algorithmics

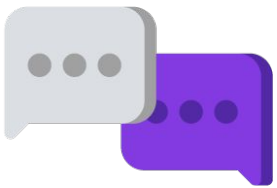
Module 2. Lesson 1.

Conditional statements

Link to the methodological guidelines

Discussion:

Programming smart recommendations

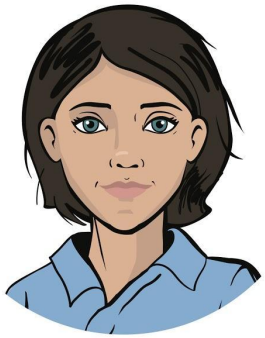


Smart recommendations

Hello, junior developers!

ProTeam has received an order from the Sweet Stories confectionery shop. Their director wants to set up **smart recommendations** for their products on the website.

This is a challenging but interesting task. Are you ready to try it?



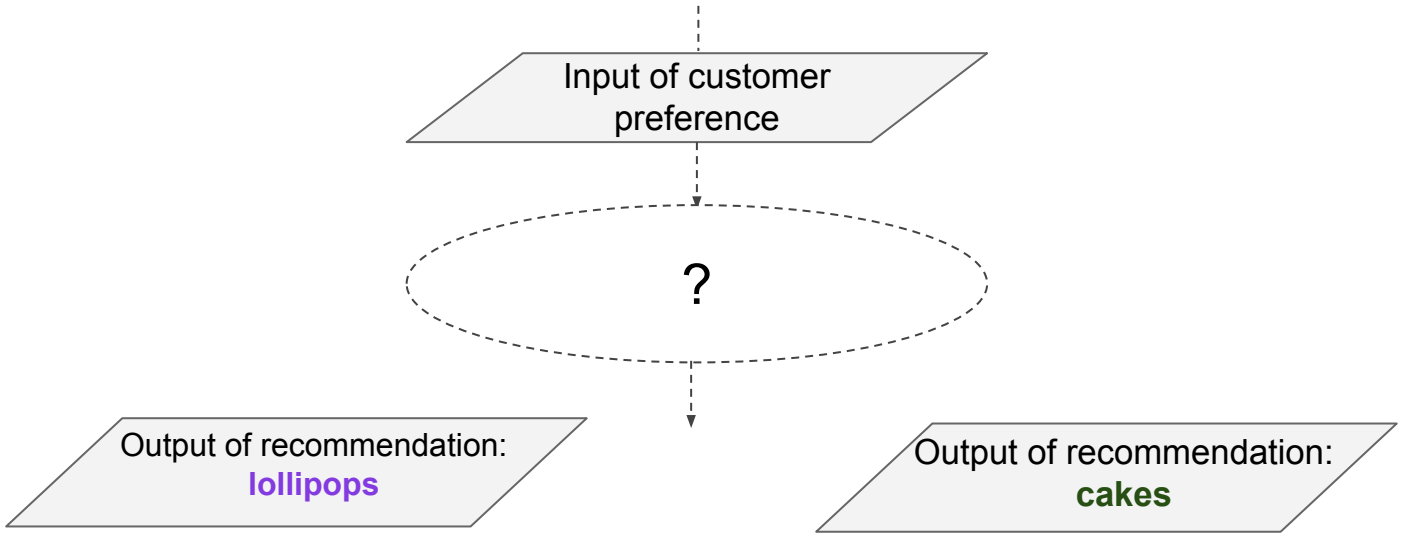
Emily,
Project Manager

Discussion of the tasks

Smart recommendations

Simplified task.

The program knows two recommendations: **lollipops** and **cakes**. The user enters a preference: **candy**. How should we set up smart recommendations?



