: Python

Table of contents

Python

1

1.1 1.

- **uv**
- **Git GitHub**
- **VS Code**

1.2 2. Python

-
-
-

1.3 3.

- :
- :
- :
- :

1.4 4.

- : if elif else
- : for while
- : break continue

1.5 5.

-
-
-
-

1.6 6.

- try-except
-
- finally

1.7 7.

-
-
-

2

2.1 1:

```
#  :
students = [
    {" ": "  ", " ": 85, " ": 92, " ": 78},
    {" ": "  ", " ": 93, " ": 87, " ": 95},
    {" ": "  ", " ": 76, " ": 84, " ": 89}
]

# 1.
print("===     ===")
for student in students:
    name = student[" "]
    scores = [student[" "], student[" "], student[" "]]
    average = sum(scores) / len(scores)
    print(f"{name}: {average:.1f} ")

# 2.
subjects = [" ", " ", " "]
print("\n===     ===")
for subject in subjects:
    total = sum(student[subject] for student in students)
    average = total / len(students)
    print(f"{subject}: {average:.1f} ")

# 3.
print("\n===     ===")
for subject in subjects:
    max_score = 0
    top_student = ""
    for student in students:
        if student[subject] > max_score:
            max_score = student[subject]
```

```
            top_student = student[" "]
    print(f"{subject}: {top_student} ({max_score} )")
```

```
===     ===
  : 85.0
  : 91.7
  : 83.0

===     ===
 : 84.7
 : 87.7
 : 87.3

===      ===
 :    (93 )
 :    (92 )
 :    (95 )
```

2.2 2:

```
def calculate_bmi(weight, height):
    """BMI (Body Mass Index)     """
    try:
        # BMI =  (kg) /  (m)^2
        bmi = weight / (height ** 2)
        return round(bmi, 2)
    except ZeroDivisionError:
        return "  :   0    "
    except TypeError:
        return "  :       "

def get_bmi_category(bmi):
    """BMI       """
    if isinstance(bmi, str):  #
        return bmi

    if bmi < 18.5:
        return "  "
    elif bmi < 25:
```

```
        return " "
    elif bmi < 30:
        return "  1"
    else:
        return "  2 "

#
test_data = [
    {" ": " ", " ": 70, " ": 1.75},
    {" ": " ", " ": 55, " ": 1.60},
    {" ": " ", " ": 85, " ": 1.80},
    {" ": "   ", " ": 70, " ": 0},  #
]

print("=== BMI    ===")
for person in test_data:
    name = person[" "]
    weight = person[" "]
    height = person[" "]

    bmi = calculate_bmi(weight, height)
    category = get_bmi_category(bmi)

    if isinstance(bmi, str):  #
        print(f"{name}: {bmi}")
    else:
        print(f"{name}: BMI={bmi},  ={category}")
```

```
=== BMI    ===
 : BMI=22.86,  =
 : BMI=21.48,  =
 : BMI=26.23,  =  1
  :   :   0
```

2.3 3:

```
import json
from datetime import datetime, timedelta
```

```python
#
sample_data = [
    {" ": "2024-01-01", " ": 120000, "  ": 45},
    {" ": "2024-01-02", " ": 98000, "  ": 38},
    {" ": "2024-01-03", " ": 145000, "  ": 52},
    {" ": "2024-01-04", " ": 87000, "  ": 33},
    {" ": "2024-01-05", " ": 167000, "  ": 61},
]

def analyze_sales_data(data):
    """       """
    if not data:
        return {"  ": "    "}

    #
    total_sales = sum(day[" "] for day in data)
    total_customers = sum(day["  "] for day in data)
    average_sales = total_sales / len(data)
    average_customers = total_customers / len(data)

    #
    best_day = max(data, key=lambda x: x[" "])
    worst_day = min(data, key=lambda x: x[" "])

    #
    customer_avg_spending = []
    for day in data:
        if day["  "] > 0:
            avg = day[" "] / day["  "]
            customer_avg_spending.append(avg)

    overall_avg_spending = sum(customer_avg_spending) / len(customer_avg_spending)

    return {
        "   ": f"{data[0][' ']}  {data[-1][' ']}",
        "  ": f"{total_sales:,} ",
        "   ": f"{total_customers:,} ",
        "    ": f"{average_sales:,.0f} ",
        "     ": f"{average_customers:.1f} ",
        "    ": f"{best_day[' ']} ({best_day[' ']:,} )",
        "    ": f"{worst_day[' ']} ({worst_day[' ']:,} )",
        "      ": f"{overall_avg_spending:,.0f} "
```

```
    }

#
analysis = analyze_sales_data(sample_data)

print("===      ===")
for key, value in analysis.items():
    print(f"{key}: {value}")
```

```
===      ===
   : 2024-01-01   2024-01-05
  : 617,000
   : 229
    : 123,400
     : 45.8
    : 2024-01-05 (167,000 )
    : 2024-01-04 (87,000 )
      : 2,682
```

