Zadanie kwalifikacyjne: W oparciu o dane dotyczące kursów walut wyciągane za pomocą API NBP (http://api.nbp.pl/ (http://api.nbp.pl/)) prosimy o przygotowanie programu automatyzującego analizę inwestycji w 3 wybrane waluty. Inwestowaną kwotą jest 1000 zł, które ma być w całości przeznaczone na zakup 3 walut przechowywanych przez okres 30 dni.

• Data rozpoczęcia może być parametrem wejściowym ew. procedury.

• Procentowy podział 3 walut może być również określony losowo lub jako parametr (np. 30% USD, 40% EUR, 30%HUF). Celem zadania jest automatyzacja tej analizy i prezentacja wniosków (max. 1 strona / 1 slajd) omawiająca historię tej inwestycji:

A. Prezentacja może być oparta na wykresach powstających automatycznie w Twoim programie

B. Prezentacja (slajd lub dashboard) powinna odpowiadać na przykładowe pytania:

- jak procentowo rozłożona była inwestowana kwota ?
- jak zmieniała się wartość Twojego portfela ?
- jak wyglądał procentowy udział walut na końcu okresu inwestycji ?

Ważne!

2. Inwestycja nie musi być dochodowa – poszukiwanie najbardziej dochodowej inwestycji nie jest celem tego zadania
3. Dla uproszczenia analizy można posłużyć się średnim kursem i nie musisz uwzględniać różnych cen zakupu i sprzedaży
4. Waluty są kupowane we wskazanej dacie i po 30 dniach jest oceniana wartość portfela – w tym okresie nie są dokonywane żadne dodatkowe transakcje kupna/sprzedaży.

In [1]:

```python
import datetime
import ipywidgets as widgets
import matplotlib.pyplot as plt
from IPython.display import clear_output
import requests
import random
import time
import sys

TabelaA = requests.get("http://api.nbp.pl/api/exchangerates/tables/A/")
TabelaA = TabelaA.json()

currency_list = []
for i in range(0, len(TabelaA[0]["rates"])):
    currency_list.append(TabelaA[0]["rates"][i]["code"])

#widgets 1-3, currency list
w1=widgets.Dropdown(
    options=currency_list,
    value='THB',
    description='currency:',
    disabled=False)
w2=widgets.Dropdown(
    options=currency_list,
    value='USD',
    description='currency:',
    disabled=False)
w3=widgets.Dropdown(
    options=currency_list,
    value='AUD',
    description='currency:',
    disabled=False)

#widgets 4, stock
w4=widgets.IntText(
    value=1000,
    description='Stock [PLN]:',
    disabled=False
)
```

In [2]:

```
#widgets 1a-3a, currency percentage
w1a=widgets.FloatSlider(
    value=0.333,
    min=0,
    max=1,
    step=0.001,
    description='Percentage:',
    disabled=False,
    continuous_update=False,
    orientation='horizontal',
    readout=True,
    readout_format='.3f',
)
w2a=widgets.FloatSlider(
    value=0.333,
    min=0,
    max=1,
    step=0.001,
    description='Percentage:',
    disabled=False,
    continuous_update=False,
    orientation='horizontal',
    readout=True,
    readout_format='.3f',
)
w3a=widgets.FloatSlider(
    value=0.334,
    min=0,
    max=1,
    step=0.001,
    description='Percentage:',
    disabled=False,
    continuous_update=False,
    orientation='horizontal',
    readout=True,
    readout_format='.3f',
)
```

In [3]:

```
#widgets 1a-3a, currency percentage, sum to 100%, slider changes.
def handle_slider_change1(change):
    w2a.value = w2a.value-(change.new-change.old)
    if w1a.value + w2a.value + w3a.value != 1:
        w3a.value=1-w2a.value-w1a.value

w1a.observe(handle_slider_change1, names='value')

def handle_slider_change2(change):
    w3a.value = w3a.value-(change.new-change.old)
    if w1a.value + w2a.value + w3a.value != 1:
        w2a.value=1-w3a.value-w1a.value

w2a.observe(handle_slider_change2, names='value')

def handle_slider_change3(change):
    #w1a.value = w1a.value-(change.new-change.old)#removed
    if w1a.value + w2a.value + w3a.value != 1:
        w3a.value=1-w2a.value-w1a.value

w3a.observe(handle_slider_change3, names='value')

button_random = widgets.Button(description="Random currency!",disabled=False)

#generate random percentage stock for sliders
def on_button_random(b):
    button_random.disabled=True
    x = []
    for i in range(3):
        x.append(random.random())
    y=[]
    for i in x:
        y.append(i/sum(x))
    for i in range(0,3):
        widget_nr = random.randint(0,1)#widget number 3 it only computate by sum
        if widget_nr==0:
            w1a.value=y[0]
        if widget_nr==1:
            w2a.value=y[1]
    w1.value=currency_list[random.randint(0,len(currency_list)-1)]
    w2.value=currency_list[random.randint(0,len(currency_list)-1)]
    w3.value=currency_list[random.randint(0,len(currency_list)-1)]
    time.sleep(1)
    button_random.disabled=False

button_random.on_click(on_button_random)
```

