CHEATSHEET COMPILATION

Generative AI for Work & Research Productivity

Course: AIW | The GRAPH Courses

Compiled: February 13, 2026

A comprehensive collection of quick-reference guides for tools and technologies used in the Generative AI for Work & Research Productivity course.

TABLE OF CONTENTS

1. CHATGPT & LLM PROMPTING BEST PRACTICES

The AUTOMAT Framework

Component	Description
Assign Role	Give the AI a specific role or persona to guide responses
Audience	Specify who the response is for (experts, beginners, etc.)
Task Goal	Clearly state the main objective or desired output
Information	Provide necessary context, background, and constraints
Communication Style	Specify tone (formal, casual, technical, etc.)
Edge Cases	Tell the model how to handle special scenarios
Topics	Highlight key areas to focus on or avoid

Key Prompting Techniques

Zero-Shot Prompting: Ask the model directly without examples
Few-Shot Prompting: Provide 1-3 examples before the actual task
Chain-of-Thought: Ask the model to 'think step by step'
Role-Playing: Assign the AI a persona ('You are an expert...)'
Retrieval Augmented Generation (RAG): Provide documents for the AI to reference
Temperature Settings: Lower (0.2-0.5) = more predictable; Higher (0.8-1.0) = more creative
Token Limits: Set max_tokens to control response length

Prompt Engineering Tips

✓ Be specific and detailed in your instructions
✓ Provide context about the task and audience
✓ Show examples of desired output format
✓ Break complex tasks into smaller steps
✓ Use clear, structured instructions
✓ Avoid ambiguous language or assumptions
✓ Specify what to do with edge cases or errors
✓ Iterate: refine prompts based on results

2. PYTHON FOR DATA ANALYSIS

Pandas Basics

```
import pandas as pd

# Read data
df = pd.read_csv('file.csv')
df = pd.read_excel('file.xlsx')

# Basic info
df.head()              # First 5 rows
df.info()              # Column info
df.describe()          # Summary statistics
df.shape               # (rows, columns)
df.columns             # Column names

# Selecting data
df['column_name']      # Single column
df[['col1', 'col2']]   # Multiple columns
df.loc[0]              # By row label
df.iloc[0]             # By row position
df[df['col'] > 5]      # Boolean indexing

# Data manipulation
df.sort_values('col')  # Sort
df.drop('col', axis=1) # Drop column
df.rename({'old':'new'})  # Rename
df.fillna(0)           # Fill missing values
df.dropna()            # Remove NaN

# Aggregation
df.groupby('col').sum()  # Group and sum
df.groupby('col').mean() # Group and average
df['col'].value_counts()  # Count occurrences
```

NumPy Basics

```
import numpy as np

# Creating arrays
arr = np.array([1, 2, 3])
arr = np.zeros((3, 3))  # 3x3 zeros matrix
arr = np.ones((2, 4))   # 2x4 ones matrix
arr = np.arange(0, 10, 2)  # [0, 2, 4, 6, 8]
arr = np.linspace(0, 1, 5)  # 5 evenly spaced

# Array operations
arr.shape              # Dimensions
arr.dtype              # Data type
arr.reshape(2, 5)      # Reshape array
arr.flatten()          # Convert to 1D
arr.sum(), arr.mean()  # Aggregate
arr + 10               # Arithmetic (broadcasting)

# Indexing & slicing
arr[0]                 # First element
```

```
arr[1:3]                # Elements 1-2
arr[arr > 5]            # Boolean mask
arr[:, 0]               # First column (2D)

# Math functions
np.sqrt(arr)            # Square root
np.exp(arr)             # Exponential
np.log(arr)             # Natural log
np.std(arr)             # Standard deviation
```

3. R FOR DATA ANALYSIS

dplyr - Data Manipulation

```
library(dplyr)

# Pipe operator: %>% (or |> in R 4.1+)
df %>%
  filter(col > 5) %>%
  select(col1, col2) %>%
  arrange(col1)

# Main dplyr verbs
filter()       # Keep rows matching conditions
select()       # Choose columns
mutate()       # Create/modify columns
arrange()      # Sort rows
summarize()    # Create summary statistics
group_by()     # Group for operations
join()         # Merge data frames

# Examples
df %>%
  filter(age > 18) %>%
  select(name, age)

df %>%
  group_by(category) %>%
  summarize(avg = mean(value),
            total = sum(value))

df %>%
  mutate(new_col = old_col * 2) %>%
  arrange(desc(new_col))
```

ggplot2 - Data Visualization

```
library(ggplot2)

# Basic structure
ggplot(data = df, aes(x = col1, y = col2)) +
  geom_point() +
  labs(title = "Title", x = "X axis", y = "Y axis")

# Common geoms
geom_point()     # Scatter plot
geom_line()      # Line plot
geom_bar()       # Bar chart
geom_histogram() # Histogram
geom_boxplot()   # Box plot
geom_smooth()    # Trend line

# Customization
+ theme_minimal()      # Minimal theme
+ theme_classic()      # Classic theme
+ scale_color_manual() # Custom colors
+ facet_wrap(~category)  # Subplots
+ coord_flip()         # Flip axes

# Example
ggplot(data = df, aes(x = category, y = value, fill = category)) +
  geom_boxplot() +
  theme_minimal() +
  labs(title = "Distribution by Category")
```

