axis_plot_lwidth   = 4

    y_tol = 0.1
    x_tol = 0.005

    fig = plt.figure()
    rect = fig.patch
    rect.set_facecolor('white')

    """
    Q:
      We have only a single subplot in the figure generated
        in this member function. It seems weird to have the
        statement below. Is there any other alternative?
    """
    ax = fig.add_subplot(111)
    x  = self.indices_t
    y  = self.x_t

    if   self.plot_type == 'stem':
      markerline, stemlines, baseline = \
        ax.stem( x, y, linefmt='grey', markerfmt='o', bottom=0.,
```

397.satırın cevabı:

"ax ve ax_last" aynı şeye işaret ettikleri için ax_last üzerinden değişim yaptığımızda sonuçta ax'in gösterdiği şey de değişmiş oluyor.

426.satırın cevabı:

431.satıra "ax = fig.add_subplot()" yazsaydık aynı şeyi elde edecektik default olarak subplot (1,1,1) olarak belirlenmiştir.

```
437         ax.stem( x, y, linefmt='grey', markerfmt='o', bottom=0.,
438                  use_line_collection=True )
439       markerline.set_markerfacecolor('red')
440       markerline.set_markersize(20)
441       markerline.set_mec('black')
442       markerline.set_mew('2')
443       baseline.set_c ('blue')
444       baseline.set_lw(5.)
445       for item in stemlines:
446         item.set_lw(5.)
447     elif self.plot_type == 'plot':
448       ax.plot( x , y , linewidth = axis_plot_lwidth )
449     else:
450       pass
451
452     x_val_low, x_val_high, y_val_low, y_val_high \
453       = fft_coefs_manipulator.compute_plot_min_max(
454           x, y, x_tol, y_tol)
455
456     ax.set_ylim( [ y_val_low , y_val_high ] )
457     ax.set_xlim( [ x_val_low , x_val_high ] )
458
459     for loc in ['top', 'right', 'bottom', 'left']:
460       ax.spines[loc].set_linewidth(axis_border_lwidth)
461
462     ax.tick_params(labelsize=ticks_labelsize)
463
464     ax.xaxis.grid()
465     ax.yaxis.grid()
466
467     ax.set_title ( title_str ,
468                    fontsize = label_fontsize , y = 1.01 )
469     ax.set_xlabel( 'Discrete Time Indices' ,
470                    fontsize = label_fontsize )
471     ax.set_ylabel( 'Signal' ,
472                    fontsize = label_fontsize )
473
474     if self.flag_plot_maximize:
475       figManager = plt.get_current_fig_manager()
476       figManager.window.showMaximized()
477
    """
```

```
478   """
479   NOTE: On how to permute columns of ndarrays, see the following URL:
480     https://stackoverflow.com/questions/20265229/rearrange-columns-of-numpy-2d-array
481   """
482
483   def to_freq(self):
484     self.x_f = np.fft.fft(self.x_t) / self.N
485     """
486     Q:
487       Is there an alternative order of operations through to
488         the discrete frequency representation of the signal?
489       Explain the fftshift operation carried out below
490         (correct for an odd number of samples).
491     """
492     permuted_col_indices \
493       = range(self.K, self.N) \
494       + range(0, self.K)
495     """
496     Q:
497       What type does the "range" function return?
498       Is the "+" above an overloaded one?
499       Would the call above work correctly or even compile
500         with the "numpy.arange()" function utilized instead of
501         "range"?
502     """
503     #print permuted_col_indices
504     idx = np.empty_like(permuted_col_indices)
505     idx[permuted_col_indices] \
506       = np.arange(len(permuted_col_indices))
507     """
508     Q:
509       If the resulting array and the original one were not
510         of the same number of elements, would the call below
511         generate errors?
512       Can the operation below be carried out in place? If so,
513         what is the advantage?
514     """
515     self.x_f[:] = self.x_f[idx]
516
517   def to_time(self):
518     self.x_t = self.x_f * self.N
```

496.satırın cevabı:

“print(type(range(10)))” sonucu bize range verecektir yani range fonksiyonu bir range tipinde bir şey return edecektir.

492.satır python3 kullanımı sonucu hata vermiştir iki range() fonksiyonu için bir “+” operatörü overload edilmemiştir, sorun şurada python2’de range() bir liste return ediyor fakat pyhon3’de bir range tipi return ediyor. “list + list” overloading’i mevcuttur ve 492.satır sorun çıkarmamaktadır çünkü kod python2 versiyonunda yazılmıştır.

“np.arange(0,10) -> [0,1, ... 9]” same as “range(10)” dolayısı ile birbirleri yerine kullanılabilirler fakat “np.arange()” ndarray return edecektir. “range” ise liste return edecektir dolayısıyla bunu unutmamak gerekir.

```
519     """
520     Q:
521       Is there an alternative order of operations through to
522         the discrete time representation of the signal?
523       Explain the fftshiftback operation carried out
524         below (correct for an odd number of samples).
525     """
526     permuted_col_indices \
527       = range(1+self.K, self.N) \
528       + range(0, 1+self.K)
529     #print permuted_col_indices
530     idx = np.empty_like(permuted_col_indices)
531     idx[permuted_col_indices] \
532       = np.arange(len(permuted_col_indices))
533     self.x_t[:] = self.x_t[idx]
534     self.x_t = np.fft.ifft(self.x_t)
535     """
536     Q:
537       If the signal is real in discete time, and operations
538         carried out do not modify this aspect, then why would we
539         still need to obtain the real parts of the samples
540         in the discrete time domain as done below?
541     """
542     if self.flag_real:
543       self.x_t = np.real(self.x_t)
544
```