Think OS SDK - Complete Beginner's Guide

Software Development Kit for Mod Creation

Version 1.0

Table of Contents

---

1. Introduction

Welcome to the Think OS SDK (Software Development Kit). This guide will teach you how to use the SDK to create powerful mods quickly and easily.

The SDK provides pre-built functions that handle common tasks like: - Displaying formatted screens - Getting user input with validation - Creating menus - Saving and loading data - And much more

This guide assumes you have basic Python knowledge and have read the "Mod Installation Guide" first.

---

2. What is the SDK?

Without SDK (Hard Way)

```python
def run_mod(os_instance):
    os_instance.clear_screen()
    print("=" * 50)
    print("        MY MOD")
    print("=" * 50)
    print()

    while True:
        try:
            age = int(input("Enter age: "))
            if age < 1 or age > 120:
                print("Invalid age!")
                continue
            break
        except ValueError:
            print("Please enter a number!")

    print(f"You are {age} years old")
```

With SDK (Easy Way)

```python
from SDK import ModSDK

def run_mod(os_instance):
    sdk = ModSDK(os_instance)

    sdk.screen.header("MY MOD")
    age = sdk.input.get_int("Enter age: ", min_val=1, max_val=120)
    print(f"You are {age} years old")
```

The SDK does the hard work for you!

---

3. Installing the SDK

Step 1: Obtain the SDK File

The SDK is a single Python file named `SDK.py`

Step 2: Place the SDK File

The SDK must be placed in the SAME folder as Think-OS1.0.py:

```
Main Folder/
├── Think-OS1.0.py       (Think OS)
├── SDK.py               (SDK - place here!)
└── think_os_mods/       (Your mods)
    └── my_mod.py
```

IMPORTANT: Do NOT place SDK.py inside the think_os_mods folder!

Step 3: Verify Installation

Create a test mod to verify the SDK works:

```python
# File: think_os_mods/sdk_test.py

from SDK import ModSDK

def run_mod(os_instance):
    sdk = ModSDK(os_instance)
    sdk.screen.header("SDK TEST")
    print("SDK is working correctly!")
    sdk.input.wait()
```

If this mod runs without errors, the SDK is installed correctly.

---

4. SDK Components Overview

The SDK provides five main components:

ModSDK (Main Class)

Initializes and manages all SDK components

Screen

Display utilities for formatting output

Input

User input with automatic validation

Menu

Easy menu creation

Storage

Save and load data

Utils

General utility functions

---

5. Your First SDK Mod

Let's create a simple greeting mod step-by-step.

Step 1: Import the SDK

Every SDK mod starts with this line:

```
from SDK import ModSDK
```

Step 2: Create the run_mod Function

```
def run_mod(os_instance):
    # Your code goes here
    pass
```

Step 3: Initialize the SDK

```
def run_mod(os_instance):
    sdk = ModSDK(os_instance)
```

Now you can use all SDK features through the `sdk` variable.

Step 4: Add Your Code

```python
from SDK import ModSDK

def run_mod(os_instance):
    # Initialize SDK
    sdk = ModSDK(os_instance)

    # Display a header
    sdk.screen.header("GREETING MOD")

    # Get user's name
    name = sdk.input.get_string("What is your name? ")

    # Display greeting
    print(f"Hello, {name}!")
    print()

    # Wait before returning
    sdk.input.wait()
```

Step 5: Save and Test

1. Save the file as `greeting.py` in think_os_mods/
2. Run Think OS
3. Install the mod
4. Run it!

---

6. Screen Utilities

The Screen component provides display formatting functions.

6.1 Clear Screen

```
sdk.screen.clear()
```

Clears the terminal screen.

6.2 Display Header

```
sdk.screen.header("MY MOD")
```

Output:

```
==================================================
        MY MOD
==================================================
```

Custom width:

```
sdk.screen.header("MY MOD", width=60)
```

6.3 Display Box

```
sdk.screen.box("This is a message")
```

Output:

```
+------------------------------------------------+
| This is a message                              |
+------------------------------------------------+
```

Multi-line text:

```
text = "Line 1\nLine 2\nLine 3"
sdk.screen.box(text)
```

6.4 Display Divider

```
sdk.screen.divider()
```

Output:

```
================================================
```

Custom character:

```
sdk.screen.divider(width=40, char="-")
```

Output:

```
----------------------------------------
```

6.5 Centered Text

```
sdk.screen.centered("Important Message")
```

Centers text within the specified width.

