University of Oum El Bouaghi
Faculty of Exact Sciences & Natural & Life Sciences
Department of Mathematics & Computer science

Master 1: AI and Data Science. 14th Jan, 2025. 1h 30min

Full Name :

Group :

Exam 01: Advanced Python Language

Exercise 01: (05pts)

1. **What are the three main types of data in data science? (1pt)**

☐. Simple, Complex, Raw
■. Structured, Semi-structured, Unstructured
☐. Tabular, Graphical, Analytical
☐. Binary, Categorical, Continuous

2. **Why is NumPy faster than regular Python lists? (1pt)**

☐. It is written in Java.
☐. It has more memory than Python lists.
☐. It converts lists into dictionaries.
■. It uses low-level C implementations for computations.

3. **What does a Pandas DataFrame represent? (1pt)**

☐. 1D labeled array
☐. Tabular array without labels
■. 2D labeled data structure
☐. Visualization of data

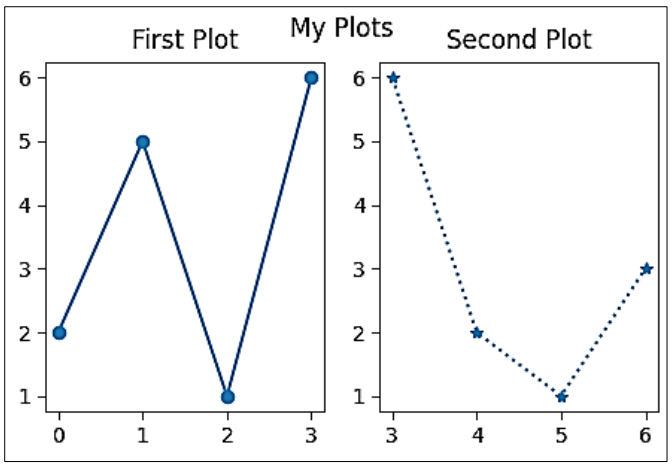
4. **Describe (or Usage of) the functions bellow: (2pts)**

Function	*Description*	*Function*	*Description*
fillna()	Replaces NaN values with `a number`	train_test_split()	Splits data into training and testing sets.
dropna()	Removes all rows with NaN values	linspace()	generates evenly spaced numbers over a specified range
plot()	Create visualizations such as line plots	fit()	train a model
grid()	add a grid to the plot	flatten()	converts a multi-dimensional array to 1D

Exercise 02: What are the outputs of the following snippets: (6pts)

data = { 'A': [1, 2], 'B': [3, 4] } **df = pd.DataFrame(data)** **print(df['A'])**	0 1 1 2 Name: A, dtype: int64	**arr = np.array([10, 20, 30, 40])** **print(arr[arr > 20])**	[30 40]
arr1 = np.array([[1, 2], [3, 4]]) **arr2 = np.array([[5, 6], [7, 8]])** **print(arr1 + arr2)**	[[6 8] [10 12]]	**data = ['Name': ['Alice', 'Bob'], 'Age': [25, 30]]** **df = pd.DataFrame(data)** **print(df)**	SyntaxError / use "{" for dictionaries.
nums = [1, 2, 3, 4] **result = [x**2 for x in nums if x % 2 == 0]** **print(result)**	[4, 16]	**toDO = lambda s : s[: : -1]** **print(toDO("lambda"))**	adbmal

Exercise 03: Write the python code of the following plot. (4pts)




```
import matplotlib.pyplot as plt      (0.5)
import numpy as np                   (0.5)

#plot 1:
x = np.array([0, 1, 2, 3])           (0.25)
y = np.array([2, 5, 1, 6])           (0.25)
plt.subplot(1, 2, 1)                 (0.25)
plt.plot(x,y,"o-")                   (0.25)
plt.title("First Plot")              (0.25)

#plot 2:
x = np.array([3, 4, 5, 6])           (0.25)
y = np.array([6, 2, 1, 3])           (0.25)
plt.subplot(1, 2, 2)                 (0.25)
plt.plot(x,y,"*:")                   (0.25)
plt.title("Second Plot")             (0.25)

plt.suptitle("My Plots")             (0.25)
plt.show()                           (0.25)
```

Exercise 04: Complete the missing in the following code (5pts)

```
import numpy as np
from sklearn.datasets import load_iris                    (0.5pts)
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler          (0.5pts)
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target                                           (0.5pts)
# Build a pipeline
pipeline = Pipeline ([                                    (0.25pts)
    ("scaler", StandardScaler() ),                        (0.25pts)
    ("classifier", LogisticRegression() )                 (0.25pts)
])
# Define the parameter grid for Grid Search
param_grid = {
    "classifier__C": [0.1, 1, 10],
    "classifier__solver": ["liblinear", "lbfgs"]
}
```

```
#Set up GridSearchCV with 10-fold cross-validation
grid_search = GridSearchCV(
    estimator= pipeline,                                  (0.25pts)
    param_grid= param_grid ,                              (0.25pts)
    cv= 10 ,                                              (0.25pts)
    scoring="accuracy",
    verbose=1
)
# Fit GridSearchCV to the data
grid_search.fit( X , y )                                  (1pts)
# Output the best parameters and the corresponding score
print("Best Parameters:", grid_search.best_params_ )
(0.5pts)
print("Best Cross-Validation Accuracy:",
grid_search.best_score_ )                                 (0.5pts)
```