Learn Python the Fun Way

Liana Bakradze

liana.bakradze@jetbrains.com
@med_vector
About Me

Saint Petersburg
Why Games?

- No fear of failure
- Reward
- Competition
- Visualisation

Icons made by Freepik, Gregor Cresnar from www.flaticon.com are licensed by CC 3.0 BY
1. Code normally executes in the order it's written.
2. Loops repeat a block of code multiple times.
3. Use tab or 4 spaces to indent the lines under the loop.
4. while True:
6. hero.moveRight()
7. # Add a moveLeft command to the loop here
8. hero.moveLeft()
1. Code normally executes in the order it's written.
2. Loops repeat a block of code multiple times.
3. Use tab or 4 spaces to indent the move lines under the loop.
4. while true:
5.   hero.moveRight()
6. # Add a moveLeft command to the loop here
7. hero.moveLeft()
# Code Combat

- Code normally executes in the order it's written.
- Loops repeat a block of code multiple times.
- Use tab or 4 spaces to indent the lines under the loop.

```plaintext
while True:
    hero.moveRight()

# Add a moveLeft command to the loop here
hero.moveLeft()
hero.moveLeft()
```

- MoveDown() commands:
  - while-true loop
  - findNearestEnemy()...
1. Code normally executes in the order it's written.
2. Loops repeat a block of code multiple times.
3. Use tab or 4 spaces to indent the move lines under the loop.
4. while(true):
5. hero.moveRight;
6. # Add a moveLeft command to the loop here
7. hero.moveLeft();
Dodge the fireballs forever.

Under 4 statements.

While loop:

while True:
   hero.moveRight()
   # Add a moveLeft command to the loop here

Move commands:

moveDown()
movLeft()
movRight()
movUp()

Attack command:

attack(target)
Defeat the ogres. (2/3)

- Your hero must survive.
- Bonus: no code problems.

GOALS: FAILING

Fix Your Code

Line 8: attack's argument target should have type unit, but got string: "Ack1". Attack "Ack", not "Ack1".

Your hero will need more armor to win this fight! Go buy the tarnished bronze breastplate.

Open Item Shop
<table>
<thead>
<tr>
<th>Role</th>
<th>Home</th>
<th>Team</th>
</tr>
</thead>
<tbody>
<tr>
<td>Archmage Home</td>
<td>Coders!</td>
<td></td>
</tr>
<tr>
<td>Artisan Home</td>
<td>Builders!</td>
<td></td>
</tr>
<tr>
<td>Adventurer Home</td>
<td>Testers!</td>
<td></td>
</tr>
<tr>
<td>Scribe Home</td>
<td>Scribblers!</td>
<td></td>
</tr>
<tr>
<td>Diplomat Home</td>
<td>Translators!</td>
<td></td>
</tr>
<tr>
<td>Ambassador Home</td>
<td>Supporters!</td>
<td></td>
</tr>
<tr>
<td>Category</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------</td>
<td></td>
</tr>
<tr>
<td>Archmage Home</td>
<td>Coders!</td>
<td></td>
</tr>
<tr>
<td>Artisan Home</td>
<td>Builders!</td>
<td></td>
</tr>
<tr>
<td>Adventurer Home</td>
<td>Testers!</td>
<td></td>
</tr>
<tr>
<td>Scribe Home</td>
<td>Scribblers!</td>
<td></td>
</tr>
<tr>
<td>Diplomat Home</td>
<td>Translators!</td>
<td></td>
</tr>
<tr>
<td>Ambassador Home</td>
<td>Supporters!</td>
<td></td>
</tr>
<tr>
<td>Archmage Home - Coders!</td>
<td>Artisan Home - Builders!</td>
<td>Adventurer Home - Testers!</td>
</tr>
<tr>
<td>-------------------------</td>
<td>--------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td><img src="image1" alt="Archmage Home - Coders!" /></td>
<td><img src="image2" alt="Artisan Home - Builders!" /></td>
<td><img src="image3" alt="Adventurer Home - Testers!" /></td>
</tr>
<tr>
<td>Scribe Home - Scribblers!</td>
<td>Diplomat Home - Translators!</td>
<td>Ambassador Home - Supporters!</td>
</tr>
<tr>
<td><img src="image4" alt="Scribe Home - Scribblers!" /></td>
<td><img src="image5" alt="Diplomat Home - Translators!" /></td>
<td><img src="image6" alt="Ambassador Home - Supporters!" /></td>
</tr>
</tbody>
</table>
Onboarding

