EaD Comprehensive Lesson Plans

or 0248043888

https://www.TeachersAvenue.net
https://TrendingGhana.net
https://www.mcgregorinriis.com

BASIC 9

WEEKLY LESSON PLAN – WEEK 8

Strand:	Productive Software	**Sub-Strand:**	Introduction to Electronic Spreadsheet		
Content Standard:	B9.2.4.1. Demonstrate How to Use Spreadsheet (Advanced Operations)				
Indicator (s)	B9.2.4.1.2. Demonstrate how to use styles, themes, templates and macros	**Performance Indicator;** Learners can differentiate between themes, templates and macros.			
Week Ending	01-03-2024				
Class	B.S.9	**Class Size:**		**Duration:**	
Subject	Computing				
Reference	Computing Curriculum, Teachers Resource Pack, Learners Resource Pack				
Teaching / Learning Resources	Personal Computer, Microsoft Excel Application, Poster, Charts, YouTube Videos	**Core Competencies:**	• Operational skills • Manipulative skills		

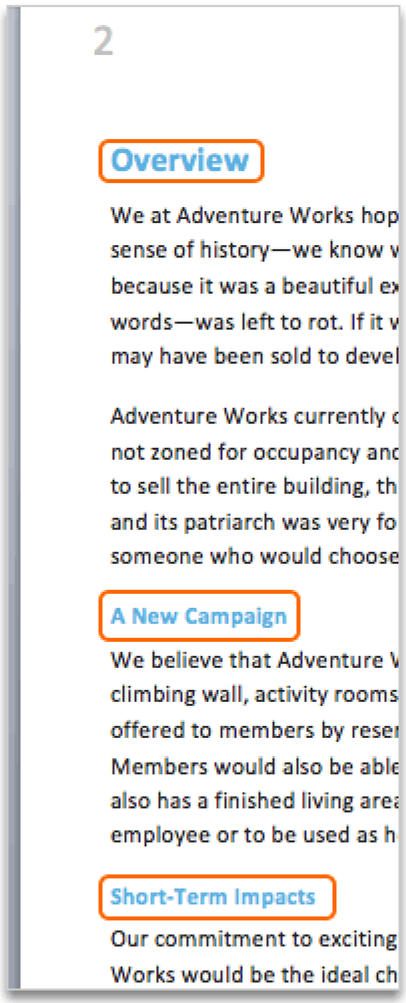
DAY/DATE	**PHASE 1 : STARTER**	**PHASE 2: MAIN**	**PHASE 3: REFLECTION**
TUESDAY	Assist Learners to differentiate between templates, themes and macros as used in spreadsheet.	1. Demonstrate on creating new spreadsheet documents from predefined templates in MS Excel. 2. Assist Learners to practice creating new spreadsheet documents from predefined templates. 3. Learners in small groups to discuss and practice the use of styles and themes on sample worksheets. **Template** Templates are files that help you design interesting, compelling, and professional-looking documents. They contain content and design elements that you can use as a starting point when creating a document. All the formatting is complete; you add what you want to them. Examples are resumes, invitations, and newsletters.	Through questions and answers, conclude the lesson. **Exercise;** Differentiate between templates, themes and macros.

Theme

To give your document a designer-quality look — a look with coordinating theme colors and theme fonts — you'll want to apply a theme. You can use and share themes among the Office for Mac applications that support themes, such as Word, Excel, and PowerPoint. For example, you can create or customize a theme in PowerPoint, and then apply it to a Word document or Excel sheet. That way, all of your related business documents have a similar look and feel.

Word styles

Themes provide a quick way to change the overall color and fonts. If you want to change text formatting quickly, Word styles are the most effective tools. After you apply a style to different sections of text in your document, you can change the formatting of this text simply by changing the style. Word includes many types of styles, some of which can be used to create reference tables in Word. For example, the Heading style, which is used to

		create a Table of Contents?  2 Overview We at Adventure Works hop sense of history—we know v because it was a beautiful ex words—was left to rot. If it v may have been sold to devel Adventure Works currently c not zoned for occupancy and to sell the entire building, th and its patriarch was very fo someone who would choose A New Campaign We believe that Adventure V climbing wall, activity rooms offered to members by reser Members would also be able also has a finished living area employee or to be used as h Short-Term Impacts Our commitment to exciting Works would be the ideal ch	
FRIDAY	Learners brainstorm to explain the term "table visualization" in Excel.	1. Demonstrate on formatting a dataset by applying styles and themes. 2. Assist learners to practice formatting a dataset by applying styles and themes. 3. Discuss with the Learners on how to format displays and values in Excel. 4. Assist Learners to practice concatenating data frame outputs in Excel. **Formatting the Display** **Formatting Values** The Styler distinguishes the *display* value from the *actual* value, in both data values and index or columns headers. To control the display value, the text is printed in each cell as a string, and we can use the .format() and .format_index() methods to manipulate this according to a format spec string or a callable that	Learners brainstorm to practice how to hide values in spreadsheet. **Exercise** State the steps to follow to format values in Excel.

takes a single value and returns a string. It is possible to define this for the whole table, or index, or for individual columns, or MultiIndex levels. We can also overwrite index names.

