



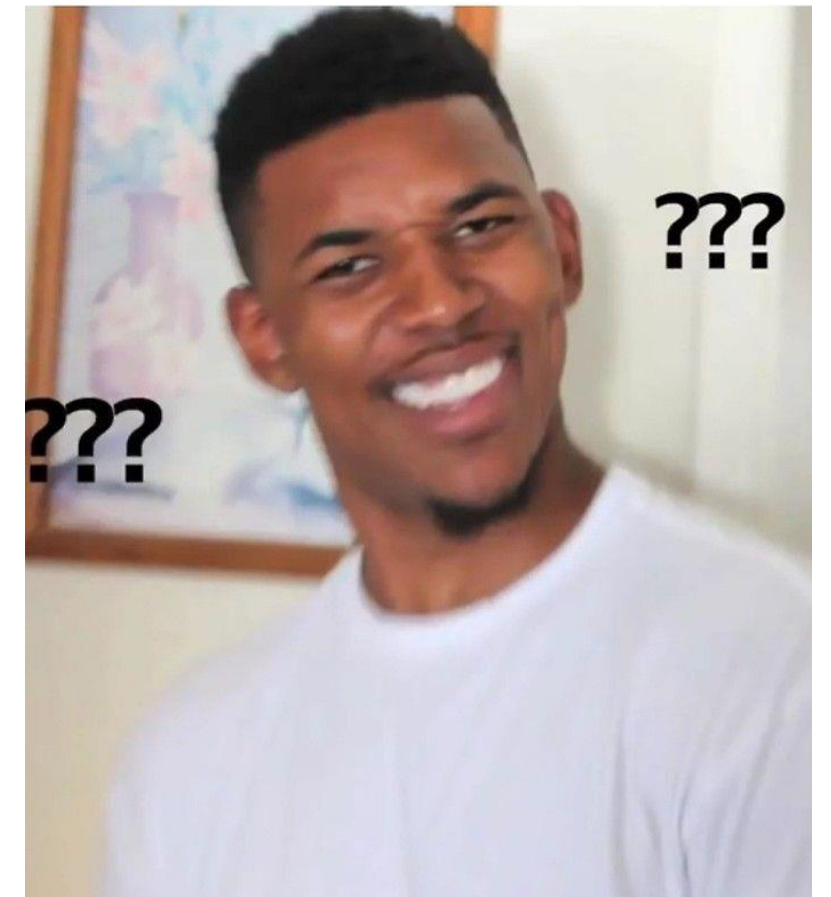
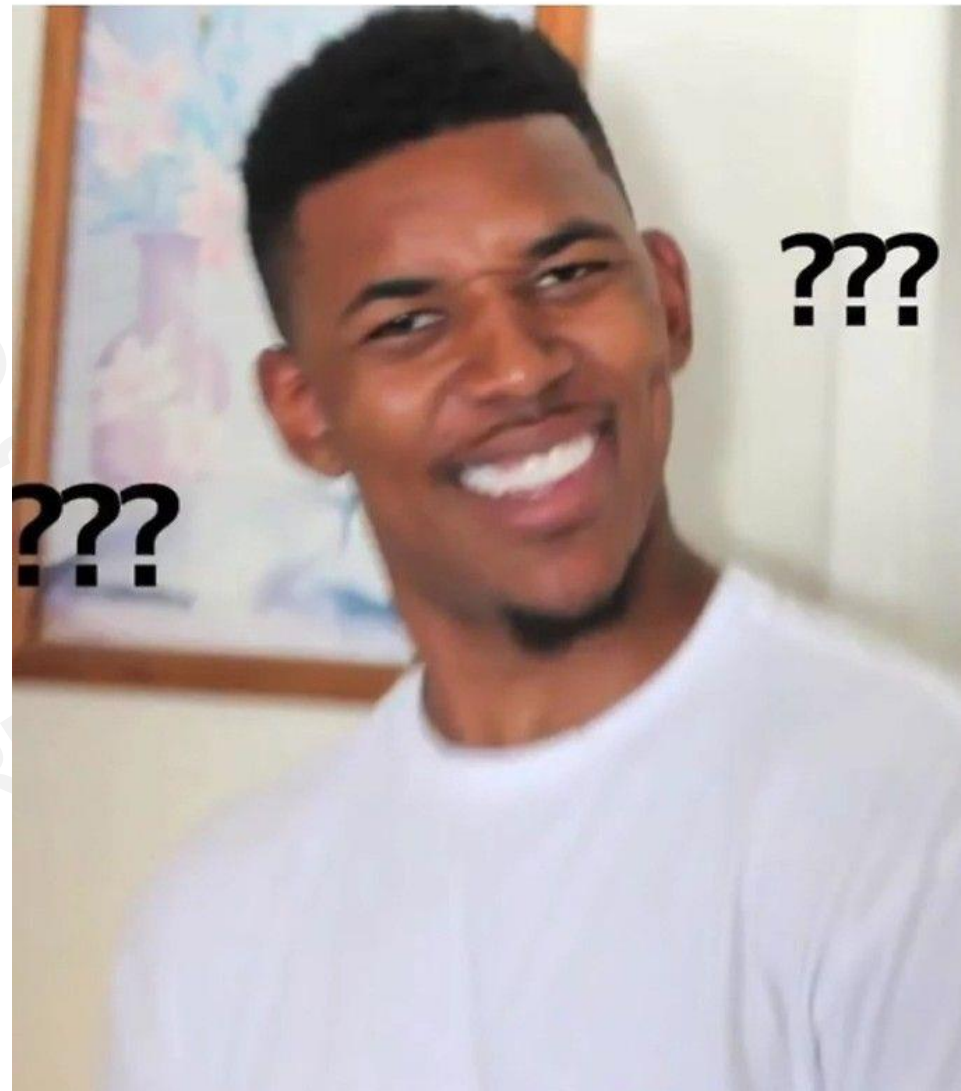
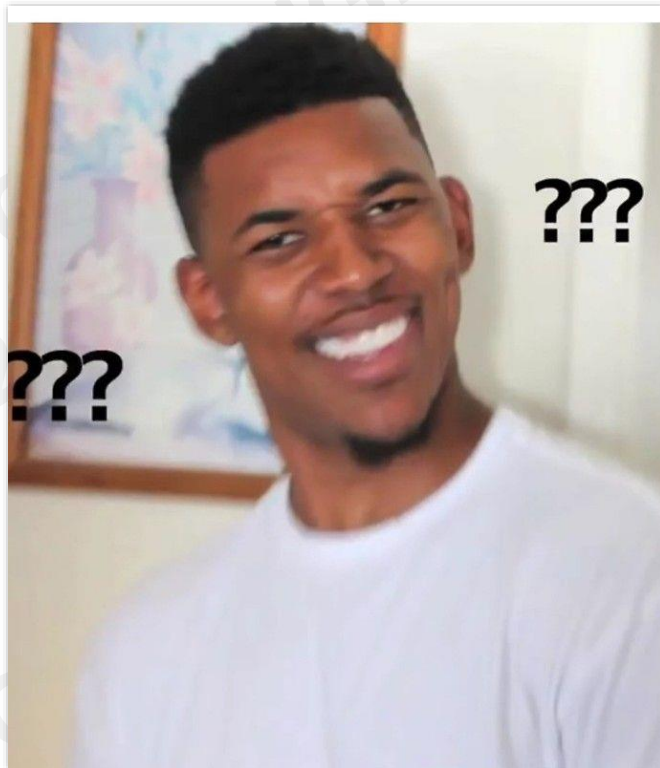
# TECHNION