3

3.1 1:

```
class HouseholdBudget:
    """    """

    def __init__(self):
        self.transactions = []
        self.categories = {
            " ": [" ", "  ", "   "],
            " ": [" ", "  ", "  ", "  ", "   "]
        }

    def add_transaction(self, date, category, amount, description=""):
        """   """
        transaction = {
            " ": date,
            "  ": category,
            " ": amount,
            " ": description,
            "ID": len(self.transactions) + 1
        }
        self.transactions.append(transaction)
        return f"     : {description} ({amount:,})"

    def get_balance(self):
        """   """
        income = sum(t[" "] for t in self.transactions
                     if t["  "] in self.categories[" "])
        expense = sum(t[" "] for t in self.transactions
                      if t["  "] in self.categories[" "])
        return income - expense
```

```
    def get_summary_by_category(self):
        """    """
        summary = {}
        for transaction in self.transactions:
            category = transaction["  "]
            amount = transaction[" "]
            if category in summary:
                summary[category] += amount
            else:
                summary[category] = amount
        return summary

    def get_monthly_report(self, year, month):
        """    """
        monthly_transactions = [
            t for t in self.transactions
            if t[" "].startswith(f"{year}-{month:02d}")
        ]

        if not monthly_transactions:
            return f"{year} {month}      "

        income = sum(t[" "] for t in monthly_transactions
                     if t["  "] in self.categories[" "])
        expense = sum(t[" "] for t in monthly_transactions
                      if t["  "] in self.categories[" "])

        return {
            " ": f"{year} {month} ",
            " ": f"{income:,} ",
            " ": f"{expense:,} ",
            " ": f"{income - expense:,} ",
            "  ": len(monthly_transactions)
        }

#
budget = HouseholdBudget()

#
print("===      ===")
print(budget.add_transaction("2024-01-01", " ", 300000, " "))
print(budget.add_transaction("2024-01-03", " ", -5000, "     "))
```

```
print(budget.add_transaction("2024-01-05", "  ", -80000, " "))
print(budget.add_transaction("2024-01-10", "  ", -12000, "   "))
print(budget.add_transaction("2024-01-15", "   ", 50000, " "))

#
print(f"\n   : {budget.get_balance():,} ")

#
print("\n===      ===")
summary = budget.get_summary_by_category()
for category, amount in summary.items():
    print(f"{category}: {amount:,} ")

#
print("\n===      ===")
report = budget.get_monthly_report(2024, 1)
if isinstance(report, dict):
    for key, value in report.items():
        print(f"{key}: {value}")
else:
    print(report)
```

```
===       ===
      :    (300,000 )
      :        (-5,000 )
      :    (-80,000 )
      :      (-12,000 )
      :    (50,000 )

   : 447,000

===       ===
  : 300,000
  : -5,000
   : -80,000
   : -12,000
    : 50,000

===      ===
  : 2024 1
  : 350,000
  : -97,000
```

: 447,000
 : 5

4

4.1 1 ( )

- □
- □ print
- □

4.2 2 ( )

- □ if for
- □
- □

4.3 3 ( )

- □
- □
- □

4.4 4 ( )

- □
- □
- □

5

1. :
2. :
3. : Web
4. : pandas requests Flask

6

Python

6.1

- :
- :
- :
- :