ATOMCAMP

Python Mini-Project: Space Mission Control
Scenario:
You're working as a Python developer for Intergalactic Space Mission Control (ISMC).
Your job is to write Python programs to help astronauts and engineers track their missions, crew details, and spaceship performance using the given dataset (space_missions.csv).
This project will test your knowledge of Python basics, data structures, control flow, functions, and exception handling.

Instructions
Work in **Google Colab** or **Jupyter Notebook**.
Use the provided file **space_missions.csv**.
Answer all **20 questions** below.
Add **comments** in your code explaining each step.
Submit the **.ipynb** file once completed.

```
import pandas as pd
```

```
data=pd.read_csv("/content/space_missions.csv")
```

```
data.head()
```

	mission_id	mission_name	crew_count	destination	duration_days	success	fuel_level	commander	year
0	1	Orion-Alpha	5	Moon	12	True	78.4	Amelia Vega	2005
1	2	Apollo-X	3	Mars	340	False	12.7	Liam Carter	2010
2	3	Voyager-Pulse	6	ISS	45	True	54.2	Noah Hassan	2018
3	4	Nova-7	4	Jupiter	720	False	0.0	Ayesha Rahman	2021
4	5	Luna-Hope	2	Moon	14	True	65.8	Sophia Malik	2012

Next steps: Generate code with data | New interactive sheet

PART A: PYTHON FOUNDATIONS

1. Print a welcome message: **"Welcome to ISMC Mission Control!"**

```
print("Welcome to ISMC Mission Control")
```

```
Welcome to ISMC Mission Control
```

2. Store the number of missions in a variable and print it. **(Hint: count rows in the CSV.)**

```
number_of_mission=len(data)
print("Number of missions:",number_of_mission)
```

```
Number of missions: 300
```

3. Ask the user to input a destination **(e.g., "Moon")** and print a message like: **"Searching missions to Moon..."**

```
destination=input("Enter Destination e.g Moon ")
print(f"Searching Mission to {destination}...")
```

```
Enter Destination e.g Moon Mars
Searching Mission to Mars...
```

4. Calculate the **average duration of all missions** and print it.

```
average_duration_of_all_mission=data["duration_days"].mean()
print("Average Duration of All Mission:",average_duration_of_all_mission)
```

```
Average Duration of All Mission: 265.28333333333336
```

5. Check if the number of missions to Mars is greater than missions to the Moon. Print **True or False.**

```
mars_mission=data[data["destination"]=="Mars"].sum()
moon_mission=data[data["destination"]=="Moon"].sum()
print(mars_mission>moon_mission)
```

```
mission_id       False
mission_name     False
crew_count       False
destination      False
duration_days     True
success          False
fuel_level       False
commander         True
year             False
dtype: bool
```

PART B: DATA STRUCTURES & CONTROL FLOW

6. Create a list of all unique destinations and print it.

```
destination_list=data["destination"].unique().tolist()
print(destination_list)
```

```
['Moon', 'Mars', 'ISS', 'Jupiter', 'Saturn', 'Venus']
```

7. Use slicing to print the first 5 mission names.

```
mission_names = data.loc[:4, "mission_name"]
print(mission_names)
```

```
0      Orion-Alpha
1         Apollo-X
2    Voyager-Pulse
3           Nova-7
4        Luna-Hope
Name: mission_name, dtype: object
```

8. Use a dictionary to store mission name its assigned commander. Print the dictionary for the first 10 missions.

```
mission_commander_dict = dict(zip(data["mission_name"].head(10), data["commander"].head(10)))
print("\n Mission-Commander Mapping (First 10):")
print(mission_commander_dict)
```

```
 Mission-Commander Mapping (First 10):
{'Orion-Alpha': 'Amelia Vega', 'Apollo-X': 'Liam Carter', 'Voyager-Pulse': 'Noah Hassan', 'Nova-7': 'Ayesha Rahman', 'Luna-Hope': 'Sophia Malik', 'Starlight-3': 'Ethan Kim',
```

9. If a mission has no crew, print "Unmanned mission". If there's just one crew member, print "Solo mission". Otherwise, print "Crewed mission". Do this for the first 20 missions.

```
print("\n Crew Classification (First 20 Missions):")
for idx, row in data.head(20).iterrows():
    if row["crew_count"] == 0:
        print(f"{row['mission_name']}: Unmanned mission")
    elif row["crew_count"] == 1:
        print(f"{row['mission_name']}: Solo mission")
    else:
        print(f"{row['mission_name']}: Crewed mission")
```

```
 Crew Classification (First 20 Missions):
Orion-Alpha: Crewed mission
Apollo-X: Crewed mission
Voyager-Pulse: Crewed mission
Nova-7: Crewed mission
Luna-Hope: Crewed mission
Starlight-3: Unmanned mission
Aurora: Crewed mission
Cosmo-9: Crewed mission
Gemini-Prime: Solo mission
Zenith-1: Crewed mission
Atlas: Crewed mission
Odyssey: Crewed mission
Orion-Beta: Crewed mission
Starfire: Crewed mission
Comet-Quest: Crewed mission
Phoenix-2: Crewed mission
Luna-Rise: Crewed mission
Solaris: Crewed mission
Infinity: Crewed mission
Exo-Prime: Solo mission
```

