Start

- Build Interactive Tabs

Bubble Sort

- Build

Selection Sort

- Build

Insertion Sort

- Build

Merge Sort

- Build

Merge Sort

- Build

< Table of Contents | References >

Build Interactive Tabs

In [1]:
```
exit()
```

In [1]:
```
%%script bash
# clean local URLs
rm -rf ./rc/Partition/*
rm -rf ./rc/Process/*
rm -rf ./rc/Sorted/*
rm -rf ./rc/Unsorted/*
```

In [2]:
```
from sys import path; path.insert(1,"../src")
```

In [3]:
```
def fResetLoopCounter():
    """Initialise loop counter name to zero.

Input:
Process: assign zero to global integer name nSwap
Output:
"""
    global nSwap
    nSwap=0
# --- END ---
```

In [4]:
```
def fResetLocalURL():
    """Initialise collection of local URLs.

Input:
Process: assign zero elements to global collection name nImagesLocalURL
Output:
"""
    global nImagesLocalURL
    nImagesLocalURL=[]
# --- END ---
```

In [5]:
```
def fAddSwapCount():
    """Count swap operations.

Input:
Process: increment global integer name nSwap
Output:
"""
    global nSwap
    nSwap+=1
# --- END ---
```

In [6]:
```
def fAddLocalURL(nParOuterIndex,nParInnerIndex,nParRC="Process"):
    """Append plot local URLs.

Input: nParOuterIndex (loop outer index); nParInnerIndex (loop inner index);
    nParRC (preassigned "Process" - inputs: "Unsorted" or "Partition" or "Sort
Process: append filepath URLs to global collection name nImagesLocalURL (appe
Output: collection of filepath URLS for use in fAddPlotTabs
"""
    nImagesLocalURL.append(f"./rc/{nParRC}/Outer{nParOuterIndex}Inner{nParInne
# --- END ---
```

In [7]:
```
def fAddBarLabels(nParCollection):
    """Add text (s) to the axes at coordinates x and y.

Input: nParCollection
Process: (len; range; matplotlib.pyplot.text)
Output:
"""
    from matplotlib.pyplot import text
    for nIndex in range(len(nParCollection)):
        text(x=nIndex,y=nParCollection[nIndex],
             s=nParCollection[nIndex],
             ha="center",va="bottom",fontsize=12)
    return nParCollection
# --- END ---
```

In [8]:
```
def fAddBarBlue(nParCollection,nParIndex):
    """Initialise collection of bar plot colours.

Input: nParCollection; nParIndex
Process: unsorted Input=nParCollection bar colours "gray";
    sorted Input=nParCollection bar colours "green";
    comparison process bar colours "blue" (len; sorted; pop; insert)
Output: collection of colours for use in fAddPlot
"""
    from matplotlib.pyplot import xlabel

    # module fubar - function fAddBarLabels
    fAddBarLabels(nParCollection=nParCollection) # label over bar

    nComparison=f"{nParCollection[nParIndex]}>{nParCollection[nParIndex+1]}"
    nTrueFalse=nParCollection[nParIndex]>nParCollection[nParIndex+1] # label

    nColor=[]
    if nParIndex==-1: # flag unsorted collection
        nColor=["gray"]*len(nParCollection) # bar colors "gray"
    elif nParCollection==sorted(nParCollection): # sorted collection
        nColor=["green"]*len(nParCollection) # bar colors "green"
    else: # algorithm walk-through process
        # package matplotlib - module pyplot
        xlabel(xlabel=f"Comparison: {nComparison} = {nTrueFalse} - Swap Count
        nColor=["gray"]*len(nParCollection) # reset bar colors
        nColor.pop(nParIndex) # remove element "gray"
        nColor.insert(nParIndex,"blue") # insert element "blue"
        nColor.pop(nParIndex+1) # remove adjacent "gray"
        nColor.insert(nParIndex+1,"blue") # insert adjacent "blue"
    return nColor # collection bar colours
# --- END ---
```

In [9]:
```
def fAddBarGreen(nParCollection,nParIndex):
    """Initialise collection of bar plot colours.

Input: nParCollection; nParIndex
Process: unsorted Input=nParCollection bar colours "gray";
    sorted Input=nParCollection bar colours "green";
    sorted partition bar colours "green" (len; append)
Output: collection of colours for use in fAddPlot
"""
    # module fubar - function fAddBarLabels
    fAddBarLabels(nParCollection=nParCollection)

    nColor=[]
    if nParIndex==len(nParCollection)-1: # determine final position
        nColor=["gray"]*nParIndex # bar colors "gray"
        nColor.append("green") # final position "green"
    else:
        nColor=["gray"]*nParIndex # bar colors "gray"
        nColor+=["green"]*(len(nParCollection)-nParIndex) # sorted partition
    return nColor # collection bar colours
# --- END ---
```

In [10]:
```
def fAddPlot(nParCollection,nParColor):
    """Make a bar plot at coordinate x with dimension height.
    Bars coloured based on some criteria.

Input: nParCollection; nParColor
Process: (matplotlib.pyplot.box,yticks,bar; len; range)
Output:
"""
    from matplotlib.pyplot import box,yticks,bar
    box(False); # remove plot box
    yticks([]) # remove y-axis ticks
    bar(x=range(len(nParCollection)), # x-axis collection indices
        height=nParCollection, # y-axis actual collection
        color=nParColor, # bar color
        tick_label=range(len(nParCollection))) # ticks represent indices
# --- END ---
```

In [11]:
```
def fBubbleSortTabs():
    """Populate algorithm Bubble Sort tabs for interative tab display.

Input:
Process: name nTabLabels global collection of string labels - len(nTabLabels)=
    name nTabText global collection of strings - len(nTabText)==len(nTabLabels
    name nTabOrder global collection of tab ordering - default is alphabetical
Output: collections for use in fAddPlotTabs
"""

    global nTabLabel,nTabText,nTabOrder
    nTabLabel=["Collection"]+["4"]*5+["3"]*4+["2"]*3+["1"]*2+["Sorted"] # plo
    nTabText=["Fig. 1.",
              "Fig. 2. Start pass 0.",
              "Fig. 3. Eq. (3) requires E swap.",
              "Fig. 4. Eq. (3) requires E swap.",
              "Fig. 5. Eq. (3) requires E swap.",
              "Fig. 6. End pass 0.",
              "Fig. 7. Start pass 1.",
              "Fig. 8. Eq. (3) requires E swap.",
              "Fig. 9. Eq. (3) requires E swap.",
              "Fig. 10. End pass 1.",
              "Fig. 11. Start pass 2.",
              "Fig. 12. Eq. (3) requires E swap.",
              "Fig. 13. End Pass 2.",
              "Fig. 14 .Start Pass 3.",
              "Fig. 15. End Pass 3.",
              "Fig. 16."
             ] # report referral text
    nTabOrder=["Collection","4","3","2","1","Sorted"] # default alphabeticall
# --- END ---
```