Introduction to Machine Learning in Engineering Science

National Cheng Kung University Department of Engineering Science

Instructor: Chi-Hua Yu

HW 1

Programming, Due 9:00, Tuesday, October 4th, 2021

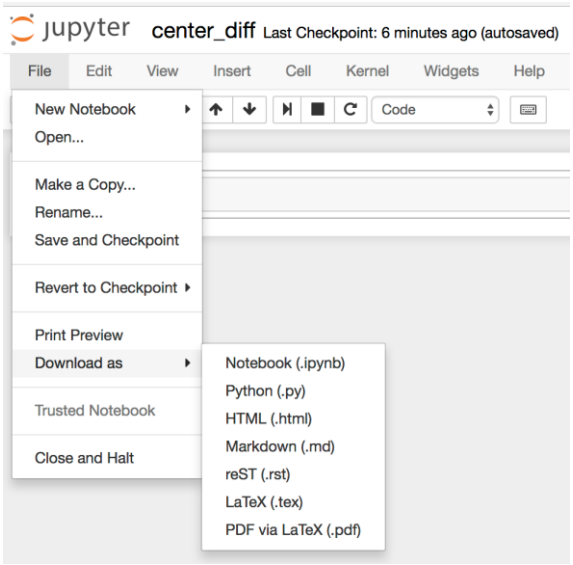
Late submission within 24 hours: score*0.9.

Late submission before post of solution: score*0.8 (the solution will usually be posted within a week); no late submission after the post of solution

---

Lab Submission Procedure (請仔細閱讀)

1. You should submit your Jupyter notebook and Python script (`*.py`, in Jupyter, click File, Download as, Python (`*.py`)).



2. Name a folder using your student id and lab number (e.g., n96081494_HW1), put all the python scripts into the folder and zip the folder (e.g., n96081494_ HW1.zip).

3. Submit your lab directly through the course website.

Total 100%

1. (20%) Name your Jupyter notebook `find_users` and Python script `find_users.py`. Consider the following `dict` of `users`, each contains user's id and `name` and a `list` of pairs (`user_id`, `interest`) that represents the technical topics he or she is interested:

```
users = {
    6: "Hicks", 7: "Devin", 8: "Kate", 0: "Hero",
    8: "Kate", 9: "Klein", 1: "Dunn", 3: "Chi",
    2: "Sue", 4: "Thor", 5: "Clive"
}
```

```
interests = [
    (0, "Hadoop"), (0, "Big Data"), (0, "HBase"), (0, "Java"),
    (0, "Spark"), (0, "Storm"), (0, "Cassandra"),
    (1, "NoSQL"), (1, "MongoDB"), (1, "Cassandra"), (1, "HBase"),
    (1, "Postgres"), (2, "Python"), (2, "scikit-learn"), (2, "scipy"),
    (2, "numpy"), (2, "statsmodels"), (2, "pandas"), (3, "R"), (3,
"Python"),
    (3, "statistics"), (3, "regression"), (3, "probability"),
    (4, "machine learning"), (4, "regression"), (4, "decision trees"),
    (4, "libsvm"), (5, "Python"), (5, "R"), (5, "Java"), (5, "C++"),
    (5, "Haskell"), (5, "programming languages"), (6, "statistics"),
    (6, "probability"), (6, "mathematics"), (6, "theory"),
    (7, "machine learning"), (7, "scikit-learn"), (7, "Mahout"),
    (7, "neural networks"), (8, "neural networks"), (8, "deep
learning"),
    (8, "Big Data"), (8, "artificial intelligence"), (9, "Hadoop"),
    (9, "Java"), (9, "MapReduce"), (9, "Big Data")
]
```

Write a simple query function `users_who_like` to the notebook that will return a list of all users who like the specific topic. Below are some sample runs:

```
def user_who_like(x):
```

```
1 user_who_like('Python')
```

```
['Sue', 'Chi', 'Clive']
```

```
1 user_who_like(['Java', 'Big Data'])
```

```
['Klein', 'Hero']
```

2. **(20%)** Name your Jupyter notebook `sin_approx` and Python script `sin_approx.py`. Using a `while` loop to implement an approximation of sine function with polynomial:

$$\sin(x) \approx x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

Compute the right-hand side for powers up to $N = 25$. Below is a sample output:
(You can choose to use the `factorial` written by yourself or import `factorial` from `math`.)
(The value of the result only corresponds to 7 decimal places.)

```
def sin(x):
    return y
```

```
1 x = 1.6
2 y = sin(x)
3 print ('sin({}) approximation is {}'.format(x, y))
```

