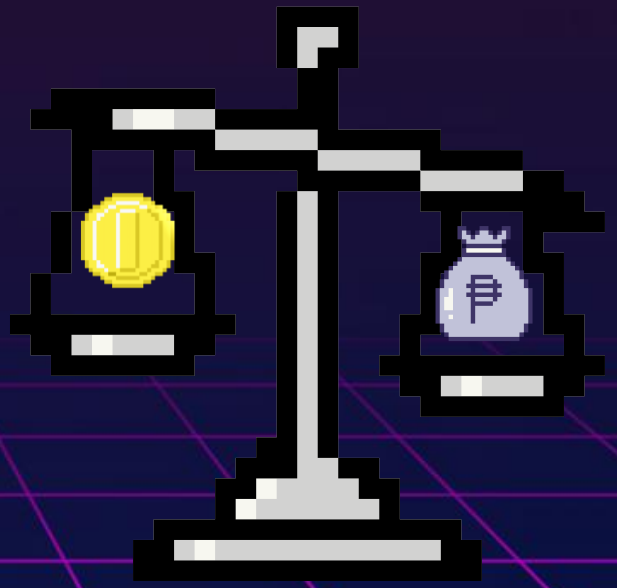
2.2
MAKING DECISIONS

which magic door should we go through?

2.2 MAKING DECISIONS

- Decision-making is an important part of our thinking process.
- We often make decisions based on answers to questions, for example, "Do I have any lives yet?".
- Making these kind of decisions help us take a particular path.
- Computers can make simple decisions by comparing two values.



2.2 MAKING DECISIONS

An answer to a question can have one of two values True or False.

A question is a Boolean expression that makes one or more comparisons.

We can create variables to label and store answers to these questions.

False | variables | one | Boolean

BOOLEAN EXPRESSIONS

```
player_age = 7
are_they_equal = (player_age == 10)
print(are_they_equal)
```

Equal to

We can compare two values by using logical operators. This is the equal to == operator.

```
False
```

False

GUESS THE OUTPUT

```
spells = 11
is_enchanter = (spells > 10)
print(is_enchanter)
```

question

A. True

B. False

C. 22

Multiple Choice

LOGICAL OPERATORS

==

Equal To

Are the two values equal to each other?

!=

Not Equal To

Are the two values not equal to each other?

<

Less Than

Is the value on the left less than the value on the right?

>

Greater Than

Is the value on the left greater than the value on the right?

State the output

```
monsters = 3
coins = 4

print(coins < monsters)
```

Less Than

Word Cloud

MORE COMPLEX LOGIC

To ask more complex questions in one statement we can use logic **comparisons**

Two examples of these are _ **'and'** and 'or'.

We can use these to join multiple **gates** __ and get one answer.

equal to

What is happening?

```
monsters = 3
coins = 4
is_game_over = (monsters == 4) or (coins == 0)
print(is_game_over)
```

Logic Gate

False

Short Answer

When using or, only one of the comparisons needs to be True for the final answer to be True. In this example, both comparisons are False, so the final answer is False.

INTERPRET CODE LIKE A PRO

For the game to be over the value of monters needs to be 4 or the number of coins needs to be 0. We create a variable to store the answer of two comparisons joined using the or operator. For the game to be over only one of the comparisons needs to be True.

State the output

```
monsters = 3
coins = 4
free_weapon = (monsters == 0) and (coins > 50)
print(free_weapon)
```

Logic Gate

When using and both comparisons need to be True for the final answer to be True.

Word Cloud

LESSON CHALLENGE

- Time to put the theory into practice.
- You build small components of trivial games like

  Rock, Paper, Scissors.
- You must use all you learned so far.
- Find your tasks!

2.2 MAKING DECISIONS

The questions help computers decide which code snippets need to run.