Azrieli Continuing Education and  
External Studies Division

# Module 5.2.1: Conditional Execution

# Conditional Execution..

WHAT ARE **CONDITIONAL EXECUTIONS**???



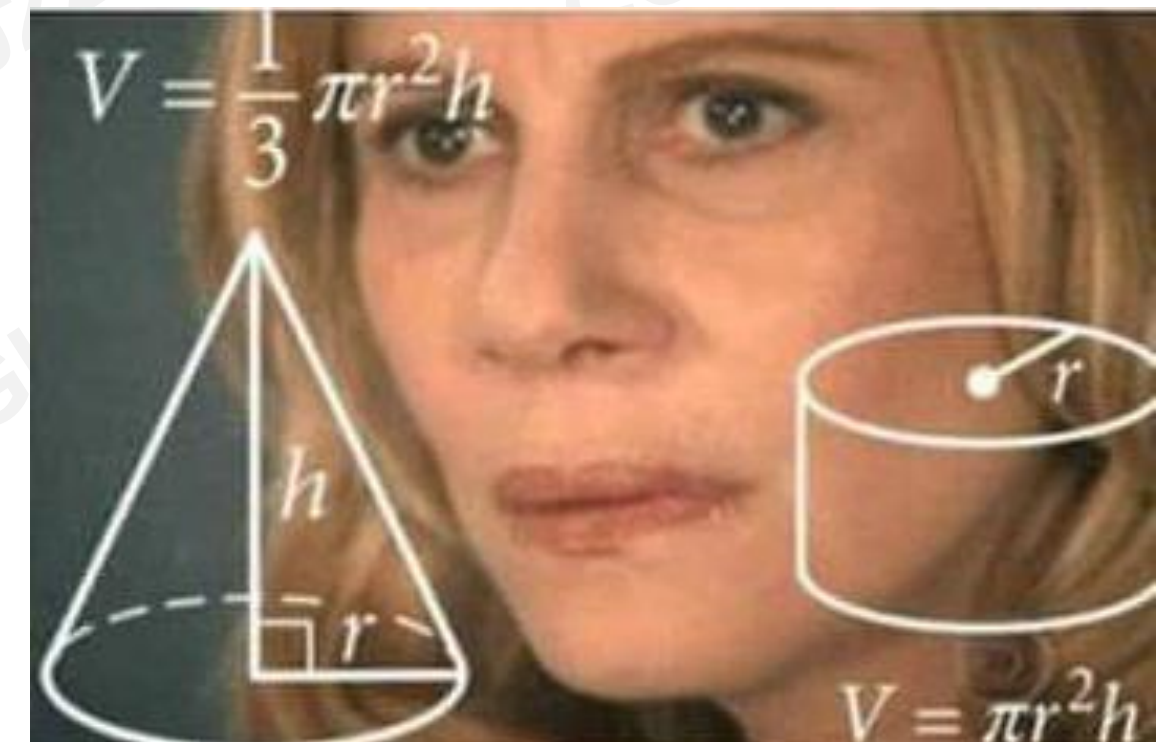
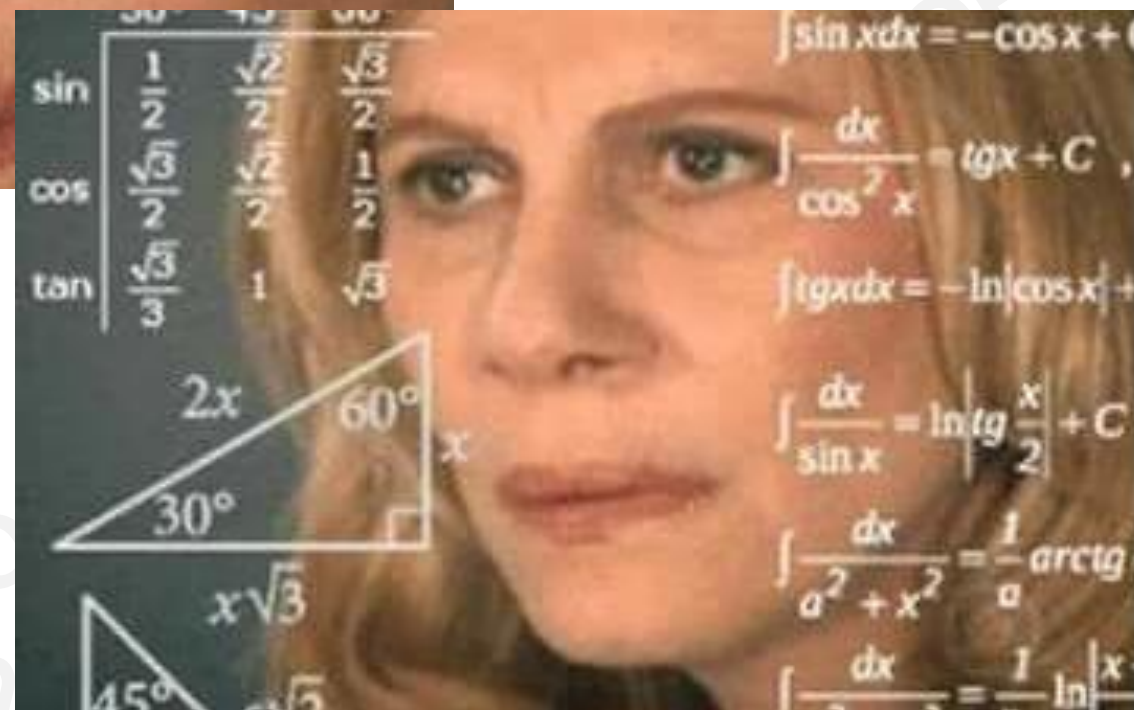
# Conditional Execution

- Conditional execution controls whether a specific-block of code will be executed or not.
- To write useful programs, we almost always need the ability to check conditions and change the program's behavior accordingly!