Python3

```python
import sys
import math

# CodinGame planet is being attacked by slimy insectoid aliens.
# :-(
# Hint: To protect the planet, you can implement the pseudo-code provided in the

# game loop
while True:
    enemy_1 = int(input())  # name of enemy 1
    dist_1 = int(input())  # distance to enemy 1
    enemy_2 = int(input())  # name of enemy 2
    dist_2 = int(input())  # distance to enemy 2

    # Write an action using print
    # To debug: print("Debug messages...", file=sys.stderr)

    # You have to output a correct ship name to shoot ("Buzz", enemy1, enemy2)
    if dist_1 < dist_2:
        print(enemy_1)
    else:
        print(enemy_2)
```

The Goal

Your program must destroy the enemy ships by shooting the closest enemy on each

Console output

Game information, Debug...

Standard Output Stream:
> Sectoid
Game information:
Sectoid has been targeted
Threats within range:
HardHat 50m
Buzz 50m

Standard Output Stream:
> HardHat
Game information:
HardHat has been targeted
The Goal

Your armor must destroy the enemy ships by shooting the closest enemy on each side.

Console output: Game information, Debug...

Standard Output Stream:
> Sectoid
  Game information:
  Sectoid has been targeted
  Threats within range:
  HardHat 50m
  Buzz 50m

Standard Output Stream:
> HardHat
  Game information:
  HardHat has been targeted

Python3
```python
import sys
import math

# CodinGame planet is being attacked by alien insectoid aliens.
# ...
# Hint: To protect the planet, you can implement the pseudo-code provided in the
# game loop.

while True:
  enemy_1 = input()  # name of enemy 1
  dist_1 = int(input())  # distance to enemy 1
  enemy_2 = input()  # name of enemy 2
  dist_2 = int(input())  # distance to enemy 2

  # Write an action using print
  # To debug: print("Debug message...", file=sys.stdout)

  if dist_1 < dist_2:
    print(enemy_1)
  else:
    print(enemy_2)
```

Test cases:

- 01: imminent danger
- PLAY TESTCASE
- 05/16
- 06/16

Actions:
- PLAY ALL TESTCASES
- SUBMIT
Onboarding

The Goal

Your name.. must destroy the enemy ships by shooting the closest enemy on each

Console output

Standard Output Stream:
> Sectorid
Game information:
Sectorid has been targeted
Threats within range:
Hardhat 50m
Buzz 50m

Standard Output Stream:
> Hardhat
Game information:
Hardhat has been targeted

Test cases

01 imminent danger

Actions

PLAY TESTCASE

SUBMIT
Onboarding

The Goal

Your mission is to destroy the enemy ships by shooting the closest enemy on each turn.

Console output

Standard Output Stream:
> Sectoid
Game information:
Sectoid has been targeted
Threats within range:
HardHat 50m
Buzz 50m

Standard Output Stream:
> HardHat
Game information:
HardHat has been targeted

Python

```python
import sys
import math

# CodinGame planet is being attacked by alien insectoid aliens.
# In this mission:
# - Destroy the planet: you can implement the pseudo-code provided in the
#   code editor:
#   - while True:
#       enemy_1 = input() # name of enemy 1
#       dist_1 = int(input()) # distance to enemy 1
#       # enemy_2 = input() # name of enemy 2
#       dist_2 = int(input()) # distance to enemy 2
#   # Write an action using print
#   # To debug: print("Debug messages...", file=sys.stdout)
#   if dist_1 < dist_2:
#       print(enemy_1)
#   else:
#       print(enemy_2)
```