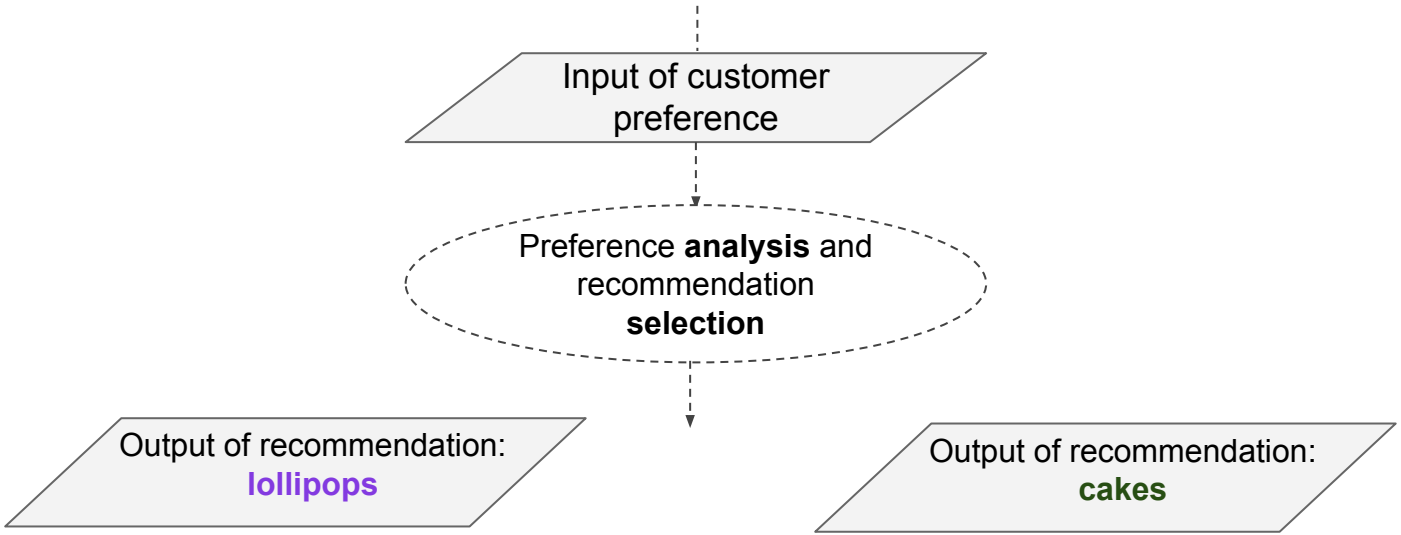
According to the rule of order, we must have some command here. Which one?

Discussion of the tasks

Smart recommendations

Simplified task.

The program knows two recommendations: **lollipops** and **cakes**. The user enters a preference: **candy**. How do we set up smart recommendations?