4. GIT & GITHUB VERSION CONTROL

Essential Git Commands

```
# Setup
git config --global user.name "Your Name"
git config --global user.email "email@example.com"

# Initialize & clone
git init                    # Initialize local repo
git clone <url>             # Clone remote repo

# Status & viewing
git status                  # Show changed files
git log                     # Show commit history
git diff                    # Show changes (unstaged)
git diff --staged           # Show staged changes

# Stage & commit
git add <file>              # Stage specific file
git add .                   # Stage all changes
git commit -m "message"     # Commit staged changes

# Branches
git branch                  # List branches
git branch <name>           # Create branch
git switch <branch>         # Switch to branch (or: git checkout)
git merge <branch>          # Merge branch into current

# Remote
git remote add origin <url> # Add remote
git push -u origin main     # Push to remote
git pull                    # Fetch & merge from remote
git fetch                   # Fetch without merging

# Undo
git restore <file>          # Discard changes
git reset HEAD <file>       # Unstage file
git revert <commit>         # Undo commit (safe)
```

GitHub Workflow

1. Fork the repository (if contributing to open source)
2. Clone: **git clone**
3. Create branch: **git checkout -b feature-name**
4. Make changes and commit: **git add . && git commit -m "message"**
5. Push: **git push origin feature-name**
6. Create Pull Request on GitHub
7. Review and merge
8. Delete branch: **git branch -d feature-name**

5. MARKDOWN QUICK REFERENCE

Syntax	Result
# Heading 1	Large heading
## Heading 2	Medium heading
### Heading 3	Small heading
bold or __bold__	Bold text
italic or _italic_	Italic text
~~strikethrough~~	Crossed-out text
`code`	Inline code
```\ncode block\n```	Code block
[text](url)	Hyperlink
![alt](image.png)	Image
- item 1\n- item 2	Bulleted list
1. item 1\n2. item 2	Numbered list
> quote	Blockquote
---	Horizontal line
| A | B |\n|---|---|\n| 1 | 2 |	Table

Common Markdown Patterns

~~~
# Title
## Subtitle

**Bold** and *italic* text with `inline code`.

- Bullet point 1
- Bullet point 2
  - Nested point

1. First step
2. Second step

[Link text](https://example.com)

> Important quote

```python
# Code block with syntax highlighting
print("Hello World")
```

| Header 1 | Header 2 |
|----------|----------|
| Cell 1   | Cell 2   |
~~~

6. HTML & CSS BASICS

HTML Structure

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width">
  <title>Page Title</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <header>
    <h1>Main Heading</h1>
    <nav>Navigation</nav>
  </header>

  <main>
    <article>
      <h2>Article Title</h2>
      <p>Paragraph text</p>
    </article>
  </main>

  <footer>Footer content</footer>
  <script src="script.js"></script>
</body>
</html>

<!-- Common elements -->
<a href="url">Link</a>
<img src="image.png" alt="description">
<button>Click me</button>
<input type="text" placeholder="Enter text">
<form> <input> <textarea> <select> </form>
```

CSS Fundamentals

```
/* Selectors */
p { }                     /* Element */
.classname { }            /* Class */
#idname { }               /* ID */
h1, h2 { }                /* Multiple */
p.highlight { }           /* Combined */

/* Text styling */
font-size: 16px;
font-weight: bold;
font-style: italic;
color: #333333;
text-align: center;
line-height: 1.5;

/* Box model */
margin: 10px;             /* Outside space */
padding: 10px;            /* Inside space */
```

```
border: 1px solid black; /* Border */
width: 100%;
height: 50px;

/* Colors & backgrounds */
background-color: #f5f5f5;
background-image: url('image.png');

/* Positioning & layout */
display: flex;           /* Flexbox */
display: grid;           /* CSS Grid */
position: relative;      /* static, relative, absolute, fixed */
float: left;             /* Old method */

/* Common properties */
opacity: 0.5;
transform: rotate(45deg) scale(1.5);
box-shadow: 0 4px 8px rgba(0,0,0,0.2);
border-radius: 8px;
```