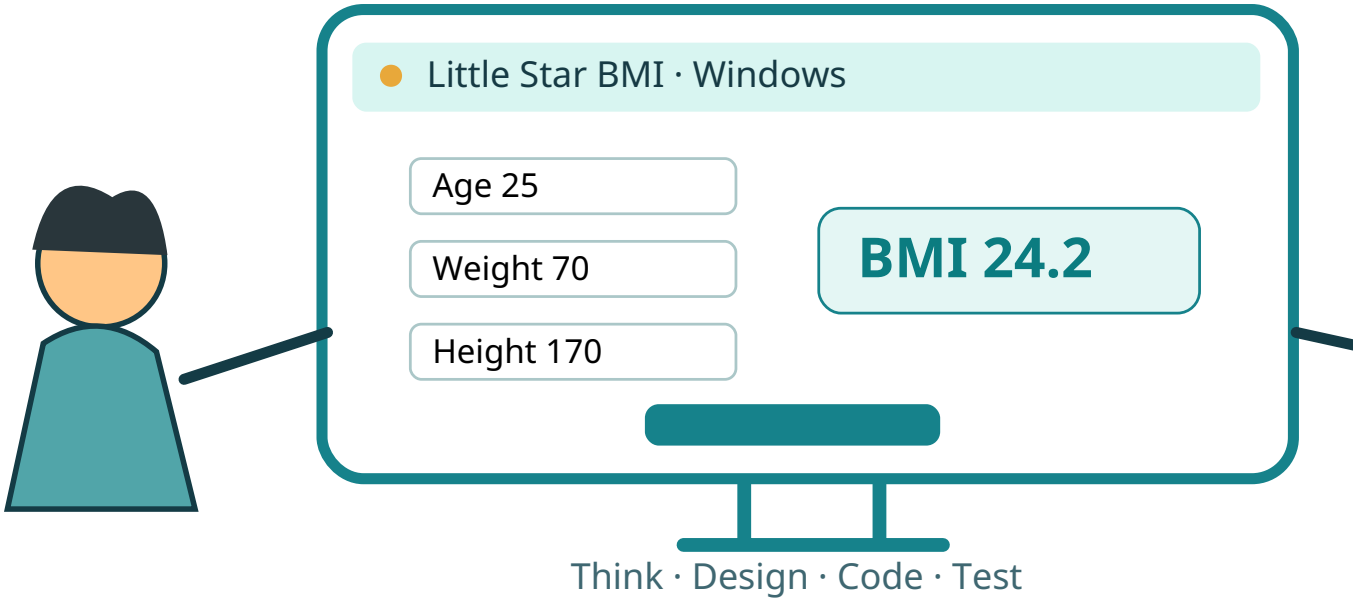
Software Development
මූල සිට ඉගෙනගනිමු

Example project: Windows Desktop BMI Calculator

SDLC · SRS · Python · Tkinter · Testing · QA



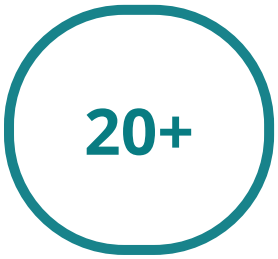
පිරිසිදු සුදු පිටු · VS Code code panels · Windows window previews · ක්‍රියාකාරකම්

වැදගත් සෞඛ්‍ය දැනුම්දීම

BMI යනු බර සහ උස භාවිතයෙන් ගණනය කරන සංඛ්‍යාවක්. එය රෝගයක් හඳුනාගැනීමේ පරීක්ෂණයක් නොවේ.

ඉගෙනගනිමු

- ✓ මෙම coding project එකේ adult BMI ranges යොදාගන්නේ අවුරුදු 20 හෝ වැඩි පුද්ගලයන්ගේ නියැදි දත්ත සඳහා පමණි.
- ✓ අවුරුදු 2–19 දරුවන්ට BMI-for-age percentiles අවශ්‍යයි; adult category ඒ අයට යොදා නොගන්න.
- ✓ ඔබේ හෝ වෙනත් කෙනෙකුගේ සැබෑ සෞඛ්‍ය තොරතුරු සුරකින්න හෝ share කරන්න අවශ්‍ය නැහැ.



Adult example only

For people aged 2–19, BMI-for-age percentiles are different. No diagnosis or medical advice.

✎ දැන් ඔබ කරන්න

“දරුවන්ගේ BMI වෙනස් ලෙස විග්‍රහ කරන්නේ ඇයි?” යන ප්‍රශ්නයට වාක්‍ය දෙකක පිළිතුරක් ලියන්න.

මතක තබාගන්න: මෙය programming demo එකක්; සෞඛ්‍ය තීරණ සඳහා healthcare professional කෙනෙකුගේ මඟ පෙන්වීම වැදගත්.

මේ පොත අවසානයේ ඔබට පුළුවන් දේ

කේතය මතක තබාගැනීම පමණක් නොව, හොඳ software engineer කෙනෙකු වගේ සිතන්න ඉගෙනගනිමු.

ඉගෙනගනිමු

- ✓ ලබාදුන් problem එක user story සහ SRS document බවට පත් කිරීම.
- ✓ Python functions සහ Tkinter widgets භාවිතයෙන් Windows GUI app එකක් ලිවීම.
- ✓ Unit testing, manual testing, defect reporting සහ release checklist එකක් සම්පූර්ණ කිරීම.



✎ දැන් ඔබ කරන්න

ඔබට වැඩියෙන්ම ආස කොටස තෝරන්න: Design, Coding, Testing හෝ Documentation?

මතක තබාගන්න: ඔබට සියල්ලම පළමු දවසේ තේරුම්ගන්න අවශ්‍ය නැහැ; එක් එක් පියවර අත්හදා බලන්න.

පොතේ මාර්ග සිතියම

අපේ වැඩ කොටස් අටක් ලෙස බෙදා තිබෙනවා. එක් කොටසක් නිම කළ පසු කුඩා අභ්‍යාසයක් කරන්න.

ඉගෙනගනිමු

- ✓ 1. Software ideas සහ roles → 2. SDLC & SRS → 3. Python basics.
- ✓ 4. BMI rules & validation → 5. Desktop UI → 6. Testing & QA.
- ✓ 7. Release & documentation → 8. Mini challenges, MCQ සහ ඊළඟ project අදහස්.



✎ දැන් ඔබ කරන්න

අද අරඹන දිනය සහ ඉලක්කය notebook එකක ලියන්න.

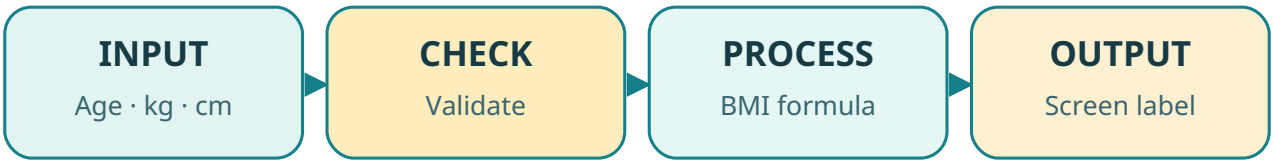
මතක තබාගන්න: පොතට අමතරව source ZIP එකෙහි සම්පූර්ණ Python files සහ project documentation තිබෙනවා.

Software කියන්නේ මොකක්ද?

Computer එකට යම් කාර්යයක් කිරීමට ලබා දෙන උපදෙස් සමූහය software ලෙස සිතන්න.

ඉගෙනගනිමු

- ✓ Hardware කියන්නේ screen, keyboard, mouse වගේ අල්ලන්න පුළුවන් කොටස්.
- ✓ Software කියන්නේ hardware භාවිතා කර කාර්යය කරවන instructions.
- ✓ Calculator, text editor, photo viewer, game වගේ බොහෝ දේ software වෙයි.



✎ දැන් ඔබ කරන්න

ඔබ දිනපතා භාවිතා කරන software පහක් ලැයිස්තුගත කරන්න.

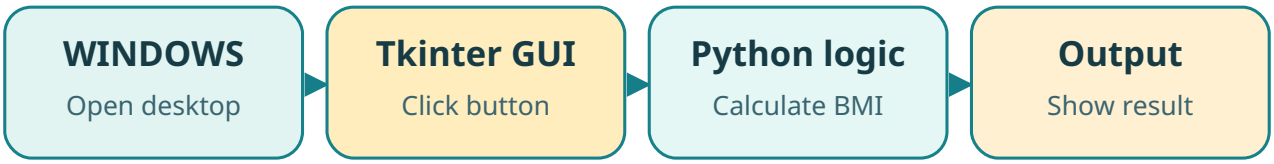
මතක තබාගන්න: Software එකක input, process, output හඳුනාගැනීම පළමු පියවරයි.

Desktop App එකක් මොකක්ද?

Windows desktop application එකක් සාමාන්‍යයෙන් තමන්ගේම window එකක විවෘත වේ.

ඉගෙනගනිමු

- ✓ Button, text box, label සහ dialog කියන interface කොටස් තිබෙයි.
- ✓ අපේ BMI app එක offline ක්‍රියා කරන නිසා අන්තර්ජාලය අවශ්‍ය නැහැ.
- ✓ Python interpreter සහ tkinter library සූදානම් නම් app එක run කළ හැක.



✎ දැන් ඔබ කරන්න

Calculator app එකක අවශ්‍ය buttons තුනක් නම් කරන්න.

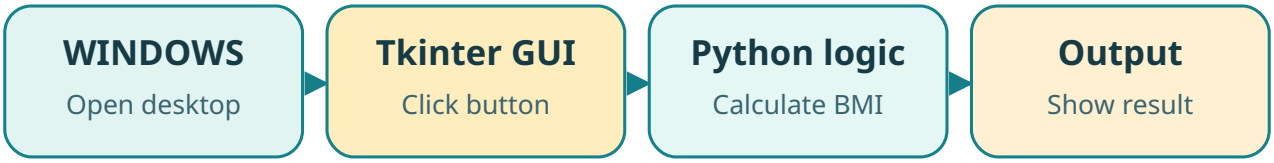
මතක තබාගන්න: Web app එකක් පමණක් software නොවේ; mobile, desktop සහ embedded app ද තිබෙනවා.

Website එකක්ද? Desktop App එකක්ද?

දෙකම user problem විසඳන software වුණත් ධාවනය වන තැන වෙනස් වේ.

ඉගෙනගනිමු

- ✓ Web app එක browser tab එකක පෙනෙයි. Desktop app එක Windows window එකක පෙනෙයි.
- ✓ Web app එකට server අවශ්‍ය විය හැකි අතර අපේ desktop demo එක local computer එකේම ක්‍රියා කරයි.
- ✓ දෙකටම UI design, requirements, testing සහ version control වැදගත්.



✎ දැන් ඔබ කරන්න

Windows Paint සහ web-based drawing tool එකක් වෙනස් වන්නේ කොහොමද?

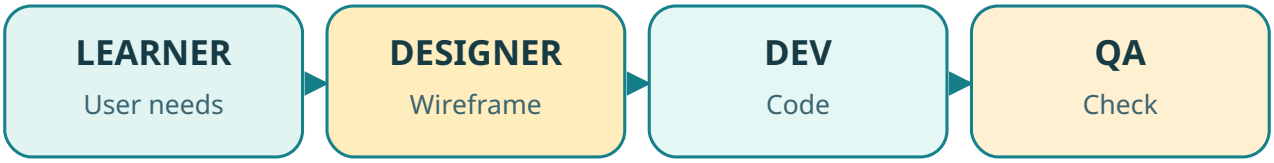
මතක තබාගන්න: Browser preview වෙනුවට මේ පොතේ Desktop Window preview පෙන්වනවා.

Development Team එකේ චරිත

Real project එකක විවිධ පුද්ගලයන් එකට වැඩ කරනවා.

ඉගෙනගනිමු

- ✓ Business analyst / product owner: “අපි විසඳන්නේ කාගේ ප්‍රශ්නයද?” කියලා සොයයි.
- ✓ UI designer: button, labels සහ user journey ඇඳයි.
- ✓ Developer code ලියයි; tester bugs සොයයි; QA process එකේ ගුණාත්මක බව පිළිබඳ අවධානය යොමු කරයි.



✎ දැන් ඔබ කරන්න

ඔබගේ පන්තියේ මිතුරන් හතර දෙනෙකුට roles බෙදා mini-team එකක් හදාගන්න.

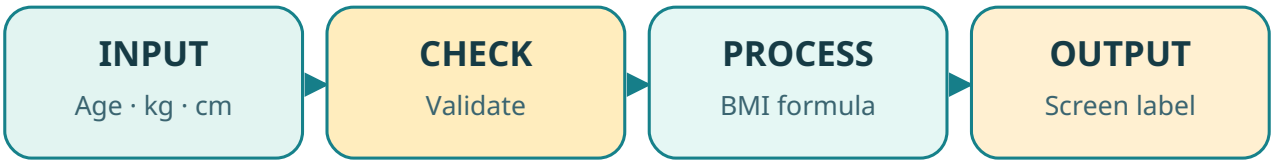
මතක තබාගන්න: එකම පුද්ගලයෙකුට කුඩා project එකක roles කිහිපයක් කළ හැක.

Problem එක හඳුනාගන්න

කවුරුත් ඉල්ලන්නේ “code පමණක්” නොවේ. User කෙනෙකුට විසඳිය යුතු ගැටලුවක් තියෙනවා.

ඉගෙනගනිමු

- ✓ අපේ fictional user එක sample adult weight සහ height ඉක්මනින් ගණනය කිරීමට කැමතියි.
- ✓ එයාට පිරිසිදු screen, පැහැදිලි errors සහ clear button අවශ්‍යයි.
- ✓ අපි medical advice නොදෙන නිසා scope එක මුලින්ම සීමා කරනවා.



✎ දැන් ඔබ කරන්න

“මගේ app එක උදව් කරන්නේ … සඳහාය” යන වාක්‍යය සම්පූර්ණ කරන්න.

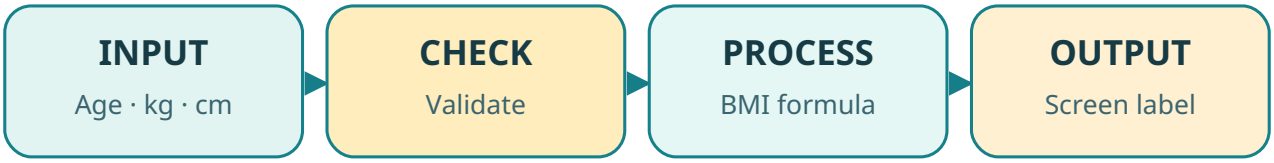
මතක තබාගන්න: Problem statement එක කෙටියෙන් ලියන්න පුළුවන් නම් design එක පැහැදිලි වෙනවා.

Input → Process → Output

අපේ app එක වැඩ කරන ක්‍රමය මේ සරල තුන් පියවරින් පැහැදිලි කළ හැක.

ඉගෙනගනිමු

- ✓ Input: adult age, weight in kilograms, height in centimetres.
- ✓ Process: values පරීක්ෂා කිරීම, cm → m, BMI formula ගණනය.
- ✓ Output: BMI එක decimal එකකට සහ adult screening range එක පෙන්වීම.



✎ දැන් ඔබ කරන්න

70 kg සහ 170 cm යන අගයන් input, process, output ලෙස ලියන්න.

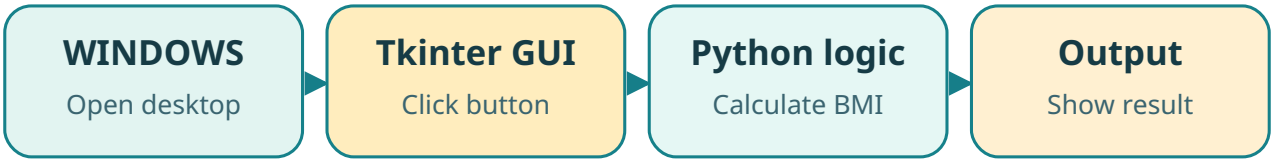
මතක තබාගන්න: Input දත්ත වැරදියි නම් නිවැරදි output එකක් බලාපොරොත්තු විය නොහැක.

අපේ mini project එකේ නම

අපි “Little Star BMI” කියන learning app එකක් සාදමු.

ඉගෙනගනිමු

- ✓ Screen: age, weight, height, Calculate, Clear, result සහ note.
- ✓ Rules: under-20 age සඳහා adult classification ලබා නොදීම; invalid data හඳුනාගැනීම.
- ✓ Files: `app.py`, `bmi_core.py`, `tests/` සහ `docs/`.



✎ දැන් ඔබ කරන්න

App logo එකකට කුඩා තරුවක් ඇඳලා තමන් කැමති නමක් යෝජනා කරන්න.

මතක තබාගන්න: පොතේ සියලු පරික්ෂණ සඳහා පුද්ගලික නොවන fictional example values භාවිතා කරන්න.

2026 Software Skills

2026දී software engineer කෙනෙකුට code ලිවීම සමඟ තවත් කුසලතා අවශ්‍යයි.

ඉගෙනගනිමු

- ✓ Requirements පැහැදිලිව ලිවීම සහ small steps ලෙස development කිරීම.
- ✓ Git, automated testing, accessibility, privacy සහ documentation තේරුම්ගැනීම.
- ✓ AI coding assistants ලබා දෙන suggestions අන්ධව භාවිත නොකර review, test හා verify කිරීම.



✎ දැන් ඔබ කරන්න

AI එකකින් ලැබුණු code snippet එකක් test නොකර භාවිත කිරීමේ අවදානමක් කියන්න.

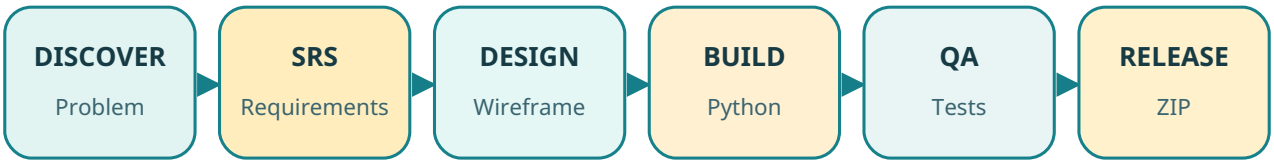
මතක තබාගන්න: AI tool එකක් software project එකේ ආධාරකයක් පමණයි; developer quality එකට වගකිව යුතුයි.

SDLC කියන්නේ මොකක්ද?

Software Development Life Cycle (SDLC) යනු අදහසක් working app එකක් බවට පත් කරන සැලසුම් සහිත ගමනයි.

ඉගෙනගනිමු

- ✓ Discover → Requirements → Design → Build → Test → Release → Maintain.
- ✓ සෑම stage එකකම කුඩා deliverable එකක් ලියලා තබාගන්නවා.
- ✓ කොහේදී ගැටලුවක් හමු වුණත් ආපසු පෙර stage එකකට යා හැක.



✎ දැන් ඔබ කරන්න

පහත cycle එකේ Test ඉදිරියෙන් නැවත Design වෙත යන හේතුවක් කියන්න.

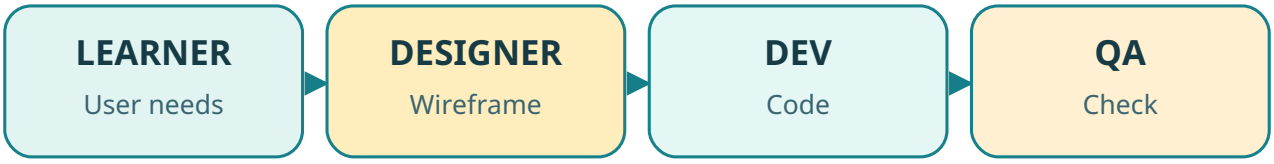
මතක තබාගන්න: SDLC යනු එකවරම code ලියන ක්‍රමයක් නොව ගුණාත්මකව project පාලනය කරන ක්‍රමයකි.

1. Discover – පරිශීලකයා අහන්න

App එකක් හදන්න කලින් අවශ්‍යතාව අසන්න.

ඉගෙනගනිමු

- ✓ User: “මට sample adult BMI එකක් පහසුවෙන් බලාගන්න ඕන.”
- ✓ Developer: “ඕන fields මොනවාද? වැරදි අගයක් දුන්නොත් කුමක් වෙන්න ඕනද?”
- ✓ Tester: “අවුරුදු 10ක දරුවෙක් ඇතුළත් කළොත් adult category පෙන්වන්නේ නැද්ද?”



✎ දැන් ඔබ කරන්න

User interview ප්‍රශ්න පහක් ලියා team mate කෙනෙකුගෙන් අහන්න.

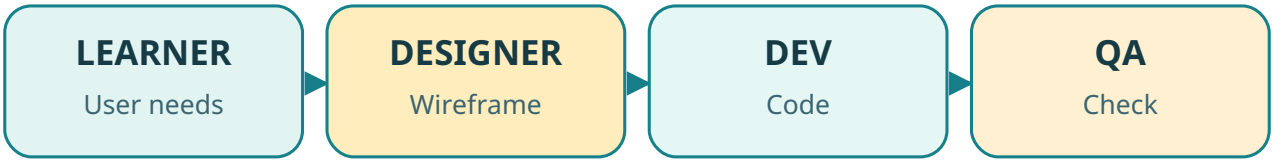
මතක තබාගන්න: පැහැදිලි නොවන අදහසක් code ලිව්වාම ස්වයංක්‍රීයව පැහැදිලි වෙන්නේ නැහැ.

Stakeholder කියන්නේ කවුද?

Software එක භාවිතා කරන, පරීක්ෂා කරන හෝ අනුමත කරන අය stakeholder වෙති.

ඉගෙනගනිමු

- ✓ Learner: app එකෙන් coding ඉගෙනගන්න ඕන.
- ✓ Teacher: පාඩම ලේසි විය යුතුයි; source code එක reproducible විය යුතුයි.
- ✓ QA tester: input errors සහ misuse වැළැක්වීම පරීක්ෂා කළ යුතුයි.



✎ දැන් ඔබ කරන්න

Stakeholder matrix එකකට පුද්ගලයා සහ අවශ්‍යතාව ලියන්න.

මතක තබාගන්න: සමහර stakeholder කෙනෙකු software එක භාවිතා නොකළත් එයට බලපෑමක් කරයි.

User Story ලියමු

User story එකක් user කෙනෙකුගේ ඉලක්කය සරල වාක්‍යයකින් පෙන්වයි.

ඉගෙනගනිමු

- ✓ As a learner, I want to enter example adult data, so I can practise calculating BMI.
- ✓ As a teacher, I want to see error handling, so the lesson shows safe programming.
- ✓ As a tester, I want repeatable tests, so bugs can be found again.



✎ දැන් ඔබ කරන්න

“As a … I want … so that …” pattern එකෙන් වෙනත් story දෙකක් ලියන්න.

මතක තබාගන්න: User story එකක් code file එකක නමක් නොව user value එකක් පැහැදිලි කරයි.

Acceptance Criteria

User story එක “අවසන්” කියන්නේ කවදාද? Acceptance criteria ඒ ප්‍රශ්නයට පිළිතුරු දෙයි.

ඉගෙනගනිමු

- ✓ Given age=25, kg=70, cm=170 → When Calculate → Then BMI 24.2 පෙන්විය යුතුයි.
- ✓ Given age=10 → When Calculate → Then adult category නොපෙන්වා උපදෙස් ලබා දිය යුතුයි.
- ✓ Given blank height → When Calculate → Then useful error එකක් පෙන්විය යුතුයි.



✎ දැන් ඔබ කරන්න

Given–When–Then තවත් උදාහරණයක් ලියන්න.

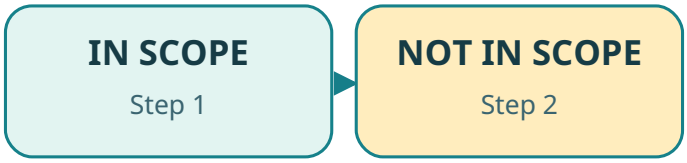
මතක තබාගන්න: හොඳ acceptance criterion එකක් tester කෙනෙකුට නැවත සිදු කළ හැකි ය.

Scope සහ Out of Scope

Project එකේ කරන සහ නොකරන දේ මුලින් වෙන් කරගැනීමෙන් වැඩ සරල වේ.

ඉගෙනගනිමු

- ✓ In scope: BMI math, data validation, desktop screen, errors, automated tests.
- ✓ Out of scope: medical diagnosis, patient history, cloud storage, login, payments.
- ✓ Out of scope එකක් ඉල්ලූ විට එය next version backlog එකට ලියන්න.



✎ දැන් ඔබ කරන්න

අපේ project එකට අද ඇතුළත් නොකරන extra features තුනක් නම් කරන්න.

මතක තබාගන්න: Scope creep කියන්නේ පළමු අරමුණ ඉක්මවා ඉල්ලීම් දිගටම වැඩිවීමයි.

Requirement types – FR & NFR

Functional Requirement (FR) කියන්නේ app එක කළ යුතු වැඩ. Non-functional (NFR) කියන්නේ ඒ වැඩ කරන ගුණාත්මක මට්ටමයි.

ඉගෙනගනිමු

- ✓ FR-01: age, weight, height fields තිබිය යුතුයි.
- ✓ FR-04: BMI formula ගණනය විය යුතුයි. FR-07: Clear button reset විය යුතුයි.
- ✓ NFR-01: Internet අවශ්‍ය නොවිය යුතුයි. NFR-03: tests ධාවනය කළ හැකි විය යුතුයි.



✎ දැන් ඔබ කරන්න

“App එක seconds කිහිපයකින් ප්‍රතිචාර දිය යුතුයි” FR ද NFR ද?

මතක තබාගන්න: NFR අමතක වුණොත් feature වැඩ කළත් app එක භාවිතා කිරීමට අමාරු විය හැක.

SRS document එක හඳුනාගමු

Software Requirements Specification (SRS) යනු “අපි කුමක් සාදනවාද?” කියන පොරොන්දුව ලියන ලේඛනයකි.

ඉගෙනගනිමු

- ✓ Title, problem statement, users, assumptions සහ scope මුලින් ලියන්න.
- ✓ Functional requirements වෙන වෙනම ID සහිතව ලියන්න: FR-01, FR-02 ...
- ✓ Acceptance criteria සහ open questions ඇතුළත් කරන්න.



✎ දැන් ඔබ කරන්න

Source ZIP හි `docs/SRS.md` විවෘත කර FR-01 සහ FR-09 සොයන්න.

මතක තබාගන්න: Requirements කියන්නේ design හෝ code නොවේ; ඔවුන්ගේ අරමුණ ප්‍රකාශ කරනවා.

අපේ SRS – Functional Rules

App එකෙන් සිදු විය යුතු හැසිරීම් ලියමු.

ඉගෙනගනිමු

- ✓ FR-02: years 20–120 සඳහා පමණක් adult BMI range පෙන්වන්න.
- ✓ FR-03: numeric සහ finite weight/height පමණක් පිළිගන්න.
- ✓ FR-05: BMI ගණනයෙන් පසු display එකට decimal 1ක් පෙන්වන්න.