6.6 Progress Bar

```
sdk.screen.progress_bar(current=50, total=100, width=30, label="Progress")
```

Output:

```
Progress: [███████████████░░░░░░░░░░░░░░░] 50%
```

Example in a loop:

```
for i in range(0, 101, 10):
    sdk.screen.clear()
    sdk.screen.header("LOADING")
    sdk.screen.progress_bar(i, 100)
    sdk.utils.sleep(0.5)
```

---

7. Input Utilities

The Input component provides validated user input.

7.1 Get String Input

```
name = sdk.input.get_string("Enter your name: ")
```

Allow empty input:

```
name = sdk.input.get_string("Optional field: ", allow_empty=True)
```

7.2 Get Integer Input

```
age = sdk.input.get_int("Enter your age: ")
```

With validation:

```
age = sdk.input.get_int("Enter your age: ", min_val=1, max_val=120)
```

The SDK automatically: - Checks if input is a number - Validates min/max values - Shows error messages - Asks again if invalid

7.3 Get Choice Input

```
choice = sdk.input.get_choice("Enter choice: ", valid_choices=['a', 'b', '
```

Only accepts specified choices. Keeps asking until valid input is received.

7.4 Yes/No Input

```
confirm = sdk.input.yes_no("Continue? (y/n): ")
```

Returns: - `True` for 'y' or 'yes' - `False` for 'n' or 'no'

Example usage:

```
if sdk.input.yes_no("Delete file? (y/n): "):
    print("File deleted!")
else:
    print("Cancelled.")
```

7.5 Wait for Enter

```
sdk.input.wait()
```

Displays "Press Enter to continue…" and waits.

Custom message:

```
sdk.input.wait("Press Enter to start...")
```

---

8. Menu Utilities

The Menu component creates interactive menus automatically.

8.1 Basic Menu

```
choice = sdk.menu.create("Main Menu", [
    "Option 1",
    "Option 2",
    "Option 3"
])
```

Displays:

```
==================================================
        MAIN MENU
==================================================

1) Option 1
2) Option 2
3) Option 3
4) Exit

Select option:
```

Return values: - `0` = First option selected - `1` = Second option selected - `2` = Third option selected - `-1` = Exit selected

8.2 Menu Without Exit Option

```
choice = sdk.menu.create("Menu", ["Option 1", "Option 2"], show_exit=False
```

8.3 Complete Menu Example

```
from SDK import ModSDK

def run_mod(os_instance):
    sdk = ModSDK(os_instance)

    while True:
        sdk.screen.header("TODO APP")

        choice = sdk.menu.create("What do you want to do?", [
            "Add task",
            "View tasks",
            "Delete task"
        ])

        if choice == -1:  # Exit selected
            break
        elif choice == 0:  # Add task
            task = sdk.input.get_string("Enter task: ")
            print(f"Added: {task}")
            sdk.input.wait()
        elif choice == 1:  # View tasks
            print("Your tasks...")
            sdk.input.wait()
        elif choice == 2:  # Delete task
            print("Task deleted")
            sdk.input.wait()
```

8.4 List Selection

```
items = ["Apple", "Banana", "Orange"]
selected = sdk.menu.list_select("Choose fruit", items)

if selected:
    print(f"You chose: {selected}")
else:
    print("Cancelled")
```

---

9. Storage Utilities

The Storage component handles data persistence.

9.1 Save Document

```
sdk.storage.save_document("my_file", "This is the content")
```

Saves to Think OS documents (accessible via Documents menu).

