B203349_data_collection_tool

June 20, 2022

0.1 Loading of initial data

0.1.1 Load requisite python packages

[ ]:
```
#Load the 'pandas' package
import pandas as pd
#Load the 'ipywidgets' package
import ipywidgets as widgets
#Load the 'IPython.display' package
from IPython.display import display
#Load the 'numpy' package
import numpy as np
```

0.1.2 Load test data

[ ]:
```
testData=pd.read_csv("../Data/ae_attendances_test.csv")
testData
```

[ ]:
```
#confirm data types
result = testData.dtypes
print("Output:")
print(result)
```

0.1.3 Set up empty data frame

[ ]:
```
dfTofill = pd.DataFrame({'index': [0],# Integer
                    'period': [pd.Timestamp('20000101')], # Date
                    'org_code': ['NA'], # String
                    'type': ['NA'], # String
                    'attendances': [0], # Integer
                    'breaches': [0], # Integer
                    'admissions': [0], # Integer
                    'consent': [False]}) # Boolean

dfTofill
```

Save the empty data frame to your working 'Data' folder (commented out out to prevent wiping file when re-running code)

[ ]:
```
#dfTofill.to_csv('../Data/CollectedData.csv', index=False)
```

Read empty 'CollectedData' dataframe

[ ]:
```
CollectData=pd.read_csv("../Data/CollectedData.csv")
CollectData
```

1 Indexing

Add the index number to the *'dfTofill'* file

[ ]:
```
index_number=11767 #Remember to change for each record.
dfTofill.iloc[0,0]=index_number
dfTofill
```

2 Widgets

2.1 Inserting consent

2.1.1 Checkbox widget

To capture the value for consent which is Boolean (i.e. True or False)

[ ]:
```
a = widgets.Checkbox(
    value=False,
    description='I consent for the data I have provided to be processed and␣
↪shared in accordance with data protection regulations with the purpose of␣
↪improving care service provision across the UK.',
    disabled=False,
    layout=widgets.Layout(width='1300px')) #layout ammended to allow display of␣
↪full text
display(a)
```

Add result from checkbox to 'dataTofill' dataframe

[ ]:
```
dfTofill.iloc[0,7]=a.value
dfTofill
```

3 Inserting the date

3.0.1 DatePicker widget

To input the period which is an object (string)

[ ]:
```
testData.head(n=1)
```

[ ]:
```
b = widgets.DatePicker(
    description='Period',
    disabled=False
)
display(b)
```

[ ]:
```
dfTofill.iloc[0,1]=b.value
dfTofill
```

3.1 Inserting *org_code* and *type*

Compute descriptive statistics for testData to identify how many unique options there will be for the variables *org_code* and *type*

[ ]:
```
testData.describe(include='all')
```

Obtain the unique ODS code for the organisations in the testData

3.1.1 Selection widget

Selection widget for the 'org_code' was chosen as there are 11 options

Apply the `unique()` function to create the object 'org_code' which contains the options for the 'org_code' variable

[ ]:
```
org_code=list(testData['org_code'].unique())
org_code
```

Apply the **Selection** widget

[ ]:
```
c=widgets.Select(
    options=org_code,
    value='C82010',
    rows=len(org_code),
    description='ODS code:',
    disabled=False
)
display(c)
```

[ ]:
```
dfTofill.iloc[0,2]=c.value
dfTofill
```

3.1.2 Radio button widget

Radio button for the 'Type Variable' as this is one of only three options

Apply the `unique()` function to create the object 'type' which contains the options for the 'Type Variable'

[ ]:
```
type=list(testData['type'].unique())
type
```

Apply the **Radio button** widget

[ ]:
```
d=widgets.RadioButtons(
    options=type,
    value='other',
    description='Type:',
    disabled=False
)
display(d)
```

[ ]:
```
dfTofill.iloc[0,3]=d.value
dfTofill
```

3.2 Inserting Attendances, Breaches, and Admissions

These are all whole numbers, therefore the **IntText** widget will be used

3.2.1 IntText widget

3.2.2 Attendances

Insert value for ED attendances

[ ]:
```
e=widgets.IntText(
    value=0,
    description='Attendances:',
    disabled=False)
display(e)
```

[ ]:
```
dfTofill.iloc[0,4]=e.value
dfTofill
```

3.2.3 Breaches

Insert value for 4hr breaches for the defined period

[ ]:
```
f=widgets.IntText(
    value=0,
    description='Breaches:',
    disabled=False)
display(f)
```

[ ]:
```
dfTofill.iloc[0,5]=f.value
dfTofill
```

3.2.4 Admissions

Insert value for admissions from the ED

[ ]:
```
g=widgets.IntText(
    value=0,
    description='Admissions:',
    disabled=False)
display(g)
```

[ ]:
```
dfTofill.iloc[0,6]=g.value
dfTofill
```

4 Saving the collected data

4.1 Concatenating the collected data to the CollectData data frame.

`concat()` function is used to append the CollectData and dfTofill data frames.

[ ]:
```
CollectData  = pd.concat([CollectData, dfTofill])
display(CollectData)
```

4.2 Confirming that consent is obtained

[ ]:
```
CollectData=CollectData[CollectData['consent'] == True]
display(CollectData)
```

4.3 Saving the CollectData data frame

Saving the data collected by your data-capture tool to the working data folder:

[ ]:
```
CollectData.to_csv('../Data/CollectedData.csv', index=False)
```

Once all captured data collected, then save captured test data to your 'RawData' folder.

[ ]:
```
CollectData.to_csv('../RawData/CollectedDataFinal.csv', index=False)
```

[ ]:
```

```