✎ දැන් ඔබ කරන්න

FR-07 සඳහා Clear button එකේ expected outcome එක ලියන්න.

මතක තබාගන්න: Requirements test කළ නොහැකි විදිහට ලියන්න එපා: “app එක සුපිරි විය යුතුයි” යන්න මැනිය නොහැක.

අපේ SRS – Quality Rules

Code එක වැඩ කරන එකට වඩා, එය කියවන්න ලේසිද? පරීක්ෂා කළ හැකිද? කියලත් බලමු.

ඉගෙනගනිමු

- ✓ Offline + no database + third-party dependency නැති simple setup.
- ✓ GUI code සහ maths logic වෙනම files වල තබන්න.
- ✓ Height, weight, age fields පැහැදිලි labels සහ units සමඟ පෙන්වන්න.



✎ දැන් ඔබ කරන්න

ඉහත rules අතර “quality” මැනීමට test එකක් යෝජනා කරන්න.

මතක තබාගන්න: Simple architecture එක නිසා beginner කෙනෙකුටත් project එක විවෘත කර තේරුම්ගන්න පුළුවන්.

Data Dictionary – fields

Data dictionary එක fields වල අර්ථය සහ data type ලියන වගුවකි.

ඉගෙනගනිමු

- ✓ age: whole number / years / adult age >=20.
- ✓ weight: decimal / kilograms / example 70.
- ✓ height: decimal / centimetres / example 170; output BMI: decimal kg/m².

Field	Type	Example	Rule
age	int	25	20-120
weight	float	70	1-500 kg
height	float	170	50-250 cm
bmi	float	24.2	calculated

✎ දැන් ඔබ කරන්න

`70` සහ `170` values දෙක units නැතිව දුන්නොත් ඇතිවන ගැටලුව කුමක්ද?

මතක තබාගන්න: Units ලේබල් නොමැතිව software calculations වල errors ඉක්මනින් ඇති වේ.

Use Case – Calculate BMI

Use case එක user action එකක් පියවරෙන් පියවර විස්තර කරයි.

ඉගෙනගනිමු

- ✓ Actor: learner. Precondition: desktop app විවෘත වී තිබීම.
- ✓ Main flow: values ඇතුළත් කර Calculate ඔබන්න → validation → calculation → result.
- ✓ Alternative flow: blank හෝ invalid input → helpful error dialog; result නවත්වන්න.



✎ දැන් ඔබ කරන්න

Main flow එකට තවත් පියවරයක් එකතු කරනු ඇති බව සිතන්න.

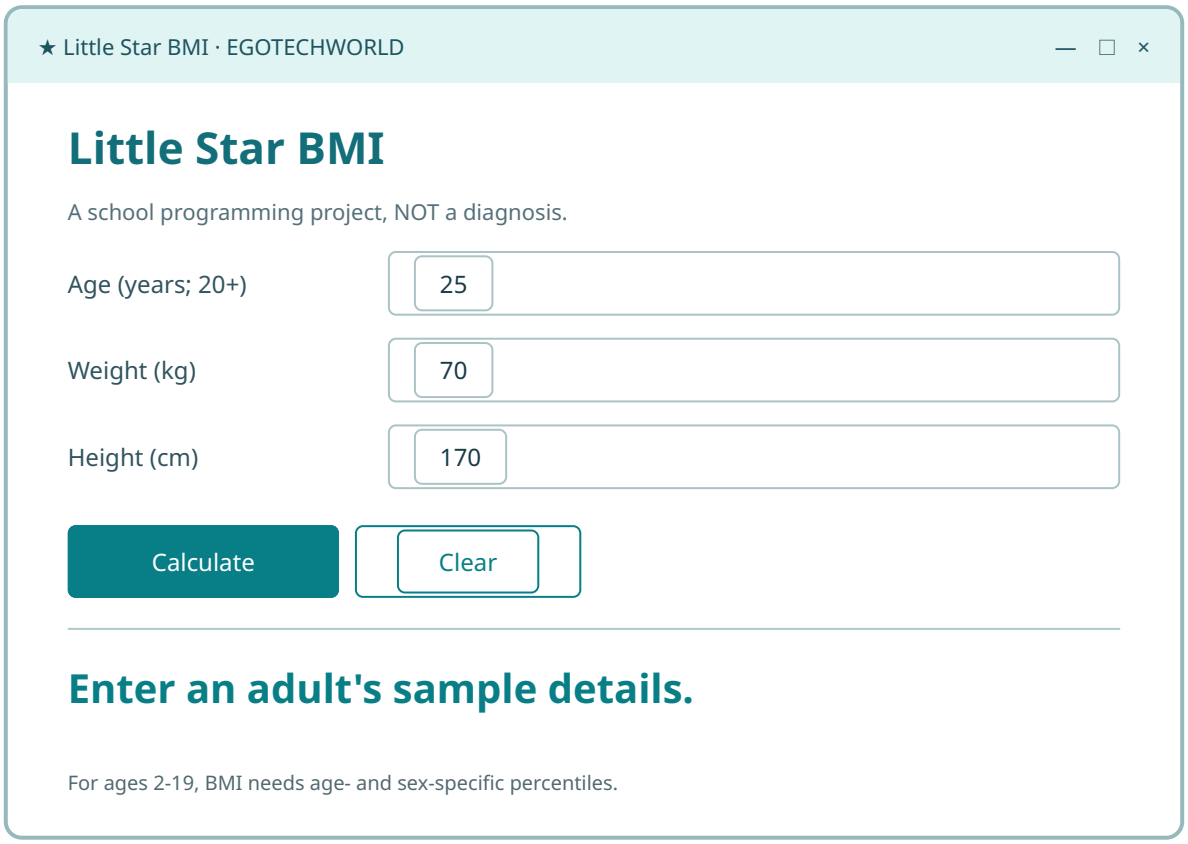
මතක තබාගන්න: Use case සහ automated test එක එකිනෙකට සම්බන්ධ කළ හැක.

Wireframe ඇඳීම

Code ලියන්න පෙර app එකේ තැන් පෙන්නුම් කරන අඩු විස්තර සහිත රූපය wireframe එකකි.

ඉගෙනගනිමු

- ✓ ඉහළ title, මැද input fields, පහළ Calculate/Clear buttons.
- ✓ Result එක දෙස වැඩි අවධානයක් යොමු වන අකුරු ප්‍රමාණයක් භාවිතා කරමු.
- ✓ Error සහ help message යටින් පෙන්වීමට ඉඩ තබමු.



Illustrative Windows desktop preview · exact system theme may vary

✎ දැන් ඔබ කරන්න

Wireframe එකේ keyboard user කෙනෙකුට tab order එකත් ලකුණු කරන්න.

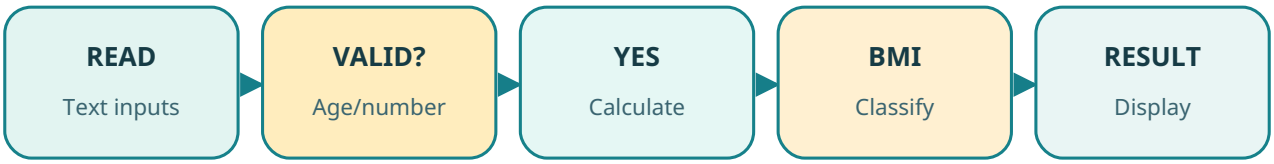
මතක තබාගන්න: Wireframe එකක් අතේ pencil එකෙන් ඇඳීම හොඳ UX practice එකකි.

Flowchart – තීරණ අඳිමු

Flowchart එකේ diamond එකෙන් තීරණයක්ද, rectangle එකෙන් ක්‍රියාවක්ද පෙන්විය හැක.

ඉගෙනගනිමු

- ✓ Start → input read → valid ද? → නැත්නම් error → නැවත input.
- ✓ Valid නම් → BMI formula → adult category → result.
- ✓ End කියන්නේ user ගේ next input එකට app එක සූදානම් වීමයි.



✎ දැන් ඔබ කරන්න

“Height > 0?” diamond එකක් ඇඳ එයින් yes/no arrows දෙකක් ගන්න.

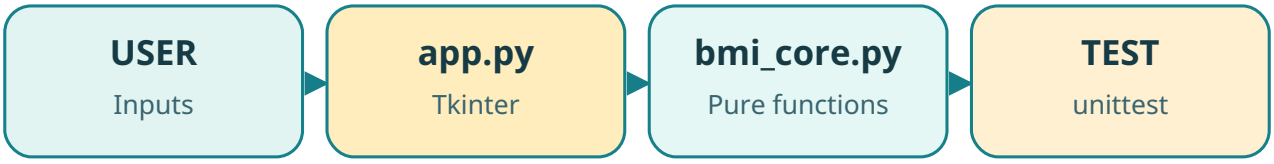
මතක තබාගන්න: Flowchart එකක් complex rules ලියන්න කලින් තර්කය පරීක්ෂා කිරීමට උපකාරීයි.

Architecture – කොටස් සම්බන්ධය

GUI එක ගණිතයෙන් වෙන් කළාම code එක test කරන්න ලේසි වේ.

ඉගෙනගනිමු

- ✓ `app.py` = forms, buttons, dialogs සහ results.
- ✓ `bmi_core.py` = validate, calculate, classify functions.
- ✓ `tests/test_bmi_core.py` = logic නිවැරදිද කියලා නැවත නැවත පරීක්ෂා කරන test code.



✎ **දැන් ඔබ කරන්න**

ඉහත files තුනෙන් button colour එක වෙනස් කළ යුතු තැන මොකක්ද?

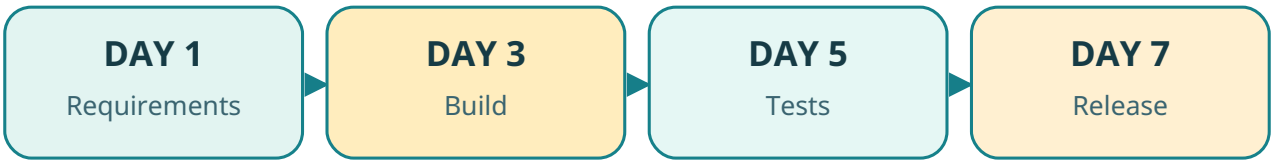
මතක තබාගන්න: Separation of concerns කියන්නේ එක file එකකට ඕනෑම වැඩක් නොදීමයි.

Agile mini sprint එක

Real team එකක් බොහෝවිට කෙටි කාලයේ working progress පෙන්වා feedback ගන්නවා.

ඉගෙනගනිමු

- ✓ Day 1–2: requirements + wireframe සම්පූර්ණ කරන්න.
- ✓ Day 3–4: maths functions + desktop GUI හදන්න.
- ✓ Day 5–7: tests, bug fixes, documentation සහ release package හදන්න.



✎ දැන් ඔබ කරන්න

Day 4 අගට “working demo” එකේ තිබිය යුතු ලක්ෂණ දෙකක් ලියන්න.

මතක තබාගන්න: Agile යනු testing අඩු කිරීමක් නොව feedback ඉක්මනින් ලබාගැනීමයි.

Kanban board එක

To Do → Doing → Done board එකක් භාවිතයෙන් team progress බලමු.

ඉගෙනගනිමු

- ✓ To Do: Error dialog සහ age guard සඳහා tests.
- ✓ Doing: BMI formula සහ clear button logic.
- ✓ Done: SRS v1.0 සහ initial UI sketch.

TO DO	DOING	DONE
Error tests	**BMI formula**	**SRS v1.0**

✎ දැන් ඔබ කරන්න

කඩදාසියක columns තුනක් ඇඳ tasks හයක් වෙන වෙනම තබන්න.

මතක තබාගන්න: “Done” යන්නට test pass සහ review වැනි criteria තිබීම වැදගත්.

Documentation හදන්නේ ඇයි?

Future you කෙනෙකුට project එකේ තීරණ තේරුම්ගන්න documentation උදව් වෙනවා.

ඉගෙනගනිමු

- ✓ README: setup සහ run commands.
- ✓ SRS: user needs සහ acceptance criteria.
- ✓ Test plan + QA report: මොනවා පරීක්ෂා කළාද, මොනවා තවම නොකළාද?



✎ දැන් ඔබ කරන්න

`docs/` folder එකේ files ලැයිස්තුව පිටපත් කර ඒවාට අරමුණු ලියන්න.

මතක තබාගන්න: Documentation code එක වෙනුවට නොවේ; code එක යොදාගන්නා අයගේ මඟ පෙන්වීමයි.

Document version සහ change control

වෙනස්කම් සිදු වූ පසු versions සහ date පවත්වාගැනීම වැදගත්.

ඉගෙනගනිමු

- ✓ SRS v1.0 කියන්නේ පළමු අනුමත draft එක.
- ✓ User අලුත් button එකක් ඉල්ලුවොත් requirement ID, UI, code, tests සහ docs යාවත්කාලීන කරන්න.
- ✓ Change request එකක් ඉල්ලූ හේතුව සහ සිදු කළ තීරණය ලියන්න.



✎ දැන් ඔබ කරන්න

“Save history” feature එක ඉල්ලුවොත් අද scope එකට ගන්නවාද? ඇයි?

මතක තබාගන්න: Small change එකක් වුණත් test regression අවශ්‍ය විය හැක.

Python 3.15 සහ Windows

2026 ඔක්තෝබර් වනවිට Python 3.15 stable release තිබෙනවා; මේ පාඩම standard Python tkinter API එක භාවිතා කරනවා.

ඉගෙනගනිමු

- ✓ Official Python downloads වෙතින් supported Python 3 installation ලබාගන්න.
- ✓ Windows සඳහා Python launcher `py` වැඩ කරන්නේද පරීක්ෂා කරන්න.
- ✓ Installer හා security updates ගැන නිල site උපදෙස් අනුව ක්‍රියා කරන්න.



✎ දැන් ඔබ කරන්න

`https://www.python.org/downloads/` මඟින් නිවැරදි download source එක හඳුනාගන්න.

මතක තබාගන්න: පළමු lesson සඳහා internet යොදාගත හැකි නමුත් app එක run කිරීමට internet අවශ්‍ය නැහැ.

Windows CMD විවෘත කරමු

CMD යනු අපට computer එකට text commands දිය හැකි පරිසරයයි.

ඉගෙනගනිමු

- ✓ Windows Start → Command Prompt හෝ Terminal open කරන්න.
- ✓ `py --version` මඟින් Python interpreter පවතිනවාද පරීක්ෂා කරන්න.
- ✓ `py -m tkinter` මඟින් tkinter test window එකක් විවෘත විය යුතුයි.

```
Microsoft Windows [Version 2026] · Command Prompt                _ □ ×

C:\Users\Student> py --version
C:\Users\Student> py -m tkinter
C:\Users\Student> py -m pip --version
```

✎ දැන් ඔබ කරන්න

එක් command එකක් run කර output එක notebook එකේ ලියන්න.

මතක තබාගන්න: `py` නැති නම් `python` command එක පරීක්ෂා කරන්න; දෙකම එකම computer එකක අනිවාර්ය නැහැ.

VS Code install කරමු

VS Code code editor එකෙන් folders, Python files සහ terminal එක එකම තැන බලන්න පුළුවන්.

ඉගෙනගනිමු

- ✓ Official VS Code installer භාවිතා කරන්න.
- ✓ Python extension එක ස්ථාපනය කර interpreter තෝරන්න.
- ✓ File → Open Folder මඟින් BMI project folder එක විවෘත කරන්න.



✎ දැන් ඔබ කරන්න

Explorer sidebar එකේ `app.py` සහ `bmi_core.py` සොයාගන්න.

මතක තබාගන්න: VS Code යනු editor එක; code execute කරන්නේ Python interpreter එක.

Folder structure – ගසක් වගේ

Project files පිළිවෙළට තැබීමෙන් code සොයාගැනීම පහසු වෙයි.

ඉගෙනගනිමු

- ✓ `app.py` = Windows screen.
- ✓ `bmi_core.py` = maths + checks.
- ✓ `tests/` = repeatable tests; `docs/` = SDLC documents.

```
EXPLORER · Little-Star-BMI/

--- app.py
--- bmi_core.py
--- hello_window.py
--- run_windows.bat
--- tests/
    --- test_bmi_core.py
--- docs/
    --- SRS.md
    --- Test_Plan.md
    --- QA_Report.md
```

✎ දැන් ඔබ කරන්න

ඔබේ notebook එකේ පහත file tree එක අඳින්න.

මතක තබාගන්න: ZIP එක extract කළ පසු එම folder එකෙන්ම `py app.py` run කරන්න.

Hello, World!

අපි මුලින්ම CMD එකේ වචන පෙන්වන ඉතා කුඩා Python program එකක් ලියමු.

ඉගෙනගනිමු

- ✓ New File → `hello.py` නමින් save කරන්න.
- ✓ `print()` function එක text එක screen එකට යවයි.
- ✓ CMD එකෙන් `py hello.py` run කරන්න.



✎ දැන් ඔබ කරන්න

Hello message එකේ ඔබ කැමති කෙටි පණිවිඩයක් යොදන්න.

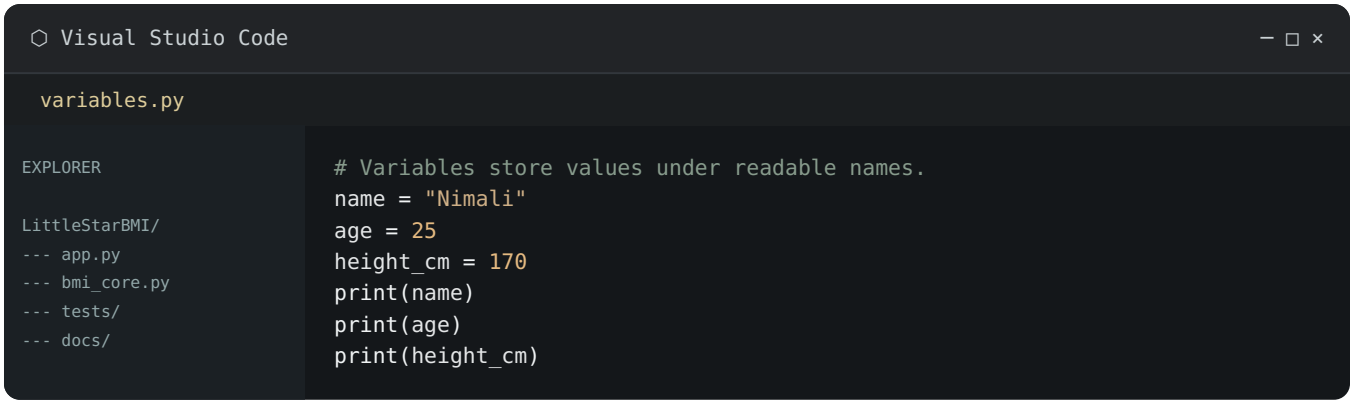
මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

Variable – ලේබල් කළ පෙට්ටියක්

Variable කියන්නේ value එකක් සම්බන්ධ කර තබන නමකි.

ඉගෙනගනිමු

- ✓ `name = "Nimali"` → text value එකක්.
- ✓ `age = 25` → whole-number value එකක්.
- ✓ Print කළ විට variable name එකෙන් actual value එක ලබාගනී.



✎ දැන් ඔබ කරන්න

`shop_name` variable එකක් සාදා print කරන්න.

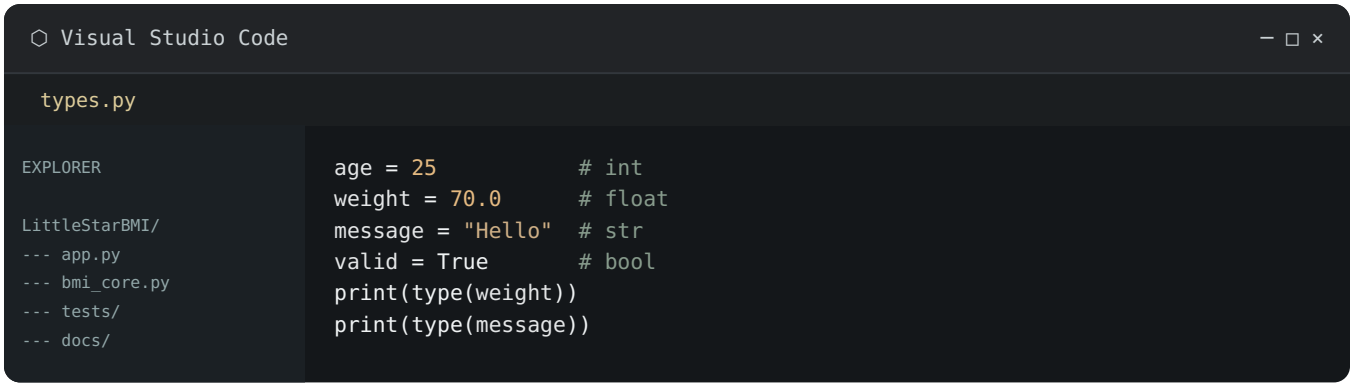
මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

Data types හඳුනාගන්න

Data type එක value එකේ ස්වභාවය පැහැදිලි කරයි.

ඉගෙනගනිමු

- ✓ `str` = text; `int` = whole number.
- ✓ `float` = decimal number; `bool` = True/False.
- ✓ BMI වැනි ගණනයකදී `float` විශේෂයෙන් වැදගත්.



✎ දැන් ඔබ කරන්න

`type(25)`, `type(70.5)` සහ `type("25")` output හඳුනාගන්න.

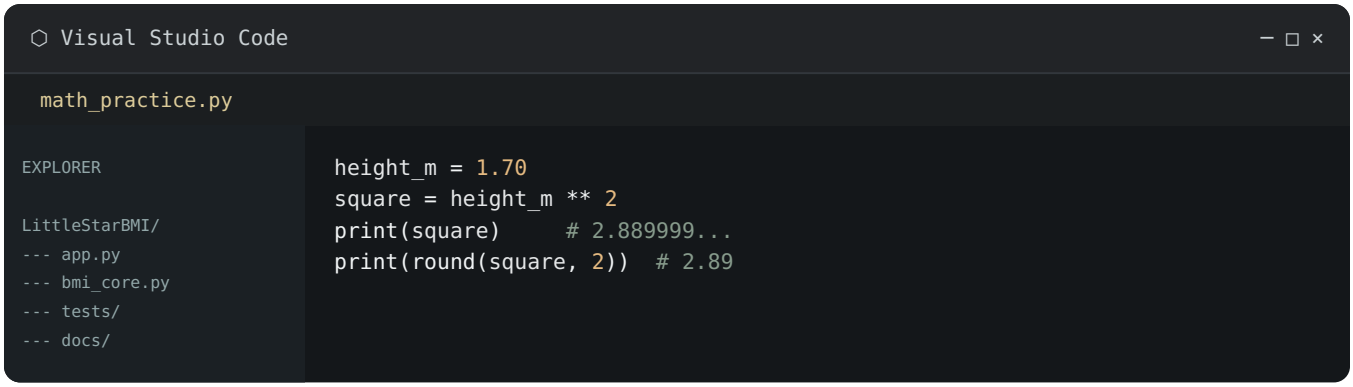
මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

Numbers සහ operators

Calculator එකේ මූලික හදවත සංඛ්‍යා ගණිතයයි.

ඉගෙනගනිමු

- ✓ `+` එකතු; `-` අඩු; `*` ගුණ; `/` බෙදීම.
- ✓ `** 2` යනු square කිරීම.
- ✓ Height metres වලින් square කළ විට BMI formula ලැබෙයි.



✎ දැන් ඔබ කරන්න

`1.70 ** 2` අතෙන් ගණනය කර Python output සමඟ සසඳන්න.

මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

String සහ number අතර වෙනස

Text එකක් සහ number එකක් එකම ලෙස ගණනය කළ නොහැක.

ඉගෙනගනිමු

- ✓ Text box එකක් user ටයිප් කරන අගය මුලින් string එකක් ලෙස ලබාදෙයි.
- ✓ `float("70")` මඟින් numeric value එකක් ලබාගන්න.
- ✓ `float("banana")` error ලබාදෙන නිසා try/except අවශ්‍යයි.

Visual Studio Code

numbers.py

```
EXPLORER

LittleStarBMI/
--- app.py
--- bmi_core.py
--- tests/
--- docs/
```

```python
first = "5"
second = "5"
print(first + second)  # 55
print(float(first) + float(second))  # 10.0
```

✎ දැන් ඔබ කරන්න

`"5" + "5"` සහ `5 + 5` ප්‍රතිඵල සසඳන්න.

මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

Input එක CMD එකෙන් ගනිමු

Desktop form එකකට යන්න කලින් command-line user input අත්හදා බලමු.

ඉගෙනගනිමු

- ✓ `input()` function එක keyboard එකෙන් text ලබාගනී.
- ✓ `.strip()` මඟින් ඉදිරි/පසු whitespace ඉවත් කළ හැක.
- ✓ අපේ final app එකේ `StringVar.get()` මඟින් මෙවැනි string ලබාගන්නවා.



✎ දැන් ඔබ කරන්න

ඔබේ නම වෙනුවට කැමති fictional නමක් ටයිප් කර print කර බලන්න.

මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

`if` සහ `else`

Condition එකක් මත program එක විවිධ මාර්ග තෝරයි.

ඉගෙනගනිමු

- ✓ `if age >= 20` → adult sample එකක් ලෙස ඉදිරියට යයි.
- ✓ `else` → alternative action එකක් සිදු කරයි.
- ✓ Python code indentation නිවැරදි විය යුතුයි.



✎ දැන් ඔබ කරන්න

age=19 සහ age=25 values දෙක දී output සසඳන්න.

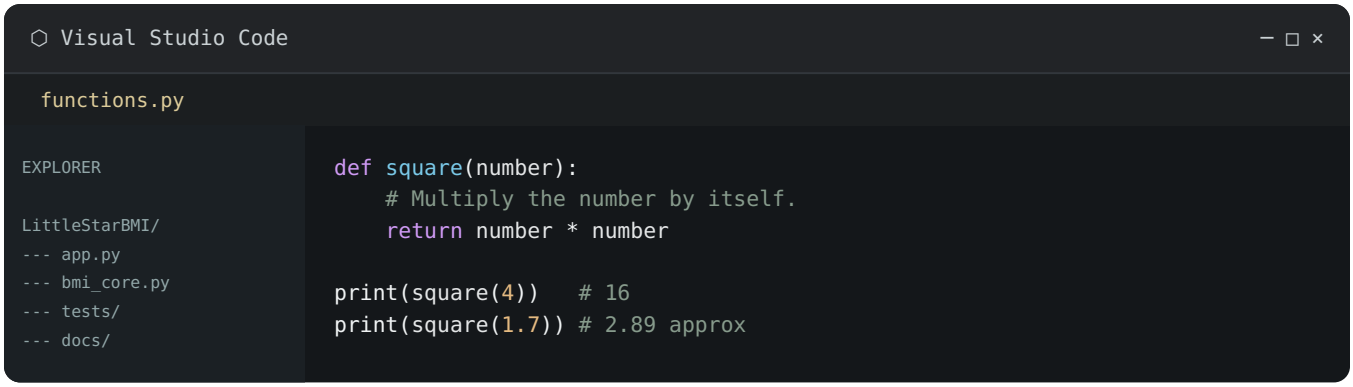
මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

Functions – නැවත භාවිතා කළ හැකි වැඩ

Function එකක් යම් කාර්යයක් සඳහා නම් කළ instructions group එකකි.

