FAT Prep

Created by	Ⓨ Yash Bishnoi
Status	Not started

Python Exam Answers

1.

a) **Program to find the largest prime factor of an integer n:**

```
n = int(input("Enter an integer: "))
def largest_prime_factor(n):
    i = 2
    while i * i <= n:
        if n % i:
            i += 1
        else:
            n //= i
    return n
print("Largest prime factor:", largest_prime_factor(n))
```

b) **Program to find minimum number of desks required (3 students per desk):**

```
a, b, c = map(int, input("Enter students in 3 classes: ").split())
def desks_needed(x):
    return (x + 2) // 3
print("Minimum desks:", desks_needed(a) + desks_needed(b) + desks_nee
ded(c))
```

2.

a) **Convert tuple to dictionary:**

```
data = (("IDP", 101), ("Name", "Alice"), ("Age", 23), ("IDP", 110), ("IDP", 700),
("Name", "Tom"))
d = {}
for k, v in data:
```

```
    d.setdefault(k, []).append(v)
print(d)
```

b) **Extract abnormal BP values (SP/DP != 120/80):**

```
bp = [(110, 90), (120, 80), (130, 70), (118, 78)]
abnormal = [b for b in bp if b != (120, 80)]
print("Abnormal BP values:", abnormal)
```

c) **Add SpO2 if HR > 100:**

```
patients = [{"Name": "Tom", "HR": 110}, {"Name": "Alice", "HR": 95}]
for p in patients:
    if p["HR"] > 100:
        p["SpO2"] = int(input(f"Enter SpO2 for {p['Name']}: "))
print(patients)
```

3.

a) **Predict output:**

```
import re
S = "832-3928.23 382.28378 5 92,000 32-928 +32 -32 ABC"
R1 = re.findall("[0-9.]+", S)  # Extracts all digit and dot combos
print(R1)
# Output: ['832', '3928.23', '382.28378', '5', '92', '000', '32', '928', '32', '3
2']
R2 = re.findall("[^0-9.]+", S)  # Everything else
print(R2)
# Output: ['-', ' ', ' ', ' ', ',', ' -', ' +', ' -', ' ABC']
```

b) **Fix regex for valid emails:**

```
import re
text = "Contact us at support@example.com or sales@shop123.com. Avoid
some.gone and test@com."
pattern = r"\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}\b"
emails = re.findall(pattern, text)
```

```
print(emails)
# Output: ['support@example.com', 'sales@shop123.com']
```

4.Ways to pass values to functions & using functions as modules:

```
# Positional arguments
def greet(name):
    print("Hello", name)
greet("Tom")

# Keyword arguments
def student(name, age):
    print(f"{name} is {age} years old")
student(age=22, name="Alice")

# Default values
def hello(name="User"):
    print("Hi", name)
hello()

# Variable-length args
def total(*args):
    return sum(args)
print(total(2, 4, 6))

# Function as module
# module.py
"""
def square(x):
    return x * x
"""
# main.py
"""
import module
print(module.square(5))
"""
```

5.

```python
import pandas as pd

df = pd.read_csv("transactions.csv")

# a) Total Revenue
print("Total Revenue:", df['TransAmt'].sum())

# b) Top 5 Transactions
print("Top 5 Transactions:\n", df.nlargest(5, 'TransAmt'))

# c) Summary report
with open("report.txt", "w") as f:
    f.write("Total Revenue: " + str(df['TransAmt'].sum()) + "\n")
    f.write("Top 5 Transactions:\n")
    f.write(df.nlargest(5, 'TransAmt').to_string())
```

6.Output Prediction and Explanation:

```python
class A:
    def _init_(self):
        self.value = 10
    def show(self):
        return self.value * 2

class B(A):
    def _init_(self):
        super()._init_()
        self.value = 20
    def show(self):
        return self.value + 5

class C(B):
    def __init__(self):
        super()._init_()
        self.value = 30
```

```
obj = C()
print(obj.show())  # Output: 35 (uses B.show, self.value is 30)
```

If `super()._init_()` is removed from class C:

- `self.value = 20` from class B will not be initialized.
- So `self.value` will remain 30 only, no impact here since C overrides it.
- But if `value` wasn't explicitly set in C, it would have remained undefined.

7.Pandas Import/Export CSV & Excel:

```
import pandas as pd

# Import CSV
df = pd.read_csv("data.csv")

# Export CSV
df.to_csv("new_data.csv", index=False)

# Import Excel
df2 = pd.read_excel("data.xlsx")

# Export Excel
df2.to_excel("output.xlsx", index=False)
```

8.ATM Simulation with Exception Handling:

```
accounts = {"123": 1000, "456": 1500}
try:
    acc = input("Enter account number: ")
    if acc not in accounts:
        raise KeyError("Account not found")
    action = input("Withdraw or Deposit? ").lower()
    amount = int(input("Enter amount: "))
    if amount <= 0:
        raise ValueError("Amount must be positive")
    if action == "withdraw":
```

```
        if accounts[acc] < amount:
            raise PermissionError("Insufficient funds")
        accounts[acc] -= amount
    elif action == "deposit":
        accounts[acc] += amount
    else:
        raise Exception("Invalid action")
    print("Updated balance:", accounts[acc])
except Exception as e:
    print("Error:", e)
```

9.

(a)

Inheritance and Polymorphism Example:

```
class Account:
    def withdraw(self, amount):
        raise NotImplementedError

class Savings(Account):
    def withdraw(self, amount):
        return f"Savings: Withdrawn {amount} with 4% penalty"

class Checking(Account):
    def withdraw(self, amount):
        return f"Checking: Withdrawn {amount}"

accounts = [Savings(), Checking()]
for acc in accounts:
    print(acc.withdraw(100))
```

(b) **Multiple Inheritance Risks and Resolution:**

- Risks: Method Resolution Order (MRO) confusion, ambiguity.
- Resolution:

```
class A:
    def greet(self): print("Hello from A")
```

```
class B:
    def greet(self): print("Hello from B")
class C(A, B): pass
obj = C()
obj.greet()  # Uses MRO: A > B
```

- Use `super()` properly and inspect `C.__mro__` for order.

10.
(a)
Colormaps in Matplotlib:

```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(0, 10, 100)
y = np.sin(x)

# Jet colormap
plt.scatter(x, y, c=y, cmap='jet')
plt.colorbar()
plt.title("Jet Colormap")
plt.show()

# Viridis colormap
plt.scatter(x, y, c=y, cmap='viridis')
plt.colorbar()
plt.title("Viridis Colormap")
plt.show()

# Plasma colormap
plt.scatter(x, y, c=y, cmap='plasma')
plt.colorbar()
plt.title("Plasma Colormap")
plt.show()
```

(b) **JSON Movie Details + Browser View:**

```
<script>
const movies = [
  {"title": "Inception", "rating": 8.8},
  {"title": "Interstellar", "rating": 9.0},
  {"title": "The Matrix", "rating": 8.7}
];
document.write("<ul>");
movies.forEach(m => {
  document.write(`<li>${m.title} - Rating: ${m.rating}</li>`);
});
document.write("</ul>");
</script>
```

End of Answers