⚡ ODDTI™ — ODDTI2 The Dual Platform Edition: A Game inspired from old school hand cricket for Ti Graphing Calculators with Python and PC

Version 2.3 ODDTI2 (Lite Interactive Build)

Designed by: Ganesh P. Nair (⚡GPN⚡)

Engine & Implementation: ChatGPT (GPT-5 Co-Developer)

Contents

🎯 Overview

ODDTI2 is the latest evolution of ODDTI™ — The Odd/Even Hand-Cricket Game, rebuilt from the ground up for both

💻 Desktop Python (3.8 +) and 🧱 TI-84 Plus CE Python calculators.
It runs natively and identically on both environments — no feature stripped, no external modules required.

The game re-creates the nostalgic school-bench odd/even cricket format and turns it into a live Python experience featuring:

- Dynamic score updates and run-chase tracking
- Interactive toss and innings flow
- Best-of-three series mode with score history
- Full compatibility across TI OS and modern Python interpreters

Its purpose remains the same: revive a childhood memory through structured code — a blend of simplicity, logic, and heritage.

🧠 System Requirements

Platform	Requirement
TI-84 Plus CE Python	TI-OS 5.6 + with built-in Python App and min 15KB free space
macOS / Windows / Linux	Python 3.8 or newer
Transfer Tool (TI)	TI-Connect CE Software

File Name: ODDTI2.py (≤ 40 KB, plain text)
Libraries: (built-in random only)

———

⚙️ Installation Guide

🧩 For TI-84 Plus CE Python

1. Install TI-Connect CE (from education.ti.com).
2. Connect calculator → wait for it to appear in sidebar.
3. Drag and drop ODDTI2.py → choose Python Programs → Archive Memory.
4. Send file → wait for transfer → disconnect safely.
5. Open Python App → verify ODDTI2 appears in list.

💻 For Desktop Python (Thonny / PyCharm / Terminal)

1. Save ODDTI2.py to any folder.
2. Run directly by double-clicking in Thonny(or an IDE) or Right click ODDTI2.py and open it with any IDE (installed in your system that runs python 3.8+
   Or copy paste python code from ODDTI2.py (or this manual) into an IDE and run.
3. No installation or packages needed — it runs out of the box.

———

🚀 How to Run

▶ Manual Launch (For Ti)
Pls enter the following prompt after running the code for starting the game…..>-
>>> import ODDTI2
>>> ODDTI2.main()

⚡ Auto-Start Option

Add the following to the end of the file if you want instant launch on import:

```
if __name__ == "__main__":
    main()
```

Now simply type import ODDTI2 and it starts automatically.

▶For PC
- Open the ODDTI2.py file using an IDE like pycharm or Thony (that supports python 3.8 or above)
- Hit run command and Enjoy…….

• If it doesn't start by default pls use manual launch option given above (it works for PC also)

🕹️ Gameplay Instructions

⚾ 1️⃣ Toss Phase

- Choose odd or even.
- Pick a number (0 – 6).
- CPU chooses its own number.
- Sum decides winner:
- If parity matches your call → You win the toss.
- Else → CPU wins.

If you win → choose to bat or bowl.
If CPU wins → it randomly decides.

🏏 2️⃣ Batting / Bowling Phase

Batting:

- Enter number (0 – 6).
- CPU bowls (random 0 – 6).
- If numbers match → OUT!
- Else → add your chosen number to score.

Bowling:

- CPU bats (random 0 – 6).
- You bowl (enter 0 – 6).
- Match = CPU OUT.
- Else → CPU adds runs.

🧱 3️⃣ Score & Target

- Live score display after each ball.
- Shows both numbers (batter & bowler).
- If chasing, displays required runs remaining.
- Target = Opponent Score + 1.

🏆 4️⃣ Results & Series Mode

- Choose Best of 3 Series from menu.
- Game tracks matches won and prints scorecards.
- First to win two matches 🏆 becomes series champion.

📊 Example Gameplay Log

```
=== ODDTI2 v2.3 ===
1) Single Match
2) Best of 3
3) Quit
```

```
Choose: 1

Toss call (odd/even): odd
Your toss num 0–6: 3
CPU: 4  Sum=7 → odd
You win toss → choose to bat.

--- PLAYER INNINGS ---
Bat 4, CPU bowls 1 → +4 (Total 4)
Bat 6, CPU bowls 6 → OUT!
Final: 4
--- CPU CHASE ---
CPU bats 3, You bowl 2 → +3 (Total 3)
CPU bats 2, You bowl 5 → +2 (Total 5)
CPU reached target!
Result: CPU wins by 1 run
```