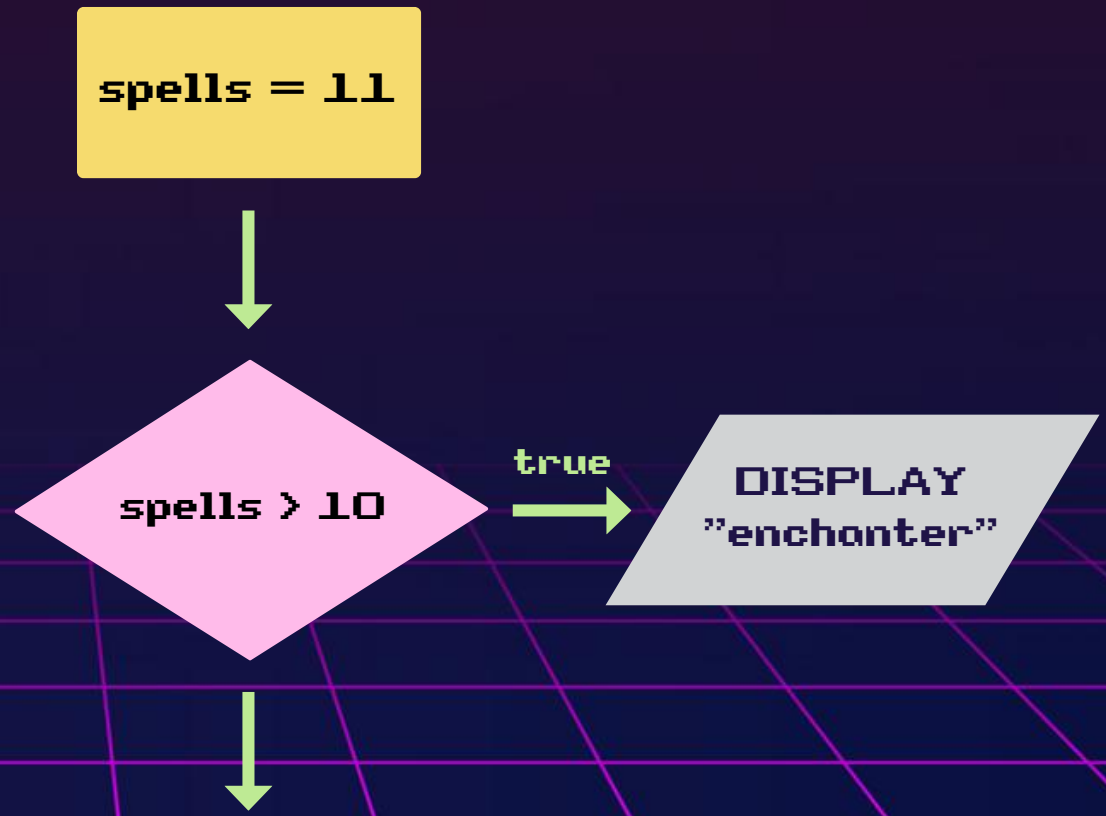
A piece of code will run when a condition or question is True

To select one option out of multiple outcomes we need to use the branching.

or operator

IF STATEMENT

```
spells = 11
if spells > 10:
    print("Enchanter Unlocked")
# more code ...
```



One Branch

The condition is spells > 10 and when True the program prints "Enchanter Unlocked"

