Scripting & Automation Cheatsheet

A comprehensive cheat sheet covering essential scripting and automation concepts, tools, and techniques.

Scripting Fundamentals

Basic Concepts

Scripting: Writing a sequence of commands to automate tasks.
Automation: Using scripts and tools to perform tasks automatically, reducing manual intervention.
Key Benefits: Increased efficiency, reduced errors, and improved consistency.

Shebang: `#!` - Specifies the interpreter for the script (e.g., `#!/bin/bash` for Bash, `#!/usr/bin/env python3` for Python).
Variables: Used to store and manipulate data.
Control Flow: Statements like `if`, `else`, `for`, and `while` to control the execution flow.

Functions: Reusable blocks of code to perform specific tasks.
Comments: `#` (Bash, Python, PowerShell) - Used to add explanatory notes to the code.

Scripting Languages

Bash	Primarily used for Unix-like operating systems. Great for system administration tasks.
Python	Versatile language suitable for web development, data analysis, and general-purpose scripting.
PowerShell	Designed for Windows system administration and automation. Includes powerful cmdlets.

Input/Output

Standard Input (stdin): Input from the keyboard or redirected from a file.
Standard Output (stdout): Output displayed on the screen or redirected to a file.
Standard Error (stderr): Error messages displayed on the screen or redirected to a file.

Redirection:

- `>` Redirect stdout to a file (overwrites).
- `>>` Redirect stdout to a file (appends).
- `2>` Redirect stderr to a file.
- `&>` or `2>&1` Redirect both stdout and stderr to a file.

Piping:

- `|` Connect the stdout of one command to the stdin of another.

Example: `ls -l | grep 'myfile.txt'`

Bash Scripting

Variables and Operators

Variable Assignment:	`variable_name="value"` **Example:** `name="John"`
Accessing Variables:	`$variable_name` or `${variable_name}` **Example:** `echo "Hello, $name!"`
Arithmetic Operators:	`+`, `-`, `*`, `/`, `%` **Example:** `result=$((5 + 3))` `echo $result`
String Operators:	`=` (equal), `!=` (not equal), `-z` (empty), `-n` (not empty) **Example:** `if [ "$name" = "John" ]; then echo "Match!"; fi`

Control Structures

If Statement:

```
if [ condition ]; then
  # Code to execute if condition is true
elif [ condition2 ]; then
  # Code to execute if condition2 is true
else
  # Code to execute if all conditions are false
fi
```

For Loop:

```
for variable in list;
do
  # Code to execute for each item in the list
done
```

While Loop:

```
while [ condition ]; do
  # Code to execute while the condition is true
done
```

Functions

Function Definition:

```
function_name() {
  # Code to execute
  return value
}
```

Calling a Function: `function_name`

Example:

```
greet() {
  echo "Hello, $1!"
}

greet "World"
```

Python Scripting

Basic Syntax

Variables: `variable_name = value`

Example:

```
name = "Alice"
```

Data Types: `int`, `float`, `str`, `bool`, `list`, `tuple`, `dict`

Example:

```
age = 30
price = 99.99
```

Operators: `+`, `-`, `*`, `/`, `%`, `**` (exponentiation), `//` (floor division)

Example:

```
result = 5 + 3
```

Comments: `# This is a comment`

Control Flow

If Statement:

```
if condition:
  # Code to execute if condition is true
elif condition2:
  # Code to execute if condition2 is true
else:
  # Code to execute if all conditions are false
```

For Loop:

```
for variable in iterable:
  # Code to execute for each item in the iterable
```

While Loop:

```
while condition:
  # Code to execute while the condition is true
```

Functions

Function Definition:

```
def function_name(parameters):
  # Code to execute
  return value
```

Calling a Function: `function_name(arguments)`

Example:

```
def greet(name):
  print(f"Hello, {name}!")

greet("World")
```

Modules

Importing Modules:

`import module_name`

`from module_name import specific_item`

`import module_name as alias`

Example:

```
import math
print(math.sqrt(16))

from datetime import datetime
print(datetime.now())
```

PowerShell Scripting

Basic Concepts

Cmdlets: Commands in PowerShell (e.g., `Get-Process`, `Write-Host`).

Variables: Start with a `$` (e.g., `$name = "John"`).

Piping: Use `|` to pass objects between cmdlets (e.g., `Get-Process | Where-Object {$_.CPU -gt 10}`).

Variables and Data Types

Variable Assignment: `$variable_name = value`

Example:

```
$name = "John"
```

Data Types: `[int]`, `[string]`, `[bool]`, `[array]`, `[hashtable]`

Example:

```
[int]$age = 30
```

Arrays:

```
$myArray = @("item1", "item2", "item3")
```

Hashtables:

```
$myHash = @{Name="John"; Age=30}
```

Control Structures

If Statement:

```
if (condition) {
  # Code to execute if condition is true
} elseif (condition2) {
  # Code to execute if condition2 is true
} else {
  # Code to execute if all conditions are false
}
```

For Loop:

```
foreach ($item in $collection) {
  # Code to execute for each item in the collection
}
```

While Loop:

```
while (condition) {
  # Code to execute while the condition is true
}
```

Functions

Function Definition:

```
function function_name {
  param (
    $parameter1,
    $parameter2
  )
  # Code to execute
  return value
}
```

Calling a Function: `function_name -parameter1 value1 -parameter2 value2`

Example:

```
function Greet {
  param (
    $Name
  )
  Write-Host "Hello, $Name!"
}

Greet -Name "World"
```