ඉගෙනගනිමු

- ✓ `def function_name(...):` මඟින් function define කරයි.
- ✓ `return` result එක වෙනත් code එකකට ලබාදෙයි.
- ✓ Functions කුඩා නම් test කිරීමත් පහසුයි.



✎ දැන් ඔබ කරන්න

Square කරන `square(n)` function එකක් සාදන්න.

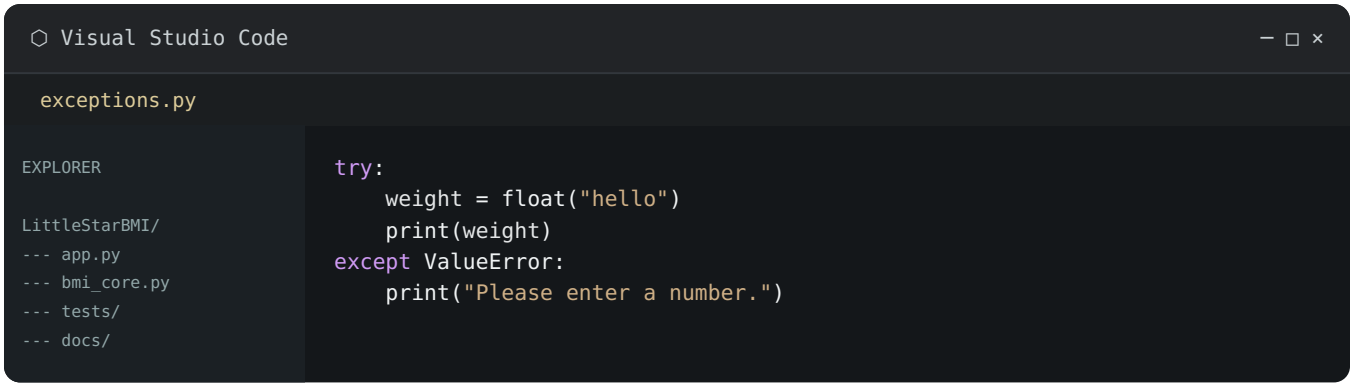
මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

Error එකක් හඳුනාගමු

Bad input එකක් දුන්නා කියලා desktop app එක crash විය යුතු නැහැ.

ඉගෙනගනිමු

- ✓ `try:` තුළ error ඇතිවිය හැකි operation කරන්න.
- ✓ `except ValueError:` තුළ user-friendly response එකක් දෙන්න.
- ✓ අපේ final program එක `InputError` exception එක භාවිතා කරනවා.



✎ දැන් ඔබ කරන්න

`float("hello")` උත්සාහ කර error එක ආරක්ෂිතව අල්ලාගන්න.

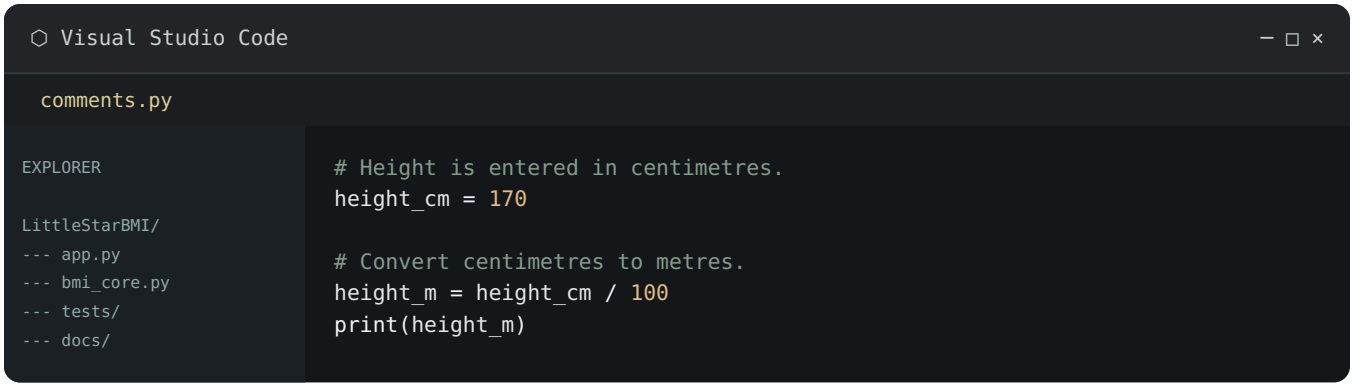
මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

Comments සහ readable names

හොඳ code එකක් කියවන කෙනෙකුට එහි අදහස තේරිය යුතුයි.

ඉගෙනගනිමු

- ✓ `#` පසු කොටස comment එකක්; Python එය execute නොකරයි.
- ✓ `weight_kg` වගේ නමක් `x` ට වඩා පැහැදිලියි.
- ✓ Magic numbers සහ duplicate code අවම කරන්න.



✎ දැන් ඔබ කරන්න

`x = 170` වෙනුවට වඩා පැහැදිලි variable name එකක් යෝජනා කරන්න.

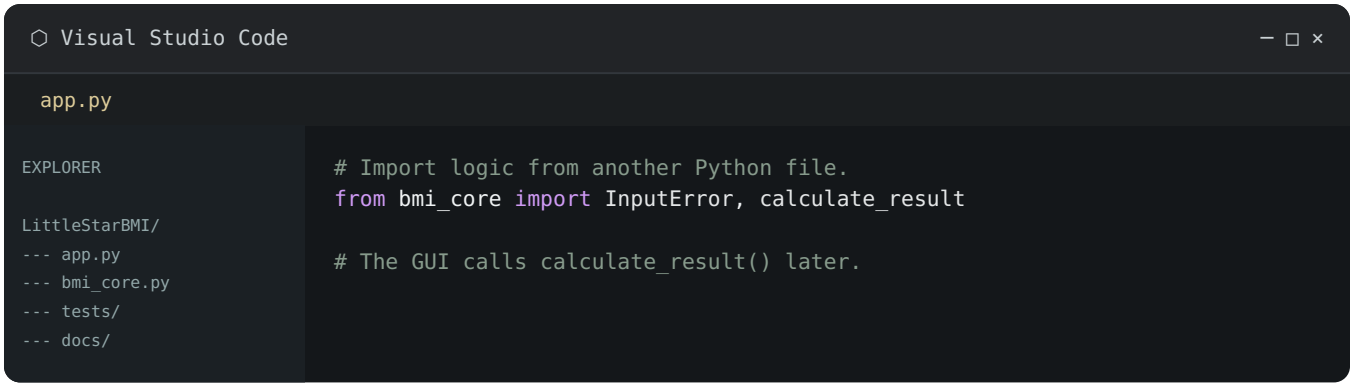
මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

Modules කියන්නේ files

Python code එක files කිහිපයකට බෙදා භාවිතා කළ හැක.

ඉගෙනගනිමු

- ✓ `bmi_core.py` හි calculate function ඇත.
- ✓ `app.py` හි `from bmi_core import ...` භාවිතා කරයි.
- ✓ මෙයින් user interface වෙනස් කළත් formula එක වෙනම test කළ හැක.



✎ දැන් ඔබ කරන්න

`app.py` file එකේ `from bmi_core` line එක source ZIP එකෙන් සොයන්න.

මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

Tkinter තිබෙනවාද බලමු

Python හි standard tkinter library එකෙන් අපට Windows-style desktop windows හදාගන්න පුළුවන්.

ඉගෙනගනිමු

- ✓ CMD එකට `py -m tkinter` ටයිප් කරන්න.
- ✓ කුඩා sample window එකක් open වුණොත් Tkinter වැඩ කරනවා.
- ✓ එය නොපෙනුණොත් Python installation සහ optional Tcl/Tk components පරීක්ෂා කරන්න.

```
Microsoft Windows [Version 2026] · Command Prompt                _ □ ×

C:\Users\Student> py -m tkinter
C:\Users\Student> py hello_window.py
```

✎ දැන් ඔබ කරන්න

Test window එක විවෘත වූ බව notebook එකේ check කරගන්න.

මතක තබාගන්න: Tkinter සඳහා මේ example එකේ external pip dependency අවශ්‍ය නැහැ.

අපේ පළමු Desktop Window

`Tk()` window එක හදනවා; `mainloop()` එය user actions සඳහා විවෘතව තබනවා.

ඉගෙනගනිමු

- ✓ නව `hello_window.py` file එක open කරන්න.
- ✓ Window title සහ geometry වෙනස් කර බලන්න.
- ✓ Label එකක text එක display වීම නිරීක්ෂණය කරන්න.

Visual Studio Code

hello_window.py

```
EXPLORER

LittleStarBMI/
--- app.py
--- bmi_core.py
--- tests/
--- docs/
```

```python
"""First Tkinter lesson: make a tiny Windows window."""
import tkinter as tk

window = tk.Tk()
window.title("My First Windows App")
window.geometry("400x220")

label = tk.Label(window, text="Hello, software developer!")
label.pack(pady=50)

window.mainloop()
```

✎ දැන් ඔබ කරන්න

Window width 400 වෙනුවට 480 කර run කරන්න.

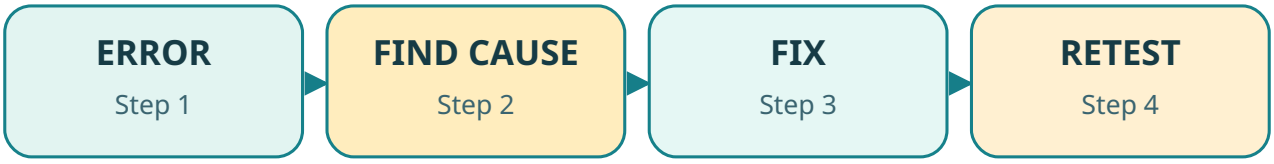
මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

Syntax error vs logic error

Error එකක් ඇතිවීමට හේතු විවිධයි; debugging ක්‍රමයක් අවශ්‍යයි.

ඉගෙනගනිමු

- ✓ Syntax error: colon හෝ bracket අමතක වීම.
- ✓ Runtime error: invalid conversion හෝ zero division.
- ✓ Logic error: program run වුණත් BMI formula එක වැරදිව තිබීම.



✎ දැන් ඔබ කරන්න

කෙටි code එකක bracket එකක් අඩු කර message එක කියවා යළි සකස් කරන්න.

මතක තබාගන්න: Bug fix එකකට පසු කලින් වැඩ කළ tests නැවත run කරන්න.

Terminal folders සමඟ වැඩ

`cd` යනු change directory; command එක නිවැරදි folder එකේ ධාවනය කරන්න.

ඉගෙනගනිමු

- ✓ VS Code Terminal → New Terminal open කරන්න.
- ✓ `dir` command එකෙන් files බලන්න.
- ✓ Project folder එකේ `py app.py` සහ `py -m unittest ...` run කරන්න.

```
Microsoft Windows [Version 2026] · Command Prompt                _ □ ×

C:\Users\Student> cd Desktop\EGOTECHWORLD_BMI_Desktop_Project_2026
C:\Users\Student> dir
C:\Users\Student> py app.py
```

✎ දැන් ඔබ කරන්න

`dir` output එකේ `bmi_core.py` තිබෙනවාද බලන්න.

මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

BMI formula එක තේරුම් ගනිමු

BMI = weight (kg) ÷ height (metres)². උස metres ලෙස ලබාගැනීම අත්‍යවශ්‍යයි.

ඉගෙනගනිමු

- ✓ Height 170 cm → 1.70 m.
- ✓ Square = 1.70 × 1.70 = 2.89.
- ✓ BMI = 70 ÷ 2.89 ≈ 24.22; display එකේ 24.2.

$$\text{BMI} = \frac{\text{Weight (kg)}}{\text{Height (m)} \times \text{Height (m)}}$$

$$70\ \text{kg} \div 1.7 \times 1.7 = \mathbf{24.2}$$

Fictional adult sample: age 25 · kg 70 · cm 170

✎ දැන් ඔබ කරන්න

65 kg සහ 165 cm සමඟ formula එක notebook එකේ ගණනය කරන්න.

මතක තබාගන්න: BMI යනු screening measure එකක්; සෞඛ්‍ය නිගමනයක් නොවේ.

Centimetres → Metres

Unit conversion එකක් වැරදි නම් මුළු result එක වැරදි වෙන්න පුළුවන්.

ඉගෙනගනිමු

- ✓ Metres = centimetres / 100.
- ✓ 170 / 100 = 1.70; 150 / 100 = 1.50.
- ✓ BMI calculation එකට cm අගය සෘජුව දමන්න එපා.



✎ දැන් ඔබ කරන්න

`height_cm = 160` දී metres value එක print කරන්න.

මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

අතින් ගණනය – example A

Fictional adult example එකකට calculation එක අතින් කර බලමු.

ඉගෙනගනිමු

- ✓ Age 25; weight 70 kg; height 170 cm.
- ✓ BMI = 70 / (1.7 × 1.7) = 24.2214...
- ✓ Display BMI = 24.2; adult range = healthy weight range.

$$\text{BMI} = \frac{\text{Weight (kg)}}{\text{Height (m)} \times \text{Height (m)}}$$

$$70\ \text{kg} \div 1.7 \times 1.7 = \mathbf{24.2}$$

Fictional adult sample: age 25 · kg 70 · cm 170

✎ දැන් ඔබ කරන්න

Calculator සහ Python දෙකෙන්ම ප්‍රතිඵලය සසඳන්න.

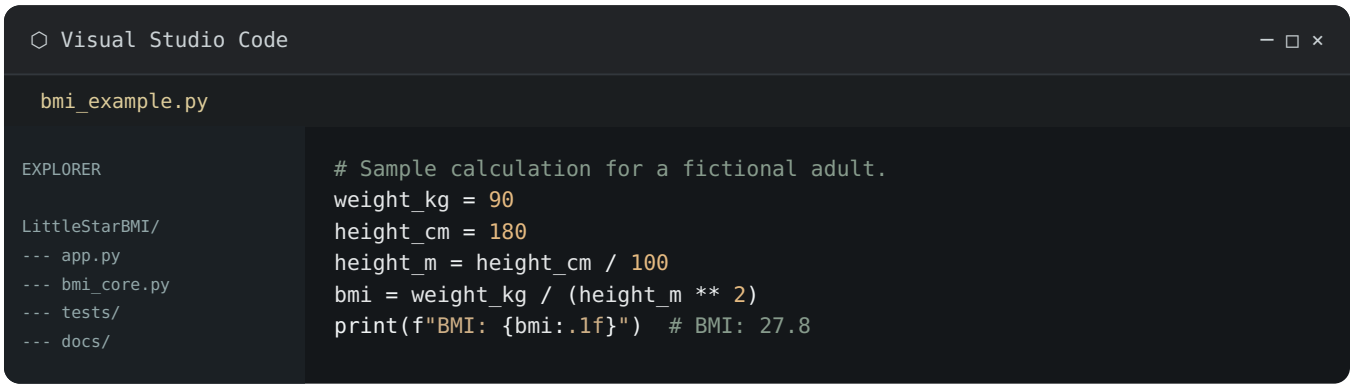
මතක තබාගන්න: Real people ගැන judgement නොකරන බව මතක තබන්න.

අතින් ගණනය – example B

වෙනත් fictional adult sample එකකට තවත් computation එකක් බලමු.

ඉගෙනගනිමු

- ✓ Age 32; weight 90 kg; height 180 cm.
- ✓ 1.8 × 1.8 = 3.24; 90 ÷ 3.24 ≈ 27.78.
- ✓ Displayed = 27.8; category determined from unrounded value.



✎ දැන් ඔබ කරන්න

`90,180` Python test එක source project එකේ සොයන්න.

මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

Rounding සිදු කරන්නේ කවදාද?

වැඩිහිටි BMI category තීරණය කරන්නේ raw calculated number එකෙන්; display සඳහා පමණක් round කරන්න.

ඉගෙනගනිමු

- ✓ BMI 24.999... නම් code එකේ `< 25` condition එක ප්‍රතිඵලය තීරණය කරයි.
- ✓ මුලින් 25.0 ලෙස round කළොත් category එක නොදැනුවත්ව වෙනස් විය හැක.
- ✓ `f"{bmi:.1f}"` යනු display formatting එකක්.



✎ දැන් ඔබ කරන්න

24.96 අගය 1 decimal දක්වා display කළ විට කුමක් පෙනෙයිද?

මතක තබාගන්න: Display precision සහ business rule එක වෙන් කිරීම හොඳ development practice එකකි.

Adult BMI screening ranges

Adults 20+ සඳහා standard ranges වල සීමා coding exercise එකට ගනිමු.

ඉගෙනගනිමු

- ✓ 18.5 ට අඩු → Underweight range.
- ✓ 18.5 සිට 25 ට අඩු → Healthy; 25 සිට 30 ට අඩු → Overweight.
- ✓ 30 හෝ ඊට වැඩි → Obesity range. මෙය diagnosis එකක් නොවේ.

Adult BMI	Code rule	Label
<18.5	bmi < 18.5	Underweight
18.5–<25	bmi < 25	Healthy range
25–<30	bmi < 30	Overweight
≥30	else	Obesity range

✎ දැන් ඔබ කරන්න

24.9, 25.0, 29.9, 30.0 සඳහා labels සොයන්න.

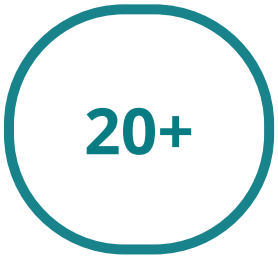
මතක තබාගන්න: මූලාශ්‍රය: CDC Adult BMI Categories. Children සඳහා මේ table එක භාවිත නොකරන්න.

දරුවන්ට වෙනම BMI ක්‍රමයක්

දරුවන්ගේ BMI number එක ගණනය කළ හැකි වුණත් adult BMI category එකෙන් තීරණය කරන්න බැහැ.

ඉගෙනගනිමු

- ✓ Ages 2–19: interpretation depends on age-and-sex-specific percentile charts.
- ✓ අපේ demo එක age <20 නම් error message දී adult label නොපෙන්වයි.
- ✓ සෞඛ්‍ය තොරතුරු, privacy සහ නිවැරදි භාවිතය coding project එකක quality requirements වෙයි.



Adult example only

For people aged 2–19, BMI-for-age percentiles are different. No diagnosis or medical advice.

✎ දැන් ඔබ කරන්න

User වයස 12 ඇතුළත් කරද්දී app එක කරන්නේ මොකක්ද කියා ලියන්න.

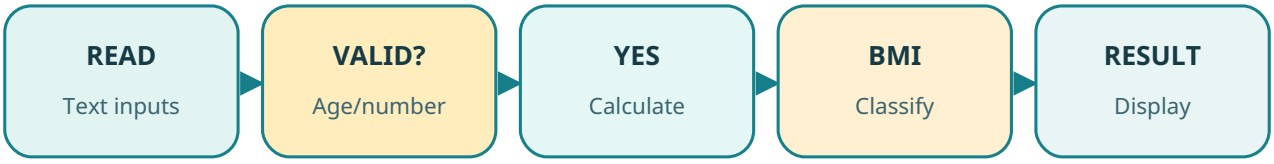
මතක තබාගන්න: CDC Child and Teen BMI Calculator යන නිල මූලාශ්‍රය නිරීක්ෂණය කරන්න.

Input validation කියන්නේ මොකක්ද?

Valid නොවන input ගණනයට පෙර නවත්වන ආරක්ෂක දොරටුව validation වේ.

ඉගෙනගනිමු

- ✓ Blank, letters, negative numbers සහ zero-height reject කළ යුතුයි.
- ✓ `nan`/`inf` වැනි special numeric values පවා reject කළ යුතුයි.
- ✓ අපේ demo ranges real clinical limits නොව beginner-friendly guardrails ය.



✎ දැන් ඔබ කරන්න

`height="0"` දුන්නොත් division කිරීමට පෙර නවත්වන්නේ ඇයි?

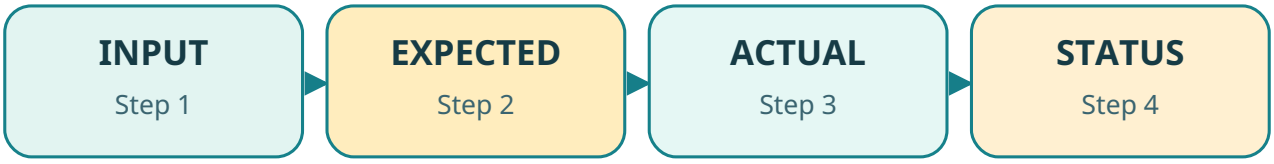
මතක තබාගන්න: Input එක trusted යැයි කිසිදා අනුමාන නොකරන්න.

Boundary Value Analysis

Rules වල 18.5, 25, 30 වැනි සීමාවල bugs වැඩිපුර ඇති විය හැක.

ඉගෙනගනිමු

- ✓ 18.4999 ↔ 18.5 වෙනස test කරන්න.
- ✓ 24.9999 ↔ 25.0 වෙනස test කරන්න.
- ✓ 29.9999 ↔ 30.0 වෙනස test කරන්න.



✎ **දැන් ඔබ කරන්න**

"Just below / on / just above" pattern එක වෙනත් number rule එකකට යොදන්න.

මතක තබාගන්න: Boundary tests යනු random tests වලට වඩා වැදගත් cases පෙන්වන ක්‍රමයකි.

`read_age()` function එක

Adult-only rule එක පළමුවම validate කරයි.

ඉගෙනගනිමු

- ✓ `.strip()` මඟින් unwanted spaces ඉවත් කරයි.
- ✓ `.isdecimal()` මඟින් whole-number digits පරීක්ෂා කරයි.
- ✓ අවුරුදු 20ට අඩු නම් helpful child-specific message එකක් ලබාදෙයි.

⬡ Visual Studio Code — □ ×

bmi_core.py (concept excerpt)

EXPLORER

LittleStarBMI/
--- app.py
--- bmi_core.py
--- tests/
--- docs/

```python
def read_age(text: str) -> int:
    text = text.strip()
    if not text.isdecimal():
        raise InputError("Enter a whole-number age.")
    age = int(text)
    if age < 20:
        raise InputError("Adult BMI categories only.")
    if age > 120:
        raise InputError("Age must be 20 to 120.")
    return age
```

✎ දැන් ඔබ කරන්න

Age=19 හා Age=20 values දෙකේ වෙනස පරීක්ෂා කරන්න.

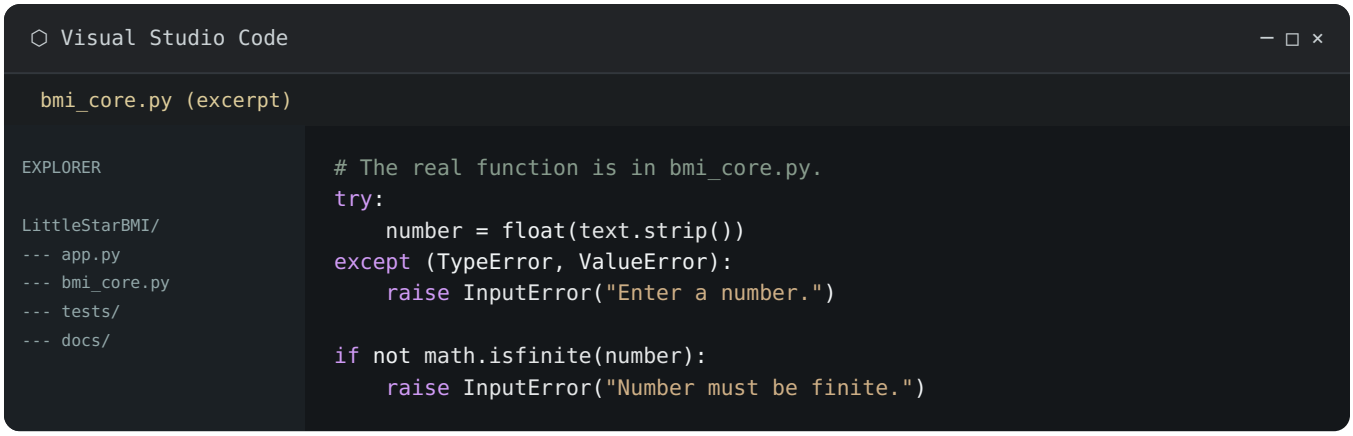
මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

`read_measurement()` function එක

Text එක safe float එකක් බවට පරිවර්තනය කර range එකත් පරීක්ෂා කරයි.

ඉගෙනගනිමු

- ✓ Try/except මඟින් invalid words අල්ලා ගනී.
- ✓ `math.isfinite()` මඟින් NaN සහ Infinity reject කරයි.
- ✓ Min/Max values නිවැරදි නම් float number return කරයි.



✎ දැන් ඔබ කරන්න

weight="banana" සහ height="250" ඇතුළත් කර output අනුමාන කරන්න.

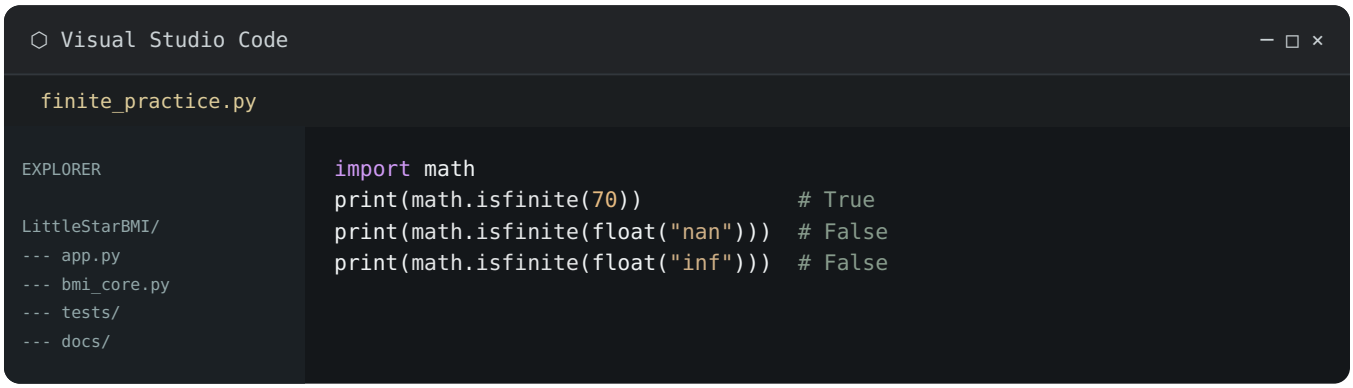
මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

Why `isfinite()`?

Python සමහර විශේෂ අකුරු සංඛ්‍යා values ලෙස ගනී; ඒවා අපේ app එකට සුදුසු නැහැ.

ඉගෙනගනිමු

- ✓ `float("nan")` යනු “not a number” විශේෂ float එකක්.
- ✓ `float("inf")` යනු infinity value එකකි.
- ✓ Valid height/weight සඳහා finite numbers පමණක් භාවිතා කරමු.