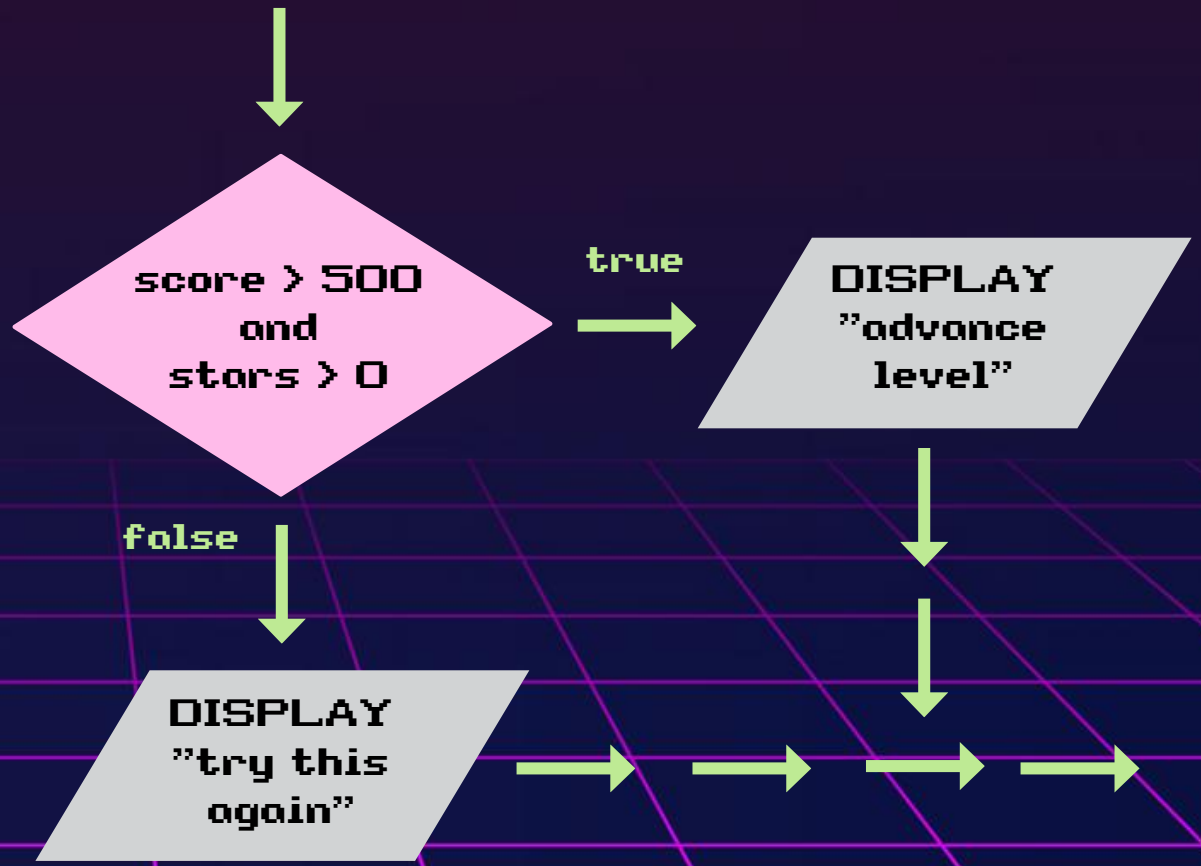
IF...ELSE BRANCHING

logical

```
if score > 500 and stars > 0:
    print("Advance a level.")
else:
    print("Try this again.")
# more code ...
```

Two Branches

We have two possibilities, the True path or the False path. This is the if-else statement.

