```
# 1. Using the aud_usd_lst and eur_aud_lst lists defined in the scaffold on the
right, perform the following tasks:

1. Create a series named aud_usd_series with non-missing quotes for the AUD/USD
exchange rate. Specifically:

The series should have dates as row labels. There should be no missing AUD/USD
values.
2. Create a series named eur_aud_series with non-missing quotes for the EUR/AUD
exchange rate. Specifically:

The series should have dates as row labels. There should be no missing EUR/AUD
values.
3. Combine the two series into a data frame named df, so it has the dates as row
labels and 'AUD/USD', 'EUR/AUD' as column labels.


CODE :main.py code

import pandas as pd
import numpy as np
from unanswered import Unanswered()

aud_usd_lst = [
    ('2020-09-08', 0.7280),
    ('2020-09-09', 0.7209),
    ('2020-09-11', 0.7263),
    ('2020-09-14', 0.7281),
    ('2020-09-15', 0.7285),
    ]

eur_aud_lst = [
    ('2020-09-08',  1.6232),
    ('2020-09-09',  1.6321),
    ('2020-09-10',  1.6221),
    ('2020-09-11',  1.6282),
    ('2020-09-15',  1.6288),
    ]

# Replace unanswered with your solution.
aud_usd_series = aud_usd_series = pd.Series(np.array(aud_usd_lst)[:,1],
index=np.array(aud_usd_lst)[:,0])
print (aud_usd_series)
eur_aud_series =  pd.Series(np.array(eur_aud_lst)[:,1], index=np.array(eur_aud_lst)
[:,0])
eur_aud_series
df = pd.DataFrame([aud_usd_series,eur_aud_series]).T
df.columns = ['AUD/USD','EUR/AUD']
df

New page code

class Unanswered:
```

```
unanswered = Unanswered()


# 2 . Code Challenge: Downloading exchange rates

In this exercise, you will create a function that takes two currency codes and
converts them to a ticker suitable for downloading data from Yahoo! Finance.

code: The currency ISO code (e.g., USD = US Dollar, AUD = Australian Dollar). You
can find a list of currency codes here.
fx:   The exchange rate, i.e. the value of one currency in terms of another
currency. We will represent an exchange rate like this:
AUD/USD : The price in US dollars of one Australian dollar
USD/AUD : The price in Australian dollars of one US dollar
AAA/BBB :  The price in currency BBB of one unit of currency AAA
from_cur: The currency which we want to price (AAA in the example above)
to_cur: The price of one unit of the from_cur (BBB in the example above)
Yahoo finance uses the following naming rules to define the ticker of the exchange
rate AAA/BBB (the price of one unit of AAA in terms of currency BBB):

1. If AAA is the USD, then the ticker is "BBB=X", i.e., the second currency code
with "=X" added at the end.

2. If AAA is not the USD, then the ticker is "AAABBB=X"

For example, the ticker for AUD/USD is "AUDUSD=X", while the ticker for USD/AUD is
"AUD=X"

The scaffold provides you with a declaration defining the function's arguments and
a  docstring that describes what the function should do. You need to fill in the
body of the function.  Note that a function called get_fx is provided as a
reference. Once your fx_code function passes the diagnostic tests, you should be
able to copy the entire file to PyCharm and use this program to download currency
data.


CODE : main.py

""" main.py
Code challenge
"""

import numpy as np
import pandas as pd
# import yfinance as yf # Uncomment this line if you are wish to work with
`yfinance` outside of Ed

# Write this function
def fx_code(from_cur, to_cur):
    """ Creates a string with the ticker required to download exchange rates
    using yfinance. The exchange rate will be the price of one unit of the
`from_cur` in terms
    of the `to_cur`.

    Parameters
    ----------
    from_cur : str
        The ISO code of the currency to be priced.
    to_cur : str
        The ISO code of the currency with the value of one unit of `from_cur`.


    Returns
    -------
        A string that meets the `yfinance` ticker standards with ALL characters in
upper case.
        The function should also be able to ignore leading and trailing spaces. For
example,
        " aud", "Aud ", and " AUD " all are treated as "AUD" internally. See the
        Notes section below for more information.

    Notes
    -----
    Yahoo finance uses the following naming rules to define the ticker of the
    exchange rate AAA/BBB:
    usd/aud

    1. If AAA is the USD, then the ticker is "BBB=X", i.e., the second currency
       code with "=X" added at the end.
    2. If AAA is not the USD, then the ticker is "AAABBB=X"

    For example, the ticker for AUD/USD is "AUDUSD=X", while the ticker for
    USD/AUD is "AUD=X"

    So, if `from_cur=AAA` and the `to_cur=BBB`, the YF ticker will be:
    1. "BBB=X" if AAA is USD
    2. "AAABBB=X" if AAA is not the USD
    """

    pass

# get_fx is provided to demonstrate how you can download currency data from
`yfinance`.
# Once your fx_code function above is correct, get_fx should work on a computer
# that has the `yfinance` package installed.
def get_fx(from_cur, to_cur, period=None, interval=None):
    """ Downloads the exchange rate between the `from_cur` and the `to_cur`.
    The exchange rate will be the price of one unit of the `from_cur` in terms
    of the `to_cur`

    Parameters
    ----------
    from_cur : str
        The ISO code of the currency to be priced

    to_cur : str
        The ISO code of the currency with the value of one unit of
        `from_cur`.

    period : str, None
        valid periods: 1d,5d,1mo,3mo,6mo,1y,2y,5y,10y,ytd,max
        (optional, default is '1mo')

    interval : str, None
        valid intervals: 1m,2m,5m,15m,30m,60m,90m,1h,1d,5d,1wk,1mo,3mo
        (optional, default is '1d')
    Returns
    -------
    df
        Dataframe with daily exchange rates from Yahoo Finance

    """
    # Defaults
    if period is None:
        period = '1mo'
    if interval is None:
        interval = '1d'

    tic = fx_code(from_cur, to_cur)

    # fetches the data
    df = yf.download(tic, period=period, interval=interval)

    return df


# 3.  Code Challenge: Manipulating data in Pandas

Consider the following quotes:

| row_id | Date       | AUD/USD | EUR/AUD |
|--------+------------+---------+---------|
| 11     | 2020-09-08 | 0.7280  | 1.6232  |
| 70     | 2020-09-09 | 0.7209  | 1.6321  |
| 99     | 2020-09-10 |         | 1.6221  |
| 4      | 2020-09-11 | 0.7263  | 1.6282  |
| 7      | 2020-09-14 | 0.7281  |         |
| 100    | 2020-09-15 | 0.7285  | 1.6288  |


where row_id uniquely identifies each row in this table (in no particular order).

Suppose this is just a small sample of what in reality is thousands of
observations. Assume that source data is stored in separate files, with row_ids.
The source files are not necessarily in order.

As an example, let's represent these files as list of tuples:

# date_info = [(row_id, date)]
date_info = [
    (11 , '2020-09-08'),
    (70 , '2020-09-09'),
    (99 , '2020-09-10'),
    (4  , '2020-09-11'),
    (7  , '2020-09-14'),
    (100, '2020-09-15'),
    ]

# aud_usd_info = [(row_id, aud/usd)]
aud_usd_info = [
    (70 ,  0.7209),
    (4  ,  0.7263),
    (11 ,  0.7280),
    (7  ,  0.7281),
    (100,  0.7285),
]


# eur_aud_info = [(row_id, eur/aud)]
eur_aud_info = [
    (70 ,  1.6321),
    (4  ,  1.6282),
    (99 ,  1.6221),
    (100,  1.6288),
    (11 ,  1.6232),
    ]
```