In [4]:

```
#widgets 4, stock = 1000
def handle_stock_change(change):
    w4.value = 1000

w4.observe(handle_stock_change, names='value')

#date from 01.02.2002 to last date - 30
date=widgets.DatePicker(description='Pick a Date',value=datetime.datetime(2023, 
mindate=datetime.date(2002, 2, 1)
maxdate=datetime.datetime.strptime(TabelaA[0]["effectiveDate"], "%Y-%m-%d").date

def date_change(change):
    if date.value < mindate:
        date.value = mindate
        with output1:
            print("Min date: ")
            print(mindate)
    if date.value >  maxdate:
        date.value = maxdate
        with output1:
            clear_all_outputs()
            print("Max date (the last published exchange rate date - 30 days): "
            print(maxdate)

date.observe(date_change, names='value')

button_date = widgets.Button(description="Random date")
def on_button_date(b):
    year = random.randint(2002,2022)
    month = random.randint(1,12)
    day = 1
    date.value = datetime.date(year, month, day)
button_date.on_click(on_button_date)

try:
  date.value=date.value.date()
except:
  print("warning, date transformed before")


def calculate_changes(currency=w1.value, date=date.value):
    stringBuilder = "http://api.nbp.pl/api/exchangerates/rates/A/" + currency + 
    try:
        Currency1 = requests.get(stringBuilder)
        Currency1  = Currency1.json()
    except:
        print("NBP: 404 Not Found")
        print("Original message:")
        print(stringBuilder)
        sys.exit(1)

    Currency1list = []
    for x in Currency1["rates"]:
      wartosc=float(x.get("mid"))
      Currency1list.append(wartosc)
    Changes = []
    for i in range(0, len(Currency1list)-1):
        Changes.append(Currency1list[i]/Currency1list[i+1])
    return Changes
```

In [5]:

```
def calculate_portfolio(currency1=w1.value, pct1=w1a.value,
                        currency2=w2.value, pct2=w2a.value,
                        currency3=w3.value, pct3=w3a.value,
                        stock=w4.value, date=date.value):
    Changes1=calculate_changes(currency=currency1, date=date)
    Changes2=calculate_changes(currency=currency2, date=date)
    Changes3=calculate_changes(currency=currency2, date=date)
    pct1=pct1*stock
    pct2=pct2*stock
    pct3=pct3*stock
    Portfolio_value=[]
    Portfolio_value.append(pct1+pct2+pct3)
    for i in range(0, len(Changes1)):
        pct1=pct1*Changes1[i]
        pct2=pct2*Changes2[i]
        pct3=pct3*Changes3[i]
        Portfolio_value.append(pct1+pct2+pct3)
    print("Stock value [PLN]: %.2f" % (pct1+pct2+pct3))
    print(currency1 + " in stock: %.2f, pct: %.2f %%" % (pct1, pct1/(pct1+pct2+p
    print(currency2 + " in stock: %.2f, pct: %.2f %%" % (pct2, pct2/(pct1+pct2+p
    print(currency3 + " in stock: %.2f, pct: %.2f %%" % (pct3, pct3/(pct1+pct2+p
    return Portfolio_value

output1 = widgets.Output()
output2 = widgets.Output()
output3 = widgets.Output()

def clear_all_outputs():
    output1.clear_output()
    output2.clear_output()
    output3.clear_output()
```

In [6]:

```
button_check = widgets.Button(description="Check investment!")

def on_button_clicked(b):
    clear_all_outputs()
    with output1:
        output1.clear_output()
        print("Output generated!")
        print("Investment report start date:")
        print(date.value)
        print("Investment report end date:")
        print(str(date.value + datetime.timedelta(days=30)))
        print("")
        print("Investment first day:")
        print("Stock value [PLN]: %.2f" % (w4.value))
        pct1=w1a.value*w4.value
        pct2=w2a.value*w4.value
        pct3=w3a.value*w4.value
        print(w1.value + " in stock: %.2f, pct: %.2f %%" % (pct1, pct1/(pct1+pct
        print(w2.value + " in stock: %.2f, pct: %.2f %%" % (pct2, pct2/(pct1+pct
        print(w3.value + " in stock: %.2f, pct: %.2f %%" % (pct3, pct3/(pct1+pct
    with output2:
        output2.clear_output()
        print("Investment last day:")
        portfolio_value = calculate_portfolio(currency1=w1.value, pct1=w1a.value,
                        currency2=w2.value, pct2=w2a.value,
                        currency3=w3.value, pct3=w3a.value,
                        stock=w4.value, date=date.value)
    with output3:
        plt.plot(portfolio_value)
        plt.title("Stock value [PLN]")
        plt.xlabel('Investment day')
        output3.clear_output()
        plt.show()
button_check.on_click(on_button_clicked)
```