# Statements ???

## WHAT ARE STATEMENTS???



# Statements

- Quotes
- Declarations
- Official statement
- Government / News / Social Media

- Decision Making

# Statements

Decision Making:

## Ice-Cream

- Flavor

- Price

- Condition

- Weather

- Brand

- Is it Chocolate?

- How much does it cost?

- Is it hardly harmed on the outside?

- Is the weather hot enough to eat it

- Ben & Jerry's (!!!)

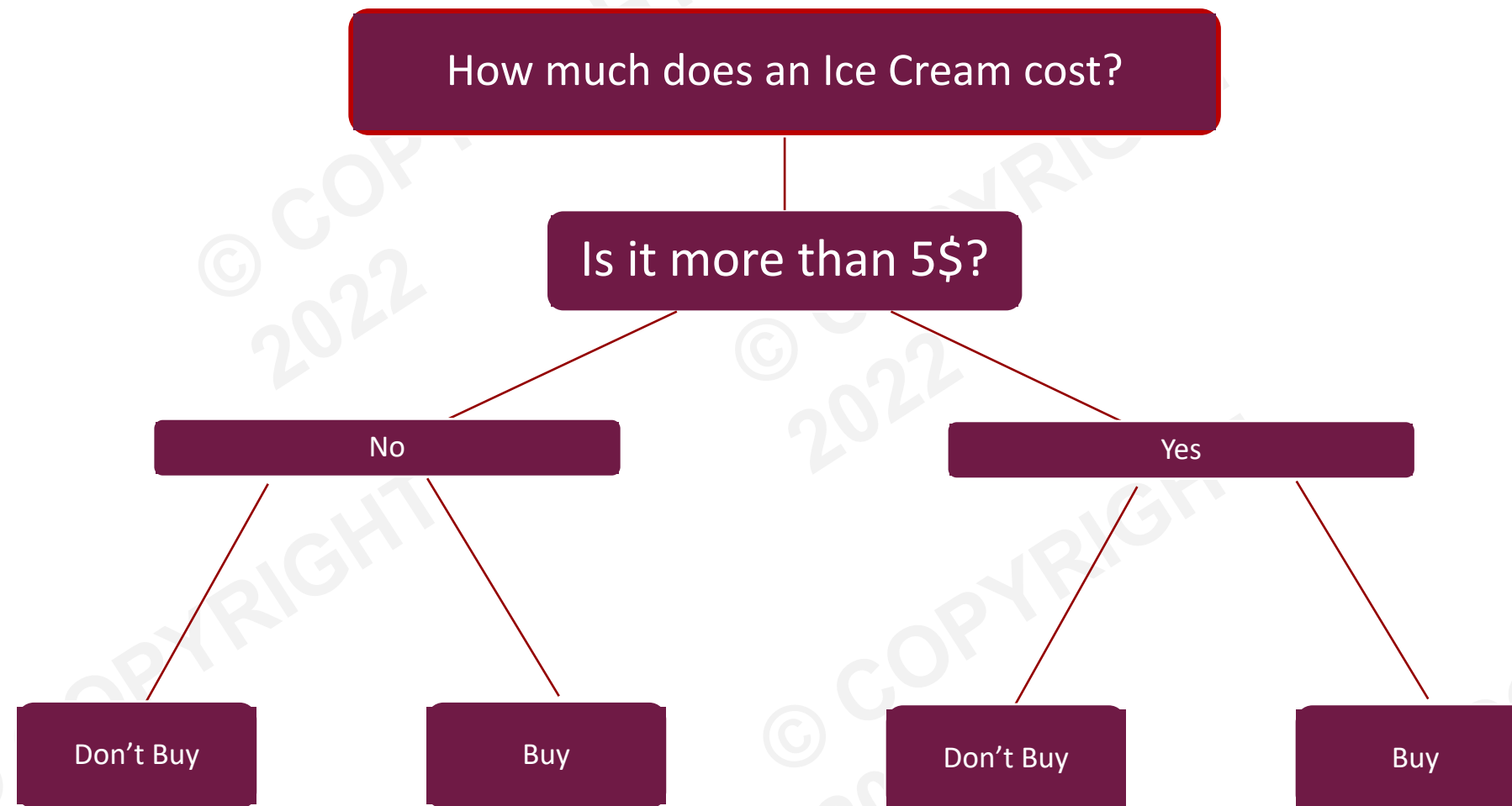
## BEN & JERRY'S FUN FACT



- Each employee gets 3 pints of Ben & Jerry's goodness a day.
- They have a graveyard where they bury discontinued flavors. There are over 300 flavors that are sent to the graveyard.
- All of the meeting rooms at Ben & Jerry's headquarters are named after ice cream flavors.
- The cow we see on Ben & Jerry's packages is named Woody, after Woody Jackson, the artist who designed her in 1983.

# Decision Making

- In modern programming, **Decision Making** is required in almost every script.
- **Decision Making** statements are used when we want a set of instructions to be executed in one situation and different instructions in another.



What if  
it's  
damaged?

What if I  
don't like  
this flavor?



# IF Statement

- The *if* Statement is used in Python for Decision Making.
- The *if* Statement helps us to evaluate a *Boolean Expression* and determine which code will be executed, respectively.