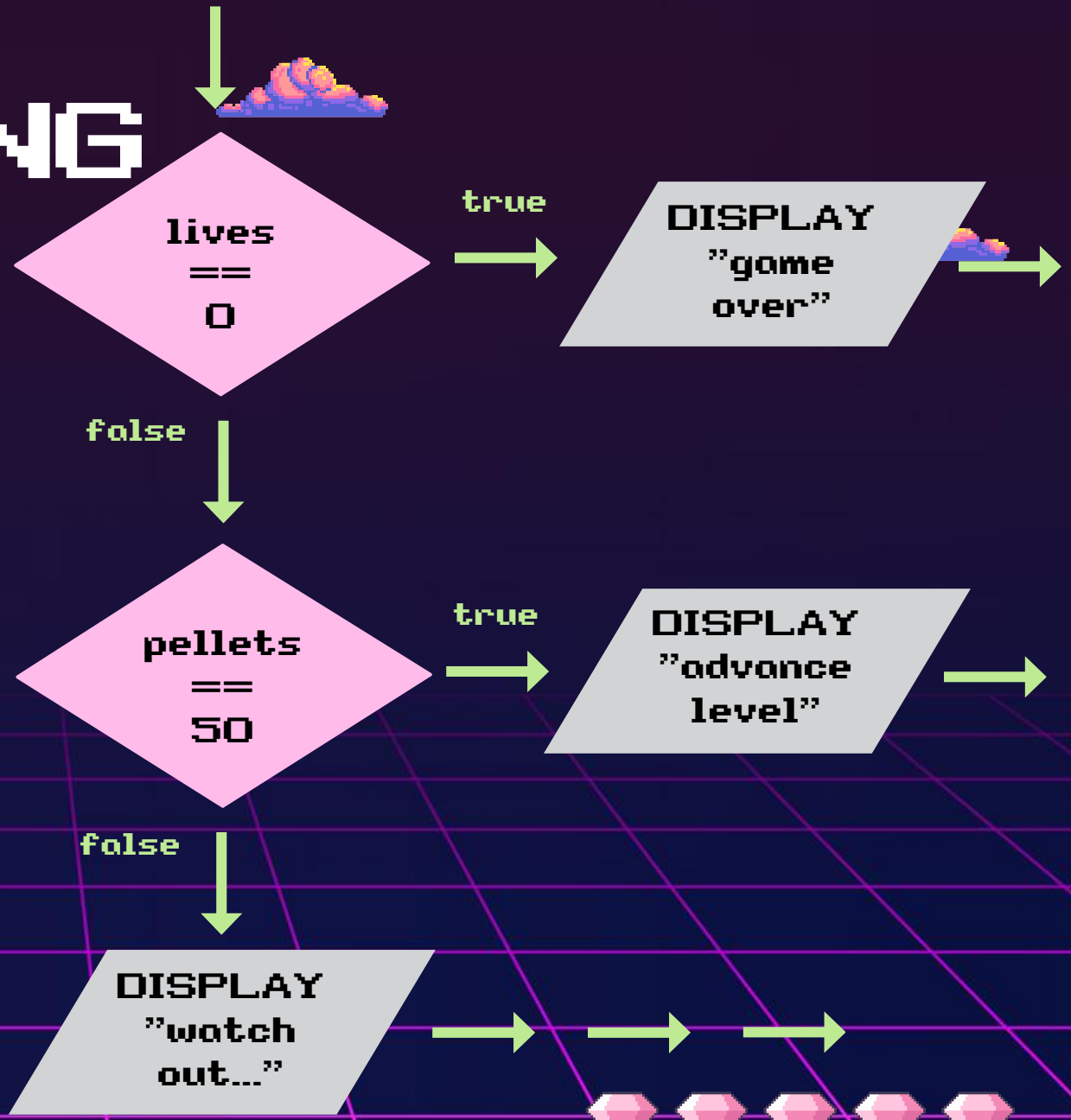


ELIF BRANCHING

```
if lives == 0:
    print("Game over.")
elif pellets == 50:
    print("Advance level.")
else:
    print("Watch out for ghosts.")
# more code ...
```