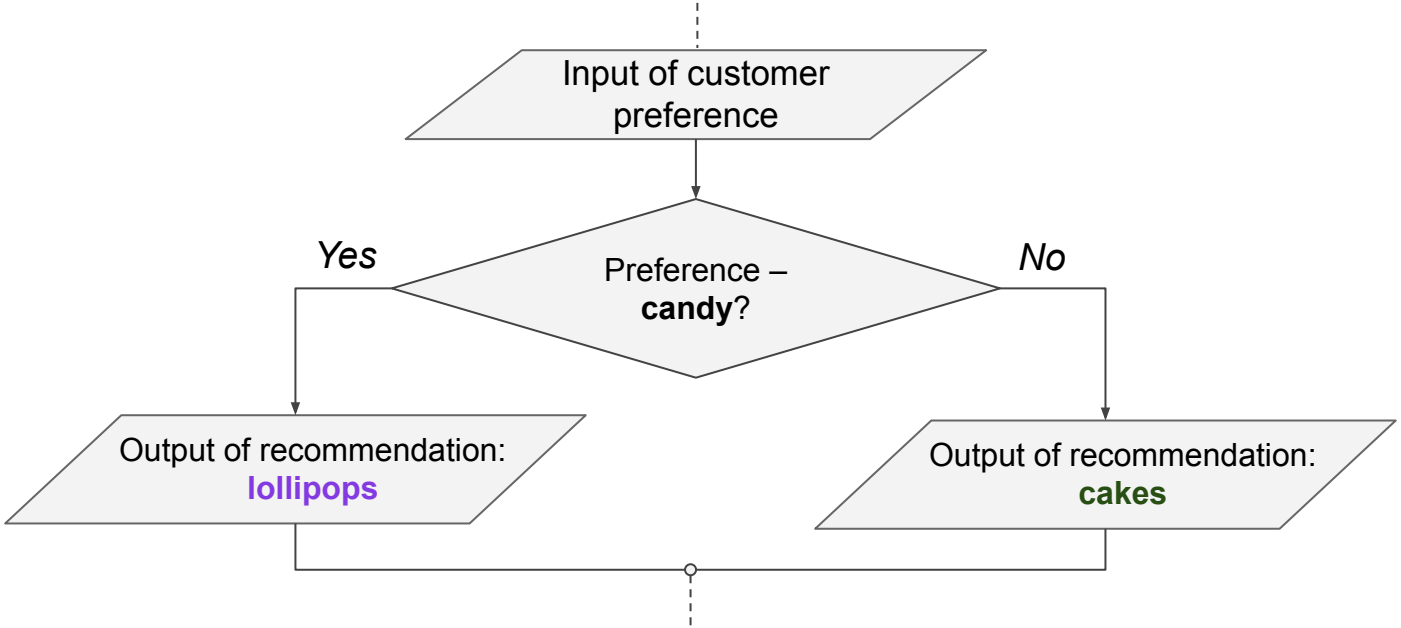


Discussion of the tasks

Smart recommendations

Simplified task.

The program knows two recommendations: **lollipops** and **cakes**. The user enters a preference: **candy**. How do we set up smart recommendations?

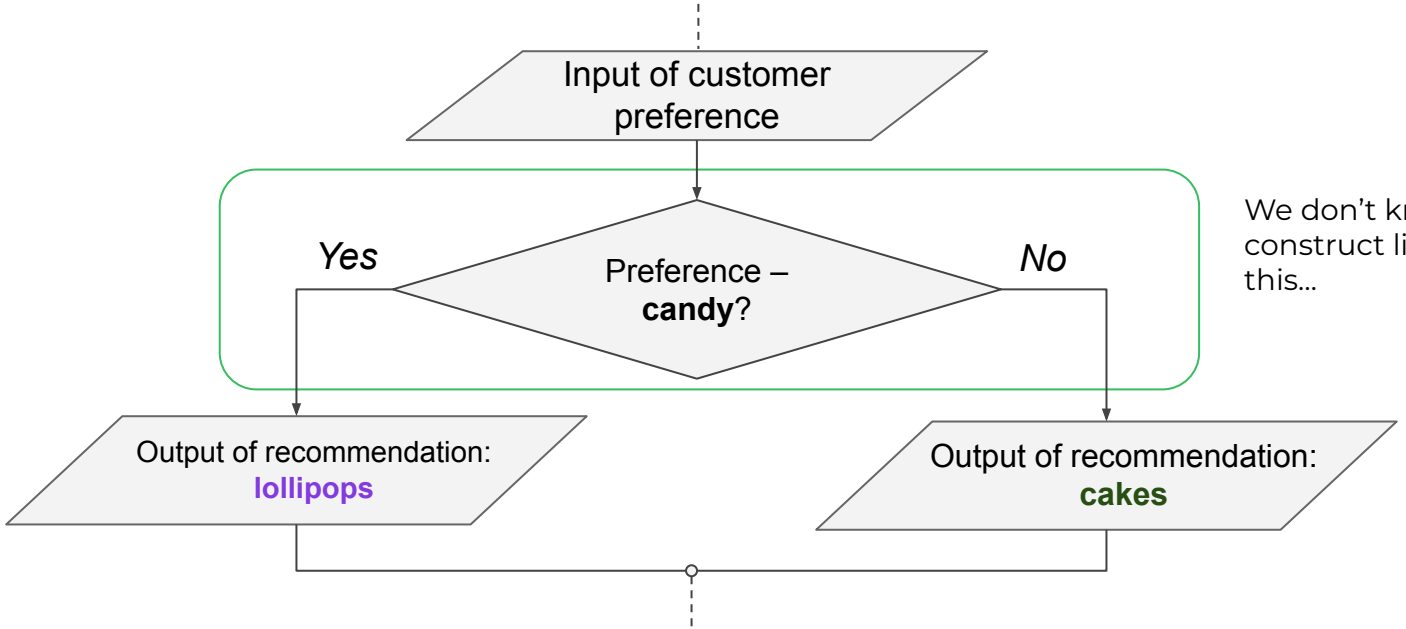


Discussion of the tasks

Smart recommendations

Simplified task.