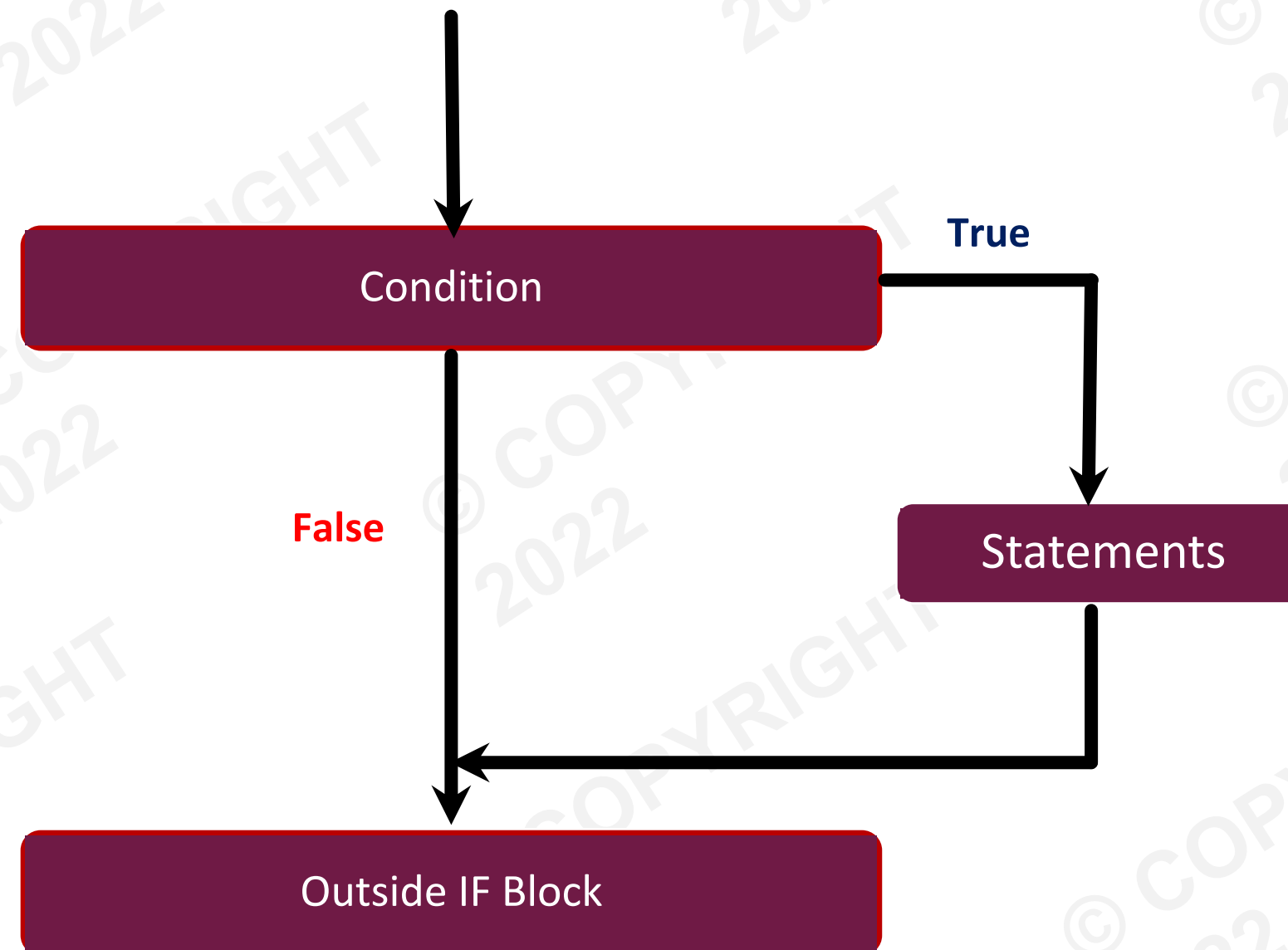
# IF Statement - Syntax

```
if (BOOLEAN EXPRESSION):  
    STATEMENTS
```

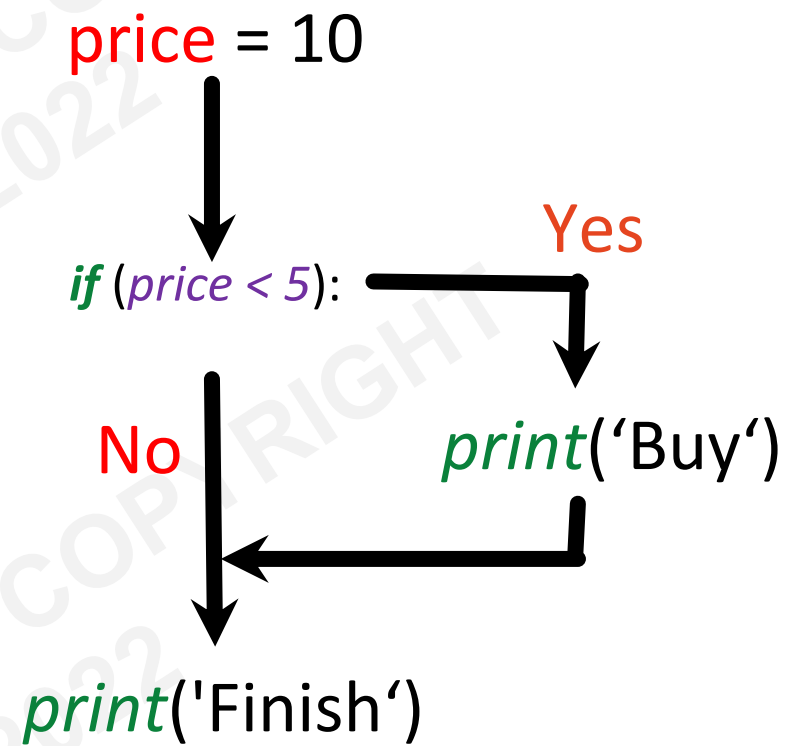
- The colon (:) is **significant** and **required**.
- The line after the colon **must** be indented. (4 Spaces)
  - Python 3 disallows mixing the use of tabs and spaces for indentation.
- All lines indented the same amount after the colon will be executed whenever the *Boolean Expression* is **true**.
- The *Boolean Expression* is called the *Condition*.

# IF Statement – Flow Chart

- If the *Boolean Condition* is *true*; then all the indented statements get executed.
- 
- If the *Boolean Condition* is *false*. Then all the indented statements will **not** execute.



# IF Statement



## Program:

```
price = 10
if (price < 5):
    print('Buy')

print('Finish')
```


Ice-Cream price is 10\$

If it's less than 5\$

Print Buy

# The IF Statement

```
x = 5
if x == 5:
    print('Equal to 5')
if x > 4:
    print('Greater than 4')
if x >= 5:
    print('Greater than or equal to 5')
if x < 6:
    print('Less than 6')
if x <= 5:
    print('Less than or equal to 5')
if x != 6:
    print('Not equal to 6')
```