✎ දැන් ඔබ කරන්න

`math.isfinite(70)` සහ `math.isfinite(float("nan"))` සසඳන්න.

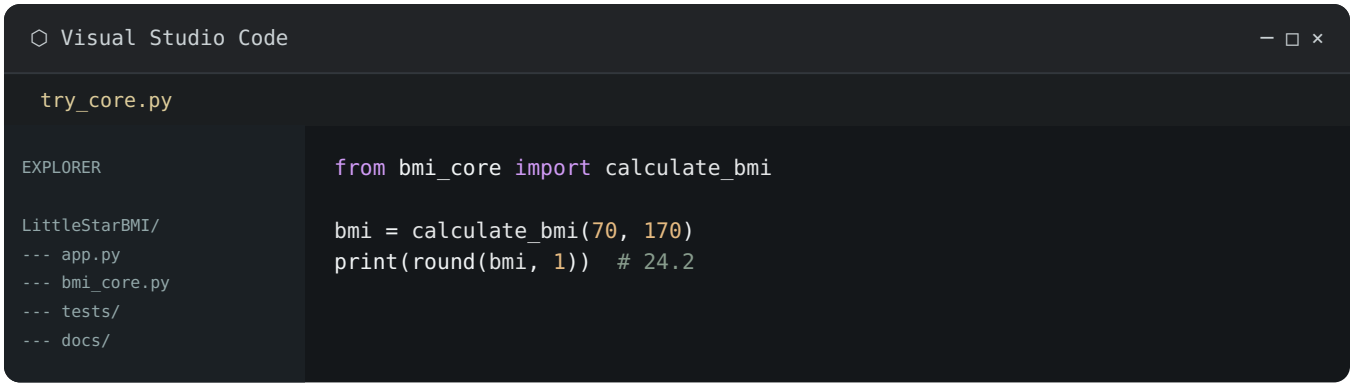
මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

`calculate_bmi()` pure function

Pure function එකක් same inputs සඳහා same output ලබාදීම නිසා testing පහසු වේ.

ඉගෙනගනිමු

- ✓ Weight සහ height ලබාගනී; no GUI actions.
- ✓ cm → m කර square කර divide කරයි.
- ✓ Result float ලෙස return කරයි; print හෝ file save නොකරයි.



✎ දැන් ඔබ කරන්න

`calculate_bmi(70,170)` output එක Python CMD එකෙන් පරීක්ෂා කරන්න.

මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

`adult_bmi_category()`

BMI value අනුව range එක තෝරන්නේ මේ function එකයි.

ඉගෙනගනිමු

- ✓ `if bmi < 18.5` නම් පළමු category එක.
- ✓ ඊළඟ `if < 25`, `if < 30` නිවැරදි order එකෙන් තිබිය යුතුයි.
- ✓ ඉතිරි values ≥30 සඳහා final `return` භාවිතා කරයි.

Visual Studio Code

bmi_core.py (logic excerpt)

EXPLORER

LittleStarBMI/
--- app.py
--- bmi_core.py
--- tests/
--- docs/

```python
def adult_bmi_category(bmi: float) -> str:
    if bmi < 18.5:
        return "Underweight range"
    if bmi < 25:
        return "Healthy weight range"
    if bmi < 30:
        return "Overweight range"
    return "Obesity range"
```

✎ දැන් ඔබ කරන්න

Code එකේ conditions order එක හරවා දැම්මොත් සිදුවිය හැක්කේ මොකක්ද?

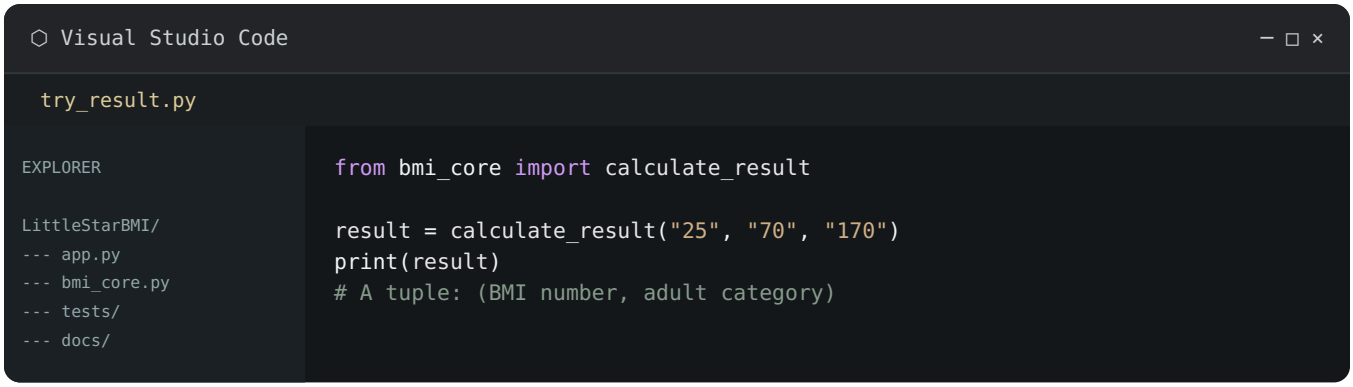
මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

`calculate_result()`

Functions කුඩා කොටස් ලෙස සකසා එකට සම්බන්ධ කළාම app එක පැහැදිලි වෙයි.

ඉගෙනගනිමු

- ✓ Age read/validate → weight validate → height validate.
- ✓ BMI calculate → adult range classify.
- ✓ GUI එකට `(bmi, category)` දෙක return කරයි.



✎ දැන් ඔබ කරන්න

`calculate_result("25","70","170")` අත්හදා බලන්න.

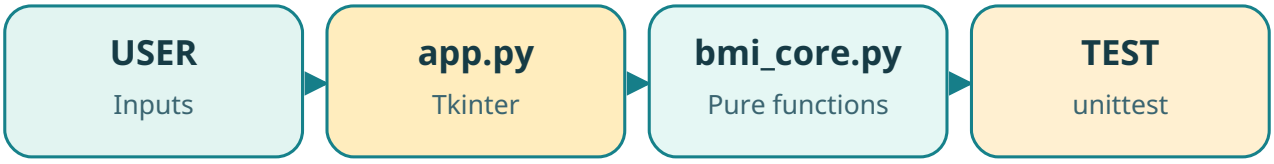
මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

BMI code structure – review

හොඳ developer කෙනෙකු code එකේ responsibility එක තනි තැනක තබයි.

ඉගෙනගනිමු

- ✓ `read_age()` = age gate.
- ✓ `read_measurement()` = number checks.
- ✓ `calculate_bmi()` = formula; `adult_bmi_category()` = labels.



✎ **දැන් ඔබ කරන්න**

Function නම් සහ ඒවායේ වගකීම diagram එකක සම්බන්ධ කරන්න.

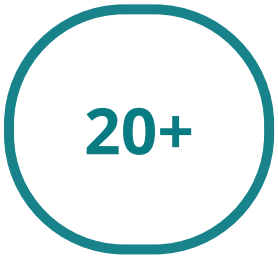
මතක තබාගන්න: Business logic වෙනම තිබීමෙන් Tkinter window එක open නොකර tests run කළ හැක.

No data saved – privacy by design

පරිශීලකයාගේ health data එකක් තිබෙන project එකක privacy එක අමතර feature එකක් නොවේ.

ඉගෙනගනිමු

- ✓ අපේ example app එක input values network එකට යවන්නේ නැහැ.
- ✓ Database හෝ save-history feature නැති නිසා personal records ගොඩනැගෙන්නේ නැහැ.
- ✓ Learners fictional numbers භාවිතා කරන්න; screenshots share කිරීමට පෙර data බලන්න.



Adult example only

For people aged 2–19, BMI-for-age percentiles are different. No diagnosis or medical advice.

✎ දැන් ඔබ කරන්න

ඔබේ app එකේ privacy statement එකක් වාක්‍යයකින් ලියන්න.

මතක තබාගන්න: Privacy by design කියන්නේ යම් අවශ්‍යතාවක් නැත්නම් data එක රැස් නොකිරීමයි.

Unit tests run කිරීමට පෙර

Code එකේ maths functions තිබෙන බවත් import කළ හැකි බවත් පරීක්ෂා කරමු.

ඉගෙනගනිමු

- ✓ CMD එකෙන් project folder එක විවෘත කරන්න.
- ✓ `py -m unittest discover -s tests -v` run කරන්න.
- ✓ “OK” message එකක් ආවොත් එම automated test suite එක සමත් බව පෙනෙයි.

```
Microsoft Windows [Version 2026] · Command Prompt                _ □ ×

C:\Users\Student> py -m unittest discover -s tests -v
C:\Users\Student> REM Expect 19 automated tests, then OK.
```

✎ දැන් ඔබ කරන්න

Command එක run කර සියලු tests සමත්ද බලන්න.

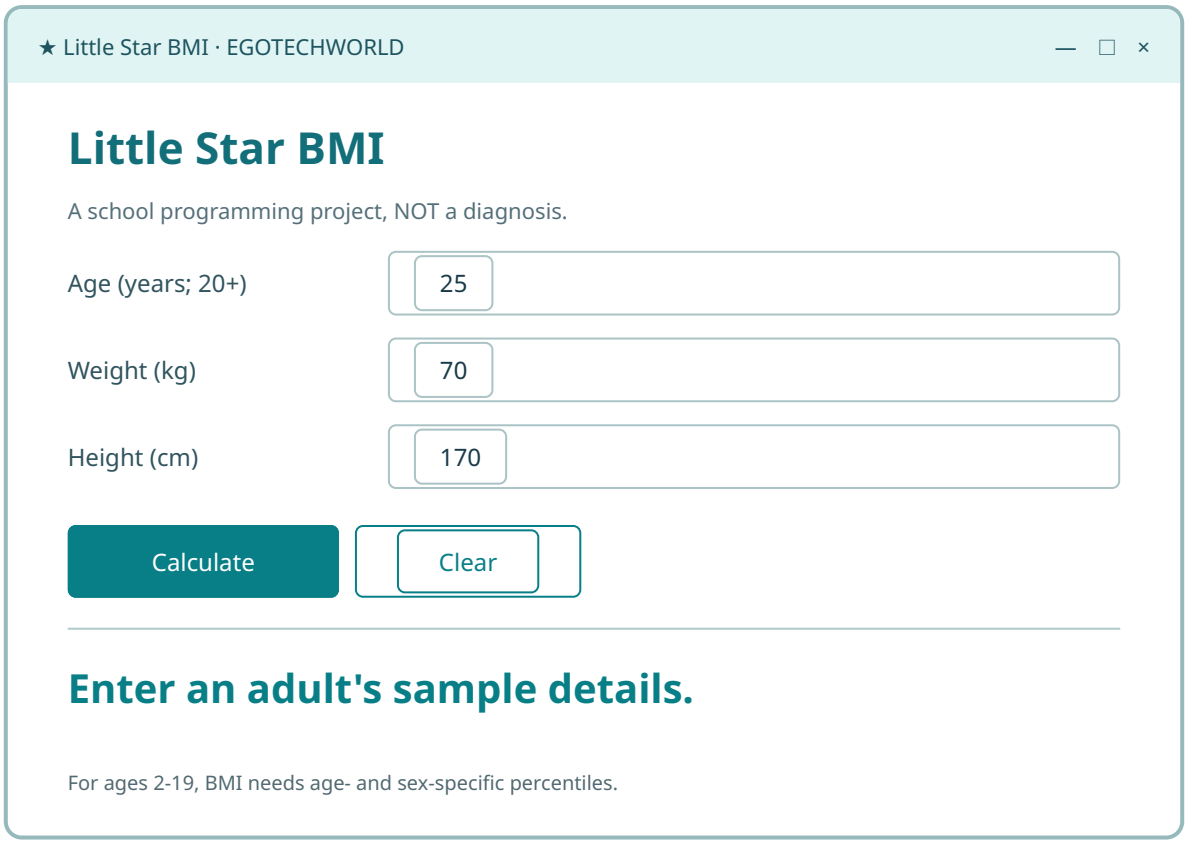
මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

UI sketch එක නැවත බලමු

දැන් requirements සහ code දෙක එකතු කර user ට පෙනෙන window එක හදමු.

ඉගෙනගනිමු

- ✓ Title → fields → buttons → result → education note යන order එක තබමු.
- ✓ Unit labels (kg/cm) සහ age 20+ warning මග නොහැරන්න.
- ✓ Focus, spacing සහ readable typography ගැන සලකා බලමු.



Illustrative Windows desktop preview · exact system theme may vary

✎ දැන් ඔබ කරන්න

UI sketch එකේ keyboard focus යන පිළිවෙළ අංකනය කරන්න.

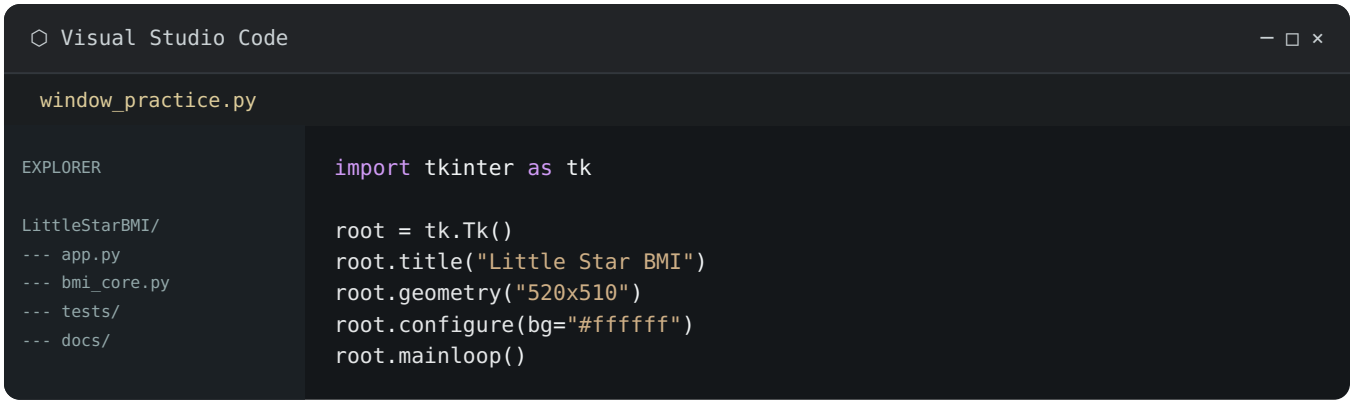
මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

Tkinter `Tk()` window

Tkinter හි root window එක app එකේ ප්‍රධාන තිරයයි.

ඉගෙනගනිමු

- ✓ `tk.Tk()` එක root object එකක් සාදයි.
- ✓ `root.title()` title bar එකට නම යොදයි.
- ✓ `root.geometry()` initial width × height ගණනය කරයි.



✎ දැන් ඔබ කරන්න

`root.geometry("520x510")` හි අගයන් වෙනස් කර බලන්න.

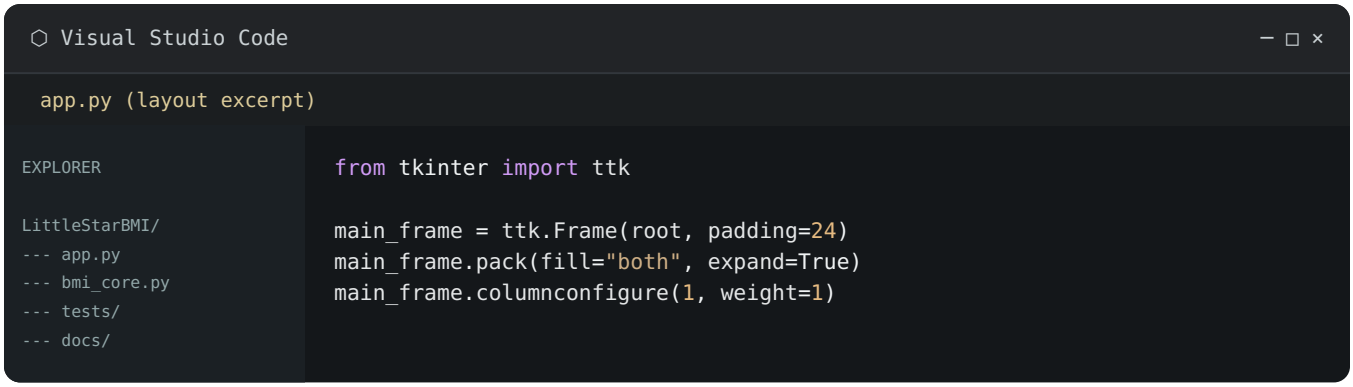
මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

Frame – UI කණ්ඩායම්

Frame එකක් screen එකේ widgets එකට තබන container එකකි.

ඉගෙනගනිමු

- ✓ `ttk.Frame()` එක root window තුළ සාදන්න.
- ✓ `padding` මඟින් widgets screen අසලම නොගැසෙන සේ දුර තබයි.
- ✓ `columnconfigure` මඟින් input column එක stretch වීමට ඉඩ දෙයි.



✎ දැන් ඔබ කරන්න

Frame padding 24 වෙනුවට 12 කර වෙනස බලන්න.

මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

Label – පැහැදිලි වචන

Label එක user ට කියවීමට text පෙන්වයි.

ඉගෙනගනිමු

- ✓ “Age (years; 20+)” කියන label එකේ name සහ unit දෙකම ඇත.
- ✓ “Weight (kg)” සහ “Height (cm)” labels වල ඒකක වෙනස් වෙයි.
- ✓ Result label එකට large font එකක් භාවිතා කරමු.

Visual Studio Code

app.py (label excerpt)

```
EXPLORER

LittleStarBMI/
--- app.py
--- bmi_core.py
--- tests/
--- docs/
```

```
ttk.Label(
    main_frame,
    text="Height (cm)",
    style="App.TLabel"
).grid(row=4, column=0, sticky="w")
```

✎ දැන් ඔබ කරන්න

“Height” පමණක් කියන label එකෙන් ඇතිවන අවුල කියන්න.

මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

Entry – value එක ටයිප් කරන්න

`ttk.Entry()` text input box එක සාදයි.

ඉගෙනගනිමු

- ✓ `tk.StringVar()` තනි field එකක text තබයි.
- ✓ `textvariable` හරහා entry එක variable සමඟ සම්බන්ධ වේ.
- ✓ `.get()` මඟින් button click අවස්ථාවේ typed text කියවයි.

Visual Studio Code

app.py (entry excerpt)

```
EXPLORER

LittleStarBMI/
--- app.py
--- bmi_core.py
--- tests/
--- docs/
```

```python
weight_var = tk.StringVar(value="70")
entry = ttk.Entry(
    main_frame,
    textvariable=weight_var,
    width=20
)
entry.grid(row=3, column=1, sticky="ew")
```

✎ දැන් ඔබ කරන්න

`weight_var.get()` ආපසු ලබාදෙන්නේ text ද number ද?

මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

Grid layout

Grid එක screen එක rows සහ columns ලෙස සකස් කරයි.

ඉගෙනගනිමු

- ✓ Row 2 → age; row 3 → weight; row 4 → height.
- ✓ Column 0 → label; column 1 → entry.
- ✓ `sticky="ew"` input box එක horizontally expand කිරීමට ඉඩ දෙයි.

ROW 2	ROW 3	ROW 4
Age	**Weight**	**Height**

✎ දැන් ඔබ කරන්න

Row=3 Column=1 හි screen එකේ තිබිය යුතු widget එක නම් කරන්න.

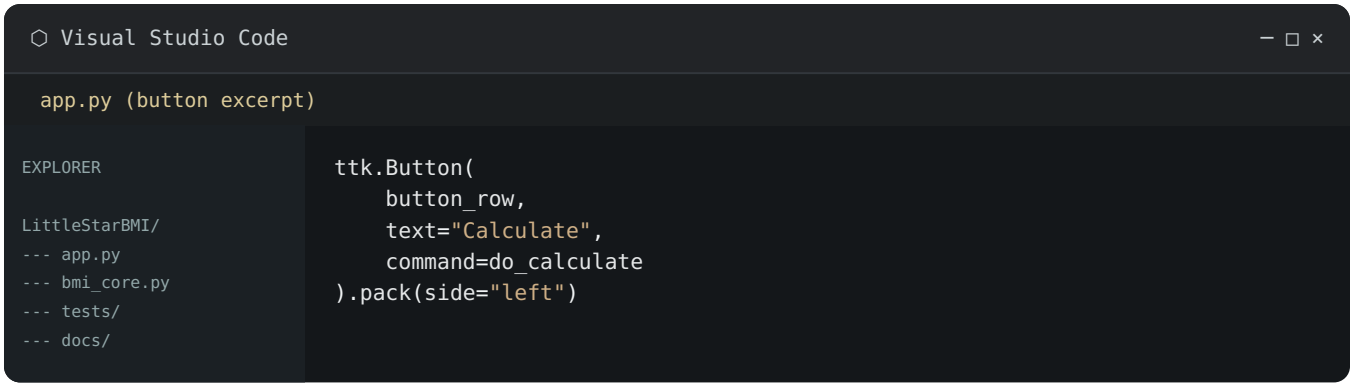
මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

Button එකක කාර්යය

User click එකකට Python function එකක් connect කරන විදිහ බලමු.

ඉගෙනගනිමු

- ✓ `command=do_calculate` function reference එක button එකට දෙයි.
- ✓ Function එක නමෙන් පසුව `()` නොදෙන්න; click අවස්ථාවේ run විය යුතුයි.
- ✓ Clear button එකට වෙනම callback එකක් දෙන්න.



✎ දැන් ඔබ කරන්න

`command=do_calculate()` කියලා දුන්නොත් විශේෂත්වය කුමක්ද?

මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

Event-driven programming

Desktop program එක user action එකක් එන තෙක් සූදානම්ව ඉඳියි.

ඉගෙනගනිමු

- ✓ Click Calculate → callback run.
- ✓ Press Enter → keyboard event run.
- ✓ Click Clear → reset callback run.



✎ දැන් ඔබ කරන්න

මවුස් click සහ keyboard Enter දෙකම same action වෙත යන diagram එකක් ඇඳන්න.

මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

Calculate callback එක

Business logic සහ UI එක මෙතැන සම්බන්ධ වේ.

ඉගෙනගනිමු

- ✓ `StringVar.get()` මඟින් values ලබාගනී.
- ✓ `calculate_result()` function එකට values යවයි.
- ✓ සාර්ථක නම් result label set කරයි; error නම් dialog එකක් පෙන්වයි.

Visual Studio Code

app.py (callback excerpt)

```
EXPLORER

LittleStarBMI/
--- app.py
--- bmi_core.py
--- tests/
--- docs/
```

```python
try:
    bmi, category = calculate_result(
        age_var.get(), weight_var.get(), height_var.get()
    )
except InputError as error:
    messagebox.showerror("Input error", str(error))
    return
result_var.set(f"Your sample BMI: {bmi:.1f}")
```

✎ දැන් ඔබ කරන්න

Callback එකේදී raw formula නැවත ලියන්නේ නැත්තේ ඇයි?

මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

Error dialog එක

User ඇතුළත් කළ අගයක් වැරදි නම් නිවැරදි කරගන්න උදව් කරන message එකක් පෙන්වන්න.

ඉගෙනගනිමු

- ✓ `messagebox.showerror()` popup dialog එකක් විවෘත කරයි.
- ✓ Technical traceback එක end user ට පෙන්වීම වෙනුවට friendly message ලබා දෙයි.
- ✓ Invalid input එකකට අවසානයේ incorrect old result එක නොතැබීම වැදගත්.



✎ දැන් ඔබ කරන්න

Weight field එකට “abc” දී මොන error එක පෙන්විය යුතුද?

මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

Result label update

Result එක window එකේම update කිරීම user ට පහසුයි.

ඉගෙනගනිමු

- ✓ `result_var.set(...)` ඉහළ BMI number label එක වෙනස් කරයි.
- ✓ `category_var.set(...)` අදාළ adult range එක පෙන්වයි.
- ✓ Formatting: number එක decimal 1කට පමණක් display කරයි.



✎ දැන් ඔබ කරන්න

`f"{24.2214:.1f}"` print කර result එක සටහන් කරන්න.

මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

Clear button එක

Clear කියන්නේ screen එකේ input සහ result එක ආපසු හිස් තත්ත්වයට පත් කිරීමයි.

ඉගෙනගනිමු

- ✓ Age, weight, height StringVar values `""` බවට පත් කරන්න.
- ✓ Result message එක default එකට ගෙනයන්න.
- ✓ Keyboard focus නැවත පළමු field එකට යවන්න.

Visual Studio Code

app.py (clear excerpt)

```
EXPLORER

LittleStarBMI/
--- app.py
--- bmi_core.py
--- tests/
--- docs/
```

```python
def do_clear() -> None:
    age_var.set("")
    weight_var.set("")
    height_var.set("")
    result_var.set("Enter an adult's sample details.")
    category_var.set("")
```

✎ දැන් ඔබ කරන්න

Clear click කළ පසු screenshot එකක පෙනිය යුතු state එක විස්තර කරන්න.

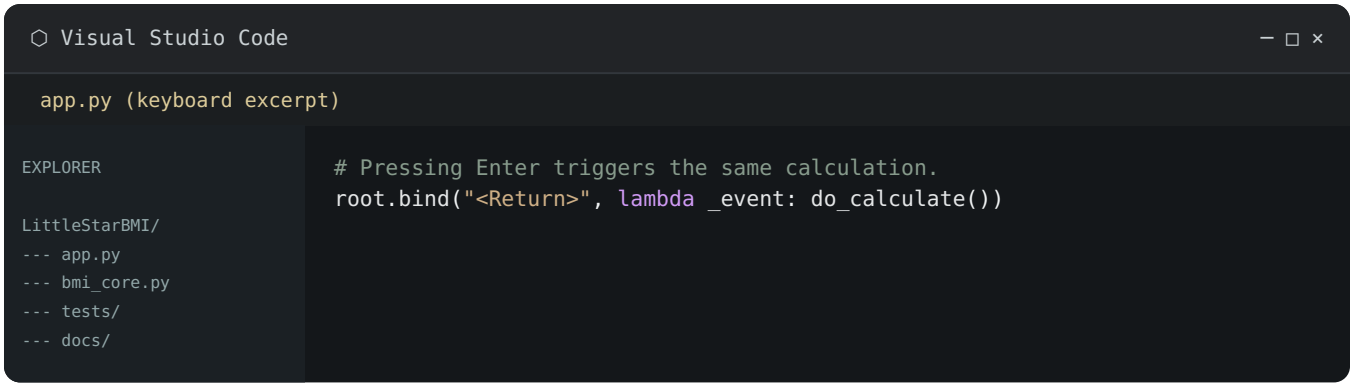
මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

Keyboard accessibility

Keyboard පමණක් භාවිතා කරන අයටත් app එකේ වැඩ කිරීමට පහසු විය යුතුයි.