10. Using a loop, count how many missions lasted more than 365 days.

```
count_long_missions = 0

for duration in data["duration_days"]:
    if duration > 365:
        count_long_missions += 1

print(" Missions lasting more than 365 days:", count_long_missions)
```

```
 Missions lasting more than 365 days: 89
```

PART C: CONTROL FLOW & FUNCTIONS

11. Write a function mission_success_rate(destination) that returns the percentage of successful missions for a given destination. Test it for "Moon", "Mars", and "Jupiter".

```
def mission_success_rate(destination):
    subset = data[data["destination"] == destination]
    if len(subset) == 0:
        return 0
    success_rate = (subset["success"] == True).sum() / len(subset) * 100
    return round(success_rate, 2)

print("Moon Success Rate:", mission_success_rate("Moon"), "%")
print("Mars Success Rate:", mission_success_rate("Mars"), "%")
print("Jupiter Success Rate:", mission_success_rate("Jupiter"), "%")
```

```
Moon Success Rate: 100.0 %
Mars Success Rate: 50.88 %
Jupiter Success Rate: 42.86 %
```

12. Write a function longest_mission() that returns the mission name and duration of the longest mission.

```
def longest_mission():
    idx = data["duration_days"].idxmax()
    return data.loc[idx, "mission_name"], data.loc[idx, "duration_days"]
print("Longest Mission:", longest_mission())
```

```
Longest Mission: ('Cosmo-9', np.int64(800))
```

13. Write a function missions_by_year(year) that returns all missions launched in a given year. Test with 2010.

```
def missions_by_year(year):
    return data[data["year"] == year]["mission_name"]
print("Missions in 2010: \n", missions_by_year(2010))
```

```
Missions in 2010:
 1          Apollo-X
21         Apollo-Z
52       Orion-Zeta
74      Comet-Blaze
96       Luna-Shine
117       Solaris-6
132        Orion-Mu
147        Cosmo-16
166        Aurora-I
189       Zenith-10
208    Gemini-Crest
219    Exo-Frontier
239     Exo-Pioneer
261       Apollo-A9
275      Phoenix-15
289       Zenith-15
Name: mission_name, dtype: object
```

14. Write a function fuel_check(fuel_level) that returns "Critical" if fuel level is less than 20, "Moderate" if fuel level is equal to 20 or between 20 and 70, and "Safe" if fuel level is greater than or equal to 70. Apply this to the last 50 missions.

```
def fuel_check(fuel_level):
    if fuel_level < 20:
        return "Critical"
    elif 20 <= fuel_level < 70:
        return "Moderate"
    else:
        return "Safe"

print("Fuel Status (Last 50 Missions):")
for idx, row in data.tail(50).iterrows():
    print(f"{row['mission_name']}: {fuel_check(row['fuel_level'])}")
```

```
Fuel Status (Last 50 Missions):
Atlas-XIII: Critical
Odyssey-13: Safe
Orion-Zeta2: Safe
Starfire-13: Safe
Comet-Dream: Critical
Phoenix-14: Safe
Luna-Peak2: Safe
Solaris-13: Critical
Infinity-13: Safe
Exo-Sky: Safe
Orion-Tau2: Safe
Apollo-A9: Critical
Voyager-Rise: Safe
Nova-20: Safe
Luna-Surge2: Critical
Starlight-16: Safe
Aurora-N: Safe
Cosmo-22: Moderate
Gemini-Quest: Safe
Zenith-14: Safe
Atlas-XIV: Critical
Odyssey-14: Moderate
Orion-Delta2: Safe
Starfire-14: Safe
Comet-Edge: Safe
Phoenix-15: Critical
Luna-Glide: Safe
Solaris-14: Safe
Infinity-14: Safe
Exo-Dawn: Critical
Orion-Phi3: Safe
Apollo-A10: Safe
Voyager-Lite: Critical
Nova-21: Safe
Luna-Rise2: Safe
Starlight-17: Safe
Aurora-O: Critical
Cosmo-23: Safe
Gemini-Orbit: Safe
Zenith-15: Critical
Atlas-XV: Moderate
Odyssey-15: Safe
Orion-Theta2: Safe
Starfire-15: Safe
Comet-Light: Critical
Phoenix-16: Safe
Luna-Crest2: Safe
Solaris-15: Critical
Infinity-15: Safe
Exo-Solar: Moderate
```