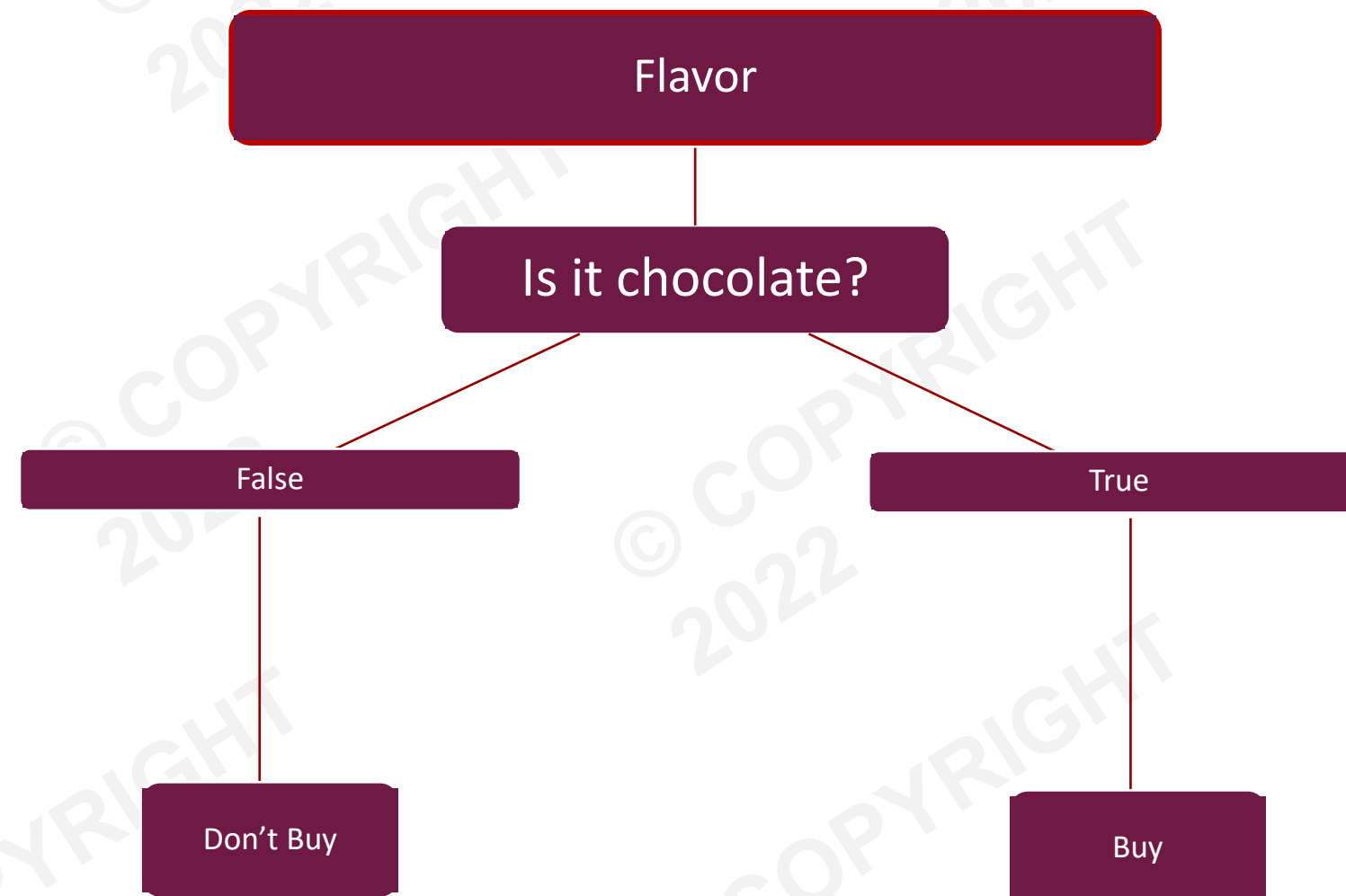
Equal to 5  
Greater than 4  
Greater than or equal to 5  
Less than 6  
Less than or equal to 5  
Not equal to 6



# Else Statement

- The *Else* Statement is used when we want to execute a particular code when our *Boolean Condition* **does not** match our condition.
- 
- Unlike the *If* Statement, which executes code if the *Boolean Condition* returns as **true**,
- the *Else* Statement can react to a false *Boolean Condition*.

# Else Statement



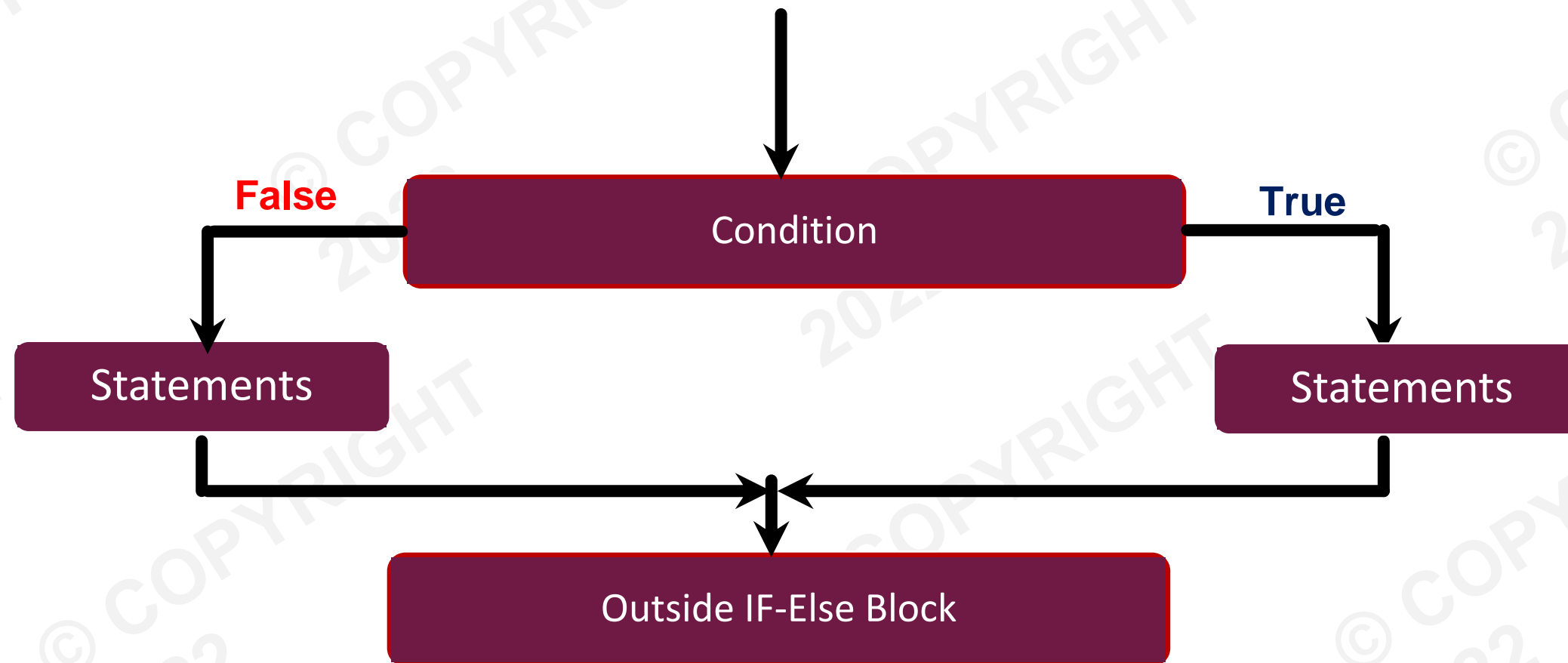
# Else Statement - Syntax

```
if (BOOLEAN EXPRESSION):  
    STATEMENTS  
else:  
    STATEMENTS
```

- The colon (:) is **significant** and **required**.
- The line after the colon **must** be indented. (4 Spaces)
- All lines indented the same amount after the colon will be executed.