The program knows two recommendations: **lollipops** and **cakes**. The user enters a preference: **candy**. How do we set up smart recommendations?



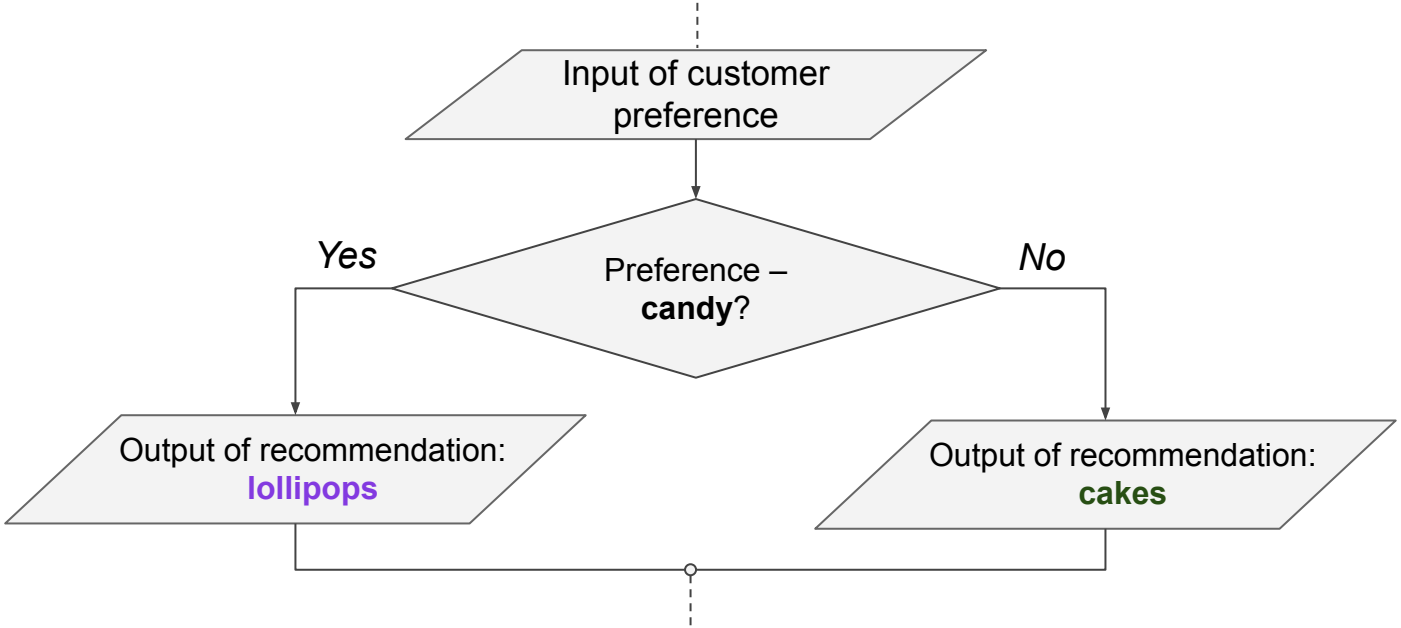
We don't know a construct like this...

Discussion of the tasks

Smart recommendations

Simplified task.

The program knows two recommendations: **lollipops** and **cakes**. The user enters a preference: **candy**. How do we set up smart recommendations?