Test cases

01: Imminent danger

Actions

PLAY ALL TESTCASES

SUBMIT
```
import sys
import math

# CodinGame planet is being attacked by slimy insectoid aliens.
# Hint: To protect the planet, you can implement the pseudo-code provided in the
# game loop:
while True:
    enemy_1 = input()  # name of enemy 1
    dist_1 = int(input())  # distance to enemy 1
    enemy_2 = input()  # name of enemy 2
    dist_2 = int(input())  # distance to enemy 2

    # Write an action using print
    # To debug: print("Debug messages...", file=sys.stderr)

    # You have to output a correct ship name to shoot ("Buzz", enemy_1, enemy_2,
    # if dist_1 < dist_2:
    #     print(enemy_1)
    # else:
    print(enemy_2)
```

**The Goal**

Your program must destroy the enemy ships by shooting the closest enemy on each

**Console output**

> Hitbot
  Game information:
  Hitbot has been targeted
  Threats within range:
  DangerDart 35m

> DangerDart
  Game information:
  DangerDart 35m

**Test cases**

 extinction danger   | PLAY TESTCASE
----------------------|-------------------
 01 Imminent danger   | PLAY TESTCASE
----------------------|-------------------

**Actions**

- PLAY ALL TESTCASES
- SUBMIT
Create your own puzzle!

The puzzles below were created by the CodinGame Community. Why don't you create yours? :)

Quaternion Multiplication by TheNinja

35 CodinGamers have completed this game

Queneau Numbers by [CG]VonRickroll
Welcome home! Take your time getting all pumped up and ready for the journey that's about to begin.

Collect points to unlock new stations

- **Non-unique Elements**
  Trim an array down to its non-unique elements
  structures

- **Median**
  Find the mathematical median in a list of numbers
  numbers statistics

- **House password**
  Check the strength of your favorite password
  text

- **Moore Neighbourhood**
A median is a numerical value separating the upper half of a sorted array of numbers from the lower half. In a list where there are an odd number of entities, the median is the number found in the middle of the array. If the array contains an even number of entities, then there is no single middle value, instead the median becomes the average of the two numbers found in the middle. For this mission, you are given a non-empty array of natural numbers (X). With it, you must separate the upper half of the numbers from the lower half and find the median.

**Input:** An array as a list of integers.

**Output:** The median as a float or an integer.

**Example:**
```python
def checkio(data):
    # These 'asserts' using only for self-checking and not necessary for auto-testing
    if __name__ == '__main__':
        assert checkio([1, 2, 3, 4, 5]) == 3, "Sorted list"
        assert checkio([1, 2, 3, 4, 5, 6]) == 4, "Not sorted list"
        assert checkio([1, 3, 400, 2, 300, 1]) == 2, "It's not an average"
        assert checkio([3, 6, 28, 99, 10, 15]) == 12.5, "Even length"
        print("Start the long test")
        assert checkio(list(range(1000000))) == 499999.5, "Long."
        print("The local tests are done.")
```

A median is a numerical value separating the upper half of a sorted array of numbers from the lower half. In a list where there are an odd number of entities, the median is the number found in the middle of the array. If the array contains an even number of entities, then there is no single middle value, instead the median becomes the average of the two numbers found in the middle. For this mission, you are given a non-empty array of natural numbers (x). With it, you must separate the upper half of the numbers from the lower half and find the median.

Input: An array as a list of integers.

Output: The median as a float or an integer.

Precondition:
1 <= len(data) <= 1000
all(0 <= x < 10 ** 6 for x in data)

LianaBakradze
I have no idea how to start solving this mission

bryukh
As first you need to define the length of the array. You can use the built-in function len().

LianaBakradze
I need some help to proceed with the mission

bryukh
With the built-in operator "modulus" you can define if a number even or odd.
```
if n % 2 == 1: # if odd
    ...
```

LianaBakradze
I am gone halfway through. Need help!

bryukh
If you are using python3, then don't forget that "/" is a real division, "/" is an integer division.

LianaBakradze
I am stuck. I need a small hint.

bryukh
Don't forget to sort the array.
def checkio(data):
    return data[0]

# These 'asserts' using only for self-checking and not necessary for auto-testing
assert checkio([1, 2, 3, 4, 5]) == 3, "Sorted list"
assert checkio([1, 3, 2, 4, 5]) == 3, "Not sorted list"
assert checkio([1, 3, 2, 4, 5, 6, 7, 8]) == 4, "It's not an average"
assert checkio([1, 3, 2, 4, 5, 6, 7]) == 4.25, "Even length"
print("Start the test")
assert checkio(list(range(1000000))) == 499999.5, "Long." 
print("The local tests are done.")
```python
def checkio(data):
    #replace this for solution
    return data[0]

#These "asserts" using only for self-checking and not necessary for auto-testing
assert checkio([1, 2, 3, 4, 5]) == 3, "Sorted list"
assert checkio([3, 1, 2, 5, 4]) == 2, "Not sorted list"
assert checkio([1, 300, 3, 200, 1]) == 2, "'It's not an average"
assert checkio([3, 6, 20, 99, 10, 15]) == 12.5, "Even length"
print("Start the long test")
assert checkio(list(range(1000000))) == 499999.5, "Long."
print("The local tests are done.")
```

A median is a numerical value separating the upper half of a sorted array of numbers from the lower half. In a list where there are an odd number of entries, the median is the number found in the middle of the array. If the array contains an even number of entries, then there is no single middle value, instead the median becomes the average of the two numbers found in the middle. For this mission, you are given a non-empty array of natural numbers (N). With it, you must separate the upper half of the numbers from the lower half and find the median.

Input: An array as a list of integers.
Output: The median as a float or an integer.

Precondition:
1 ≤ len(data) ≤ 1000
all(0 ≤ x ≤ 10 ** 6 for x in data)
def checkio(data):
    # These 'asserts' using only for self-checking and not necessary for auto-testing
    if __name__ == '__main__':
        assert checkio([1, 2, 3, 4, 5]) == 3, "Sorted list"
        assert checkio([1, 1, 1, 2, 3]) == 2, "Not sorted list"
        assert checkio([1, 3, 2, 20, 1]) == 2, "It's not an average"
        assert checkio([1, 6, 8, 9, 10, 15]) == 12.5, "Even length"
        print("Start the test")
        print(checkio(list(range(1000000))))
        print("The local tests are done.")

    return data[0]
def checkio(data):
    # These "asserts" are used for self-checking and are not necessary for auto-testing
    if __name__ == '__main__':
        assert checkio([1, 2, 3, 4, 5]) == 3, "Sorted list"
        assert checkio([3, 1, 2, 5, 3]) == 3, "Not sorted list"
        assert checkio([1, 300, 2, 200, 1]) == 2, "It's not an average"
        assert checkio([3, 6, 20, 99, 10, 15]) == 12.5, "Even length"
        print("Start the long test")
        assert checkio(list(range(1000000))) == 499999.5, "Long."
        print("The local tests are done.")

Click on "Run Code" to view results or Ctrl + /
Click on "Save" to save your code or Ctrl + S
Base CheckiO template for users task

79 commits 1 branch 0 releases 3 contributors

Branch: master New pull request

Bryukh Change jscolors

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
<th>Author</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>editor</td>
<td>Change jscolors</td>
<td>Bryukh</td>
<td>2 years ago</td>
</tr>
<tr>
<td>hints</td>
<td>typo</td>
<td>Bryukh</td>
<td>2 years ago</td>
</tr>
<tr>
<td>info</td>
<td>Change jscolors</td>
<td>Bryukh</td>
<td>2 years ago</td>
</tr>
<tr>
<td>verification</td>
<td>small referee tests adjustments</td>
<td>Bryukh</td>
<td>3 years ago</td>
</tr>
<tr>
<td>.gitignore</td>
<td>Fix error handling with new data</td>
<td>Bryukh</td>
<td>3 years ago</td>
</tr>
<tr>
<td>README.md</td>
<td>Update README.md</td>
<td>Bryukh</td>
<td>3 years ago</td>
</tr>
</tbody>
</table>

checkio-task-template
Conclusion

• 3 great projects
• try yourself
• contribute
• invent something cool

image: www.freeimages.co.uk
Thank You!

Liana Bakradze

liana.bakradze@jetbrains.com
@med_vector