ඉගෙනගනිමු

- ✓ `Tab` key එකෙන් next input/button වෙත යා හැකි විය යුතුයි.
- ✓ `Enter` key එකෙන් calculate කිරීම optional shortcut එකකි.
- ✓ Labels, contrast සහ focus indicator පරීක්ෂා කරන්න.



✎ දැන් ඔබ කරන්න

Mouse එක භාවිතා නොකර එක් result එකක් ලබාගන්න.

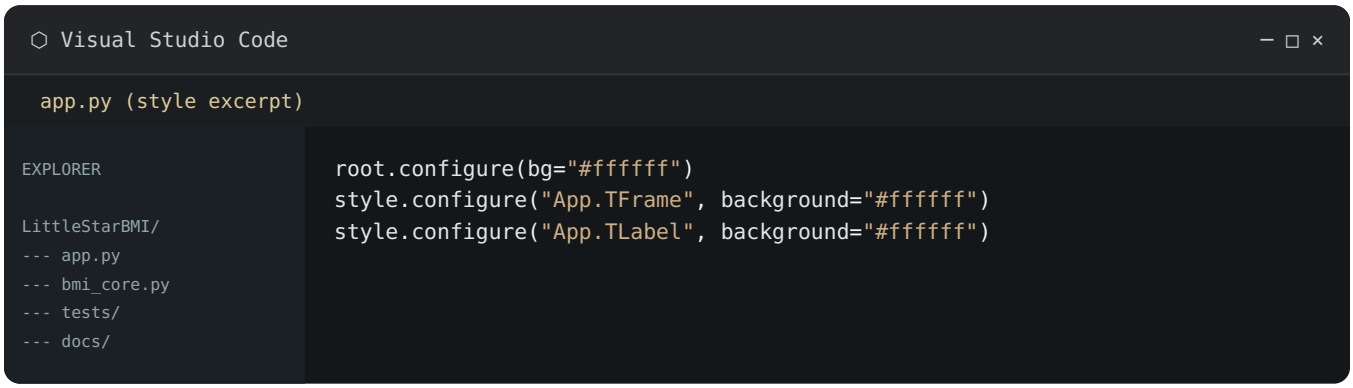
මතක තබාගන්න: Accessibility කියන්නේ “අමතර ලස්සන” නොව භාවිතා කිරීමට අයිතිය සහ පහසුවකි.

Window style සහ colours

User ඉල්ලූ clean interface එකක් සඳහා white background සහ පැහැදිලි අකුරු භාවිතා කරමු.

ඉගෙනගනිමු

- ✓ `root.configure(bg="#ffffff")` = pure white background.
- ✓ Title සහ result තරමක් වෙනස් කළත් input text පැහැදිලි විය යුතුයි.
- ✓ අධික decorations නැතිව spacing භාවිතා කරන්න.



✎ දැන් ඔබ කරන්න

අපේ PDF පිටු සුදු තැබීම සහ app window සුදු තැබීම අතර සමාන design principle එක කුමක්ද?

මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

Final app එක open කරමු

දැන් ලබාදුන් ZIP එක extract කර complete `app.py` run කරන්න.

ඉගෙනගනිමු

- ✓ ZIP එක extract කර project folder එක VS Code හි Open Folder කරන්න.
- ✓ `py app.py` කියා CMD එකෙන් run කරන්න.
- ✓ Window එකේ sample values පරීක්ෂා කර Calculate click කරන්න.

```
Microsoft Windows [Version 2026] · Command Prompt                    _ □ ×

C:\Users\Student> cd EGOTECHWORLD_BMI_Desktop_Project_2026
C:\Users\Student> py app.py
```

✎ දැන් ඔබ කරන්න

මතු වන app window එක notebook එකේ sketch එක සමඟ සසඳන්න.

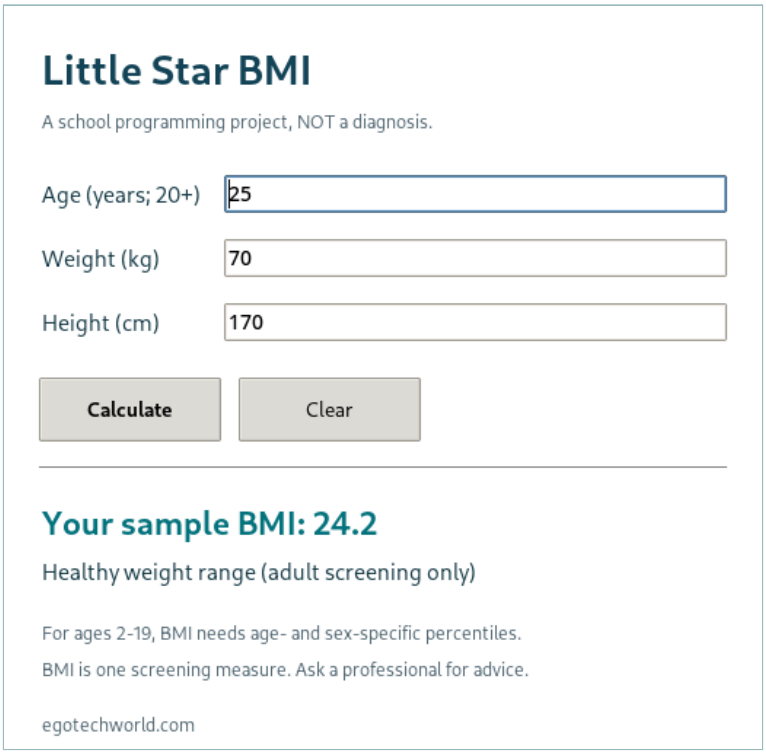
මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

Working desktop preview

මෙය වැඩසටහනේ නියැදි Windows window preview එකක්. ඔබේ Windows theme අනුව සුළු වෙනස්කම් තිබිය හැක.

ඉගෙනගනිමු

- ✓ Sample age=25, weight=70 kg, height=170 cm.
- ✓ Calculate click කළ පසු expected BMI 24.2.
- ✓ Adult range label එක සමඟ health disclaimer එකත් පෙනිය යුතුයි.



Actual application window captured in a virtual desktop test; Windows theme may differ.

✎ දැන් ඔබ කරන්න

Actual app එක run කර UI preview එකට සාපේක්ෂව වෙනස්කම් සටහන් කරන්න.

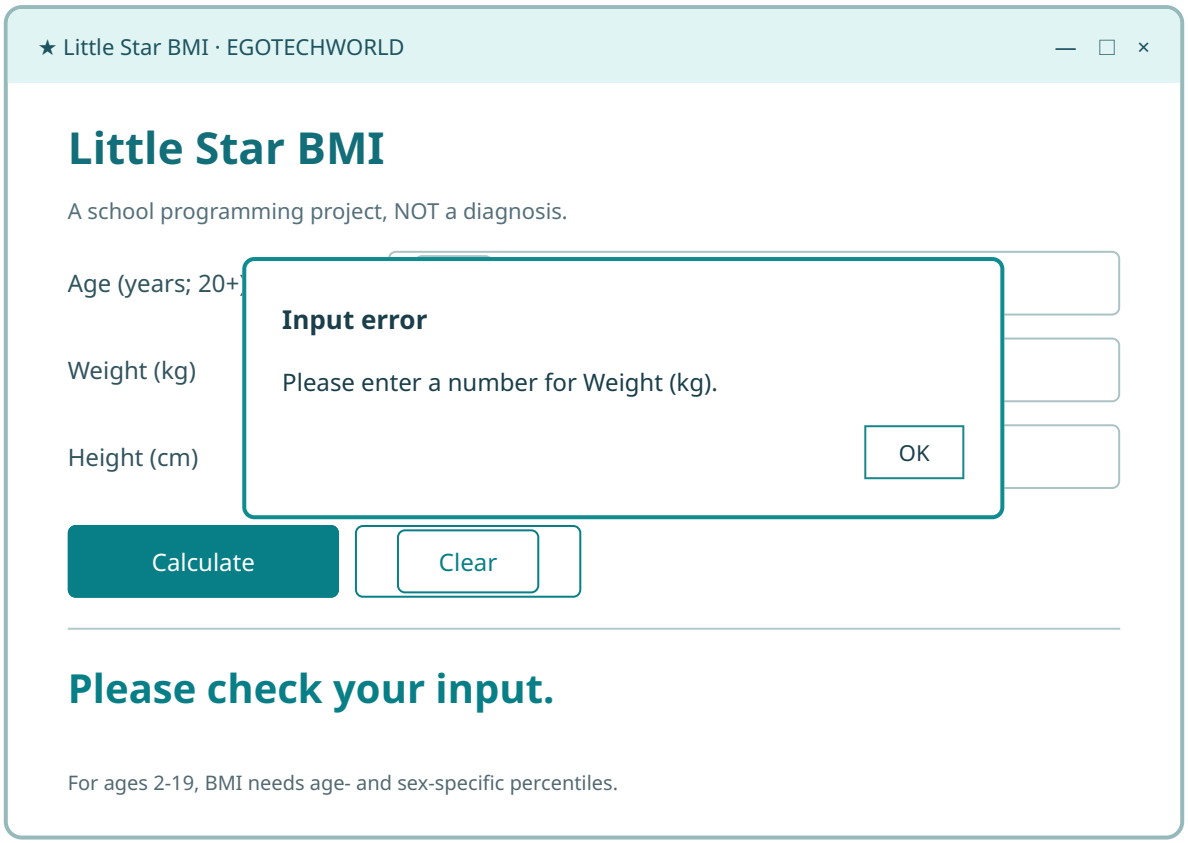
මතක තබාගන්න: පහත රූපය illustrative mockup එකක්; ඔබේ real app එක window ලෙස විවෘත වේ.

Error preview – letters

App එක crash නොවී friendly error එකක් පෙන්වීම හොඳ UX වේ.

ඉගෙනගනිමු

- ✓ Weight field එකට `abc` ලියන්න.
- ✓ Calculate click කරන්න.
- ✓ Error dialog පෙනී පරණ result clear වන බව බලන්න.



Illustrative Windows desktop preview · exact system theme may vary

✎ දැන් ඔබ කරන්න

Error message එක user ට input සකස් කරගැනීමට ප්‍රමාණවත්ද?

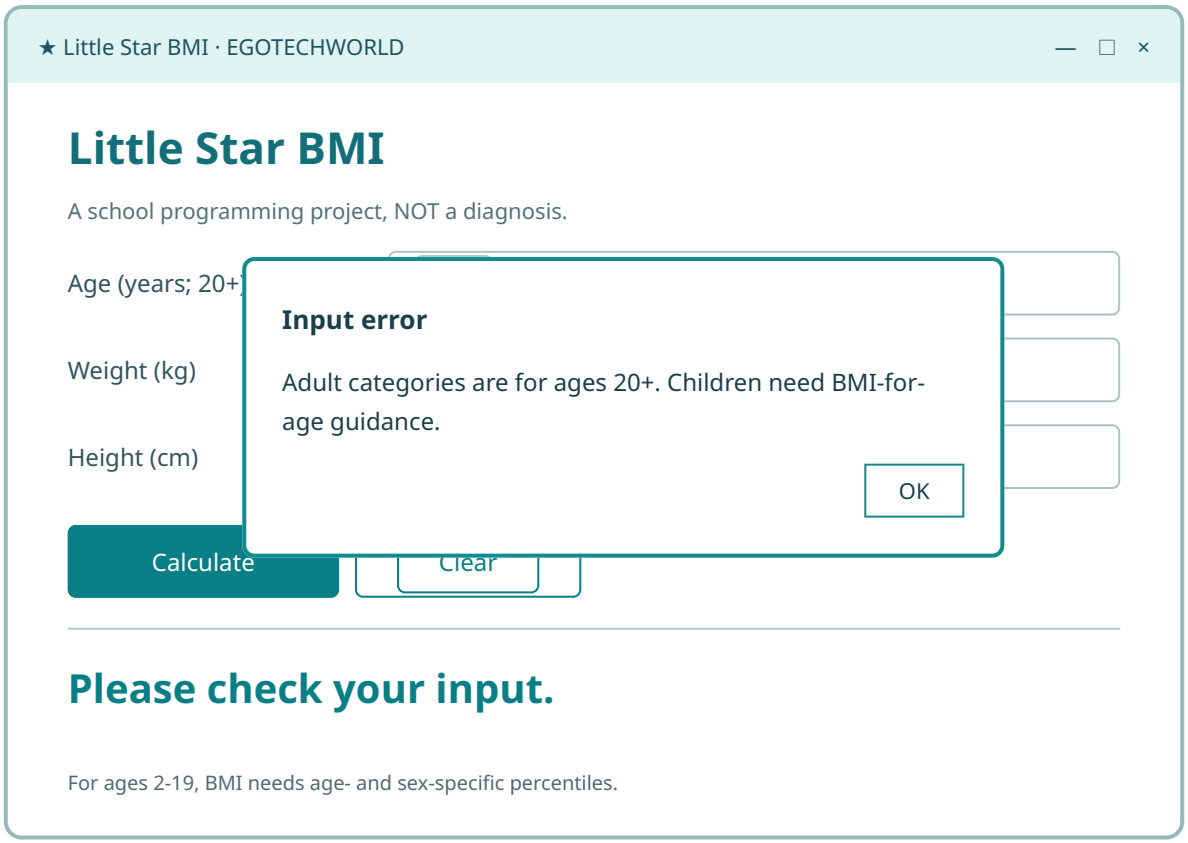
මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

Age 10 – safe handling

මේ පොත කියවන දරුවන්ට වැදගත් test එකක්.

ඉගෙනගනිමු

- ✓ Age = 10, Weight = 35, Height = 140 යන fictional values දමන්න.
- ✓ Calculate click කළ විට adult category එකක් නොපෙන්විය යුතුයි.
- ✓ Children need BMI-for-age guidance කියන message එක ලැබිය යුතුයි.



Illustrative Windows desktop preview · exact system theme may vary

✎ දැන් ඔබ කරන්න

මේ behaviour එකට SRS requirement ID එකක් යෝජනා කරන්න.

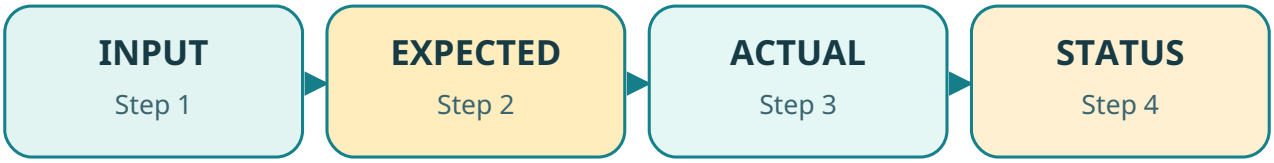
මතක තබාගන්න: මෙම guard එක learning demo එකේ safe interpretation සඳහා ඇත.

Testing කියන්නේ මොකක්ද?

Software එක කලින් සැලසුම් කළ ආකාරයට වැඩ කරනවාද කියලා evidence සමඟ පරීක්ෂා කිරීම testing වේ.

ඉගෙනගනිමු

- ✓ Test case එකට input, expected result, actual result සහ status ලියන්න.
- ✓ Happy path සමඟ invalid path ද පරීක්ෂා කරන්න.
- ✓ Testing එක ගැටලු නොමැති බව 100% ඔප්පු නොකරයි; වැරදි සොයාගැනීමට උදව් කරයි.



✎ දැන් ඔබ කරන්න

Sample TC-01 එකේ expected result ලියන්න.

මතක තබාගන්න: QA process එක, testing exercise එක සහ debugging fixing එක වෙන වෙනම සංකල්පයි.

QA vs QC vs Testing

කොටස් එකිනෙකට සම්බන්ධ නමුත් අර්ථය එකම නොවේ.

ඉගෙනගනිමු

- ✓ QA: හොඳ ක්‍රියාවලියක් පිළිපදින්න planning, reviews, standards යොදයි.
- ✓ QC: deliverable එක අවශ්‍ය තත්ත්වයේද කියලා පරීක්ෂා කරයි.
- ✓ Testing: software behaviour එකට tests ක්‍රියාත්මක කරන ක්‍රියාකාරකමකි.



✎ දැන් ඔබ කරන්න

“Code review” QA/QC දෙකටම සහාය විය හැක්කේ කොහොමද?

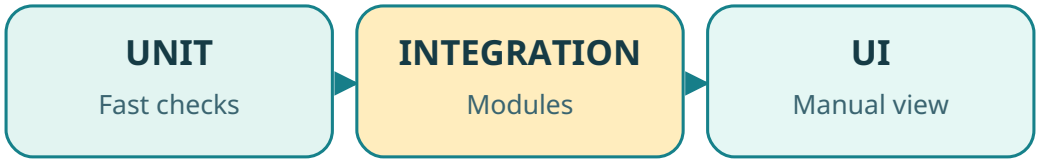
මතක තබාගන්න: Quality is a team responsibility; tester කෙනෙකුගේ වගකීම පමණක් නොවේ.

Testing pyramid

නැවත නැවත වේගයෙන් දුවන small unit tests වලින් ආරම්භ කරන්න.

ඉගෙනගනිමු

- ✓ Unit tests: function එක වෙනම පරීක්ෂා කරයි.
- ✓ Integration tests: functions/modules අතර සම්බන්ධය පරීක්ෂා කරයි.
- ✓ UI manual tests: user වගේ button සහ windows අත්හදා බලයි.



✎ දැන් ඔබ කරන්න

BMI formula එකට unit test එකක් ලියාපදිංචි කරන්න.

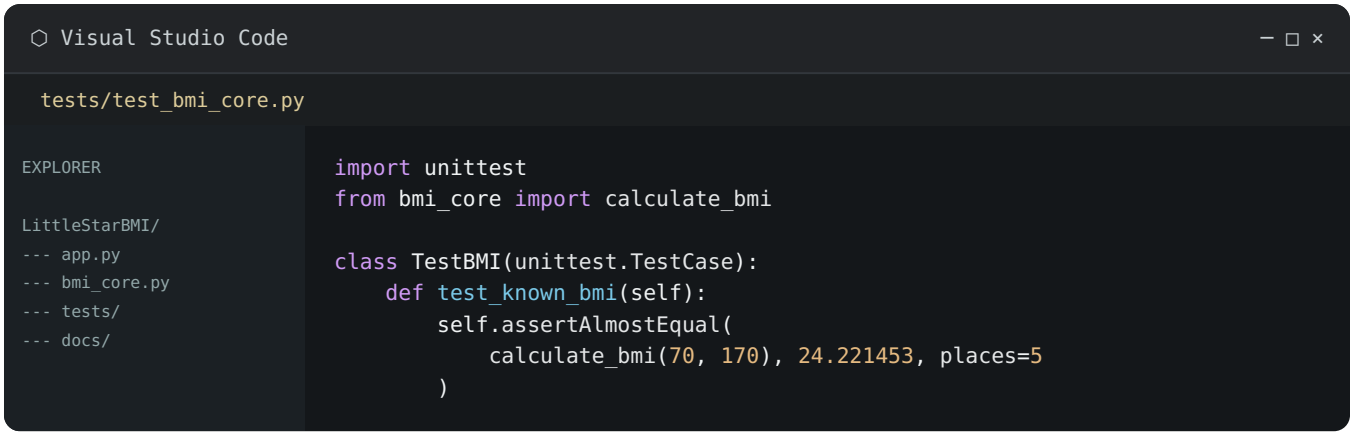
මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

Unittest කියන්නේ මොකක්ද?

Python `unittest` module එක standard library තුළ ඇත.

ඉගෙනගනිමු

- ✓ `class TestBMI(unittest.TestCase)` යන test class එක සාදයි.
- ✓ `self.assertAlmostEqual()` අඩු decimal error tolerance සමඟ maths compare කරයි.
- ✓ `assertRaises()` මඟින් invalid input නිවැරදිව reject කරනවාද බලයි.



✎ දැන් ඔබ කරන්න

`tests/test_bmi_core.py` හි known BMI test එක සොයාගන්න.

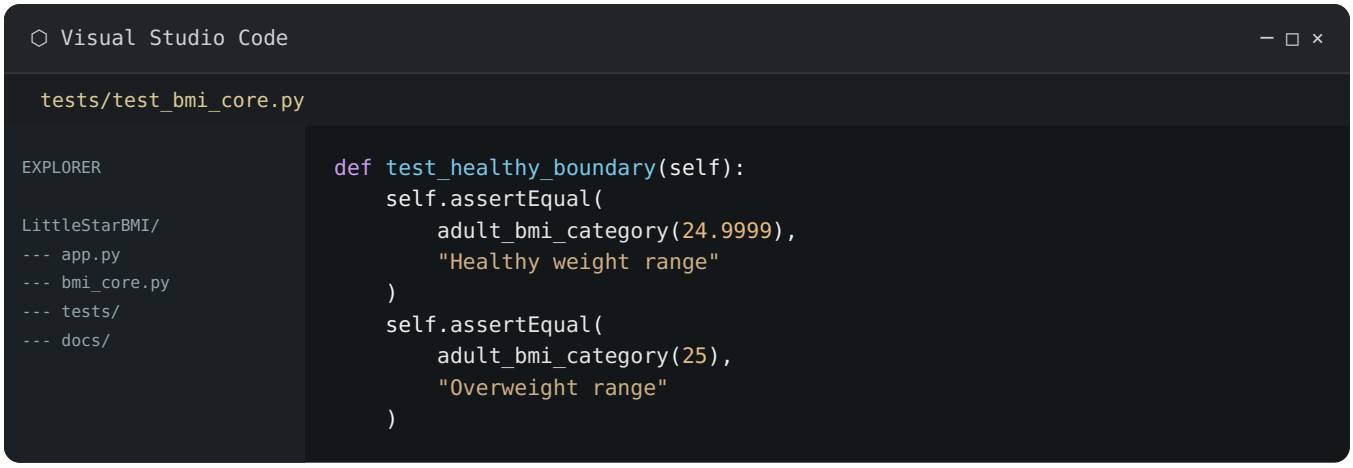
මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

Boundary tests ලියමු

සීමා අගයන් වෙනස් වූ විට category එක නිවැරදිද කියලා test කරන්න.

ඉගෙනගනිමු

- ✓ 18.4999 → Underweight; 18.5 → Healthy.
- ✓ 24.9999 → Healthy; 25 → Overweight.
- ✓ 29.9999 → Overweight; 30 → Obesity range.



✎ දැන් ඔබ කරන්න

Boundary 25.0 සඳහා test method එක ලියන්න.

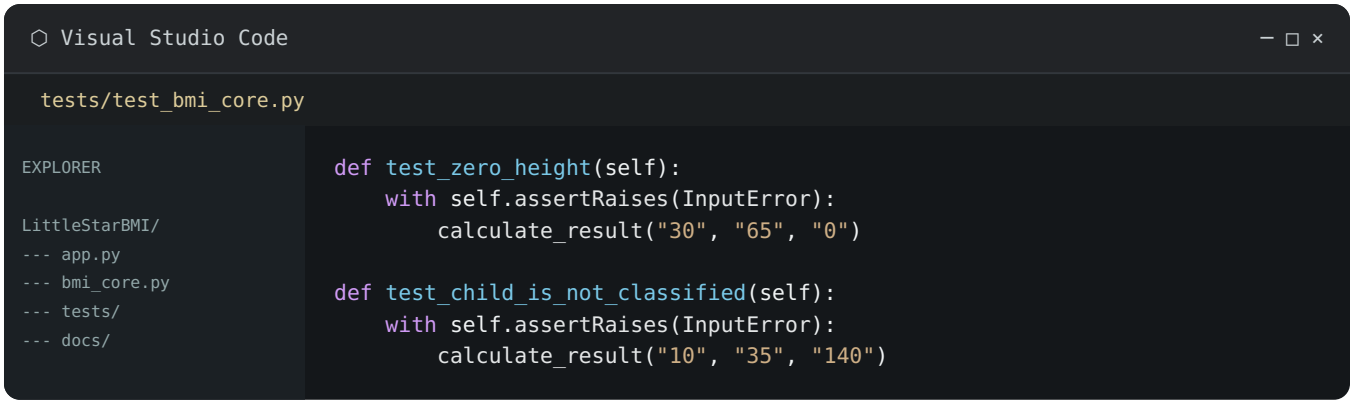
මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

Negative tests

Invalid values reject විය යුතු බව test කිරීමෙන් crash bugs අඩු කරගන්න පුළුවන්.

ඉගෙනගනිමු

- ✓ blank weight, letters, negative numbers.
- ✓ Height zero, unreasonably high demo values, NaN/Inf.
- ✓ Age below 20 or above 120.



✎ දැන් ඔබ කරන්න

Empty height එකට expected error සහ actual result සටහන් කරන්න.

මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

CMD එකෙන් සියලු tests

Automated suite එකක් සැබෑවට execute කර report එක කියවමු.

ඉගෙනගනිමු

- ✓ Project root folder එකේ `py -m unittest discover -s tests -v` run කරන්න.
- ✓ Each test line `... ok` ලෙස පෙන්විය යුතුයි.
- ✓ අවසන් `Ran 19 tests ... OK` වැනි summary එක කියවන්න.

```
Microsoft Windows [Version 2026] · Command Prompt                _ □ ×

C:\Users\Student> py -m unittest discover -s tests -v
C:\Users\Student> REM Each test should show ... ok
C:\Users\Student> REM Final line: OK
```

✎ දැන් ඔබ කරන්න

Test log එකෙන් total tests ගණන සහ status note කරන්න.

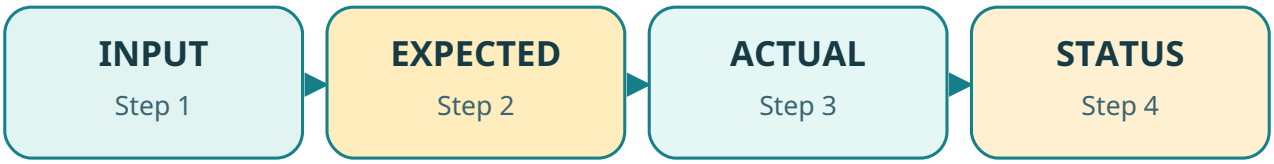
මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

Manual testing – user වගේ බලමු

Automated test එකෙන් සියලු GUI details නොපෙනේ. මිනිසෙකු window එක භාවිතා කර බලන්නත් ඕන.

ඉගෙනගනිමු

- ✓ Age/weight/height labels සහ units නිවැරදිද?
- ✓ Calculate, Clear buttons mouse සහ keyboard දෙකෙන්ම වැඩ කරනවාද?
- ✓ Error dialog එක කියවන්න පහසුද? Window resize කළොත් layout කැඩෙන්නේද?



✎ දැන් ඔබ කරන්න

Manual test sheet එකක් අතින් සාදා PASS/FAIL/NOT RUN ලියන්න.

මතක තබාගන්න: මෙම source suite එකේ Windows GUI manual tests user කෙනෙකුට ඉටු කිරීමට ඇති බව QA report එකේ කියලා තියෙනවා.

Test Plan document එක

Test plan එක test කිරීමට පෙර objectives සහ scope සටහන් කරයි.

ඉගෙනගනිමු

- ✓ Test objective: correct BMI, safe errors, age guard, working UI.
- ✓ Environment: Windows + supported Python + Tkinter.
- ✓ Entry criteria, exit criteria, test data, responsible tester සඳහන් කරන්න.