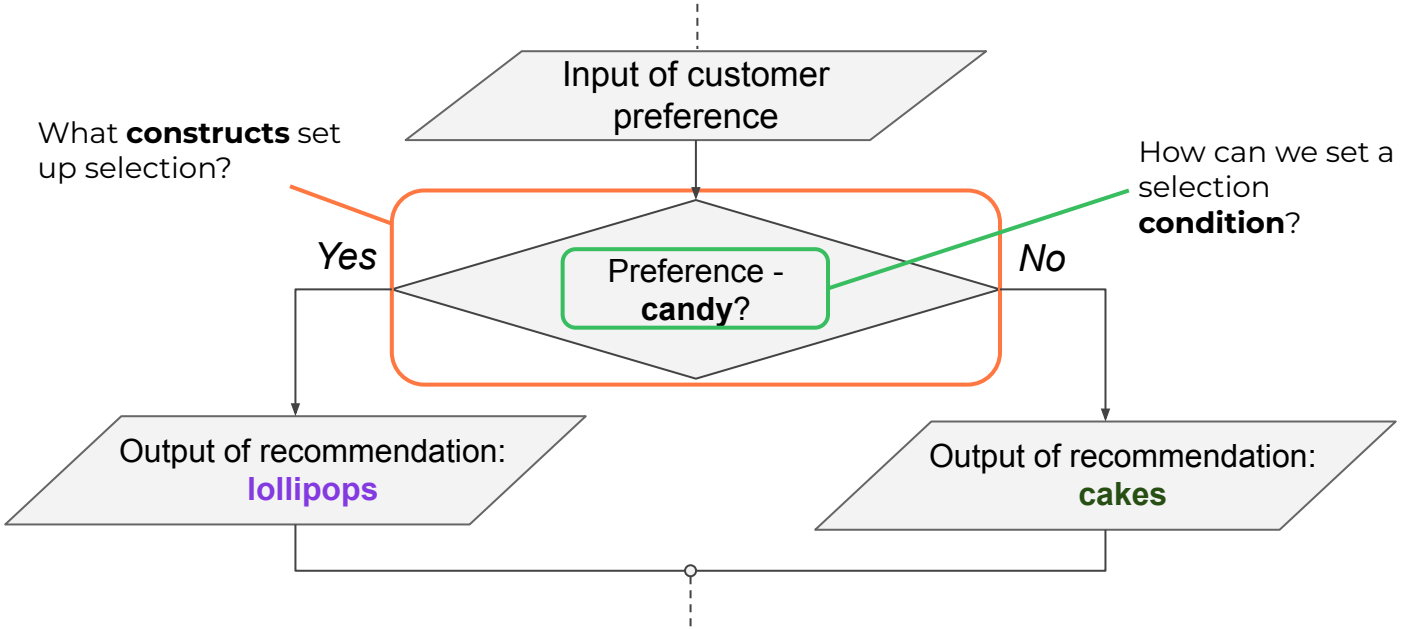
What do we need to learn to program such a construct?

Discussion of the tasks

Smart recommendations

Simplified task.

The program knows two recommendations: **lollipops** and **cakes**. The user enters a preference: **candy**. How do we set up smart recommendations?



Discussion of the tasks

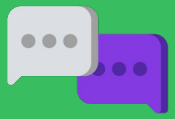
The goal of the workday is to

set up smart recommendations *for a confectionery shop.*

The customer wants the program to offer buyers products that match their preferences

Today you will:

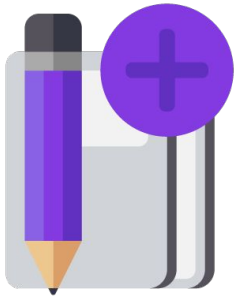
- learn that a conditional statement is a construct that analyzes a condition and selects a command to execute;
- learn how to program a condition using a new data type;
- take on your first working project.



Discussion of the tasks

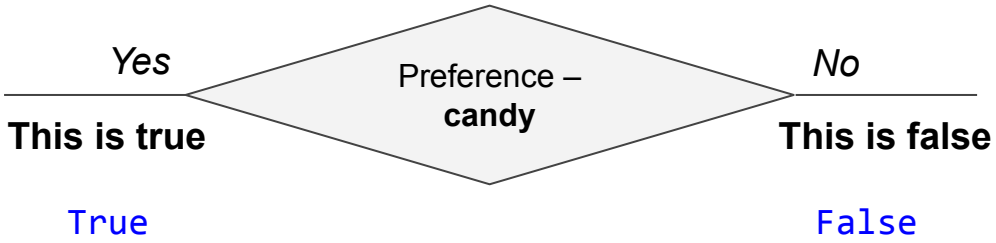
Brainstorm:

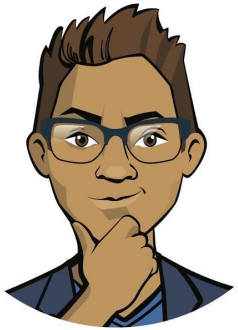
Logical data type



How do we program a condition?

