QUICK REFERENCE CARD #1: PYTHON ESSENTIALS

VARIABLES — Storing Data	
Type	**Example**
Integer (whole number)	age = 25
Float (decimal)	price = 19.99
String (text)	name = 'Ali'
Boolean (True/False)	passed = True

LISTS — Multiple Values in One Container	
Operation	**Code Example**
Create list	`scores = [78, 55, 82, 90]`
First item	`scores[0]  # 78`
Last item	`scores[-1]  # 90`
Count items	`len(scores)  # 4`
Add item	`scores.append(95)`
Remove item	`scores.remove(55)`

LOOPS — Automating Repetition

```
for score in scores:
    print(score)
```

```
# Filter: Keep only passing scores
passed = [s for s in scores if s >= 60]
```

```
# Transform: Add 5 to every score
adjusted = [s + 5 for s in scores]
```

FUNCTIONS — Reusable Code Blocks

```
def calculate_average(scores):
    return sum(scores) / len(scores)

avg = calculate_average([78, 82, 90])
print(avg)  # 83.33
```

```
def classify_grade(score):
    if score >= 90: return 'A'
    elif score >= 80: return 'B'
    elif score >= 70: return 'C'
    else: return 'F'

print(classify_grade(85))  # B
```

QUICK REFERENCE CARD #2: PANDAS DATA WRANGLING

SETUP — Import Libraries

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

LOADING DATA

Method	Code
Built-in dataset	`df = sns.load_dataset('tips')`
CSV file	`df = pd.read_csv('data.csv')`
Excel file	`df = pd.read_excel('data.xlsx')`
From dictionary	`df = pd.DataFrame({'col': [1,2,3]})`

THE ESSENTIAL 5 COMMANDS — Always Run These First!

Command	What It Does
`df.head()`	First 5 rows
`df.shape`	Number of (rows, columns)
`df.info()`	Column names, types, missing count
`df.describe()`	Mean, std, min, max, quartiles
`df.isnull().sum()`	Count missing values per column

SELECTING & FILTERING

Task	Code
One column	`df['age']`
Multiple columns	`df[['name', 'age', 'score']]`
Filter rows	`df[df['age'] > 25]`
Multiple conditions (AND)	`df[(df['age'] > 25) & (df['score'] >= 60)]`
Multiple conditions (OR)	`df[(df['age'] < 18) | (df['age'] > 65)]`
Create new column	`df['total'] = df['price'] * df['qty']`

GROUPING & AGGREGATING — The Most Powerful Operation!

Task	Code
Group and calculate mean	`df.groupby('class')['score'].mean()`
Multiple statistics	`df.groupby('class')['score'].agg(['mean', 'std', 'count'])`
Group by two columns	`df.groupby(['class', 'gender'])['score'].mean()`

QUICK REFERENCE CARD #3: VISUALIZATION & STATISTICS

SEABORN VISUALIZATIONS — 1-2 Lines Each!	
Chart Type	**Code**
Histogram	`sns.histplot(data=df, x='age', bins=30, kde=True)`
Box plot	`sns.boxplot(data=df, x='class', y='fare')`
Bar chart (counts)	`sns.countplot(data=df, x='category')`
Scatter plot	`sns.scatterplot(data=df, x='age', y='fare')`
Heatmap	`sns.heatmap(df.corr(), annot=True, cmap='coolwarm')`
Add title & show	`plt.title('My Chart')` `plt.show()`

DESCRIPTIVE STATISTICS		
Measure	**Code**	**What It Tells You**
Mean	`df['age'].mean()`	Average value
Median	`df['age'].median()`	Middle value (50th percentile)
Mode	`df['category'].mode()[0]`	Most common value
Std Deviation	`df['age'].std()`	Typical variation from mean
Variance	`df['age'].var()`	Squared std deviation
Min / Max	`df['age'].min(), df['age'].max()`	Smallest and largest values
Range	`df['age'].max() - df['age'].min()`	Spread (max - min)
Quartiles	`df['age'].quantile([0.25, 0.5, 0.75])`	25th, 50th, 75th percentiles
Skewness	`df['age'].skew()`	Shape: >0 right-skewed, <0 left-skewed

QUICK TIPS

- Mean vs Median: If skewed (mean ≠ median), use median as 'typical' value
- Std Dev > Mean: Indicates high variability and likely outliers
- Skewness > 1 or < -1: Distribution is highly skewed
- Always remove missing values first: df['col'].dropna()

QUICK REFERENCE CARD #4: HYPOTHESIS TESTING

WHICH TEST SHOULD I USE? — Decision Tree

Question Type	Data Condition	Test to Use
Compare 2 groups (numeric)	Normal distribution	Independent t-test
Compare 2 groups (numeric)	Skewed / outliers	Mann-Whitney U test
Compare 2 categories	Any	Chi-square test
Test a proportion	Binary outcome	Binomial test
Compare 3+ groups	Normal distribution	ANOVA (not covered)

TEST CODES — Copy-Paste Ready

Independent t-test:

```
from scipy import stats
t_stat, p_value = stats.ttest_ind(group1, group2)
print(f'p-value: {p_value:.4f}')
```

Mann-Whitney U test:

```
from scipy import stats
u_stat, p_value = stats.mannwhitneyu(group1, group2)
print(f'p-value: {p_value:.4f}')
```

Chi-square test:

```
from scipy import stats
ct = pd.crosstab(df['cat1'], df['cat2'])
chi2, p_value, dof, exp = stats.chi2_contingency(ct)
print(f'p-value: {p_value:.4f}')
```

Binomial test:

```
from scipy.stats import binomtest
result = binomtest(successes, trials, 0.5)
print(f'p-value: {result.pvalue:.4f}')
```

INTERPRETING p-VALUES

p-value	Interpretation
p < 0.001	Extremely significant — very strong evidence
p < 0.05	Significant — reject null hypothesis
p ≥ 0.05	Not significant — fail to reject null hypothesis

THE 5-STEP HYPOTHESIS TESTING PROCESS

1. State hypotheses: H_0 (no difference) vs H_1 (there is a difference)
2. Choose significance level: Usually $\alpha = 0.05$
3. Calculate test statistic: Run the appropriate test
4. Find p-value: Probability this happened by chance
5. Make decision: If $p < 0.05 \rightarrow$ Reject H_0 (significant difference exists)

⚠ COMMON MISTAKES TO AVOID

- Saying 'accept H_0' — Correct: 'fail to reject H_0'
- Confusing significance with importance — $p < 0.05$ doesn't mean the effect is large
- Forgetting to check assumptions — Use Mann-Whitney if data is skewed
- Using 'and' instead of & in Pandas conditions — Correct: (cond1) & (cond2)