# Else Statement – Flow Chart

- If the *Boolean Expression* returns as **false**, the entire block of *If* Statements is skipped.
- If the *Boolean Expression* returns as **true**, the entire block of *Else* Statements is skipped.



# Else Statement

```
flavor = "Chocolate"
```

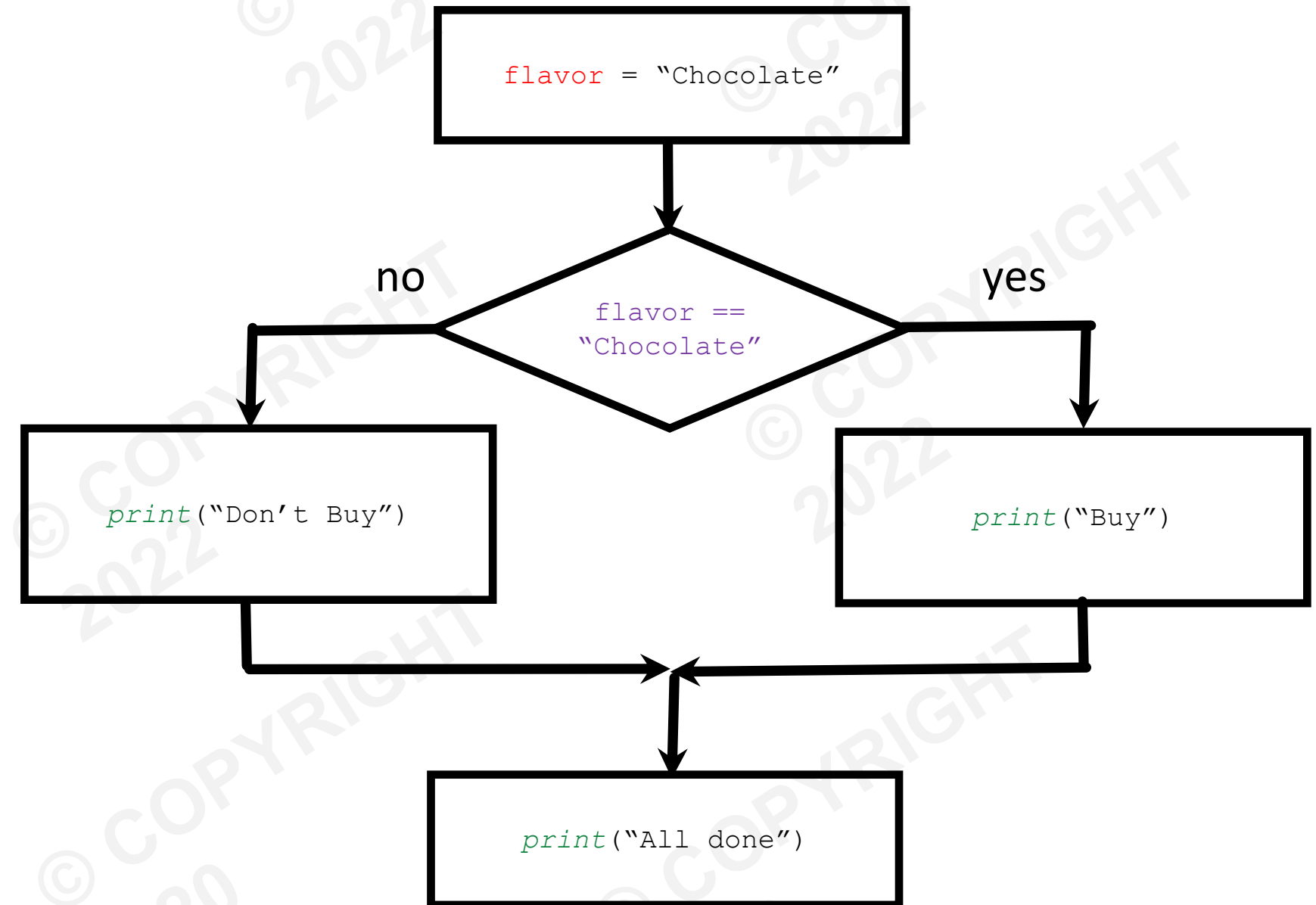
```
if (flavor == "Chocolate"):
```

```
    print("Buy")
```

```
else:
```

```
    print("Don't Buy")
```

```
print("All done")
```