In the previous task, we considered a **condition** as some kind of statement that can be **either true or false.**





Cole,
Senior Developer

Brainstorm

Logical data type

Such statements play an important role in programming. A **logical (boolean) data type** was invented for them.

Data type	*Integer*	*Logical*
Values	`-100`, `5`, `512`	`True`, `False`
Variables	`days = 31`	`is_correct = True`
Simple expressions	`daily_money * days` `price - sale`	`5 > 2` `name != 'John'`

Brainstorm

Variables and simple expressions

Variables and expressions can take values

`True` or `False`.

```
checked = True
is_sent = False
print(checked)
print(is_sent)
```

Brainstorm

Variables and simple expressions

Variables and expressions can take values:

`True` or `False`.

```
checked = True
is_sent = False
print(checked)
print(is_sent)
```


```
amount_shop = int(input('In stock:'))
booked = int(input('Bought:'))
ok = amount_shop > booked
print(ok)
```

Brainstorm

Variables and simple expressions

Variables and expressions can take values:

`True` or `False`.

```
checked = True
is_sent = False
print(checked)
print(is_sent)
```


```
amount_shop = int(input('In stock:'))
booked = int(input('Bought:'))
ok = amount_shop > booked
print(ok)
```

Logical operation

Logical expression

Brainstorm

Simple logical expression:
comparison operators

Comparison operators can be used to make logical expressions.

Integer type					
*	/	%	//	+	-
Multiplication	Division	Remainder	Quotient	Addition	Subtraction

Brainstorm

Simple logical expression:
comparison operators

Comparison operators can be used to make logical expressions.

Integer type					
*	/	%	//	+	-
Multiplication	Division	Remainder	Quotient	Addition	Subtraction

Logical type					
>	<	==	!=	<=	>=
Greater than	Less than	Equal	Not equal	Less than or equal	Greater than or equal

Brainstorm

Simple logical expression:
comparison operators

Task. Write a program that asks for the stock balance of chocolates and determines if the stock needs to be replenished. The minimum amount of sweets in stock is 50 kg.

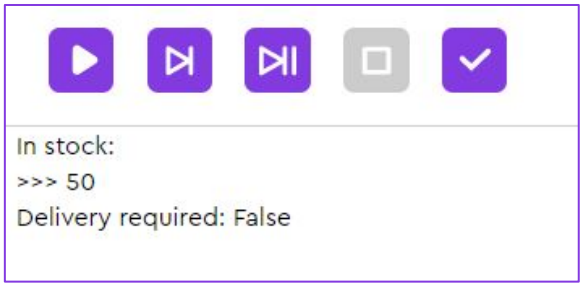
Brainstorm

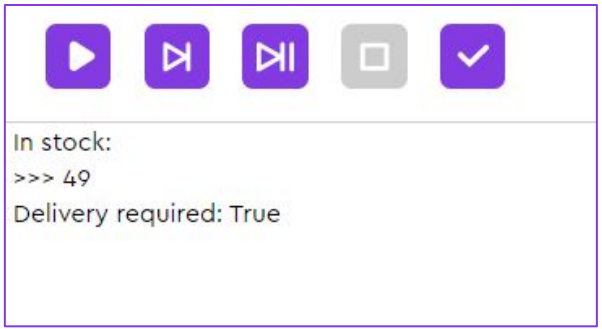
You might want to set up the delivery requirement using a logical expression.