9.2 Load Document

```
content = sdk.storage.load_document("my_file")

if content:
    print(content)
else:
    print("File not found")
```

9.3 Delete Document

```
sdk.storage.delete_document("my_file")
```

9.4 List Documents

```
all_docs = sdk.storage.list_documents()
print(f"Found {len(all_docs)} documents")
```

List with filter:

```
notes = sdk.storage.list_documents(prefix="NOTE_")
```

9.5 Save Mod Data

For mod-specific data (like settings or save files):

```
data = {
    "level": 5,
    "score": 1000,
    "name": "Player1"
}

sdk.storage.save_mod_data("my_game", data)
```

9.6 Load Mod Data

```
data = sdk.storage.load_mod_data("my_game")

if data:
    level = data.get("level", 1)
    score = data.get("score", 0)
else:
    # First time running, use defaults
    level = 1
    score = 0
```

9.7 Complete Storage Example

```
from SDK import ModSDK

def run_mod(os_instance):
    sdk = ModSDK(os_instance)

    # Load saved high score
    data = sdk.storage.load_mod_data("high_score_game") or {"high_score":

    sdk.screen.header("HIGH SCORE GAME")
    print(f"Current high score: {data['high_score']}")
    print()

    # Play game (simplified)
    score = sdk.input.get_int("Enter your score: ", min_val=0)

    # Update high score if beaten
    if score > data['high_score']:
        data['high_score'] = score
        sdk.storage.save_mod_data("high_score_game", data)
        print("New high score!")
    else:
        print("Try again!")

    sdk.input.wait()
```

---

10. General Utilities

The Utils component provides helpful functions.

10.1 Countdown Timer

```
sdk.utils.countdown(5, "Starting in")
```

Displays:

```
Starting in 5...
Starting in 4...
Starting in 3...
Starting in 2...
Starting in 1...
```

10.2 Sleep

```
sdk.utils.sleep(2)  # Pause for 2 seconds
```

10.3 Random Choice

```
options = ["Rock", "Paper", "Scissors"]
computer_choice = sdk.utils.random_choice(options)
print(f"Computer chose: {computer_choice}")
```

10.4 Random Integer

```
dice_roll = sdk.utils.random_int(1, 6)
print(f"You rolled: {dice_roll}")
```

10.5 Get Date and Time

```
current_date = sdk.utils.get_date()
current_time = sdk.utils.get_time()

print(f"Date: {current_date}")
print(f"Time: {current_time}")
```

10.6 Format Number

```
large_number = 1234567
formatted = sdk.utils.format_number(large_number)
print(formatted)  # Output: 1,234,567
```

10.7 Truncate Text

```
long_text = "This is a very long piece of text"
short = sdk.utils.truncate(long_text, length=20)
print(short)  # Output: This is a very lo...
```

10.8 Wrap Text

```
long_text = "This is a long sentence that needs to be wrapped to fit on sc
lines = sdk.utils.wrap_text(long_text, width=30)

for line in lines:
    print(line)
```

---

11. Complete Examples

Example 1: Simple Note Taking App

```python
from SDK import ModSDK

def run_mod(os_instance):
    sdk = ModSDK(os_instance)
    notes = sdk.storage.load_mod_data("notes_app") or {"notes": []}

    while True:
        sdk.screen.header("NOTES APP")

        # Show notes
        if notes["notes"]:
            print("Your notes:")
            for i, note in enumerate(notes["notes"], 1):
                print(f"{i}. {note}")
            print()
        else:
            print("No notes yet.\n")

        # Menu
        choice = sdk.menu.create("Notes Menu", [
            "Add note",
            "Delete note",
            "Clear all"
        ])

        if choice == -1:
            break
        elif choice == 0:  # Add
            note = sdk.input.get_string("Enter note: ")
            notes["notes"].append(note)
            sdk.storage.save_mod_data("notes_app", notes)
            print("Note added!")
            sdk.utils.sleep(1)
        elif choice == 1:  # Delete
            if notes["notes"]:
                idx = sdk.input.get_int("Note number: ", 1, len(notes["not
                notes["notes"].pop(idx - 1)
                sdk.storage.save_mod_data("notes_app", notes)
                print("Note deleted!")
                sdk.utils.sleep(1)
        elif choice == 2:  # Clear
            if sdk.input.yes_no("Delete all notes? (y/n): "):
                notes["notes"] = []
                sdk.storage.save_mod_data("notes_app", notes)
                print("All notes cleared!")
                sdk.utils.sleep(1)
```