Additionally, the format function has a **precision** argument to specifically help format floats, as well as **decimal** and **thousands** separators to support other locales, an **na_rep** argument to display missing data, and an **escape** and **hyperlinks** arguments to help displaying safe-HTML or safe-LaTeX. The default formatter is configured to adopt pandas' global options such as styler.format.precision option, controllable using with pd.option_context('format.precision', 2):

[2]:
```
import pandas as pd
import numpy as np
import matplotlib as mpl

df = pd.DataFrame({
    "strings": ["Adam", "Mike"],
    "ints": [1, 3],
    "floats": [1.123, 1000.23]
})
df.style \
  .format(precision=3, thousands=".", decimal=",") \
  .format_index(str.upper, axis=1) \
  .relabel_index(["row 1", "row 2"], axis=0)
```
[2]:

	STRINGS	**INTS**	**FLOATS**
row 1	Adam	1	1,123
row 2	Mike	3	1.000,230

Using Styler to manipulate the display is a useful feature because maintaining the indexing and data values for other purposes gives greater control. You do not have to overwrite your DataFrame to display it how you like. Here is a more comprehensive example of using the formatting functions whilst still relying on the underlying data for indexing and calculations.

[3]:
```
weather_df = pd.DataFrame(np.random.rand(10,2)*5,
                          index=pd.date_range(start="2021-01-01", 
periods=10),
                          columns=["Tokyo", "Beijing"])

def rain_condition(v):
    if v < 1.75:
        return "Dry"
    elif v < 2.75:
        return "Rain"
    return "Heavy Rain"

def make_pretty(styler):
    styler.set_caption("Weather Conditions")
    styler.format(rain_condition)
    styler.format_index(lambda v: v.strftime("%A"))
    styler.background_gradient(axis=None, vmin=1,
vmax=5, cmap="YlGnBu")
    return styler

weather_df
```

[3]:

	Tokyo	**Beijing**
2021-01-01	4.985092	3.473298
2021-01-02	3.264144	0.033467
2021-01-03	4.678288	4.567539
2021-01-04	2.983053	4.141140
2021-01-05	2.145126	3.784963
2021-01-06	4.197181	1.994896
2021-01-07	2.218433	1.584807
2021-01-08	3.002908	3.734839
2021-01-09	2.682287	2.879510
2021-01-10	3.343583	2.592540

[4]:

```
weather_df.loc["2021-01-04":"2021-01-
08"].style.pipe(make_pretty)
```

[4]:

Weather Conditions

	Tokyo	**Beijing**
Monday	Heavy Rain	Heavy Rain
Tuesday	Rain	Heavy Rain
Wednesday	Heavy Rain	Rain
Thursday	Rain	Dry

Friday	Heavy Rain	Heavy Rain

Hiding Data

The index and column headers can be completely hidden, as well subselecting rows or columns that one wishes to exclude. Both these options are performed using the same methods.

The index can be hidden from rendering by calling .hide() without any arguments, which might be useful if your index is integer based. Similarly column headers can be hidden by calling .hide(axis=”columns”) without any further arguments.

Specific rows or columns can be hidden from rendering by calling the same .hide() method and passing in a row/column label, a list-like or a slice of row/column labels to for the subset argument.

Hiding does not change the integer arrangement of CSS classes, e.g. hiding the first two columns of a DataFrame means the column class indexing will still start at col2, since col0 and col1 are simply ignored.

[5]:

```
df = pd.DataFrame(np.random.randn(5, 5))
df.style \
  .hide(subset=[0, 2, 4], axis=0) \
  .hide(subset=[0, 2, 4], axis=1)
```

[5]:

	1	**3**
1	0.679529	-0.397631
3	-1.765405	-1.166462

To invert the function to a **show** functionality it is best practice to compose a list of hidden items.

[6]:

```
show = [0, 2, 4]
df.style \
  .hide([row for row in df.index if row not in show],
axis=0) \
  .hide([col for col in df.columns if col not in show],
axis=1)
```

[6]:

	0	**2**	**4**

0	0.387468	0.348897	2.144013
2	-0.631555	-0.621477	2.273923
4	1.203479	-0.420140	0.946772

Concatenating DataFrame Outputs

Two or more Stylers can be concatenated together provided they share the same columns. This is very useful for showing summary statistics for a DataFrame, and is often used in combination with DataFrame.agg.

Since the objects concatenated are Stylers they can independently be styled as will be shown below and their concatenation preserves those styles.

[7]:
```
summary_styler = df.agg(["sum", "mean"]).style \
                .format(precision=3) \
                .relabel_index(["Sum", "Average"])
df.style.format(precision=1).concat(summary_styler)
```
[7]:

	0	**1**	**2**	**3**	**4**
0	0.4	0.3	0.3	-0.1	2.1
1	0.0	0.7	-2.7	-0.4	0.9
2	-0.6	-1.4	-0.6	0.7	2.3
3	-0.9	-1.8	-0.9	-1.2	0.4
4	1.2	-0.4	-0.4	1.1	0.9
Sum	0.063	-2.554	-4.271	0.121	6.726
Average	0.013	-0.511	-0.854	0.024	1.345

Name of Teacher: **School:** **District:**