Simple logical expression:
comparison operators

Task. Write a program that asks for the stock balance of chocolates and determines if the stock needs to be replenished. The minimum amount of sweets in stock is 50 kg.

```
amount_store = int(input('In stock:'))
amount_min = 50
delivery = amount_store < amount_min
print('Delivery required:', delivery)
```





Brainstorm

Compound logical expression

A **compound** logical expression can be made up of **simple expressions** by linking them using logical operators:

Operator	*Name*	*Used when needed:*
and	Logical AND	Require two simple conditions to be met at the same time
or	Logical OR	Require at least one of two simple conditions to be met

order of execution

**Subexpressions connected by logical AND are executed first, then those linked by logical OR.*

Brainstorm

Compound logical expression

Task. Write a program that notifies the user about an error in the stock of chocolates.

A stock error occurs when the storage is almost empty (less than 50 kg) or when it is full (more than 300 kg).

Try to program a stock error using a compound logical expression.

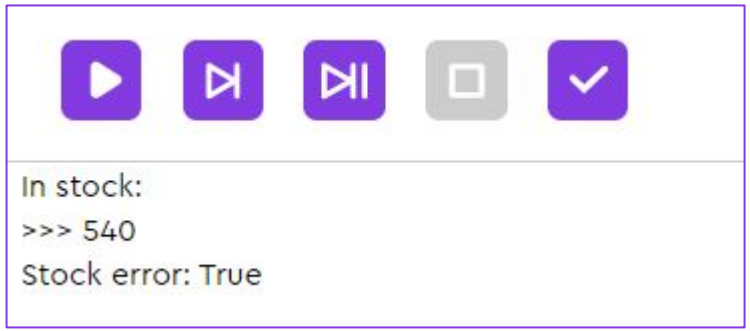
Brainstorm

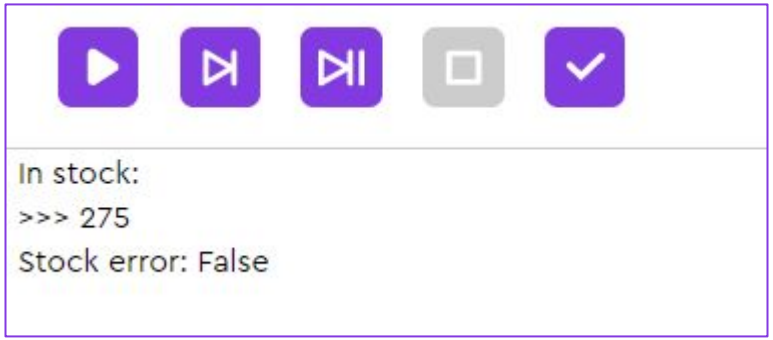
Compound logical expression

Task. Write a program that notifies the user about an error in the stock of chocolates.

A stock error occurs when the storage is almost empty (less than 50 kg) or when it is full (more than 300 kg).

```
amount_store = int(input('In stock:'))
error = amount_store < 50 or amount_store > 300
print('Stock error:', error)
```





Brainstorm

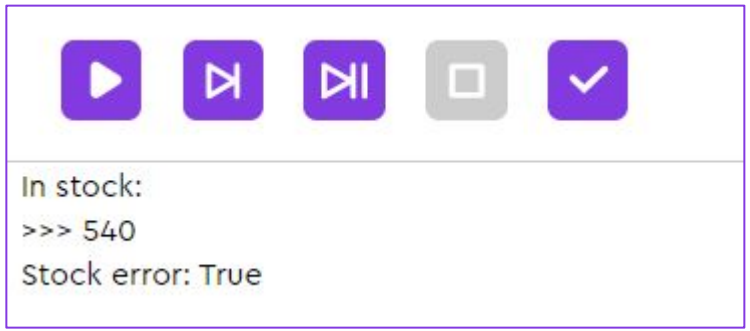
Compound logical expression

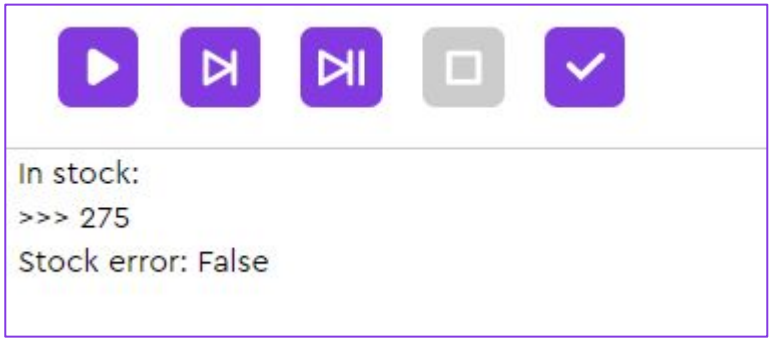
Task. Write a program that notifies the user about an error in the stock of chocolates.