Your goal is to write the functionmk_dfin the scaffold that takes these structures as arguments and returns a DataFrame. In this DataFrame, the row labels are sorted dates. Column labels are 'AUD/USD'and 'EUR/AUD'. Missing values are retained and represented by NaN. A sample of this format is shown below:

```
|            | AUD/USD | EUR/AUD |
+------------+---------+---------|
| 2020-09-08 | 0.7280  | 1.6232  |
| 2020-09-09 | 0.7209  | 1.6321  |
| 2020-09-10 | NaN     | 1.6221  |
| 2020-09-11 | 0.7263  | 1.6282  |
| 2020-09-14 | 0.7281  |  NaN    |
| 2020-09-15 | 0.7285  | 1.6288  |
```

Important notes for writing this code:

Replace pass with the appropriate statements. You do not need to modify anything else.
You should NOT use .set_index inside this function
Your code will be evaluated with different data so you should think of the lists above as test cases.These lists are provided to you in the scaffold so you may trial your function.

CODE:

```
import pandas as pd
import numpy as np

# Write this function
def mk_df(date_info, aud_usd_info, eur_aud_info):
    """ Combines the information from different sources to produce a dataframe
    with dates row labels. Row labels must be sorted

    Parameters
    ----------
    date_info : list
        date_info = [(row_id, date)]

    aud_usd_info : list
        aud_usd_info = [(row_id, aud/usd)]
    eur_aud_info : list
        eur_aud_info = [(row_id, eur/aud)]

    Returns
    -------
    df
    """
    pass


# Sample data for you to develop your function
# date_info = [(row_id, date)]
date_info = [
    (11 , '2020-09-08'),
    (70 , '2020-09-09'),
    (99 , '2020-09-10'),
    (4  , '2020-09-11'),
    (7  , '2020-09-14'),
    (100, '2020-09-15'),
    ]

# aud_usd_info = [(row_id, aud/usd)]
aud_usd_info = [
    (70 ,  0.7209),
    (4  ,  0.7263),
    (11 ,  0.7280),
    (7  ,  0.7281),
    (100,  0.7285),
]


# eur_aud_info = [(row_id, eur/aud)]
eur_aud_info = [
    (70 ,  1.6321),
    (4  ,  1.6282),
    (99 ,  1.6221),
    (100,  1.6288),
    (11 ,  1.6232),
    ]

df = mk_df(date_info, aud_usd_info, eur_aud_info)
print(df)
```