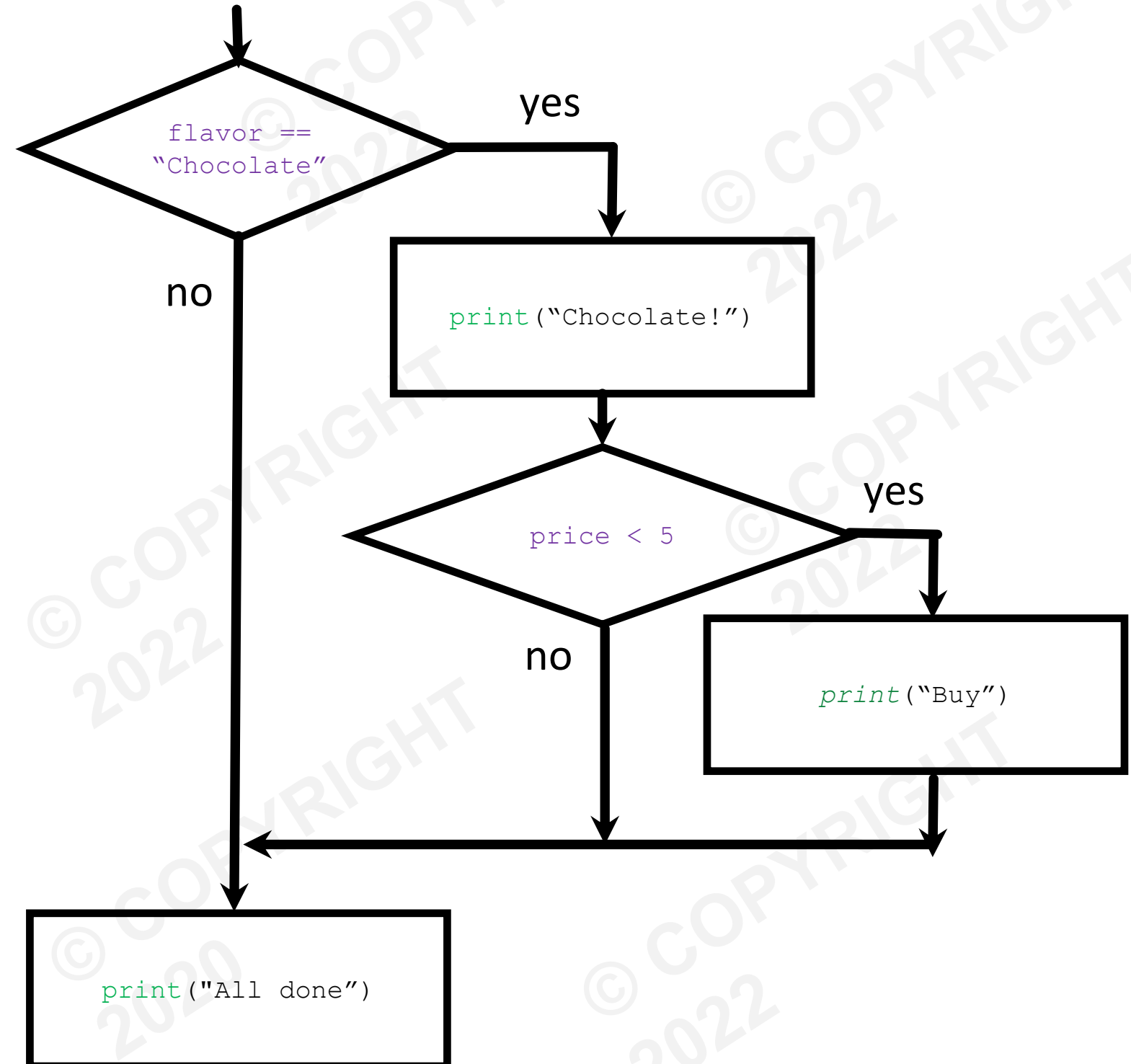
# Nested Conditions

- An *if* Statement inside an *if* Statement is called a **Nested Condition**.

```
flavor = "Chocolate"  
price = 10
```

```
if (flavor == "Chocolate"):  
    print("Yes!")  
    if (price < 5):  
        print("Buy")
```

```
print("All done")
```



# Else If Statement

- The *Else If* Statement serves its purpose when we want to execute specific code when our *Boolean Condition* does not match our previous condition, but it might match a new one.
- 
- Sometimes there are more than two possibilities, and we need more than one condition.
- 
- Using an *Else If* Statement is useful to avoid excessive indentation.
- 
- The keyword '*elif*' is short for '*Else If*'.

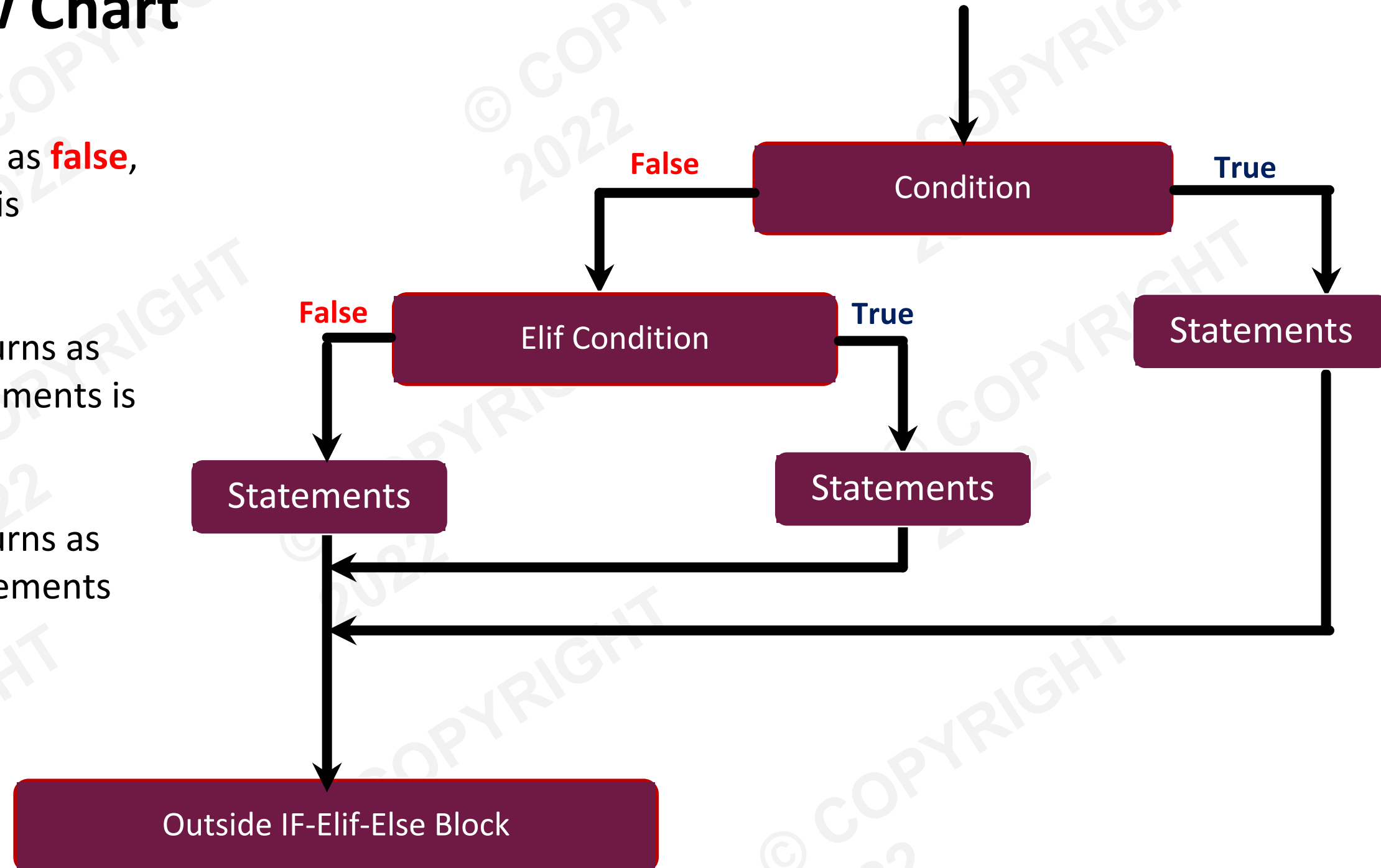
# Elif Statement - Syntax

```
if (BOOLEAN EXPRESSION):  
    STATEMENTS  
elif (BOOLEAN EXPRESSION):  
    STATEMENTS  
else:  
    STATEMENTS
```

- The colon (:) is **significant** and **required**.
- The line after the colon **must** be indented. (4 Spaces)
- All lines indented the same amount after the colon will be executed.

# Elif Statement – Flow Chart

- If the *Boolean Expression* returns as **false**, the entire block of *If* Statements is skipped.
- If the *Elif Boolean Expression* returns as **true**, the entire block of *Elif* Statements is executed.
- If the *Elif Boolean Expression* returns as **false**, the entire block of *Elif* Statements is skipped.