✎ දැන් ඔබ කරන්න

`docs/Test_Plan.md` හි TC-02 සහ TC-08 සොයන්න.

මතක තබාගන්න: Testing plan එකේ “what we did NOT test” කියලත් පැහැදිලිව සඳහන් කළ යුතුයි.

TC-01 – Happy path

සාමාන්‍ය valid data ලබාදුන් විට result එක නිවැරදි විය යුතුයි.

ඉගෙනගනිමු

- ✓ Test ID: TC-01; input age=25, weight=70, height=170.
- ✓ Action: Click Calculate.
- ✓ Expected: BMI=24.2 and Healthy weight range; no error dialog.

Field	Expected	Actual
Age	25	To fill
Weight	70 kg	To fill
Height	170 cm	To fill
Result	24.2, Healthy	To fill
Status	PASS / FAIL	To fill

✎ දැන් ඔබ කරන්න

Real Windows app එකේ TC-01 run කර actual result ලියන්න.

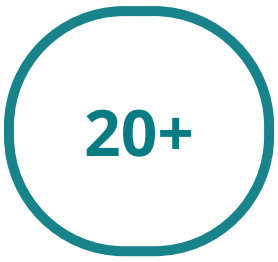
මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

TC-02 – Child age gate

Child test එක fail වුණොත් app එක අනිසි health category පෙන්විය හැක.

ඉගෙනගනිමු

- ✓ Input age=10; weight=35; height=140.
- ✓ Expected: informative error; BMI category නොපෙන්වීම.
- ✓ Actual සහ status teacher සමඟ verify කරන්න.



Adult example only

For people aged 2–19, BMI-for-age percentiles are different. No diagnosis or medical advice.

✎ දැන් ඔබ කරන්න

මෙම test එකට High priority දෙන හේතුවක් ලියන්න.

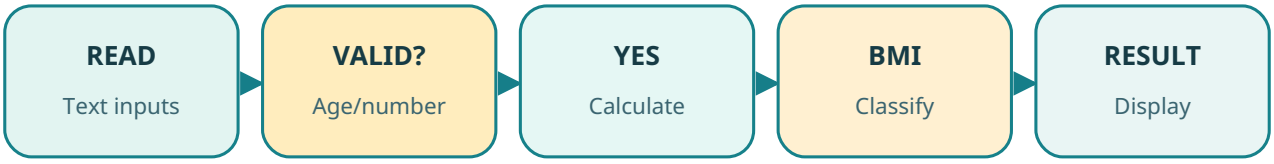
මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

TC-03 – Height zero

Zero height එක formula එකේ zero division error එකකට හේතුවිය හැක.

ඉගෙනගනිමු

- ✓ Input age=30; weight=65; height=0.
- ✓ Expected: height range validation message.
- ✓ App එක crash නොවී input නැවත නිවැරදි කිරීමට ඉඩ ලබාදිය යුතුයි.



✎ දැන් ඔබ කරන්න

TC-03 සඳහා steps, expected, actual fields සහිත bug template එකක් සකස් කරන්න.

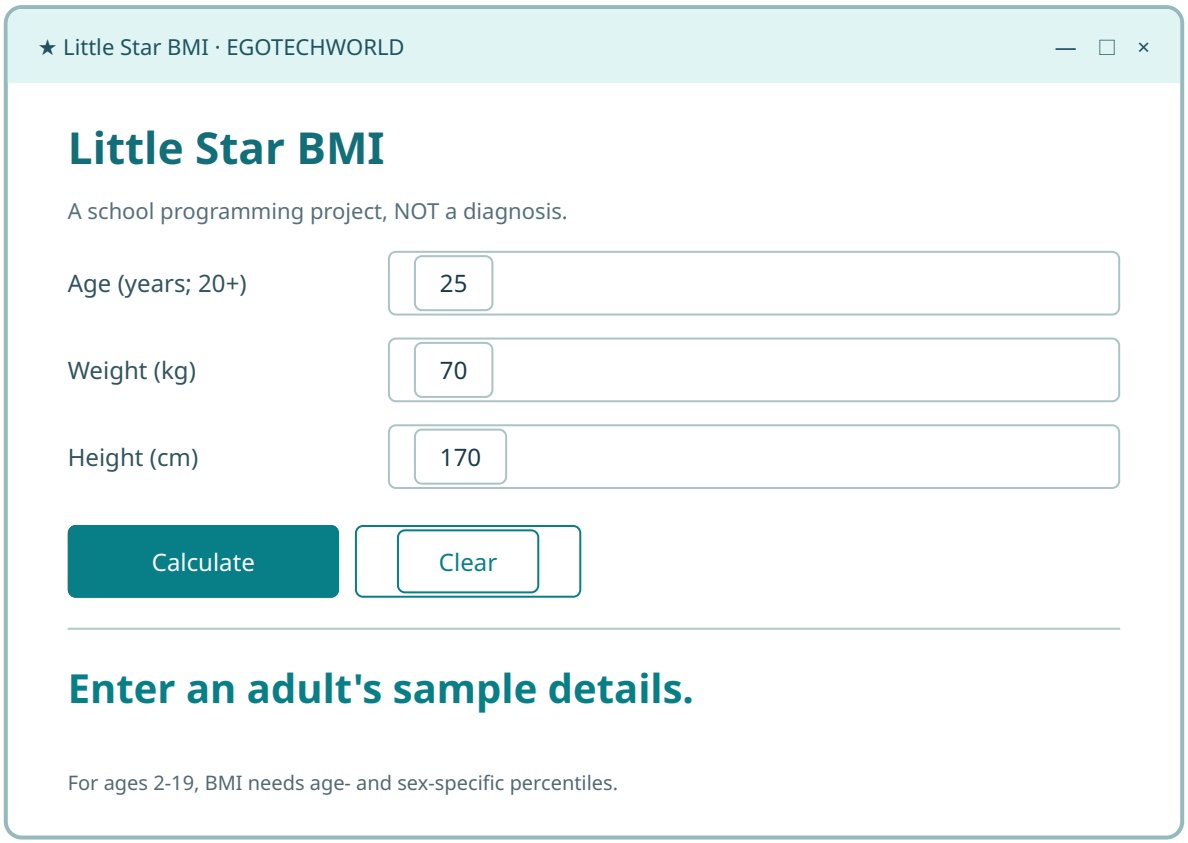
මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

TC-04 – Clear button

Clear action එකේ behaviour එකත් requirement එකක්.

ඉගෙනගනිමු

- ✓ Fields තුනම data සහ result එකක් පෙන්වන්න.
- ✓ Clear ඔබන්න.
- ✓ Expected: fields හිස්; category හිස්; default message; focus first field.



Illustrative Windows desktop preview · exact system theme may vary

✎ දැන් ඔබ කරන්න

Clear click කළ පසු screenshot එකේ පරීක්ෂා කළ යුතු detail 3ක් ලියන්න.

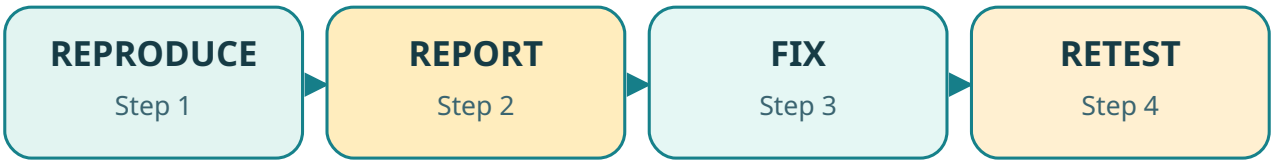
මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

Defect/Bug report ලියමු

Bug report එකක් developer කෙනෙකුට නැවත problem එක සිදු කර බැලීමට උපකාරී විය යුතුයි.

ඉගෙනගනිමු

- ✓ ID, title, severity, environment, steps to reproduce ලියන්න.
- ✓ Expected සහ actual outcome වෙන වෙනම සඳහන් කරන්න.
- ✓ Screenshot හෝ log තිබේ නම් sensitive data නැතිව එක් කරන්න.



✎ දැන් ඔබ කරන්න

`docs/Defect_Report.md` හි example bug එක කියවා heading නම් ලියන්න.

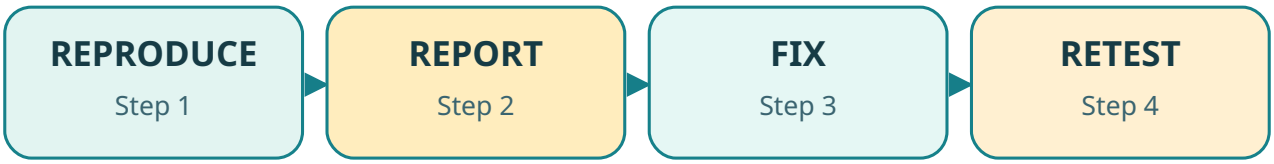
මතක තබාගන්න: Provided sample bug report එක hypothetical learning example එකක්; actual delivered bug එකක් නොවේ.

Severity සහ Priority

Severity කියන්නේ bug එකේ බලපෑම. Priority කියන්නේ එය ඉක්මනින් fix කිරීමේ අවශ්‍යතාවයි.

ඉගෙනගනිමු

- ✓ Crash bug → potentially high severity.
- ✓ Button text typo → lower severity, user context අනුව priority වෙනස් විය හැක.
- ✓ Child age gate failure → incorrect interpretation නිසා high-priority review අවශ්‍ය විය හැක.



✎ දැන් ඔබ කරන්න

Examples දෙකක් ලියා severity සහ priority ගැළපීම් යෝජනා කරන්න.

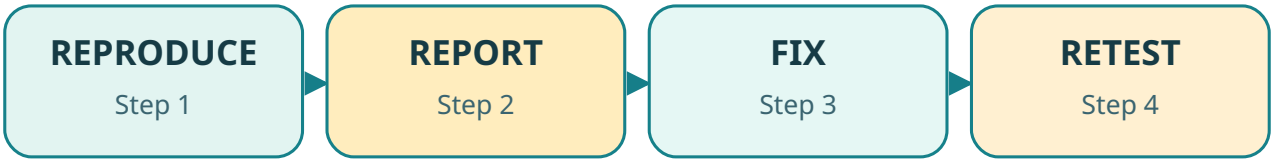
මතක තබාගන්න: Impact සහ urgency එකම මැනුමක් නොවේ.

Bug lifecycle

Found → Reported → Assigned → Fixed → Retested → Closed වැනි states තිබේ.

ඉගෙනගනිමු

- ✓ QA tester bug reproduce කර expected/actual ලියයි.
- ✓ Developer code fix කර unit test එකක් එකතු කරයි.
- ✓ QA tester same scenario සහ regression suite නැවත run කරයි.



✎ දැන් ඔබ කරන්න

Closed කරන පෙර evidence වශයෙන් තබන දේවල් දෙකක් නම් කරන්න.

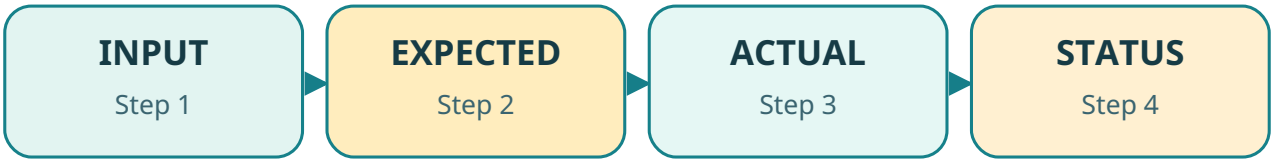
මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

Regression Testing

අලුත් fix එකක් කලින් වැඩ කළ feature එකක් කඩනවාද කියලා නැවත පරීක්ෂා කිරීම regression වේ.

ඉගෙනගනිමු

- ✓ Height zero bug fix කළත් normal 70/170 result එක තවම නිවැරදිද?
- ✓ Age validation fix කළ පසු valid age=25 තවම accept වෙනවාද?
- ✓ සෑම change එකකට පසු unit tests run කරන්න.



✎ දැන් ඔබ කරන්න

බොත්තමක colour වෙනස් කළ පසුත් functional test දෙකක් run කරන්න ඕනද? ඇයි?

මතක තබාගන්න: Automated suite එකක් තිබීම maintenance cost අඩු කරයි.

RTM – Requirements Traceability

Requirement එකකට code සහ test link කිරීමෙන් කිසි feature එකක් පරීක්ෂා නොවී ඉතිරි නොවන ලෙස බලමු.

ඉගෙනගනිමු

- ✓ FR-04 BMI calculation → bmi_core.calculate_bmi → test_known_bmi.
- ✓ FR-02 adult-only guard → read_age → test_child_is_not_classified.
- ✓ FR-07 Clear button → app.do_clear → manual TC-08.

Req	Implementation	Verification
FR-02	read_age	Unit child test
FR-04	calculate_bmi	Unit formula test
FR-07	do_clear	Manual TC-08
FR-09	no save/network	Source review

✎ දැන් ඔබ කරන්න

ඔබගේම requirement එකක් RTM row එකක් ලෙස ලියන්න.

මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

QA report සකස් කරමු

Testing කළ පසු සිදු කළ දේ සහ ඉතිරි දේ වාර්තා කරන්න.

ඉගෙනගනිමු

- ✓ Automated tests: මේ project source එකේ 19 tests PASS.
- ✓ Python syntax compile: PASS.
- ✓ Windows GUI manual verification: එය Windows machine එකක කරන තෙක් NOT RUN ලෙස පැහැදිලිව සඳහන් කරන්න.



✎ දැන් ඔබ කරන්න

`docs/QA_Report.md` කියවා “Not yet executed” කොටස හඳුනාගන්න.

මතක තබාගන්න: බලලා නැති දෙයක් PASS කියලා ලියන එක quality report එකකට සුදුසු නැහැ.

Quality evidence කියන්නේ මොකක්ද?

“මම හිතන්නේ වැඩ” වෙනුවට නැවත පරීක්ෂා කළ හැකි සාක්ෂි එකතු කරමු.

ඉගෙනගනිමු

- ✓ Unit test output log.
- ✓ Manual test steps, date, system and PASS/FAIL status.
- ✓ Defect IDs, fixes සහ regression results.



✎ දැන් ඔබ කරන්න

ඔබ පරීක්ෂා කළ එක් case එකකට evidence list එකක් ලියන්න.

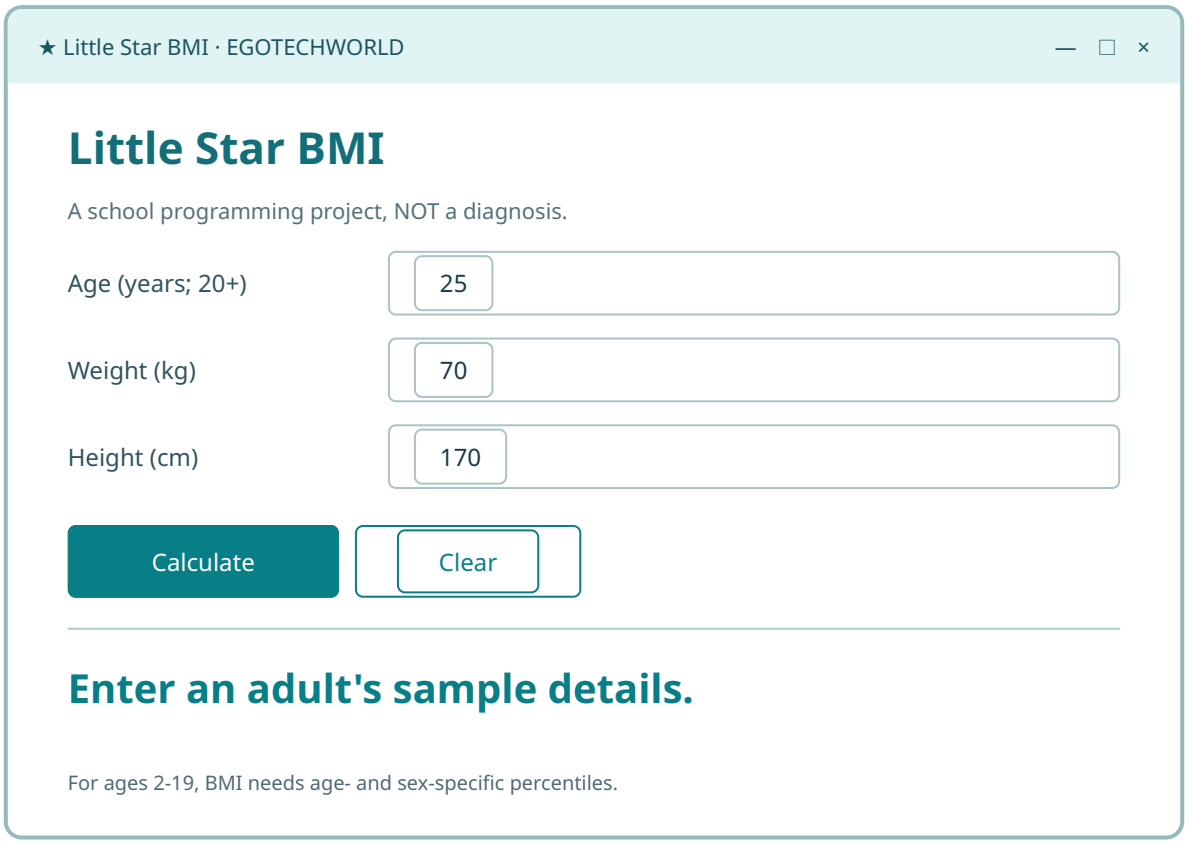
මතක තබාගන්න: QA summary එක නිවැරදිව තබාගැනීම professional ethics එකක කොටසකි.

Accessibility QA

විවිධ users ට app එක භාවිතා කළ හැකිද කියලා බලමු.

ඉගෙනගනිමු

- ✓ Keyboard navigation: Tab සහ Enter behaviour.
- ✓ Clear unit labels, readable sizes සහ contrast.
- ✓ Error message එක තේරුම්ගත හැකිද; වැරදි input එකට උපදෙස් තිබේද?



Illustrative Windows desktop preview · exact system theme may vary

✎ දැන් ඔබ කරන්න

ඔබේ app එක mouse නැතිව පරීක්ෂා කරන්න.

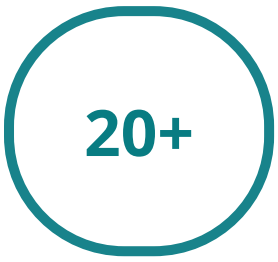
මතක තබාගන්න: Accessibility පරීක්ෂණයක් දිනපතා code review එකකට එක් කළ හැක.

Privacy සහ security review

Health සම්බන්ධ values ඇති app එකකට privacy ගැන නිරන්තර සැලකිල්ලක් අවශ්‍යයි.

ඉගෙනගනිමු

- ✓ App එක user input files වලට save කරන්නේ නැහැ.
- ✓ Network call හෝ account login නැහැ.
- ✓ Future history feature එකකට consent, retention සහ security requirements එකතු කිරීමට සිදුවේ.



Adult example only

For people aged 2–19, BMI-for-age percentiles are different. No diagnosis or medical advice.

✎ දැන් ඔබ කරන්න

සැබෑ data නොගෙන testing කළ හැකි fictional sample dataset එකක් යෝජනා කරන්න.

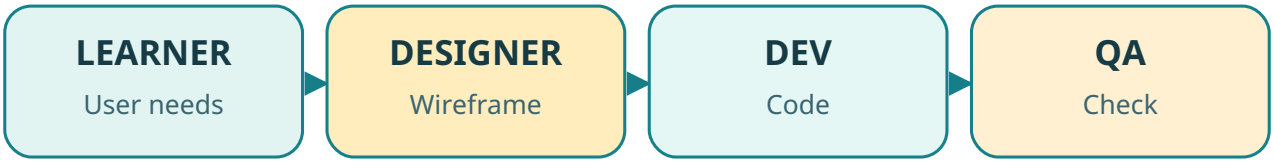
මතක තබාගන්න: “No data collected” කියන්න කලින් code එකේ file/network actions නොමැති බව review කරන්න.

UAT – user acceptance

User Acceptance Testing (UAT) කියන්නේ app එක සැබෑ අරමුණ ඉටු කරනවාද කියලා user/presenter නිරීක්ෂණය කිරීමයි.

ඉගෙනගනිමු

- ✓ Beginner කෙනෙකුට input fields තේරෙනවාද?
- ✓ Invalid input එකකට feedback පැහැදිලිද?
- ✓ Calculator result එක paper formula එක සමඟ match වෙනවාද?



✎ දැන් ඔබ කරන්න

මිතුරෙකුට app එක demo කර වැඩිදියුණු කිරීමක් අහන්න.

මතක තබාගන්න: UAT feedback නැවත backlog එකට යවන්න.

Git සහ version control

Git මඟින් code changes වල history තබා පසුගිය working version එකකට යාම පහසු වේ.

ඉගෙනගනිමු

- ✓ `git init`, `git add`, `git commit` යන basic flow එක ඉගෙනගන්න.
- ✓ Meaningful commit messages ලියන්න: “Add age validation”.
- ✓ Public repository එකකට personal data හෝ secrets upload නොකරන්න.

```
Microsoft Windows [Version 2026] · Command Prompt                    _ □ ×

C:\Users\Student> git init
C:\Users\Student> git add .
C:\Users\Student> git commit -m "Build safe BMI calculator demo"
C:\Users\Student> git status
```

✎ දැන් ඔබ කරන්න

`git status` command එකෙන් project status පරීක්ෂා කරන්න.

මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

README – user manual

README එකෙන් වෙනත් කෙනෙකුට ඔබේ project එක run කිරීමට steps ලැබෙයි.

ඉගෙනගනිමු

- ✓ Required Python version සහ operating system සඳහන් කරන්න.
- ✓ Extract ZIP → open folder → `py app.py` steps ලියන්න.
- ✓ How to test, known limitations සහ privacy statement එකතු කරන්න.



✎ දැන් ඔබ කරන්න

README එකේ Quick Start section එක කියවා අනෙකෙකුට run කරන්න උපදෙස් දෙන්න.

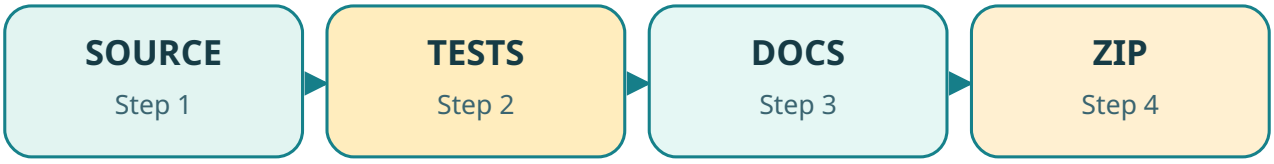
මතක තබාගන්න: Good README එකක් user support ප්‍රශ්න අඩු කරයි.

Release checklist

App එක share කිරීමට පෙර final review එකක් අවශ්‍යයි.

ඉගෙනගනිමු

- ✓ Source files, docs, tests සියල්ල ZIP එකට අයත්ද?
- ✓ Unit tests PASS; Windows GUI manual tests checked?
- ✓ Child age guard සහ no-medical-advice note පැහැදිලිද?



✎ දැන් ඔබ කරන්න

Release checklist එකට ඔබේම check item දෙකක් එකතු කරන්න.

මතක තබාගන්න: Windows EXE installer එකක් මෙහි ඇතුළත් නැහැ; source එක Python installed PC එකක run කළ යුතුයි.

ZIP project folder එක

Source ZIP එකෙන් අනෙක් කෙනෙකුට folder structure එක නොනැසී ලබාගත හැක.

ඉගෙනගනිමු

- ✓ ZIP extract කළ පසු `app.py` root එකේ තිබිය යුතුයි.
- ✓ `tests/` සහ `docs/` subfolders එම මට්ටමේම තිබිය යුතුයි.
- ✓ External database/password files අවශ්‍ය නැහැ.

```
EXPLORER · Little-Star-BMI/

--- app.py
--- bmi_core.py
--- hello_window.py
--- run_windows.bat
--- tests/
    --- test_bmi_core.py
--- docs/
    --- SRS.md
    --- Test_Plan.md
    --- QA_Report.md
```

✎ දැන් ඔබ කරන්න

ZIP එක අලුත් folder එකකට extract කර README අනුව run කරන්න.

මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

Windows CMD – final run

දැනට ලබාදුන් project එක Windows computer එකක start කිරීමට කෙටි පියවර.

ඉගෙනගනිමු

- ✓ CMD එකේ project folder වෙත `cd` කරන්න.
- ✓ `py -m tkinter` check කරන්න; `py app.py` run කරන්න.
- ✓ Application window එක විවෘත වුණාම fictional adult sample එක calculate කරන්න.

```
Microsoft Windows [Version 2026] · Command Prompt                _ □ ×

C:\Users\Student> py --version
C:\Users\Student> py -m tkinter
C:\Users\Student> py app.py
C:\Users\Student> py -m unittest discover -s tests -v
```

✎ දැන් ඔබ කරන්න

`run_windows.bat` file එකෙන් start කළ හැකිද බලන්න.

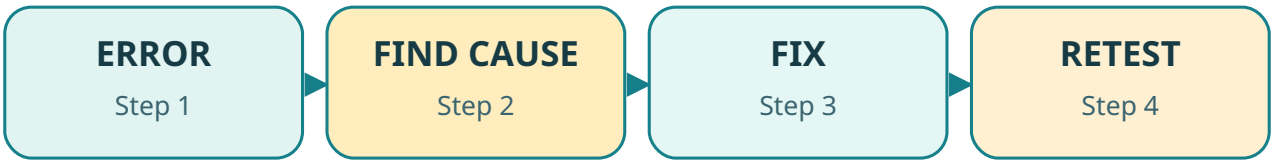
මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

Common error 01 – Python not found

`py` recognized නොවේ නම් Python installation හෝ launcher configuration පරීක්ෂා කරන්න.

ඉගෙනගනිමු

- ✓ Official installer එක භාවිතා කර Python install වී තිබේද?
- ✓ `python --version` alternate command එක පරීක්ෂා කරන්න.
- ✓ Editor එකේ Python interpreter selection නිවැරදිද බලන්න.



✎ දැන් ඔබ කරන්න

CMD error message එක copy කර යළි සකස් කරන steps ලියා තබන්න.

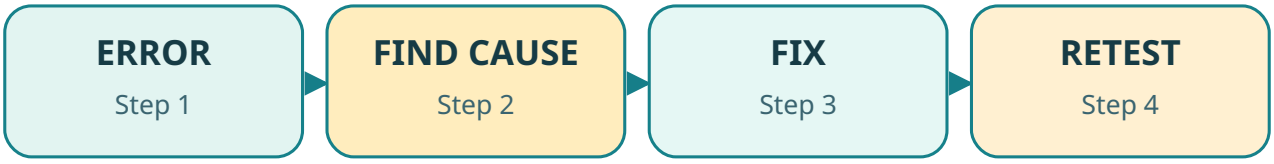
මතක තබාගන්න: Random web downloads වලින් interpreters ගන්න එපා; official source භාවිතා කරන්න.