In [7]:

```
clear_all_outputs()

display(w4,
        w1, w1a,
        w2, w2a,
        w3, w3a,
        button_random,
        date, button_date,
        button_check,
        output1, output2, output3)
```

Stock [PLN]: 1000

currency: HUF

Percentage: 0.178

currency: UAH

Percentage: 0.443

currency: CAD

Percentage: 0.379

Random currency!

Pick a Date 01.06.2021

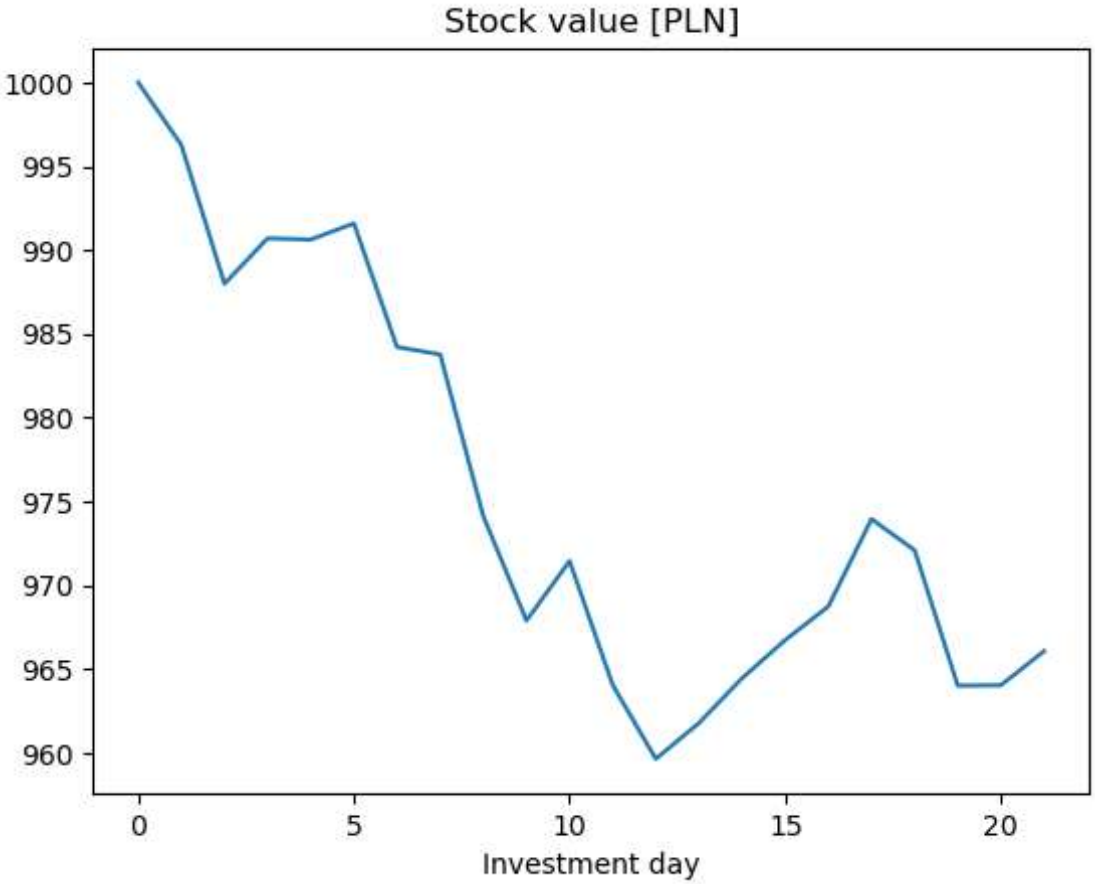
Random date

Check investment!

```
Output generated!
Investment report start date:
2021-06-01
Investment report end date:
2021-07-01

Investment first day:
Stock value [PLN]: 1000.00
HUF in stock: 178.49, pct: 17.85 %
UAH in stock: 442.84, pct: 44.28 %
CAD in stock: 378.67, pct: 37.87 %


Investment last day:
Stock value [PLN]: 966.06
HUF in stock: 178.78, pct: 18.51 %
UAH in stock: 424.39, pct: 43.93 %
CAD in stock: 362.89, pct: 37.56 %
```


Stock value [PLN]
1000
995
990
985
980
975
970
965
960
0
5
10
15
20
Investment day