Three Branches

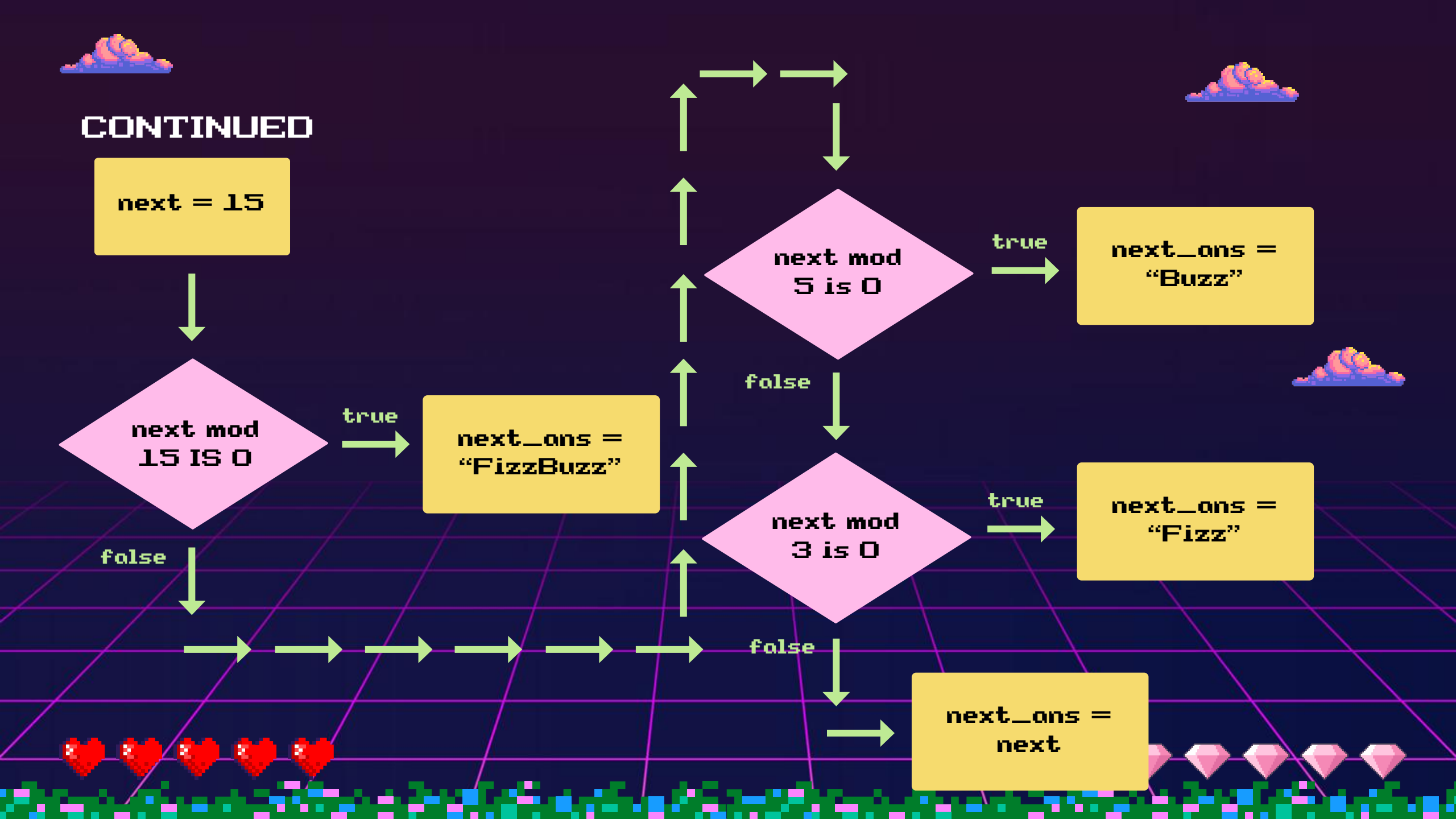
With three or more outcomes we need to use elif which is short for else-if



DID YOU UNDERSTAND?

COMPLETE THE PROGRAM

Fill in the blanks of the code snippet for a game called FizzBuzz. Imagine that the user is given the beginning of a sequence and needs to input the next correct 10 numbers one after the other to complete it. However, for multiples of 3 the user must type "Fizz", for multiples of 5 it must type "Buzz", and, for multiples of 15 the user must type "FizzBuzz".


CONTINUED
next = 15
next mod 15 IS 0
true
next_ons = “FizzBuzz”
false
next mod 5 is 0
true
next_ons = “Buzz”
false
next mod 3 is 0
true
next_ons = “Fizz”
false
next_ons = next

CONTINUED

elif

elif

next

else

==

Fill in the Blanks

```
next = 15
next_ans = ""
if next % 15 __==__ 0:
    next_ans = "FizzBuzz"
_elif_____ next % 5 == 0:
    next_ans = "Buzz"
_elif_____ next % 3 == 0:
    next_ans = "Fizz"
_else_____:
    next_ans = _next____
```

mod operator (remainder)

equal to

LESSON CHALLENGE

- Time to put the theory into practice.
- You will continue building a small word search game component.
- You must use all you learned so far.
- Find your tasks!