🧠 Troubleshooting

Problem	Cause	Fix
Bad Token	Smart quotes or MS Word copy	Re-transfer from plain text editor
Syntax Error	Tabs instead of spaces	Replace tabs with 4 spaces
Out of Memory	Too many comments	Use this Lite Edition only
Program Freezes	Stuck input loop	Press [ON] → [2ND] → [QUIT]
No module found	File not in Python memory	Ensure sent to “Python Programs”

⚡ Quick Reference Sheet

Platform	Run Command
TI-84 CE Python	import ODDTI2 → ODDTI2.main()
Desktop Python (Normall)	Run ODDTI2.py in an IDE with py 3.8 or above
Auto-Start Mode	Add if __name__=="__main__": main()

💬 Notes for Developers & Learners

- Fully written in pure Python using only random.
- Modular structure for teaching loops, conditions & functions.
- Excellent intro for STEM students learning logical flow control.
- Identical behavior on TI and Desktop → ideal for cross-platform teaching.

Code>-

```
# ============================================================
# ⚡ ODDTI™ v2.3 — Predictor Edition
# ------------------------------------------------------------
# Developer  : Ganesh P. Nair (⚡GPN⚡)
# Co-Developer (AI Engine): ChatGPT (OpenAI GPT-5)
# Build Date : 2025
# Platform   : Python 3.8+ (PC Edition)
# ------------------------------------------------------------
# © 2025 Ganesh P. Nair. All rights reserved.
#
# LICENSE NOTICE:
# This software is proprietary and distributed for educational
# and personal use only. Commercial use, redistribution, or
# modification of any portion of this code without explicit
# written permission from the author is strictly prohibited.
#
# You may:
#   • Download and run this file for personal, non-commercial use.
#   • Study and learn from the source code for educational purposes.
#
# You may NOT:
#   • Copy, edit, rebrand, or redistribute this code.
#   • Use it in any project, repository, or package without permission.
#   • Claim authorship or modify the copyright notice.
#
# DISCLAIMER:
# This software is provided "as is" without warranty of any kind.
# The author assumes no responsibility for any damage resulting
# from use or misuse of this program.
#
# ------------------------------------------------------------
# OFFICIAL TAGLINE:
#  “The classic Indian bench game, reborn with machine logic.”
# ------------------------------------------------------------
# Official Repository: (to be added after GitHub release)
# ============================================================

# ODDTI_UI.py  - compact, interactive TI-friendly edition
# Save -> transfer to TI-84 CE Python
import random

VALID = (0,1,2,3,4,5,6)

def prompt_choice(prompt, options):
    opts = [o.lower() for o in options]
    while True:
        r = input(prompt).strip().lower()
        if r in opts:
            return r
        print("Invalid. try:", "/".join(options))

def prompt_num(prompt):
```

```
    while True:
        s = input(prompt).strip()
        try:
            v = int(s)
        except:
            print("Enter number 0-6")
            continue
        if v in VALID:
            return v
        print("Enter number 0-6")

# ---------- Toss ----------
def do_toss():
    call = prompt_choice("Toss call (odd/even): ", ("odd","even"))
    p = prompt_num("Your toss num 0-6: ")
    c = random.choice(VALID)
    s = p + c
    parity = "even" if s%2==0 else "odd"
    print("You",p,"CPU",c,"Sum:",s,parity)
    if parity == call:
        print("You win toss")
        who = "player"
    else:
        print("CPU wins toss")
        who = "cpu"
    return who

# ---------- Innings (interactive) ----------
def play_innings(batting, target=None):
    # returns (score, ball_log_list)
    score = 0
    ball_log = []  # list of tuples (batter_num, bowler_num, runs_added)
    print("\n--- {} INNINGS ---".format("PLAYER" if batting=="player" else "CPU"))
    while True:
        if batting == "player":
            b = prompt_num("Bat num 0-6: ")
            bowl = random.choice(VALID)
            print("CPU bowls", bowl)
            if b == bowl:
                print("OUT! (you) Last ball:", b, "vs", bowl)
                ball_log.append((b, bowl, 0))
                break
            score += b
            ball_log.append((b, bowl, b))
            print("Runs+", b, "Total:", score)
            if target is not None:
                need = target + 1 - score
                if need <= 0:
                    print("Target achieved! 🎯")
                    break
                print("Need", need, "more")
        else:
            bat = random.choice(VALID)
            bowl = prompt_num("Bowl num 0-6: ")
            print("CPU bats", bat)
            if bat == bowl:
                print("CPU OUT! Last ball:", bat, "vs", bowl)
                ball_log.append((bat, bowl, 0))
                break
            score += bat
```

```
            ball_log.append((bat, bowl, bat))
            print("CPU +", bat, "Total:", score)