Example 2: Number Guessing Game

```python
from SDK import ModSDK

def run_mod(os_instance):
    sdk = ModSDK(os_instance)

    sdk.screen.header("GUESSING GAME")

    print("I'm thinking of a number between 1 and 100!")
    print()

    # Generate random number
    secret = sdk.utils.random_int(1, 100)
    attempts = 0

    while True:
        guess = sdk.input.get_int("Your guess: ", 1, 100)
        attempts += 1

        if guess < secret:
            print("Too low! Try again.\n")
        elif guess > secret:
            print("Too high! Try again.\n")
        else:
            print(f"Correct! You got it in {attempts} attempts!")
            break

    # Save high score
    data = sdk.storage.load_mod_data("guess_game") or {"best": 999}

    if attempts < data["best"]:
        print("NEW RECORD!")
        data["best"] = attempts
        sdk.storage.save_mod_data("guess_game", data)
    else:
        print(f"Best record: {data['best']} attempts")

    print()
    sdk.input.wait()
```

Example 3: Simple Calculator

```python
from SDK import ModSDK

def run_mod(os_instance):
    sdk = ModSDK(os_instance)

    while True:
        sdk.screen.header("CALCULATOR")

        choice = sdk.menu.create("Operation", [
            "Addition",
            "Subtraction",
            "Multiplication",
            "Division"
        ])

        if choice == -1:
            break

        num1 = sdk.input.get_int("Enter first number: ")
        num2 = sdk.input.get_int("Enter second number: ")

        print()

        if choice == 0:
            result = num1 + num2
            print(f"{num1} + {num2} = {result}")
        elif choice == 1:
            result = num1 - num2
            print(f"{num1} - {num2} = {result}")
        elif choice == 2:
            result = num1 * num2
            print(f"{num1} x {num2} = {result}")
        elif choice == 3:
            if num2 == 0:
                print("Cannot divide by zero!")
            else:
                result = num1 / num2
                print(f"{num1} / {num2} = {result}")

        print()
        sdk.input.wait()
```

Example 4: Task Manager with Progress

```
from SDK import ModSDK

def run_mod(os_instance):
    sdk = ModSDK(os_instance)
    tasks = sdk.storage.load_mod_data("task_manager") or {"tasks": []}

    while True:
        sdk.screen.header("TASK MANAGER")

        if tasks["tasks"]:
            completed = sum(1 for t in tasks["tasks"] if t["done"])
            total = len(tasks["tasks"])

            sdk.screen.progress_bar(completed, total, label="Progress")
            print()

            for i, task in enumerate(tasks["tasks"], 1):
                status = "[X]" if task["done"] else "[ ]"
                print(f"{i}. {status} {task['name']}")
            print()
        else:
            print("No tasks yet.\n")

        choice = sdk.menu.create("Task Manager", [
            "Add task",
            "Mark done",
            "Delete task"
        ])

        if choice == -1:
            break
        elif choice == 0:
            name = sdk.input.get_string("Task name: ")
            tasks["tasks"].append({"name": name, "done": False})
            sdk.storage.save_mod_data("task_manager", tasks)
        elif choice == 1:
            if tasks["tasks"]:
                idx = sdk.input.get_int("Task number: ", 1, len(tasks["tas
                tasks["tasks"][idx - 1]["done"] = True
                sdk.storage.save_mod_data("task_manager", tasks)
        elif choice == 2:
            if tasks["tasks"]:
                idx = sdk.input.get_int("Task number: ", 1, len(tasks["tas
                tasks["tasks"].pop(idx - 1)
                sdk.storage.save_mod_data("task_manager", tasks)
```

---

12. Best Practices

12.1 Always Initialize SDK

```
def run_mod(os_instance):
    sdk = ModSDK(os_instance)  # Always do this first
    # Rest of your code...
```

12.2 Use Headers for Clarity

```
sdk.screen.header("MY MOD NAME")
```

Makes your mod look professional.