15. Write a nested loop to count the number of missions per commander for each destination. Print results like: Moon: Amelia Vega: 1 mission Zara Sheikh: 1 mission Mars: Liam Carter: 2 missions ...

```
print("Missions per Commander by Destination:")
destinations = data["destination"].unique()

for dest in destinations:
```

```
    print(f"\n{dest}:")
    subset = data[data["destination"] == dest]
    commanders = subset["commander"].unique()

    for cmd in commanders:
        count = (subset["commander"] == cmd).sum()
        print(f"  {cmd}: {count} mission(s)")
```

```
Missions per Commander by Destination:

Moon:
  Amelia Vega: 1 mission(s)
  Sophia Malik: 1 mission(s)
  Isabella Torres: 1 mission(s)
  Arjun Patel: 2 mission(s)
  Emily Davis: 1 mission(s)
  Charlotte Evans: 1 mission(s)
  William Chen: 1 mission(s)
  Isaac Flores: 1 mission(s)
  Aaliyah Shah: 1 mission(s)
  Samuel White: 1 mission(s)
  Lucas Martin: 1 mission(s)
  Mohammed Ali: 1 mission(s)
  Maria Rossi: 1 mission(s)
  David Lopez: 1 mission(s)
  Lara Ahmed: 2 mission(s)
  Jack Taylor: 1 mission(s)
  Isabella Khan: 1 mission(s)
  James Evans: 1 mission(s)
  Ella Turner: 1 mission(s)
  Ethan Scott: 1 mission(s)
  Leila Khan: 1 mission(s)
  Liam Clark: 1 mission(s)
  Sofia Rahman: 1 mission(s)
  Laila Ahmed: 1 mission(s)
  Oliver Lewis: 1 mission(s)
  Leah Ali: 1 mission(s)
  Hannah Davis: 1 mission(s)
  Sara Ali: 1 mission(s)
  Oliver Brown: 1 mission(s)
  Sara Torres: 1 mission(s)
  Maya Clarke: 1 mission(s)
  Sofia Patel: 1 mission(s)
  Zoe Martin: 1 mission(s)
  Emily Scott: 1 mission(s)
  Aisha Khan: 1 mission(s)
  Isabella Smith: 1 mission(s)
  Huda Johnson: 1 mission(s)
  Ethan White: 1 mission(s)
  Liam Wilson: 1 mission(s)
  Amira Rahman: 1 mission(s)
  Hannah Brown: 1 mission(s)
  Aisha White: 1 mission(s)
  James Davis: 1 mission(s)
  Lara Torres: 1 mission(s)
  Noah Brown: 1 mission(s)
  Jacob Ali: 1 mission(s)
  Adam Carter: 1 mission(s)
  David Smith: 1 mission(s)
  Ethan Torres: 1 mission(s)
  Emma White: 1 mission(s)
  Zara Torres: 1 mission(s)
  Noah Scott: 2 mission(s)
  Leila Ali: 1 mission(s)
  Laila Khan: 1 mission(s)
  Hassan Ali: 1 mission(s)
```

PART D: EXCEPTION HANDLING & ROBUST CODE

16. Write a try-except block to safely convert user input into an integer for selecting a mission ID. If invalid, print "Invalid input. Please enter a number."

```
try:
    mission_id_input = int(input("Enter a Mission ID: "))
except ValueError:
    print("Invalid input. Please enter a number.")
```

```
Enter a Mission ID: a
Invalid input. Please enter a number.
```

17. Paste your code for Q13 (missions_by_year) and modify it to handle the case when no missions exist for a year. Return "No missions found."

```
def missions_by_year(year):
    missions = data[data["year"] == year]["mission_name"]
    if len(missions) == 0:
        return "No missions found."
    return missions

print("Missions in 1800:", missions_by_year(1800))
```

```
Missions in 1800: No missions found.
```

18. Write a function get_mission(mission_id) that returns mission details if ID exists or raises a ValueError if mission_id is not found. Handle it with try-except.

```
def get_mission(mission_id):
    if mission_id in data["mission_id"].values:
        return data[data["mission_id"] == mission_id].to_dict("records")[0]
    else:
        raise ValueError("Mission ID not found.")
try:
    mission_details = get_mission(5)
    print("Mission Details:", mission_details)
except ValueError as e:
    print(e)
```

```
Mission Details: {'mission_id': 5, 'mission_name': 'Luna-Hope', 'crew_count': 2, 'destination': 'Moon', 'duration_days': 14, 'success': True, 'fuel_level': 65.8, 'commander'
```

19. Ask user for a mission ID, and if input is not a number, print "Please enter digits only." If the mission itself does not exist at all, print "Mission not found."

```
user_input = input("\nEnter Mission ID: ")
try:
    mission_id = int(user_input)
    if mission_id in data["mission_id"].values:
        details = data[data["mission_id"] == mission_id].to_dict("records")[0]
        print("Mission Found:", details)
    else:
        print("Mission not found.")
except ValueError:
    print("Please enter digits only.")
```

```
Enter Mission ID: b
Please enter digits only.
```

20. Paste your code for Q19 and use finally to print "Mission lookup complete." after it is executed, no matter what happens.

```
user_input = input("\nEnter Mission ID: ")
try:
    mission_id = int(user_input)
    if mission_id in data["mission_id"].values:
        details = data[data["mission_id"] == mission_id].to_dict("records")[0]
        print("Mission Found:", details)
    else:
        print("Mission not found.")
except ValueError:
    print("Please enter digits only.")
finally:
    print("Mission lookup complete.")
```

```
Enter Mission ID: 1900
Mission not found.
Mission lookup complete.
```