Use Case of Operators

| ****Use Case**** | ****Operator/Variable**** | ****Brackets**** | ****Description**** | ****Example**** |
| --- | --- | --- | --- | --- |
| **Access all script arguments** | $@ | N/A | Access all arguments as separate entities | for arg in "$@"; do echo "$arg"; done |
| **Access all arguments as a string** | $\* | N/A | Access all arguments as a single string | echo "$\*" |
| **Count number of arguments** | $# | N/A | Count the number of arguments | echo "Number of arguments: $#" |
| **Exit status of last command** | $? | N/A | Get the exit status of the last command | echo "Exit status: $?" |
| **Script name** | $0 | N/A | Get the name of the script | echo "Script name: $0" |
| **Current process ID** | $$ | N/A | Get the current script's process ID | echo "Process ID: $$" |
| **Last background command PID** | $! | N/A | Get the PID of the last background command | sleep 10 & echo "PID: $!" |
| **Last argument of previous command** | $\_ | N/A | Get the last argument passed to the script | echo "Last argument: $\_" |
| **Simple test condition** | N/A | [ ] | Use for basic tests (file checks, string comparisons) | if [ -f "file.txt" ]; then ... fi |
| **Advanced test condition** | N/A | [[ ]] | Use for complex tests (pattern matching, logical ops) | if [[ "$var" == "value" ]]; then ... fi |
| **Subshell execution** | N/A | ( ) | Create a subshell for command execution | (cd /tmp && ls) |
| **Arithmetic operations** | N/A | (( )) | Use for performing arithmetic calculations | (( result = 5 + 3 )) |
| **Variable expansion** | N/A | { } | Use for variable/parameter expansion | echo "Name: ${name}" |
| **Arrays** | N/A | { } | Use for defining and accessing array elements | arr=(1 2 3); echo "${arr[1]}" |

### Notes:

* **Single brackets** [ ]: Good for simple tests and conditions. Spaces are required around the brackets.
* **Double brackets** [[ ]]: More powerful, allowing for advanced string comparisons and pattern matching. More forgiving with spaces and no need for escaping certain characters.
* **Parentheses** ( ): Used for grouping commands and creating subshells; changes made within do not affect the parent shell.
* **Double parentheses** (( )): Used specifically for arithmetic expressions, allowing you to perform calculations directly.
* **Curly braces** { }: Used for variable expansion, and defining arrays.