Common error 02 – Tkinter window

`No module named tkinter` හෝ screen open නොවන ගැටලු setup එකේ විය හැක.

ඉගෙනගනිමු

- ✓ `py -m tkinter` test එකෙන් standard toolkit ක්‍රියා කරනවාද බලන්න.
- ✓ Windows Python distribution එකේ Tcl/Tk components තිබේද පරීක්ෂා කරන්න.
- ✓ Remote/headless systems වල GUI display එකක් නොතිබිය හැක.



✎ දැන් ඔබ කරන්න

`py -m tkinter` test එකේ output සහ screenshot එක සටහන් කරන්න.

මතක තබාගන්න: Linux server එකක display නොපෙනුණා කියලා Windows desktop source code වැරදි බව අනිවාර්ය නැහැ.

Common error 03 – Import

`No module named bmi_core` කියන්නේ file path හෝ project folder ප්‍රශ්නයක් විය හැක.

ඉගෙනගනිමු

- ✓ `app.py` සහ `bmi_core.py` එකම project root එකේද බලන්න.
- ✓ Filename spelling/case නිවැරදිද බලන්න.
- ✓ CMD current directory එක ZIP එකේ main folder එකද පරීක්ෂා කරන්න.

```
Microsoft Windows [Version 2026] · Command Prompt                    _ □ ×

C:\Users\Student> dir
C:\Users\Student> REM Check these exist in the same folder:
C:\Users\Student> REM app.py
C:\Users\Student> REM bmi_core.py
```

✎ දැන් ඔබ කරන්න

`dir` command එකෙන් file names නැවත පරීක්ෂා කරන්න.

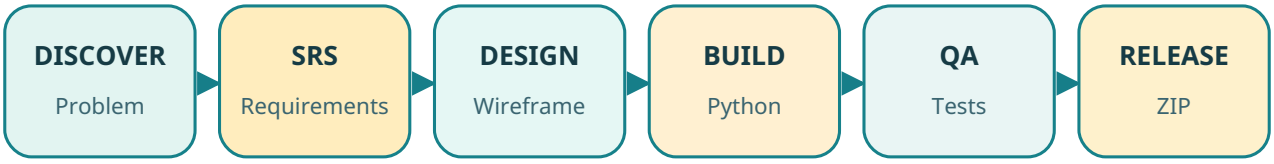
මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

Maintenance – bug fix cycle

Release එකෙන් පසුද software වැඩ ඉවර වෙන්නේ නැහැ.

ඉගෙනගනිමු

- ✓ User bug report → reproduction → root cause → fix.
- ✓ Unit test එකක් add කර regression suite run කරන්න.
- ✓ Docs සහ version label update කර new release ලබාදෙන්න.



✎ දැන් ඔබ කරන්න

Future v1.1 update එකකට feature එකක් සහ test එකක් යෝජනා කරන්න.

මතක තබාගන්න: Software development එක ඉගෙනගන්න හොඳම ක්‍රමය කුඩා වැඩිදියුණු කිරීම් කිරීමයි.

7-Day Beginner Plan

මේ පාඩම සතියක් තුළ practice කළ හැකි ලෙස බෙදාගන්න.

ඉගෙනගනිමු

- ✓ Days 1–2: problem, user story, SRS, SDLC map.
- ✓ Days 3–4: Python basics, BMI logic, function tests.
- ✓ Days 5–7: Tkinter, manual QA, bug report, README, release ZIP.



✎ දැන් ඔබ කරන්න

සෑම දවසක් අවසානයේ “අද මට පුළුවන්” කියන වාක්‍යයක් ලියන්න.

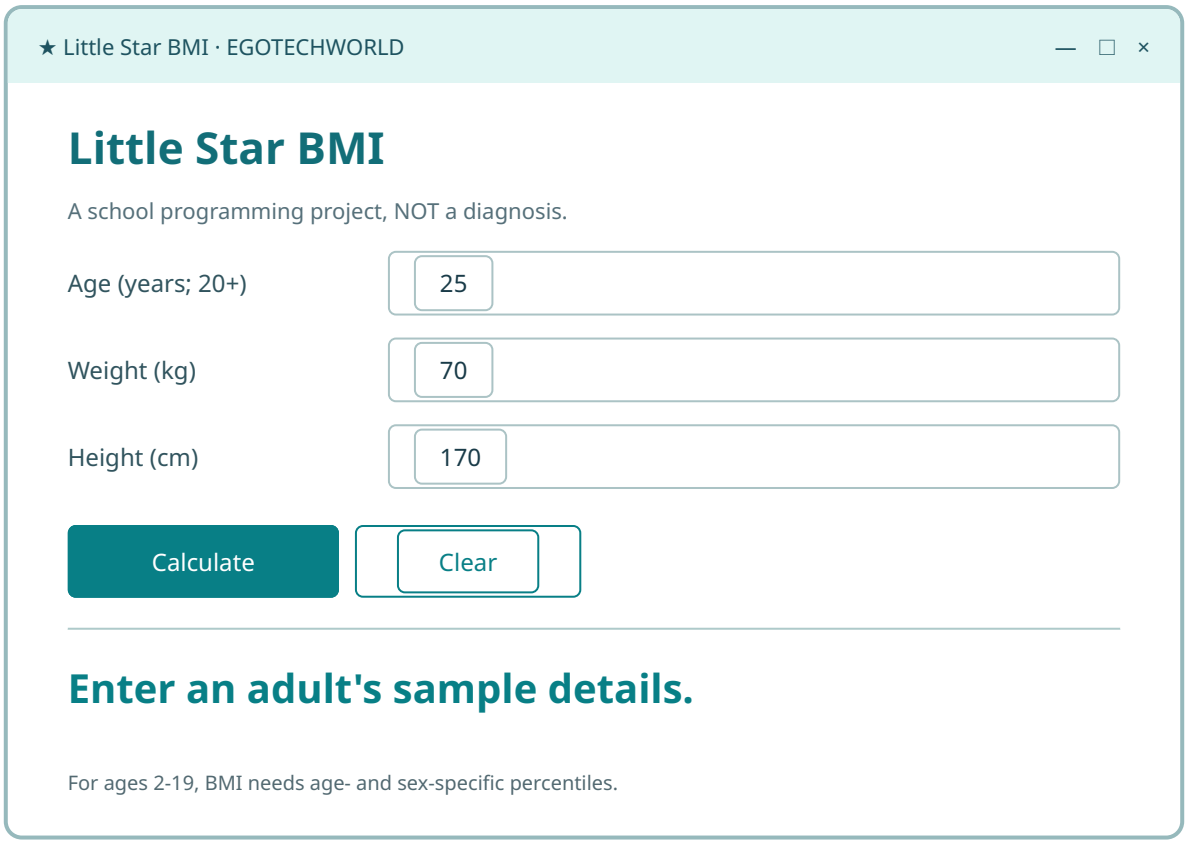
මතක තබාගන්න: ඕනෑම දවසක අමාරු නම් එම පාඩම නැවත practice කරන්න.

Mini challenge 1 – UI වෙනස් කරන්න

Ready project එකක් වෙනස් කිරීමෙන් අලුත් development skills පටන් ගන්න.

ඉගෙනගනිමු

- ✓ Window title “Little Star BMI” වෙනුවට “My Learning Calculator” කරන්න.
- ✓ Clear button text එක “Reset” කරන්න.
- ✓ Version number එකක් status label එකකට එකතු කරන්න.



Illustrative Windows desktop preview · exact system theme may vary

✎ දැන් ඔබ කරන්න

Changes තුනම tests නැවත run කිරීමෙන් පසු teacher ට demo කරන්න.

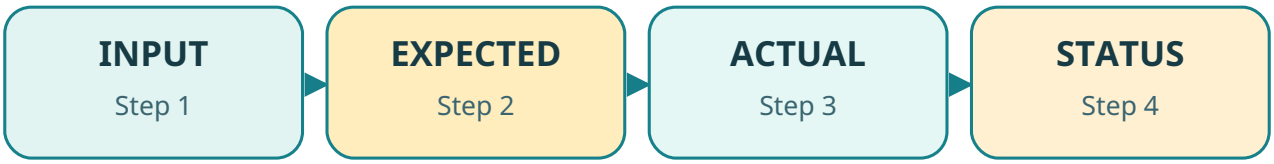
මතක තබාගන්න: Source code backup copy එකක් තබාගන්න.

Mini challenge 2 – Unit tests

කලින් unit tests තිබුණත් නව case එකක් නිර්මාණය කිරීමෙන් testing දක්ෂතාව වැඩෙයි.

ඉගෙනගනිමු

- ✓ Value: age=120, weight=65, height=160 valid ද?
- ✓ Value: age=121 invalid ද?
- ✓ Value: weight=0 error ද?



✎ දැන් ඔබ කරන්න

ඔබේ own `test_...` method එකක් ලියා suite එක run කරන්න.

මතක තබාගන්න: Test code තමන්ගේම product code එකේ ආරක්ෂක ජාලයකි.

Mini challenge 3 – SRS update

Functional change එකක් documentation සහ testing දෙකම වෙනස් කළ යුතුයි.

ඉගෙනගනිමු

- ✓ Display එකේ “BMI units kg/m²” යන caption එකක් දාන්න ඉල්ලීමක් හිතන්න.
- ✓ SRS requirement සහ acceptance criteria ලියන්න.
- ✓ Code change කිරීමෙන් පසු manual test case සහ release note update කරන්න.



✎ දැන් ඔබ කරන්න

Small change request document එකක් සකස් කරන්න.

මතක තබාගන්න: උදාහරණය වෙනස් කර බලන්න. පළමුව ප්‍රතිඵලය අනුමාන කර පසුව පරීක්ෂා කරන්න.

MCQ – Concepts

පහත ප්‍රශ්නවල නිවැරදි පිළිතුර තෝරන්න.

ඉගෙනගනිමු

- ✓ 1. SRS යනු (A) Code editor (B) Requirements document (C) Browser.
- ✓ 2. `Tk()` භාවිතා කරන්නේ (A) Window (B) Database (C) Network.
- ✓ 3. `float("70")` ලබාදෙන්නේ (A) Text (B) Number (C) Button.



✎ දැන් ඔබ කරන්න

ඔබ තෝරාගත් A/B/C පිළිතුරු notebook එකක ලියන්න.

මතක තබාගන්න: ඊළඟ පිටුවේද ප්‍රශ්න තිබෙනවා; පිළිතුරු පසුව බලන්න.

MCQ – SDLC & QA

SDLC සහ testing සංකල්ප ඔබට තේරුණාද බලමු.

ඉගෙනගනිමු

- ✓ 4. BMI formula: (A) kg/m² (B) kg×m² (C) kg+cm.
- ✓ 5. Boundary test example: (A) 18.5 (B) favourite colour (C) app logo.
- ✓ 6. Actual test කර නොමැති නම් status: (A) PASS (B) NOT RUN (C) DONE.



✎ දැන් ඔබ කරන්න

පිළිතුරු දෙමින් හේතුවක්ද ලියන්න.

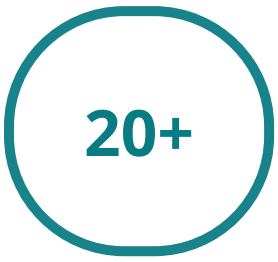
මතක තබාගන්න: ඇත්තටම test කළ නැති දෙයක් PASS කියන්න එපා.

MCQ – Safety & Privacy

Software engineering තුළ responsible design වැදගත්.

ඉගෙනගනිමු

- ✓ 7. Age 10 සඳහා adult category (A) පෙන්වන්න (B) නොපෙන්වන්න (C) random.
- ✓ 8. BMI (A) Diagnosis (B) Screening measure (C) Password.
- ✓ 9. Privacy by design (A) අනවශ්‍ය data save නොකිරීම (B) data publish කිරීම (C) secret copy.



Adult example only

For people aged 2–19, BMI-for-age percentiles are different. No diagnosis or medical advice.

✎ දැන් ඔබ කරන්න

මෙම ප්‍රශ්න 3ට පිළිතුරු සහ කෙටි හේතුවක් ලියන්න.

මතක තබාගන්න: කේතය නිවැරදි වීම සහ එය වගකීමෙන් භාවිතය දෙකම අත්‍යවශ්‍යයි.

MCQ පිළිතුරු සහ විග්‍රහය

තෝරාගත් answers සාක්ෂි සමඟ පරීක්ෂා කරන්න.

ඉගෙනගනිමු

- ✓ 1=B (SRS), 2=A (Tk window), 3=B (float).
- ✓ 4=A (kg/m²), 5=A (boundary 18.5), 6=B (NOT RUN).
- ✓ 7=B (adult category guard), 8=B (screening), 9=A (privacy).



✎ දැන් ඔබ කරන්න

ඔබ වැරදි කළ ප්‍රශ්නවල topics නැවත PDF එකෙන් ඉගෙනගන්න.

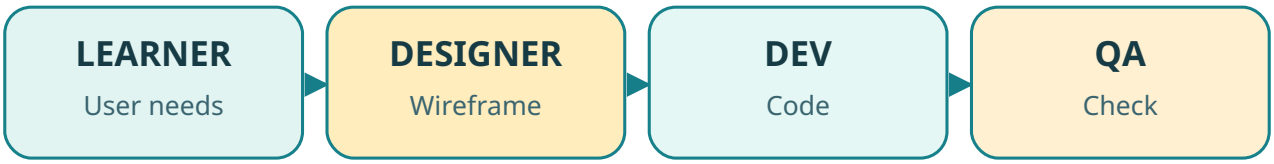
මතක තබාගන්න: Answers මතක තබාගැනීම පමණක් නොව හේතුව අවබෝධ කරගැනීම වැදගත්.

Team project demonstration

Software engineer කෙනෙකුට තම project එක පැහැදිලිව පෙන්වන්න පුළුවන් විය යුතුයි.

ඉගෙනගනිමු

- ✓ 1 min: problem + users + scope.
- ✓ 2 min: wireframe + program demo.
- ✓ 2 min: tests + QA report + known limitations.



✎ දැන් ඔබ කරන්න

අවුරුදු 10ක student කෙනෙකුට තේරෙන ලෙස ඔබේ project demo speech එක සකස් කරන්න.

මතක තබාගන්න: Demonstration එකේ fictional health values පමණක් භාවිතා කරන්න.

Self-assessment checklist

මේ දක්වා ඉගෙනගත් කුසලතා තමන්ම පරීක්ෂා කරන්න.

ඉගෙනගනිමු

- ✓ Software problem, SDLC stages, FR/NFR, SRS, design diagram තේරෙනවාද?
- ✓ Python function, exception, Tkinter event, BMI formula පැහැදිලිද?
- ✓ Unit tests run, manual cases record, defect report සහ README ලියන්න පුළුවන්ද?



✎ දැන් ඔබ කරන්න

තව practice අවශ්‍ය skills තුනක් තෝරා ඊළඟ සතියේ plan එකක් සකස් කරන්න.

මතක තබාගන්න: “නොදන්නා දෙයක්” හඳුනාගැනීමත් ඉගෙනීමේ ශක්තියකි.

2026 Software engineering habits

Project එක කුඩා වුණත් professional habits ගොඩනගාගන්න පුළුවන්.

ඉගෙනගනිමු

- ✓ Understand → design → code → test → document යන cycle එක නැවත යොදාගන්න.
- ✓ AI suggestions review කර tests නොමැතිව merge නොකරන්න.
- ✓ Accessible UI, privacy, security updates සහ honest QA evidence වැදගත්.



✎ දැන් ඔබ කරන්න

Future project එකකට habits තුනක් “මම හැමදාම …” ලෙස ලියන්න.

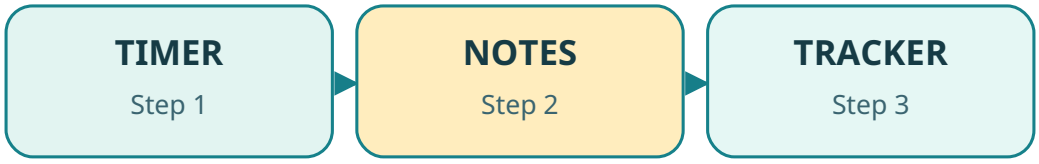
මතක තබාගන්න: විශාල software systems වලටත් මේ fundamentals යොදාගත හැක.

Next project ideas

BMI app එකෙන් ඉගෙනගත් techniques වෙනත් ගැටලු සඳහා යොදාගන්න.

ඉගෙනගනිමු

- ✓ Study timer / simple calculator / library book tracker.
- ✓ To-do list with data saving (privacy and backups සලකා බලන්න).
- ✓ Mini inventory tracker සමඟ validation සහ reports.



✎ දැන් ඔබ කරන්න

ඔබ ඊළඟට හදන app එකකට user story එකක් ලියන්න.

මතක තබාගන්න: කලින් ඉගෙනගත් SRS, tests සහ QA report අලුත් project එකටත් reuse කරන්න.

References – නිල තොරතුරු

මෙම පාඩම සැලසුම් කිරීමේදී නිල documentation මඟින් coding සහ BMI සීමා cross-check කළා.

ඉගෙනගනිමු

- ✓ Python downloads: python.org/downloads/; Tkinter: docs.python.org/3/library/tkinter.html.
- ✓ Python unittest: docs.python.org/3/library/unittest.html.
- ✓ Adult/Child BMI guidance: cdc.gov/bmi/adult-calculator/bmi-categories.html සහ cdc.gov/bmi/child-teen-calculator/.



✎ දැන් ඔබ කරන්න

මෙම links දෙකක් විවෘත කර document title සටහන් කරන්න.

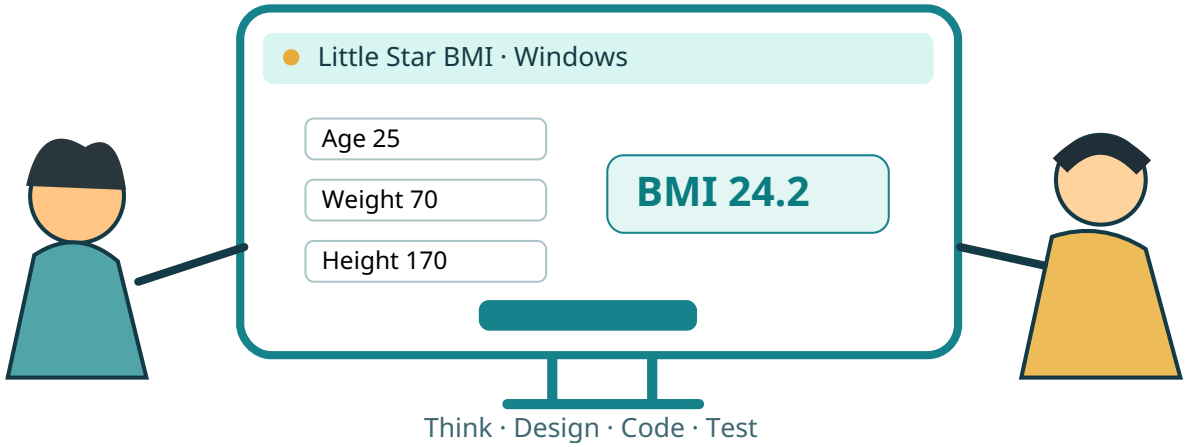
මතක තබාගන්න: 2026 version details අනාගතයේ වෙනස් විය හැකි නිසා download වෙලාවේ official site පරීක්ෂා කරන්න.

සාරාංශය – එක මනා නිමාවක්

ඔබ software development එකේ මුල් ගමන ඉගෙනගෙන, project එක testable සහ reproducible ලෙස සකස් කරලා තියෙනවා.

ඉගෙනගනිමු

- ✓ Problem → Requirements → Design → Implementation → Testing → Release.
- ✓ Windows Tkinter app source + pure Python BMI logic + tests.
- ✓ SRS, test plan, QA report, traceability, bug example සහ release checklist.



✎ දැන් ඔබ කරන්න

ඊළඟ සතියේ මිතුරෙකුට ඔබේ project එකෙන් lessons තුනක් උගන්වන්න.

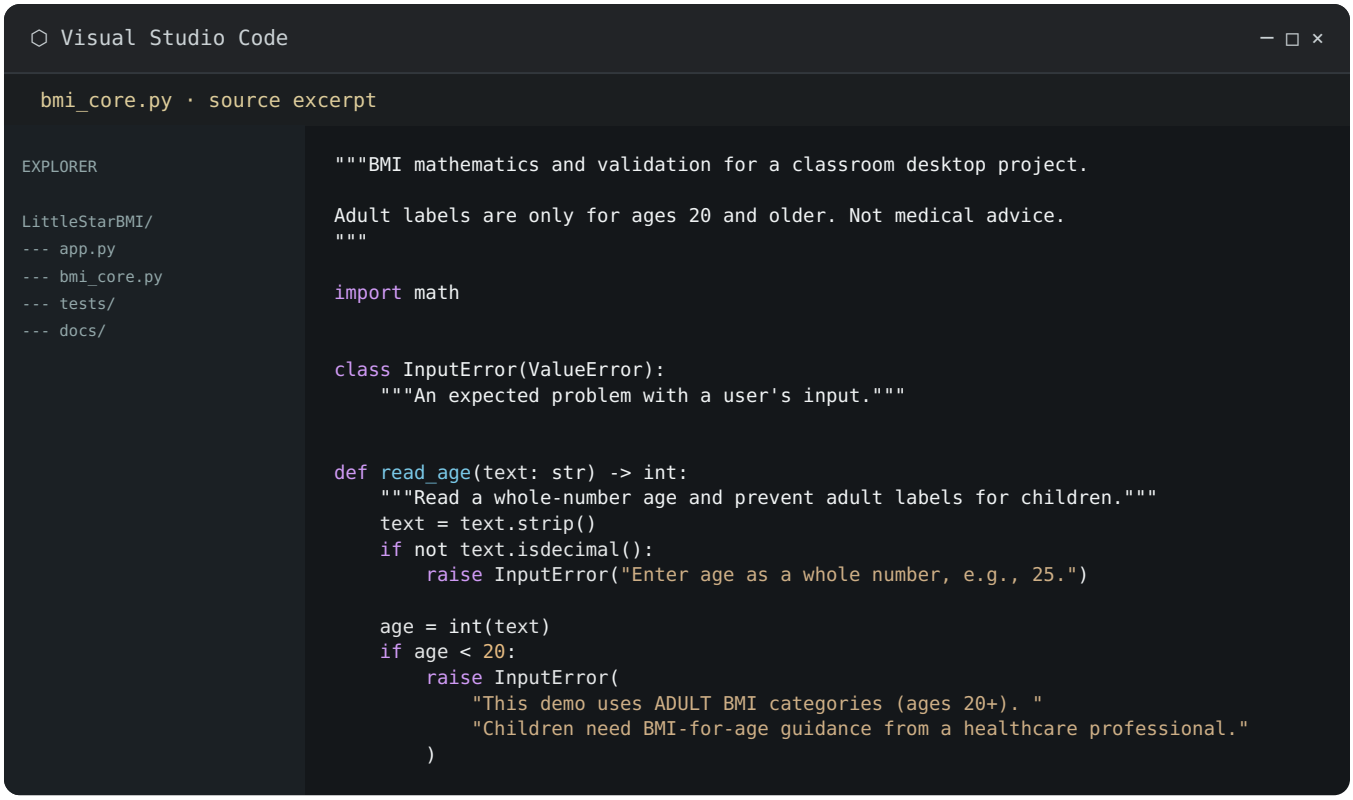
මතක තබාගන්න: ඉගෙනගැනීම නතර කරන්න එපා. සිතන්න, sketch කරන්න, code කරන්න, test කරන්න, document කරන්න!

bmi_core.py · කොටස 1/4

Project ZIP හි ඇතුළත් bmi_core.py file එකේ සත්‍ය code lines. Input validation සහ BMI calculations. මේවා එකම file එකේ ඉදිරියෙන් ඉදිරියට කියවන්න.

ඉගෙනගනිමු

- ✓ මෙම excerpt එකේ functions, names සහ comments හඳුනාගන්න.
- ✓ Code එකේ indent සහ syntax නොවෙනස්ව තබන්න.
- ✓ සම්පූර්ණ file එක source ZIP එකෙන් ලබාගන්න; මේ කොටස තනිව execute කිරීමට අදහස් නොකරයි.



✎ දැන් ඔබ කරන්න

කොටස 1 හි අලුත්ම variable හෝ function එකක් තෝරා එහි කාර්යය පැහැදිලි කරන්න.

මතක තබාගන්න: Source ZIP එක runnable project එකයි; appendix එක code reading සහ revision සඳහාය.

bmi_core.py · කොටස 2/4

Project ZIP හි ඇතුළත් bmi_core.py file එකේ සත්‍ය code lines. Input validation සහ BMI calculations. මේවා එකම file එකේ ඉදිරියෙන් ඉදිරියට කියවන්න.

ඉගෙනගනිමු

- ✓ මෙම excerpt එකේ functions, names සහ comments හඳුනාගන්න.
- ✓ Code එකේ indent සහ syntax නොවෙනස්ව තබන්න.
- ✓ සම්පූර්ණ file එක source ZIP එකෙන් ලබාගන්න; මේ කොටස තනිව execute කිරීමට අදහස් නොකරයි.

⬡ Visual Studio Code — □ ×

bmi_core.py · source excerpt

EXPLORER

LittleStarBMI/
--- app.py
--- bmi_core.py
--- tests/
--- docs/

```
        if age > 120:
            raise InputError("Enter a realistic adult age (20 to 120).")
        return age


    def read_measurement(text: str, label: str, minimum: float,
                         maximum: float) -> float:
        """Convert text into a safe finite number within a demo range."""
        try:
            number = float(text.strip())
        except (TypeError, ValueError):
            raise InputError(f"Enter a number for {label}.") from None

        if not math.isfinite(number):
            raise InputError(f"{label} must be a finite number.")
        if not minimum <= number <= maximum:
            raise InputError(f"{label} must be between {minimum:g} and {maximum:g}.")
        return number


    def calculate_bmi(weight_kg: float, height_cm: float) -> float:
        """Calculate weight / height^2 using metres; do not round early."""
        weight = read_measurement(str(weight_kg), "Weight (kg)", 1, 500)
        height = read_measurement(str(height_cm), "Height (cm)", 50, 250)
```

✎ දැන් ඔබ කරන්න

කොටස 2 හි අලුත්ම variable හෝ function එකක් තෝරා එහි කාර්යය පැහැදිලි කරන්න.

මතක තබාගන්න: Source ZIP එක runnable project එකයි; appendix එක code reading සහ revision සඳහාය.

bmi_core.py · කොටස 3/4

Project ZIP හි ඇතුළත් bmi_core.py file එකේ සත්‍ය code lines. Input validation සහ BMI calculations. මේවා එකම file එකේ ඉදිරියෙන් ඉදිරියට කියවන්න.

ඉගෙනගනිමු

- ✓ මෙම excerpt එකේ functions, names සහ comments හඳුනාගන්න.
- ✓ Code එකේ indent සහ syntax නොවෙනස්ව තබන්න.
- ✓ සම්පූර්ණ file එක source ZIP එකෙන් ලබාගන්න; මේ කොටස තනිව execute කිරීමට අදහස් නොකරයි.