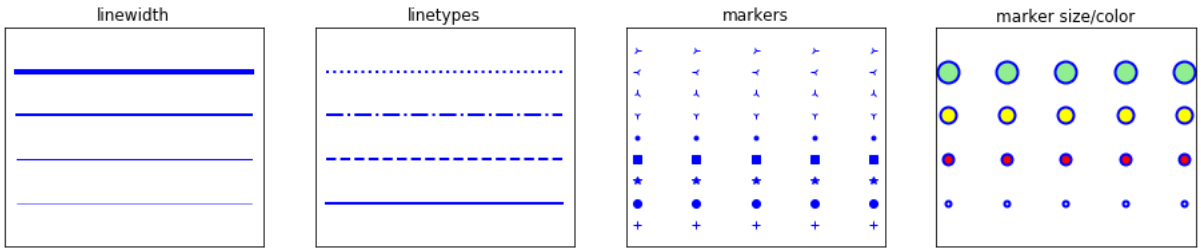
```
sin(1.6) approximation is 0.9995736030415051
```

3. **(30%)** Name your Jupyter notebook `LineProp` and Python script `LineProp.py`. Plot a figure

with four subplots to illustrate the line properties including

```
linewidths = [0.5, 1.0, 2.0, 4.0]
linestyles = ['-', '--', '-.', ':']
markers = ['+', 'o', '*', 's', '.', '1', '2', '3', '4']
markersizecolors = [(4, "white"), (8, "red"), (12, "yellow"), (16,
"lightgreen")]
```

Below is a sample output:



(**Big hint**: the following code will give you the first subplot)

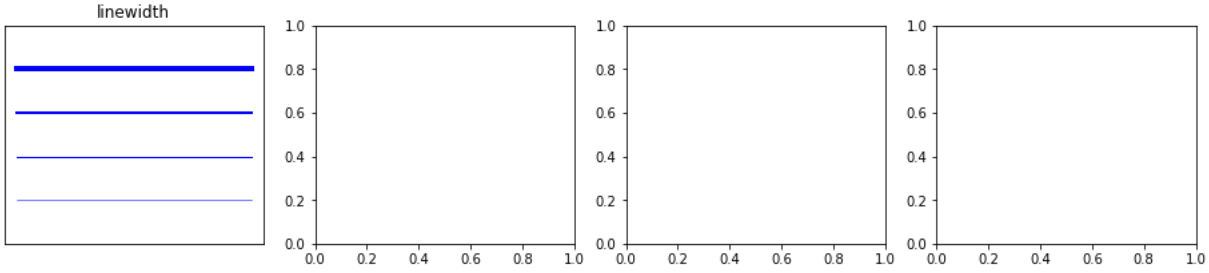
```
x = np.linspace(-5, 5, 5)
y = np.ones_like(x)

def axes_settings(fig, ax, title, ymax):
    ax.set_xticks([])
    ax.set_yticks([])
    ax.set_ylim(0, ymax+1)
    ax.set_title(title)

fig, axes = plt.subplots(1, 4, figsize=(16,3))

# Line width
linewidths = [0.5, 1.0, 2.0, 4.0]
for n, linewidth in enumerate(linewidths):
    axes[0].plot(x, y + n, color="blue", linewidth=linewidth)

axes_settings(fig, axes[0], "linewidth", len(linewidths))
```



4. (30%) Name your Jupyter notebook `SimpleRegr` and Python script `SimpleRegr.py`. For a set

of n samples $(x_0, y_0), (x_1, y_1), \cdots, (x_{n-1}, y_{n-1})$, you can easily fit a line $y = \hat{\beta}_0 + \hat{\beta}_1 x$ for these samples. This fitting is called simple linear regression. Using the least square fitting criterion, we can show:

$$\hat{\beta}_1 = \frac{\sum_{i=0}^{n-1}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=0}^{n-1}(x_i - \bar{x})^2}$$
$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

where $\bar{x}$ and $\bar{y}$ are the sample mean. Implement a function `least_squares_fit` to compute $\hat{\beta}_0$ and $\hat{\beta}_1$ from a set of samples. Report $\hat{\beta}_0$ and $\hat{\beta}_1$ and plot these samples and the fitting line. Below is a sample run:

```
def least_squares_fit(x, y):
```

```
X = np.array([1, 2, 3, 4])
Y = np.array([9, 13, 14, 18])
beta0, beta1 = least_squares_fit(X, Y)
print("From home-made linear regression model")
print('beta0 =', beta0)
print('beta1 =', beta1)
```

```
From home-made linear regression model
beta0 = 6.5
beta1 = 2.8
```

And a sample plot (`line width = 3, marker size = 20`):