# Elif Statement

```
flavor = "Chocolate"
```

```
if (flavor == "Vanilla"):
```

```
    print("Vanilla!")
```

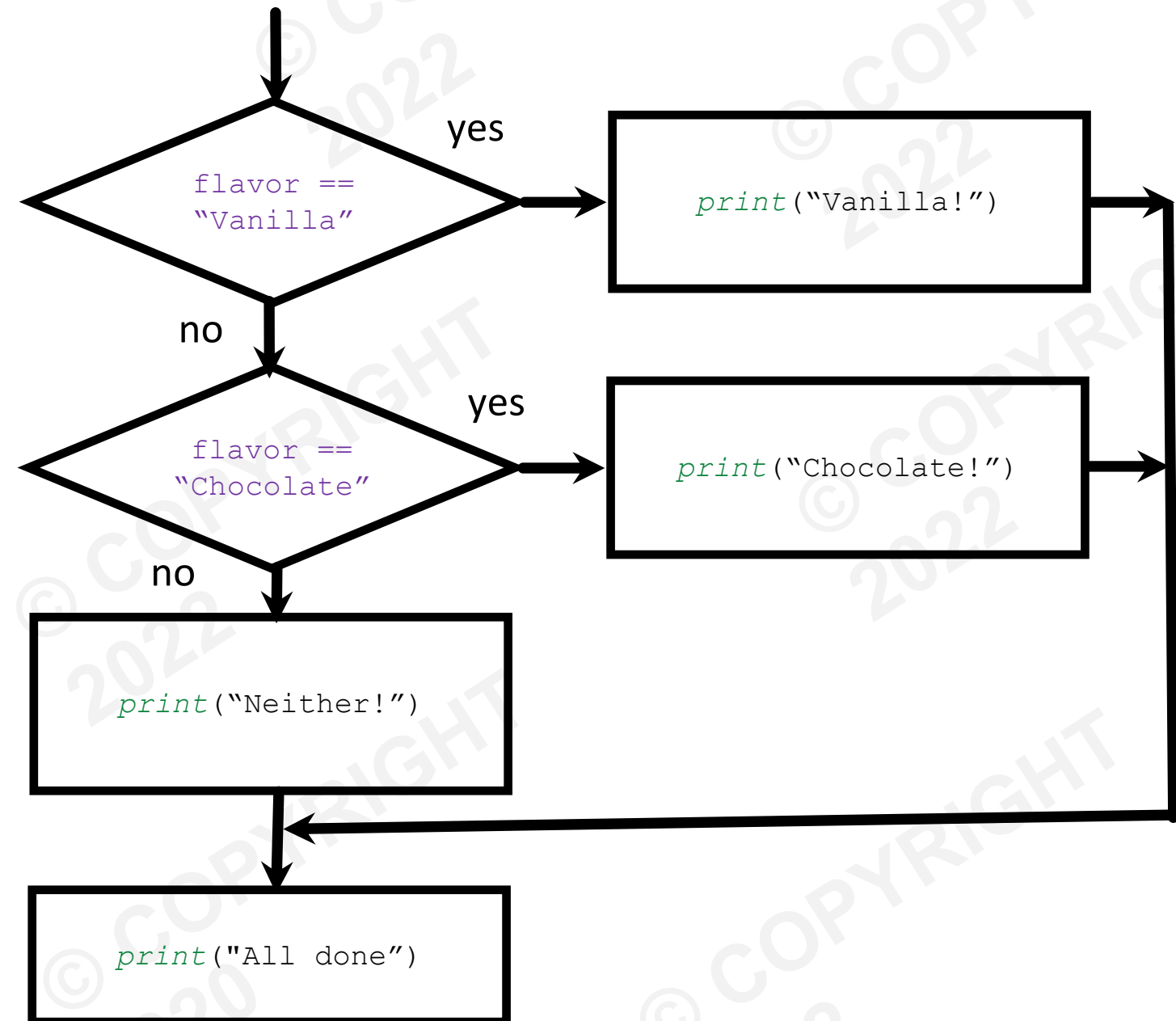
```
elif (flavor == "Chocolate"):
```

```
    print("Chocolate!")
```

```
else:
```

```
    print("Neither!")
```

```
print("All done")
```





# Logical Operators

- Logical Operators are used to combining conditional statements.
- If an operator acts on a single variable. It is called a **unary** operator.
- 
- If an operator acts on two variables. It is called a **binary** operator.

Operator	Type	Description	Example
<i>and</i>	Binary	Returns True if both statements are true	$(x < 5) \text{ and } (x < 10)$
<i>or</i>	Binary	Returns True if one of the statements is true	$(x < 5) \text{ or } (x > 10)$
<i>not</i>	Unary	Reverse the result, returns false if the result is true	$\text{not } (x < 5)$

# Logical Operators - Explained

- 1 = True
- 0 = False

AND

OR

NOT

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

conjunction  
 $A \text{ AND } B = A \cdot B = AB$

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

disjunction  
 $A \text{ OR } B = A + B$

A	NOT A
0	1
1	0

negation  
 $\text{NOT } A = \sim A = A' = \overline{A}$



**AND** is 1 if **both inputs** are 1.



**OR** is 1 if **one ore more** of the inputs are 1.



**NOT** is 1 **only** if the input is 0.

# The *in* Operator

- The *in* operator returns **True** if the first operand is contained within the second, and **False** otherwise!
- There is also a *not in* operator, which does the opposite!

# Order of Logical Operator Assessment

- **NOT** will always happen first, then **AND**, then **OR**.
- The use of ***parentheses*** - **()**, can change this order.
- **Always** use parentheses, so that the individuals code is more readable, *even when they are not needed*.

```
>>> not False and True  
True
```

```
>>> True or True and False  
True  
>>> (True or True) and False  
False
```

# Boolean Values of Variables

- Any “empty” variables are *False*.
- Any variables *with* “content” are *True*.

- Q: *What is an “empty” integer?*

- A: *0*

- Q: *What is an “empty” string?*

- A: *""*

```
>>> bool(0)
```

```
False
```

```
>>> bool(3)
```

```
True
```

```
>>> bool("")
```

```
False
```

```
>>> bool("Hello!")
```

```
True
```



# Boolean Values in Conditions

- In a condition, the casting of a variable/statement to a Boolean is redundant, so it is best practice to remove the `bool()` casting altogether.

```
>>> if 0:  
    print("This should not be printed")
```

```
>>> x = "A full string"  
>>> if x:  
    print("Non-empty strings are True!")
```

```
Non-empty strings are True!
```

```
>>>
```

# Summary

- Decision Making
- If Statement
- Else Statement
- Elif Statement
- Nested Decisions
- Logical Operators
- Boolean Variable Values