Testing Python Functionality for GlassBR

**testCalculations**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Ref** | **Test Name**  **(File Name)** | **Test Purpose** | **Traceability** | **Significant Input** | **Expected Output** | **Why is it failing?/Errors?/Stated improvements?/look further into** | **Where’s info from?** | **Is it a good test?/Notes** |
| 1 | ?  (testCalculations.py) | to make sure returns expected value of pb | Calculations.calc\_pb function uses equations from DD1 (B=…) and IM1 (Pb=…) | defaultInput.txt | *Overall expected result:* 'For the given input parameters, the glass is considered safe' | instead of equality of floats (assertEqual), should use some epsilon error | defaultInput.txt | Y 🡪 since uses `is\_safe`() (checks if Pb < Pbtol and if LR >q 🡪 is glass considered safe?) |
| 2 | ?  (testCalculations2.py) | “ | “ | testInput1.txt | 'For the given input parameters, the glass is considered safe' | “ | testInput1.txt | “ |
| 3 | ?  (testCalculations3.py) | “ | “ | testInput2.txt | 'For the given input parameters, the glass is considered safe' | “ | testInput2.txt | “ |
| 4 | ?  (testCalculations4.py) | “ | “ | testInput3.txt | 'For the given input parameters, the glass is NOT considered safe' | “ | testInput3.txt | “ |
| 5 | ?  (testCalculations5.py) | “ | “ | testInput4.txt | 'For the given input parameters, the glass is NOT considered safe' | “ | testInput4.txt | “ |
| 6 | ?  (testCalculations6.py) | “ | “ | testInput5.txt | 'For the given input parameters, the glass is NOT considered safe' | “ | testInput5.txt | “ |
| 7 | ?  (testCalculations7.py) | “ | “ | testInput6.txt | 'For the given input parameters, the glass is NOT considered safe' | “ | testInput6.txt | “ |

**testCheckConstraints**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Ref** | **Test Name** | **Test Purpose** | **Traceability** | **Significant Input** | **Expected Output** | **Why is it failing?/Errors?/Stated improvements?/look further into** | **Where’s info from?** | **Is it a good test?/Notes** |
| 8 | testCheckConstraints.py | to see if a (i.e. length) > 0 like it should be | Following A1 (glass must be of rectangular shape) 🡪 following physical constraint from Table 2 where “a > 0” and software constraint from Table 2 where “a => dmin” | a = -1600 | InputError: a and b must be greater than 0 |  | testInvalidInput1.txt | Y |
| 9 | testCheckConstraints2.py | to see if b (i.e. breadth) > 0 like it should be | “ 🡪 following physical constraint from Table 2 where “b > 0” and software constraint from Table 2 where “b => dmin” | b = -1500 | InputError: a and b must be greater than 0 |  | testInvalidInput2.txt | “ |
| 10 | testCheckConstraints3.py | to see if 1 < a/b < 5 like it should be | “  🡪 length should pertain to the longer side, following physical constraint from Table 2 where “b < a” | b = 2000 | (a/b=0.8<1)  InputError: a/b must be between 1 and 5 |  | testInvalidInput3.txt | “ |
| 11 | testCheckConstraints4.py | to see if a/b (i.e. aspect ratio) < 5 like it should be | “ 🡪 following software constraint from Table 2 where “a/b < ARmax” | b = 200 | (a/b=8>5)  InputError: a/b must be between 1 and 5 |  | testInvalidInput4.txt | “ |
| 12 | testCheckConstraints5.py | to see if input t value (i.e. nominal thickness) is one of the industrial standard thicknesses | following R1 (t description) | t = 7 | InputError: t must be in  [2.5,2.7,3.0,4.0,  5.0,6.0,8.0, 10.0,12.0,16.0,  19.0,22.0] |  | testInvalidInput5.txt | “ |
| 13 | testCheckConstraints6.py | to see if input w value (i.e. weight of charge) is > minimum permissible input charge weight | following value of wmin (4.5 kg) from Table 3 | w = 3 | InputError: wtnt must be between 4.5 and 910 |  | testInvalidInput6.txt | “ |
| 14 | testCheckConstraints7.py | to see if input w value (i.e. weight of charge) is < maximum permissible input charge weight | following value of wmax (910 kg) from Table 3 | w = 1000 | InputError: wtnt must be between 4.5 and 910 |  | testInvalidInput7.txt | “ |
| 15 | testCheckConstraints8.py | to see if input tnt value (i.e. TNT equivalent factor) | following physical constraint from Table 2 where “TNT > 0” | tnt = -2 | InputError: TNT must be greater than 0 |  | testInvalidInput8.txt | “ |
| 16 | testCheckConstraints9.py | to see if input SD (i.e. Stand off Distance) is > minimum stand off distance permissible for input | following value of SDmin (6 m) from Table 3 | sdx = 0  sdy = 1.0  sdz = 2.0 | InputError: SD must be between 6 and 130 |  | testInvalidInput9.txt |  |
| 17 | testCheckConstraints10.py | to see if input SD (i.e. Stand off Distance) is < maximum stand off distance permissible for input | following value of SDmax (130 m) from Table 3 | sdx = 0  sdy = 200  sdz = 100 | InputError: SD must be between 6 and 130 |  | testInvalidInput10.txt | “ |
| 18 | testCheckConstraints11.py | see 8 | see 8 | a = 0 | InputError: a and b must be greater than 0 |  | testInvalidInput11.txt | “ |
| 19 | testCheckConstraints12.py | see 9 | see 9 | b = 0 | InputError: a and b must be greater than 0 | RuntimeWarning: divide by zero encountered in double\_scalars params.asprat = params.a / params.b | testInvalidInput11.txt | “ |
| 20 | testCheckConstraints13.py | see 15 | see 15 | tnt = 0 | InputError: TNT must be greater than 0 |  | testInvalidInput13.txt | “ |
| 21 | testCheckConstraints14.py | see 10 | see 10 | a = 1500  b = 1500 | (a/b = 1)  "Encountered an unexpected exception" 🡪 why not the same error as 10? |  | testInput7.txt | “ |
| 22 | testCheckConstraints15.py | see 11 | see 11 | a = 7500  b = 1500 | (a/b = 5)  "Encountered an unexpected exception" |  | testInput8.txt | “ |
| 23 | testCheckConstraints16.py | see 13 | see 13 | w = 4.5 | "Encountered an unexpected exception" |  | testInput9.txt | “ |
| 24 | testCheckConstraints17.py | see 14 | see 14 | w = 910 | "Encountered an unexpected exception" |  | testInput10.txt | “ |
| 25 | testCheckConstraints18.py | “ | “ | “ | REMOVE? Or was it supposed to follow the pattern and have tnt = 0? like #15 |  | “ | “ |
| 26 | testCheckConstraints19.py | see 16 | see 16 | sdx = 0  sdy = 6  sdz = 0 | "Encountered an unexpected exception" |  | testInput11.txt | “ |
| 27 | testCheckConstraints20.py | see 17 | see 17 | sdx = 130  sdy = 0  sdz = 0 | "Encountered an unexpected exception" |  | testInput12.txt | “ |

* “Wrong thickness entered -Not a industrial standard” 🡪 due to the 10 i/o 10.0 in defaultInput.txt?
* The error messages aren’t showing up because when the input is invalid, the error is raised, which means assertRaises passes the test
* what is `context.exception.args[0]` actually doing? (from testCheckConstraints)
* look into items in red…
* not really sure whether “Is it a good test” is answered correctly
* 13+14, 8+9+18+19, 10+11+21+22, 15+20, 13+14+23+24, 16+17+26+27 } combine tests?