⬡ Visual Studio Code — □ ×

bmi_core.py · source excerpt

EXPLORER

LittleStarBMI/
--- app.py
--- bmi_core.py
--- tests/
--- docs/

```
    metres = height / 100
    return weight / (metres ** 2)


def adult_bmi_category(bmi: float) -> str:
    """Use adult 20+ BMI screening categories, never child labels."""
    if not math.isfinite(bmi) or bmi <= 0:
        raise InputError("BMI must be a positive finite number.")
    if bmi < 18.5:
        return "Underweight range"
    if bmi < 25:
        return "Healthy weight range"
    if bmi < 30:
        return "Overweight range"
    return "Obesity range"


def calculate_result(age_text: str, weight_text: str,
                     height_text: str) -> tuple[float, str]:
    """Validate all input fields and return BMI + adult category."""
    read_age(age_text)
    weight = read_measurement(weight_text, "Weight (kg)", 1, 500)
    height = read_measurement(height_text, "Height (cm)", 50, 250)
    bmi = calculate_bmi(weight, height)
```

✎ දැන් ඔබ කරන්න

කොටස 3 හි අලුත්ම variable හෝ function එකක් තෝරා එහි කාර්යය පැහැදිලි කරන්න.

මතක තබාගන්න: Source ZIP එක runnable project එකයි; appendix එක code reading සහ revision සඳහාය.

bmi_core.py · කොටස 4/4

Project ZIP හි ඇතුළත් bmi_core.py file එකේ සත්‍ය code lines. Input validation සහ BMI calculations. මේවා එකම file එකේ ඉදිරියෙන් ඉදිරියට කියවන්න.

ඉගෙනගනිමු

- ✓ මෙම excerpt එකේ functions, names සහ comments හඳුනාගන්න.
- ✓ Code එකේ indent සහ syntax නොවෙනස්ව තබන්න.
- ✓ සම්පූර්ණ file එක source ZIP එකෙන් ලබාගන්න; මේ කොටස තනිව execute කිරීමට අදහස් නොකරයි.



✎ දැන් ඔබ කරන්න

කොටස 4 හි අලුත්ම variable හෝ function එකක් තෝරා එහි කාර්යය පැහැදිලි කරන්න.

මතක තබාගන්න: Source ZIP එක runnable project එකයි; appendix එක code reading සහ revision සඳහාය.

app.py · කොටස 1/5

Project ZIP හි ඇතුළත් app.py file එකේ සත්‍ය code lines. Windows GUI, widgets සහ events. මේවා එකම file එකේ ඉදිරියෙන් ඉදිරියට කියවන්න.

ඉගෙනගනිමු

- ✓ මෙම excerpt එකේ functions, names සහ comments හඳුනාගන්න.
- ✓ Code එකේ indent සහ syntax නොවෙනස්ව තබන්න.
- ✓ සම්පූර්ණ file එක source ZIP එකෙන් ලබාගන්න; මේ කොටස තනිව execute කිරීමට අදහස් නොකරයි.

⬡ Visual Studio Code — □ ×

app.py · source excerpt

EXPLORER

LittleStarBMI/
--- app.py
--- bmi_core.py
--- tests/
--- docs/

```
"""Little Star BMI: an educational Windows desktop app (Python/Tkinter).

IMPORTANT: Adult BMI screening only (ages 20+); not a diagnosis.
No personal information is saved or transmitted.
"""

import tkinter as tk
from tkinter import ttk, messagebox
from bmi_core import InputError, calculate_result


def main() -> None:
    """Create the widgets and start the desktop event loop."""
    root = tk.Tk()
    root.title("Little Star BMI | EGOTECHWORLD")
    root.geometry("520x510")
    root.minsize(440, 450)
    root.configure(bg="#ffffff")

    style = ttk.Style(root)
    style.theme_use("clam")
    style.configure("App.TFrame", background="#ffffff")
    style.configure("App.TLabel", background="#ffffff", foreground="#173b47",
                    font=("Segoe UI", 11))
```

✎ දැන් ඔබ කරන්න

කොටස 1 හි අලුත්ම variable හෝ function එකක් තෝරා එහි කාර්යය පැහැදිලි කරන්න.

මතක තබාගන්න: Source ZIP එක runnable project එකයි; appendix එක code reading සහ revision සඳහාය.

app.py · කොටස 2/5

Project ZIP හි ඇතුළත් app.py file එකේ සත්‍ය code lines. Windows GUI, widgets සහ events. මේවා එකම file එකේ ඉදිරියෙන් ඉදිරියට කියවන්න.

ඉගෙනගනිමු

- ✓ මෙම excerpt එකේ functions, names සහ comments හඳුනාගන්න.
- ✓ Code එකේ indent සහ syntax නොවෙනස්ව තබන්න.
- ✓ සම්පූර්ණ file එක source ZIP එකෙන් ලබාගන්න; මේ කොටස තනිව execute කිරීමට අදහස් නොකරයි.

Visual Studio Code

app.py · source excerpt

EXPLORER

LittleStarBMI/
--- app.py
--- bmi_core.py
--- tests/
--- docs/

```
style.configure("Title.TLabel", background="#ffffff", foreground="#124356",
                font=("Segoe UI", 20, "bold"))
style.configure("Note.TLabel", background="#ffffff", foreground="#526773",
                font=("Segoe UI", 9))
style.configure("Result.TLabel", background="#ffffff", foreground="#04737d",
                font=("Segoe UI", 16, "bold"))
style.configure("Accent.TButton", font=("Segoe UI", 10, "bold"), padding=10)
style.configure("Plain.TButton", font=("Segoe UI", 10), padding=10)

main_frame = ttk.Frame(root, style="App.TFrame", padding=24)
main_frame.pack(fill="both", expand=True)
main_frame.columnconfigure(1, weight=1)

ttk.Label(main_frame, text="Little Star BMI", style="Title.TLabel").grid(
    row=0, column=0, columnspan=2, sticky="w", pady=(0, 6))
ttk.Label(main_frame, text="A school programming project, NOT a diagnosis.",
          style="Note.TLabel").grid(row=1, column=0, columnspan=2,
                                    sticky="w", pady=(0, 18))

age_var = tk.StringVar(value="25")
weight_var = tk.StringVar(value="70")
height_var = tk.StringVar(value="170")
result_var = tk.StringVar(value="Enter an adult's sample details.")
category_var = tk.StringVar(value="")
```

✎ දැන් ඔබ කරන්න

කොටස 2 හි අලුත්ම variable හෝ function එකක් තෝරා එහි කාර්යය පැහැදිලි කරන්න.

මතක තබාගන්න: Source ZIP එක runnable project එකයි; appendix එක code reading සහ revision සඳහාය.

app.py · කොටස 3/5

Project ZIP හි ඇතුළත් app.py file එකේ සත්‍ය code lines. Windows GUI, widgets සහ events. මේවා එකම file එකේ ඉදිරියෙන් ඉදිරියට කියවන්න.

ඉගෙනගනිමු

- ✓ මෙම excerpt එකේ functions, names සහ comments හඳුනාගන්න.
- ✓ Code එකේ indent සහ syntax නොවෙනස්ව තබන්න.
- ✓ සම්පූර්ණ file එක source ZIP එකෙන් ලබාගන්න; මේ කොටස තනිව execute කිරීමට අදහස් නොකරයි.

⬡ Visual Studio Code — □ ×

app.py · source excerpt

EXPLORER

LittleStarBMI/
--- app.py
--- bmi_core.py
--- tests/
--- docs/

```
entries = [
    ("Age (years; 20+)", age_var),
    ("Weight (kg)", weight_var),
    ("Height (cm)", height_var),
]
entry_widgets = []
for index, (caption, variable) in enumerate(entries, start=2):
    ttk.Label(main_frame, text=caption, style="App.TLabel").grid(
        row=index, column=0, sticky="w", pady=9, padx=(0, 14))
    entry = ttk.Entry(main_frame, textvariable=variable, width=20)
    entry.grid(row=index, column=1, sticky="ew", pady=9)
    entry_widgets.append(entry)

def do_calculate() -> None:
    """The Calculate button requests a result from pure business logic."""
    try:
        bmi, category = calculate_result(
            age_var.get(), weight_var.get(), height_var.get())
    except InputError as error:
        result_var.set("Please check your input.")
        category_var.set("")
        messagebox.showerror("Input error", str(error), parent=root)
        return
```

✎ දැන් ඔබ කරන්න

කොටස 3 හි අලුත්ම variable හෝ function එකක් තෝරා එහි කාර්යය පැහැදිලි කරන්න.

මතක තබාගන්න: Source ZIP එක runnable project එකයි; appendix එක code reading සහ revision සඳහාය.

app.py · කොටස 4/5

Project ZIP හි ඇතුළත් app.py file එකේ සත්‍ය code lines. Windows GUI, widgets සහ events. මේවා එකම file එකේ ඉදිරියෙන් ඉදිරියට කියවන්න.

ඉගෙනගනිමු

- ✓ මෙම excerpt එකේ functions, names සහ comments හඳුනාගන්න.
- ✓ Code එකේ indent සහ syntax නොවෙනස්ව තබන්න.
- ✓ සම්පූර්ණ file එක source ZIP එකෙන් ලබාගන්න; මේ කොටස තනිව execute කිරීමට අදහස් නොකරයි.

⬡ Visual Studio Code — □ ×

app.py · source excerpt

EXPLORER

LittleStarBMI/
--- app.py
--- bmi_core.py
--- tests/
--- docs/

```
        result_var.set(f"Your sample BMI: {bmi:.1f}")
        category_var.set(category + " (adult screening only)")

    def do_clear() -> None:
        """Remove inputs and output without storing private data."""
        age_var.set("")
        weight_var.set("")
        height_var.set("")
        result_var.set("Enter an adult's sample details.")
        category_var.set("")
        entry_widgets[0].focus_set()

    button_row = ttk.Frame(main_frame, style="App.TFrame")
    button_row.grid(row=5, column=0, columnspan=2, sticky="ew", pady=(16, 14))
    ttk.Button(button_row, text="Calculate", style="Accent.TButton",
               command=do_calculate).pack(side="left", padx=(0, 12))
    ttk.Button(button_row, text="Clear", style="Plain.TButton",
               command=do_clear).pack(side="left")

    ttk.Separator(main_frame).grid(row=6, column=0, columnspan=2,
                                   sticky="ew", pady=(3, 15))
    ttk.Label(main_frame, textvariable=result_var, style="Result.TLabel").grid(
        row=7, column=0, columnspan=2, sticky="w", pady=6)
    ttk.Label(main_frame, textvariable=category_var, style="App.TLabel").grid(
```

✎ දැන් ඔබ කරන්න

කොටස 4 හි අලුත්ම variable හෝ function එකක් තෝරා එහි කාර්යය පැහැදිලි කරන්න.

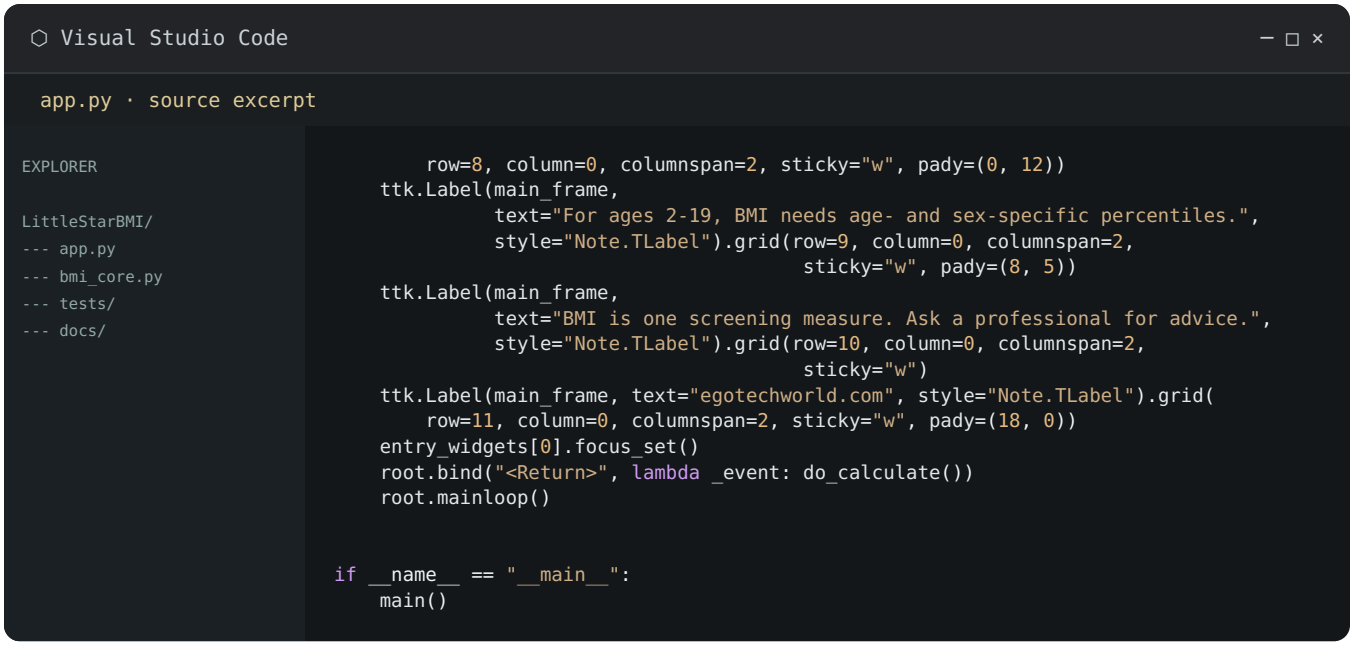
මතක තබාගන්න: Source ZIP එක runnable project එකයි; appendix එක code reading සහ revision සඳහාය.

app.py · කොටස 5/5

Project ZIP හි ඇතුළත් app.py file එකේ සත්‍ය code lines. Windows GUI, widgets සහ events. මේවා එකම file එකේ ඉදිරියෙන් ඉදිරියට කියවන්න.

ඉගෙනගනිමු

- ✓ මෙම excerpt එකේ functions, names සහ comments හඳුනාගන්න.
- ✓ Code එකේ indent සහ syntax නොවෙනස්ව තබන්න.
- ✓ සම්පූර්ණ file එක source ZIP එකෙන් ලබාගන්න; මේ කොටස තනිව execute කිරීමට අදහස් නොකරයි.



✎ දැන් ඔබ කරන්න

කොටස 5 හි අලුත්ම variable හෝ function එකක් තෝරා එහි කාර්යය පැහැදිලි කරන්න.

මතක තබාගන්න: Source ZIP එක runnable project එකයි; appendix එක code reading සහ revision සඳහාය.

tests/test_bmi_core.py · කොටස 1/4

Project ZIP හි ඇතුළත් tests/test_bmi_core.py file එකේ සත්‍ය code lines. Automated unit tests. මේවා එකම file එකේ ඉදිරියෙන් ඉදිරියට කියවන්න.

ඉගෙනගනිමු

- ✓ මෙම excerpt එකේ functions, names සහ comments හඳුනාගන්න.
- ✓ Code එකේ indent සහ syntax නොවෙනස්ව තබන්න.
- ✓ සම්පූර්ණ file එක source ZIP එකෙන් ලබාගන්න; මේ කොටස තනිව execute කිරීමට අදහස් නොකරයි.

⬡ Visual Studio Code — □ ×

tests/test_bmi_core.py · source excerpt

EXPLORER

LittleStarBMI/
--- app.py
--- bmi_core.py
--- tests/
--- docs/

```python
"""Repeatable tests for the BMI calculator's business logic."""
import math
import unittest

from bmi_core import (
    InputError, read_age, read_measurement,
    calculate_bmi, adult_bmi_category, calculate_result,
)


class TestBMI(unittest.TestCase):
    def test_known_bmi(self):
        self.assertAlmostEqual(calculate_bmi(70, 170), 24.221453, places=5)

    def test_another_bmi(self):
        self.assertAlmostEqual(calculate_bmi(90, 180), 27.777778, places=5)

    def test_healthy_result(self):
        bmi, category = calculate_result("25", "70", "170")
        self.assertEqual(category, "Healthy weight range")
        self.assertEqual(f"{bmi:.1f}", "24.2")

    def test_low_boundary(self):
        self.assertEqual(adult_bmi_category(18.4999), "Underweight range")
```

✎ දැන් ඔබ කරන්න

කොටස 1 හි අලුත්ම variable හෝ function එකක් තෝරා එහි කාර්යය පැහැදිලි කරන්න.

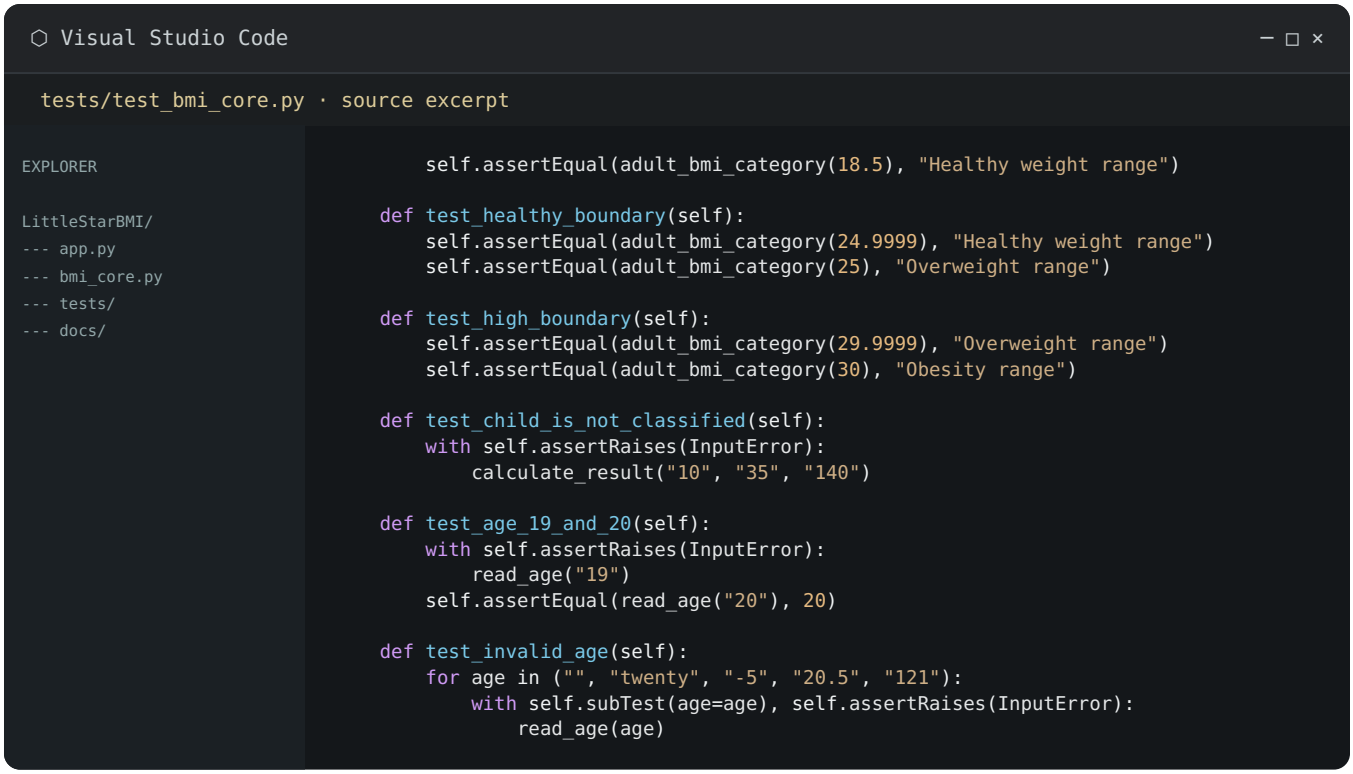
මතක තබාගන්න: Source ZIP එක runnable project එකයි; appendix එක code reading සහ revision සඳහාය.

tests/test_bmi_core.py · කොටස 2/4

Project ZIP හි ඇතුළත් tests/test_bmi_core.py file එකේ සත්‍ය code lines. Automated unit tests. මේවා එකම file එකේ ඉදිරියෙන් ඉදිරියට කියවන්න.

ඉගෙනගනිමු

- ✓ මෙම excerpt එකේ functions, names සහ comments හඳුනාගන්න.
- ✓ Code එකේ indent සහ syntax නොවෙනස්ව තබන්න.
- ✓ සම්පූර්ණ file එක source ZIP එකෙන් ලබාගන්න; මේ කොටස තනිව execute කිරීමට අදහස් නොකරයි.



✎ දැන් ඔබ කරන්න

කොටස 2 හි අලුත්ම variable හෝ function එකක් තෝරා එහි කාර්යය පැහැදිලි කරන්න.

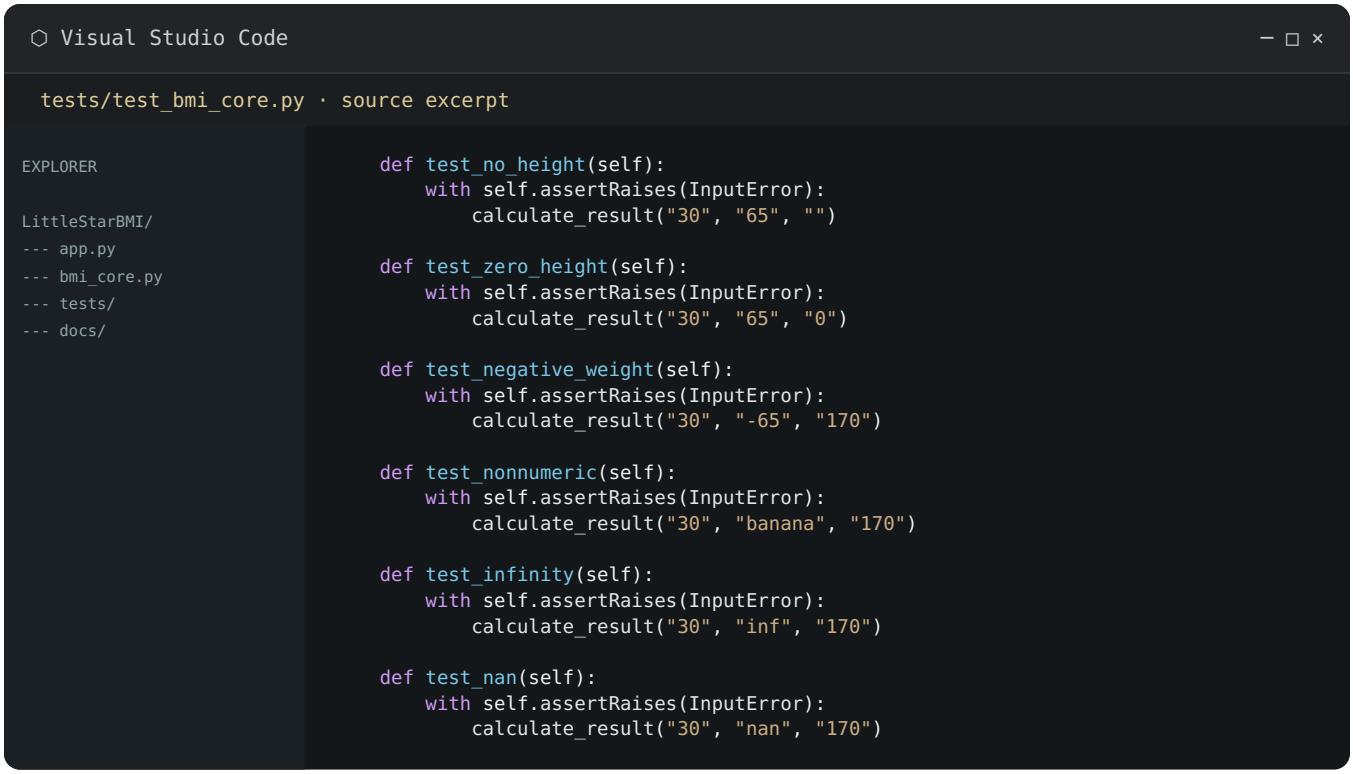
මතක තබාගන්න: Source ZIP එක runnable project එකයි; appendix එක code reading සහ revision සඳහාය.

tests/test_bmi_core.py · කොටස 3/4

Project ZIP හි ඇතුළත් tests/test_bmi_core.py file එකේ සත්‍ය code lines. Automated unit tests. මේවා එකම file එකේ ඉදිරියෙන් ඉදිරියට කියවන්න.

ඉගෙනගනිමු

- ✓ මෙම excerpt එකේ functions, names සහ comments හඳුනාගන්න.
- ✓ Code එකේ indent සහ syntax නොවෙනස්ව තබන්න.
- ✓ සම්පූර්ණ file එක source ZIP එකෙන් ලබාගන්න; මේ කොටස තනිව execute කිරීමට අදහස් නොකරයි.



✎ දැන් ඔබ කරන්න

කොටස 3 හි අලුත්ම variable හෝ function එකක් තෝරා එහි කාර්යය පැහැදිලි කරන්න.

මතක තබාගන්න: Source ZIP එක runnable project එකයි; appendix එක code reading සහ revision සඳහාය.

tests/test_bmi_core.py · කොටස 4/4

Project ZIP හි ඇතුළත් tests/test_bmi_core.py file එකේ සත්‍ය code lines. Automated unit tests. මේවා එකම file එකේ ඉදිරියෙන් ඉදිරියට කියවන්න.

ඉගෙනගනිමු

- ✓ මෙම excerpt එකේ functions, names සහ comments හඳුනාගන්න.
- ✓ Code එකේ indent සහ syntax නොවෙනස්ව තබන්න.
- ✓ සම්පූර්ණ file එක source ZIP එකෙන් ලබාගන්න; මේ කොටස තනිව execute කිරීමට අදහස් නොකරයි.



✎ දැන් ඔබ කරන්න

කොටස 4 හි අලුත්ම variable හෝ function එකක් තෝරා එහි කාර්යය පැහැදිලි කරන්න.

මතක තබාගන්න: Source ZIP එක runnable project එකයි; appendix එක code reading සහ revision සඳහාය.

මම Software Developer කෙනෙක් ලෙස ආරම්භ කළා!

මේක ඔබේ පළමු milestone එක. සාර්ථකත්වය කියන්නේ bugs නැති code එකක් නොව, පරීක්ෂා කර ඉගෙනගන්නා හුරුබුහුටි ගමනක්.

ඉගෙනගනිමු

- ✓ Project: Little Star BMI – Python Windows Desktop.
- ✓ Skills: SDLC, SRS, Python, UI, validation, testing, QA, release.
- ✓ Student name: _____ Date: _____

★

Young Software Developer

SDLC · Python · Tkinter · QA

Name: _____

✎ දැන් ඔබ කරන්න

ඔබගේ නම ලියා ඉගෙනගත් වැදගත්ම lesson එක මිතුරෙකුට පෙන්වන්න.

මතක තබාගන්න: egotechworld.com – 2026 Sinhala Software Development Basics.