            if target is not None:
                need = target + 1 - score
                if need <= 0:
                    print("CPU reached target")
                    break
                print("CPU needs", need, "more")
    print("--- innings end. Score:", score, "---\n")
    return score, ball_log

# ---------- display helpers ----------
def print_match_summary(player_score, cpu_score, player_batted_first, log_first=None,
log_second=None):
        # show last-ball logs if available (short)
    if log_first or log_second:
        print("\nBall logs (last few balls):")
        if log_first:
            print("Innings 1 last balls:", log_first[-5:])
        if log_second:
            print("Innings 2 last balls:", log_second[-5:])
    print()
    print("\n" + "="*28)
    ord_text = "Player batted 1st" if player_batted_first else "CPU batted 1st"
    print(ord_text)
    print("-"*28)
    print("Player :", player_score)
    print("CPU    :", cpu_score)
    if player_score > cpu_score:
        print("Result : Player wins by", player_score - cpu_score, "runs")
    elif cpu_score > player_score:
        print("Result : CPU wins by", cpu_score - player_score, "runs")
    else:
        print("Result : Match tied")
    print("="*28)


# ---------- Single match ----------
def single_match():
    toss_winner = do_toss()
    if toss_winner == "player":
        pick = prompt_choice("You choose bat or bowl? (bat/bowl): ", ("bat","bowl"))
        player_first = (pick == "bat")
    else:
        comp_choice = random.choice(("bat","bowl"))
        print("CPU chooses to", comp_choice)
        player_first = (comp_choice != "bat")

    if player_first:
        print("You bat first")
        p_score, log1 = play_innings("player")
        print("CPU needs", p_score + 1)
        c_score, log2 = play_innings("computer", target=p_score)
    else:
        print("CPU bats first")
        c_score, log1 = play_innings("computer")
        print("You need", c_score + 1)
        p_score, log2 = play_innings("player", target=c_score)

    print_match_summary(p_score, c_score, player_first, log1, log2)
```

```
    # return winner string for series bookkeeping
    if p_score > c_score:
        return "player", p_score, c_score
    elif c_score > p_score:
        return "cpu", p_score, c_score
    else:
        return "tie", p_score, c_score

# ---------- Best-of-3 ----------
def best_of_three():
    p_wins = 0
    c_wins = 0
    matches = []
    for i in range(1,4):
        print("\n=== Match", i, "===")
        winner, p_score, c_score = single_match()
        matches.append((winner, p_score, c_score))
        if winner == "player":
            p_wins += 1
        elif winner == "cpu":
            c_wins += 1
        print("Series:", p_wins, "-", c_wins)
        if p_wins == 2 or c_wins == 2:
            break

    print("\n=== Series summary ===")
    for idx, m in enumerate(matches, start=1):
        w, ps, cs = m
        label = "Player" if w=="player" else "CPU" if w=="cpu" else "Tie"
        print("M{}: {}  {}-{}".format(idx, label, ps, cs))
    if p_wins > c_wins:
        print("Series winner: Player", p_wins, "to", c_wins)
    elif c_wins > p_wins:
        print("Series winner: CPU", c_wins, "to", p_wins)
    else:
        print("Series ended tied", p_wins, "-", c_wins)
    print()

# ---------- Main UI ----------
def main():
    print("=== ODDTI UI v2 (lite interactive) ===")
    while True:
        print("\n1) Single match")
        print("2) Best of 3")
        print("3) Quit")
        ch = input("Choose 1/2/3: ").strip()
        if ch == "1":
            single_match()
        elif ch == "2":
            best_of_three()
        elif ch == "3":
            print("Bye ⚡GPN⚡ ")
            break
        else:
            print("Invalid")

if __name__ == "__main__":
    main()
#————————————————————————————————————————————————————————————
```

🪙 Credits

Role	Contributor
Concept & Design	Ganesh P. Nair (⚡ GPN ⚡)
Core Engine & Implementation	ChatGPT (GPT-5)
Testing Platforms	TI-84 Plus CE Python, MacBook Pro (M2)

Version: ODDTI™ v2.3 -ODDTI2 (Lite Interactive)
Tagline "The classic Indian bench game, reborn in code."

📘 License & Use

🧩 Pro Tip

Keep both ODDTI2.py and your lab scripts archived on your TI calculator —
you carry a mini engineering console in your pocket that can both play and teach Python.

End of Manual — ODDTI2 v2.3 Dual Edition
Compiled and documented by ⚡ GPN ⚡

Version Summary

Version	Highlights	Platform
v2.0	Base single match logic	TI-84 / PC
v2.1	Best-of-3 series added	TI-84 / PC
v2.2	Scorecards, runs required, UI improvements	TI-84 / PC
v2.3 Predictor edition (This version)	Predictor AI added, smarter CPU, memory system	PC only
v2.3 ODDTI2 (This version)	**V2.3 simplified to run on Ti graphing calculators with python support**	**TI-84 / PC**

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx