2.5 FIXING BUGS

bug squish frenzy

2.5 FIXING BUGS

- A bug is another way to refer to an **error**.
- Errors may be confusing at times, but they tell you what's wrong with your code.
- The **Command Prompt** or Terminal window is the go-to place for seeing error details.
- The error message will contain a **line number** such as Ln:12. This is your map to finding the error.

FIXING BUGS

In Python, errors are shown in the Command ___ prompt ___ or Terminal window.

A ____ bug ____ is another way to refer to an error.

When examining bugs, it's useful to check the ____ line number.

syntax

TYPES OF ERRORS

SYNTAX ERROR

This usually means you typed something wrong.

Maybe you misspelled a word or missed part of a statement?

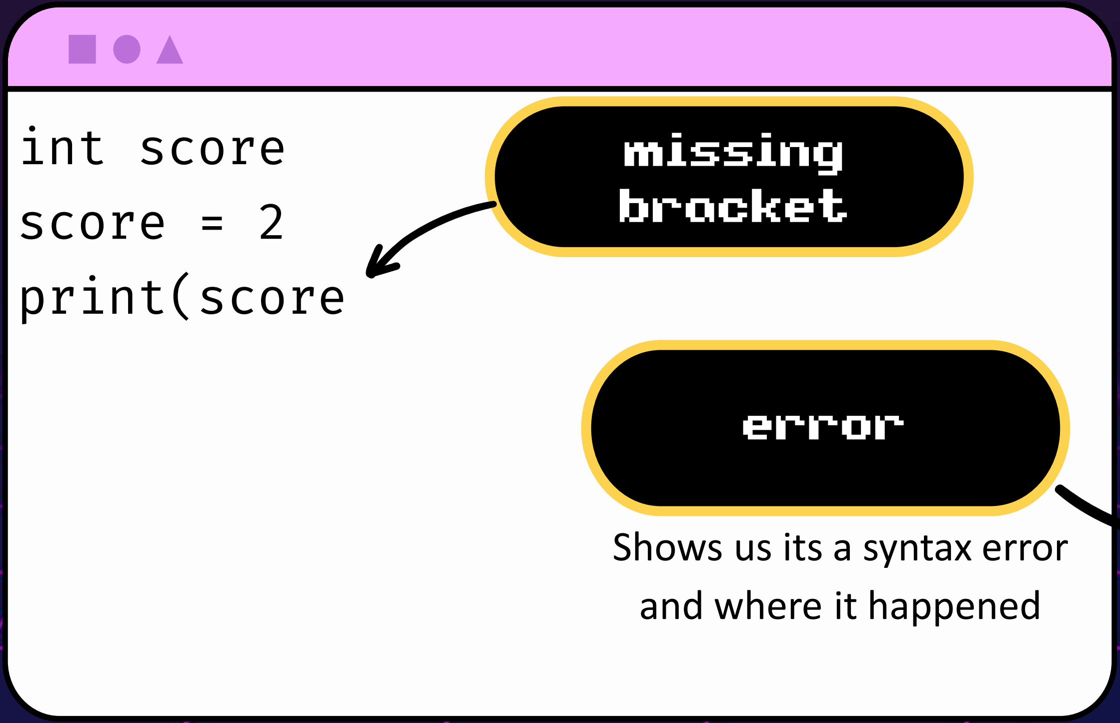
These mistakes are the easiest to fix.

INDENTATION

Python is particularly picky about indentation. You'll get an error if it's off!

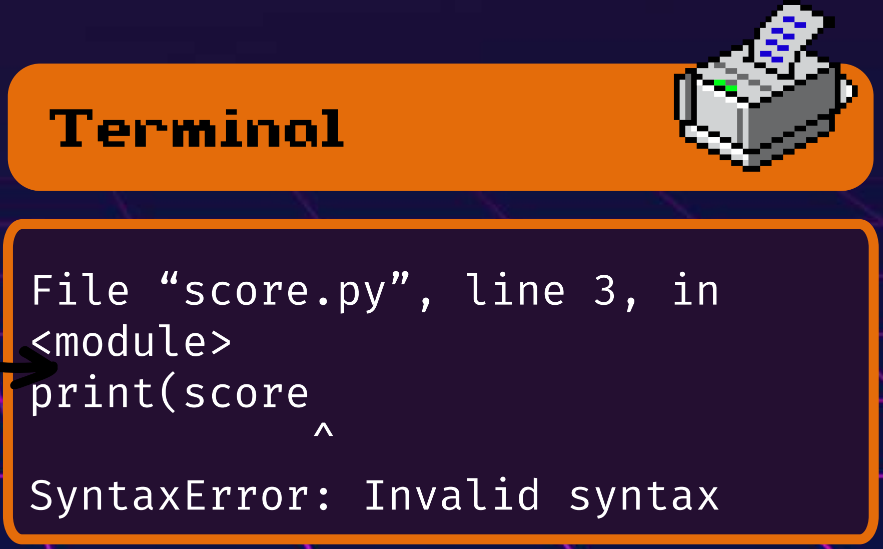
If a line of code ends with a colon, you must indent the next line.

SYNTAX ERRORS



```
int score
score = 2
print(score
```

Shows us its a syntax error and where it happened



```
File “score.py”, line 3, in
<module>
print(score
           ^
SyntaxError: Invalid syntax
```

SYNTAX ERRORS

```
int score
scre = 2
print(score)
```

misspelled variable

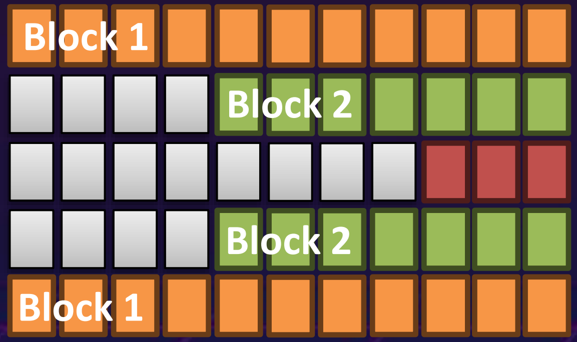
error

Shows us its a syntax error and where it happened

Terminal

```
File “score.py”, line 2, in
<module>
scre = 2
^
SyntaxError: Invalid syntax
```

INDENTATION ERRORS



```
int score
score = 2
if score>1:
print(score)
```

missing indentation

should be..

```
if score>1:
____print(score)
```

error

Shows us it’s an indentation error and where it happened

Terminal

```
File “score.py”, line 4, in
<module>
Unexpected indented block
```

BUG SWATTER

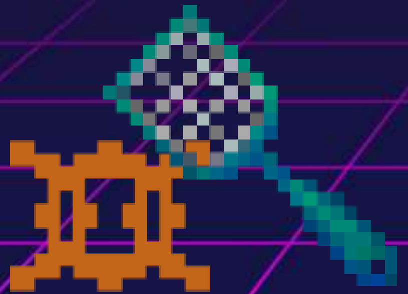
Find the line that has the bug and identify the type.

```
>>> if(score > 80):
>>> print("You passed Level 1.")
```

A. Syntax Error

B. Indentation Error

C. Type Error



Multiple Choice

BUG SWATTER

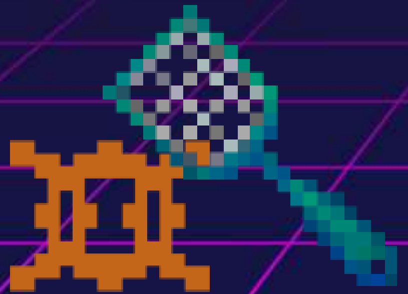
Find the line that has the bug and identify the type.

```
>>> if(lives == 0):
>>>     # Display game over message
>>>     print ("Game Over! Try again")
```

A. Syntax Error

B. Indentation Error

C. Type Error



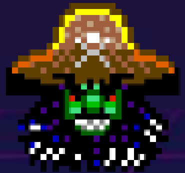
Multiple Choice

MORE TYPES OF ERRORS

TYPE ERRORS

These occur when you use data types incorrectly in the code.

Example: putting a number in a string.



LOGIC ERRORS

These are the hardest errors to fix.

They often may not even raise an error in the terminal.

The code will work but the logic will not make sense.

TYPE ERRORS

str(score)

this gives a string

```
# Player's Score Display
score = 1000
print("Your score is: " + score)
```

*

not a string

error

Shows us this is a type error

Terminal

```
File “score.py”, line 3, in
<module>
print("Your score is: " +
score)

TypeError: can only
concatenate str (not "int")
to str
```

LOGIC ERRORS

```
# Calculating Player's Health after an Attack
player_health = 100
attack_damage = 50

# Player is supposed to take damage from an attack
player_health = player_health + attack_damage
print("Player's health after the attack:",
player_health)
```

where's the error?

Although the code runs fine, this line does not make sense, since the player health is supposed to decrease not increase

Terminal

```
Player's health after the
attack: 150
```

no error message

BUG SWATTER

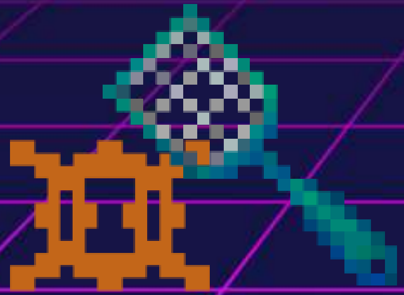
Find the line that has the bug and identify the type.

```
>>># stored as a string instead of an integer
>>>health_potions = "3"
>>>if health_potions* > 3:        int(health_potions)
>>>    print("You have enough health potions.")
>>>else:
>>>    print("You need more health potions.")
```

A. Syntax Error

B. Indentation Error

C. Type Error



Multiple Choice

Did you understand?

GUESS WHICH

Identify the right error type for this piece of code

```
# Checking the status of the player
status = "active"
if (status == "active")
    print("Player is currently active")
```

Fill in the Blanks

Terminal

```
File “score.py”, line _, in
<module>
if (status == "active")
                       ^

______Error: ....error
details...
```

Syntax

Logic

Type

Code

FIXING BUGS

The easiest errors to identify are usually _ syntax _ errors.

When we have a _ logical _ error, our code will run but the output will not be correct.

When we add a string to an int, it results in a _ type error

naming

Did you understand?

GUESS WHICH

Identify the right error type for this piece of code

```
# Checking if Player has Enough Coins
coins = "100"
if coins > 50:
    print("You have enough coins!")
```

Terminal

```
File “score.py”, line 3, in
<module>
if coins > 50:

_____Error: '>' not supported
between instances of 'str'
and 'int'
```

Syntax

Logic

Type

Code

Fill in the Blanks

LOGIC ERRORS

```
# Checking if the player has unlocked a new level
current_level = 5
experience_points = 2500

# The player needs 3000 experience points to reach the next level
if experience_points > 3000:
    current_level = current_level - 1
    print("Congratulations! You've reached level", current_level)
```

where's the error?

Terminal

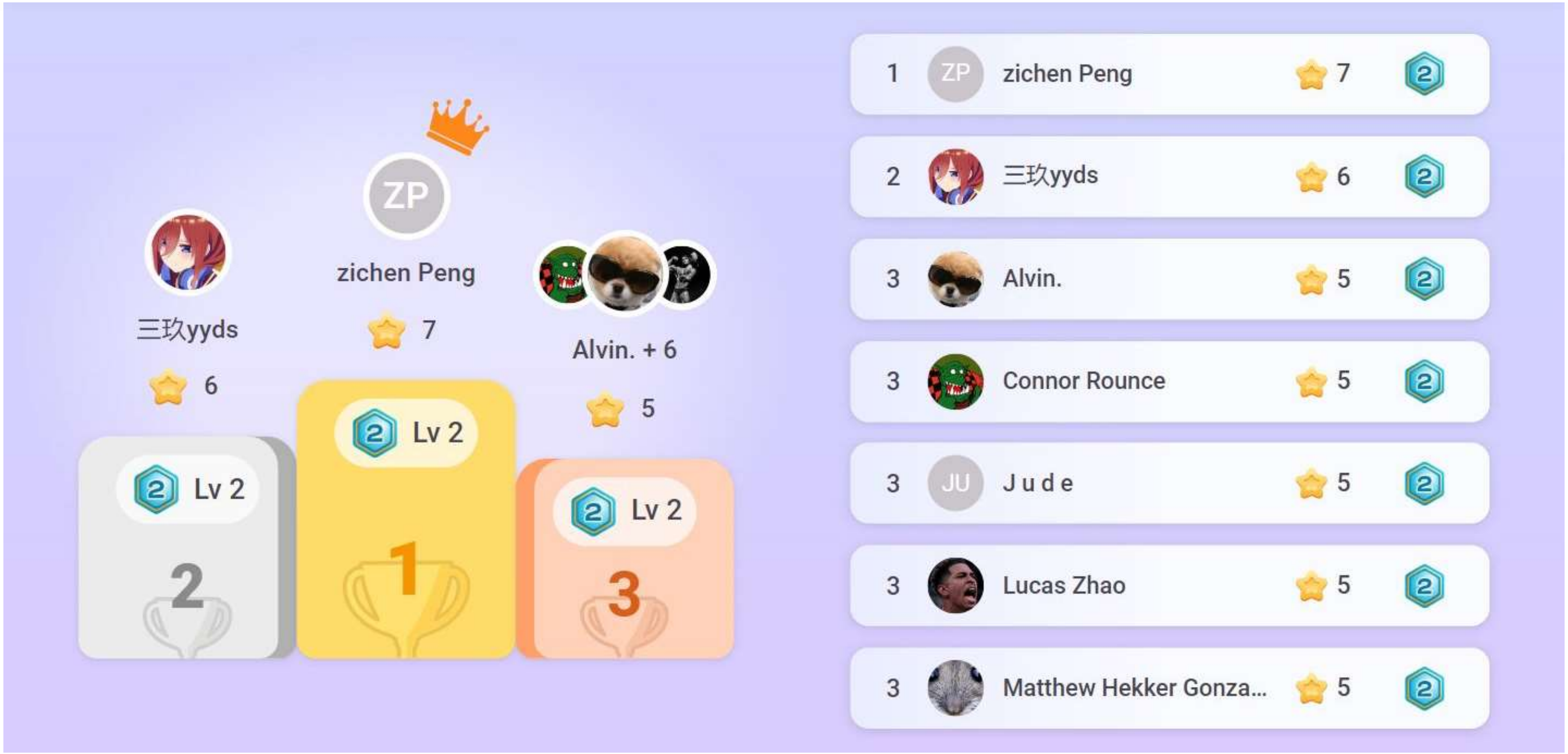
```
Congratulations! You've reached level 4
```

no error message

LESSON CHALLENGE

- Time to put the theory into practice.
- You will intentionally put bugs in an I Spy game.
- In each case, you need to have a look at the command prompt output and verify the bug.
- Find your tasks!

Leader Board



1	zichen Peng	7	2
2	三玖yyds	6	2
3	Alvin.	5	2
3	Connor Rounce	5	2
3	J u d e	5	2
3	Lucas Zhao	5	2
3	Matthew Hekker Gonza...	5	2