12.3 Save Data Regularly

```
# After making changes
sdk.storage.save_mod_data("my_mod", data)
```

12.4 Validate Input

Use SDK input functions instead of raw input():

```
# Good
age = sdk.input.get_int("Age: ", 1, 120)

# Avoid
age = int(input("Age: "))  # No validation!
```

12.5 Handle Missing Data

```
data = sdk.storage.load_mod_data("my_mod") or {"default": "value"}
```

Always provide defaults in case data doesn't exist.

12.6 Use Meaningful Variable Names

```
# Good
high_score = sdk.storage.load_mod_data("game")

# Avoid
x = sdk.storage.load_mod_data("game")
```

12.7 Add Comments

```
# Load saved settings
settings = sdk.storage.load_mod_data("settings") or {"volume": 50}

# Get user preference
volume = sdk.input.get_int("Volume (0-100): ", 0, 100)
```

12.8 Test Your Mod

Test your mod with: - Valid input - Invalid input - Empty data - Maximum values - Edge cases

---

13. Troubleshooting

Error: "No module named 'SDK'"

Problem: SDK.py is not in the correct location.

Solution: Make sure SDK.py is in the SAME folder as Think-OS1.0.py, NOT in the mods folder.

Error: "AttributeError: 'ModSDK' object has no attribute..."

Problem: Using wrong SDK function name.

Solution: Check the spelling and refer to this guide for correct function names.

Data Not Saving

Problem: Forgot to call save function.

Solution: Always call `sdk.storage.save_mod_data()` after making changes.

Input Not Validating

Problem: Using raw input() instead of SDK functions.

Solution: Use `sdk.input.get_int()` or `sdk.input.get_string()` instead.

---

14. Reference Guide

Quick Function Reference

Screen Functions

```
sdk.screen.clear()
sdk.screen.header(title, width=50)
sdk.screen.box(text, width=50)
sdk.screen.divider(width=50, char="=")
sdk.screen.centered(text, width=50)
sdk.screen.progress_bar(current, total, width=30, label="Progress")
```

Input Functions

```
sdk.input.get_string(prompt, allow_empty=False)
sdk.input.get_int(prompt, min_val=None, max_val=None)
sdk.input.get_choice(prompt, valid_choices=None)
sdk.input.yes_no(prompt)
sdk.input.wait(prompt="Press Enter to continue...")
```

Menu Functions

```
sdk.menu.create(title, options, show_exit=True)
sdk.menu.list_select(title, items, show_cancel=True)
```

Storage Functions

```
sdk.storage.save_document(name, content)
sdk.storage.load_document(name)
sdk.storage.delete_document(name)
sdk.storage.list_documents(prefix=None)
sdk.storage.save_mod_data(mod_name, data)
sdk.storage.load_mod_data(mod_name)
```

Utility Functions

```
sdk.utils.countdown(seconds, message="Starting in")
sdk.utils.sleep(seconds)
sdk.utils.random_choice(items)
sdk.utils.random_int(min_val, max_val)
sdk.utils.get_date()
sdk.utils.get_time()
sdk.utils.format_number(number)
sdk.utils.truncate(text, length, suffix="...")
sdk.utils.wrap_text(text, width=50)
```

Basic Mod Template

```python
from SDK import ModSDK

def run_mod(os_instance):
    """Mod description here"""

    # Initialize SDK
    sdk = ModSDK(os_instance)

    # Display header
    sdk.screen.header("MOD NAME")

    # Your code here
    print("Hello from my mod!")

    # Wait before returning
    sdk.input.wait()

MOD_INFO = {
    "name": "Mod Name",
    "version": "1.0",
    "author": "Your Name",
    "description": "What your mod does"
}
```

Conclusion

The Think OS SDK makes mod creation fast and easy. With just a few lines of code, you can create professional-looking mods with proper input validation, menus, and data storage.

Remember: - Always initialize the SDK first - Use SDK functions instead of raw Python - Save your data regularly - Test your mods thoroughly - Refer to this guide when needed

Happy modding!

End of Guide

For more information, visit the Think OS community or check the SDK.py source code for advanced features.