A stock error occurs when the storage is almost empty (less than 50 kg) or when it is full (more than 300 kg).

```
amount_store = int(input('In stock:'))
error = amount_store < 50 or amount_store > 300
print('Stock error:', error)
```

The values of simple expressions are calculated first. Then, they are followed by the values of compound expressions.

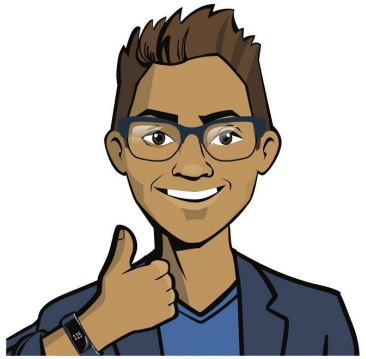




Brainstorm

Conclusions:

1. A **logical data type** is a data type used to program statements that can be true or false.
2. **Simple logical expressions** can be constructed using comparison operators.
3. **Compound logical expressions** can be constructed from simple logical expressions and logical operators.

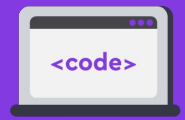


Brainstorm

Bake shop:
Conditions

Do the task on the platform

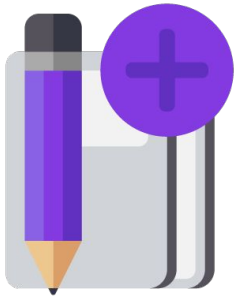
“Bake shop: conditions”



Confectionery store: conditions

Break

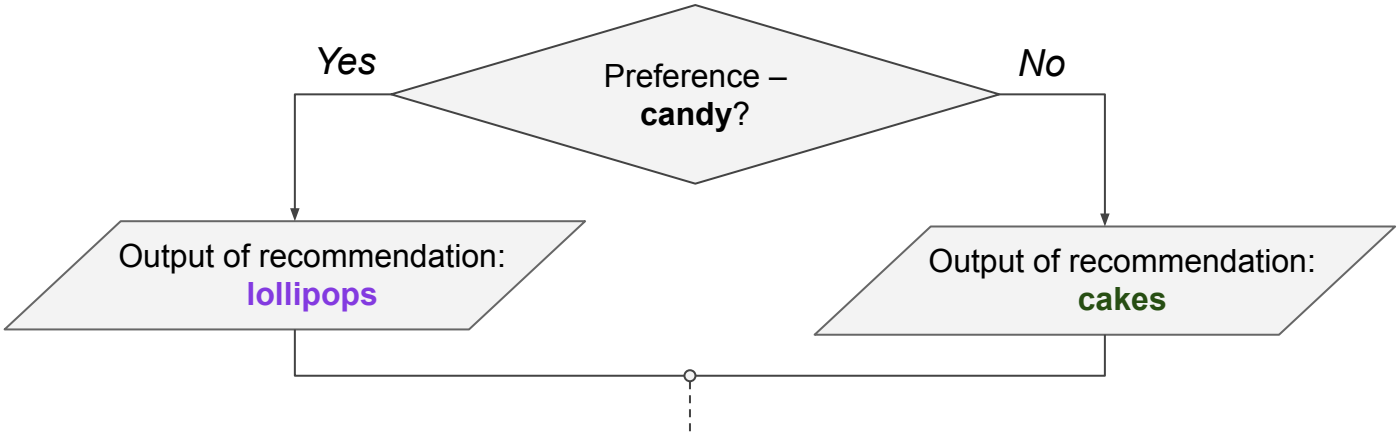
Conditional statements



How do we program a selection?

We learned how to program a condition – a statement that can be true or false.

Now, let's become familiar with a construct that selects a command to execute depending on whether the condition is true.

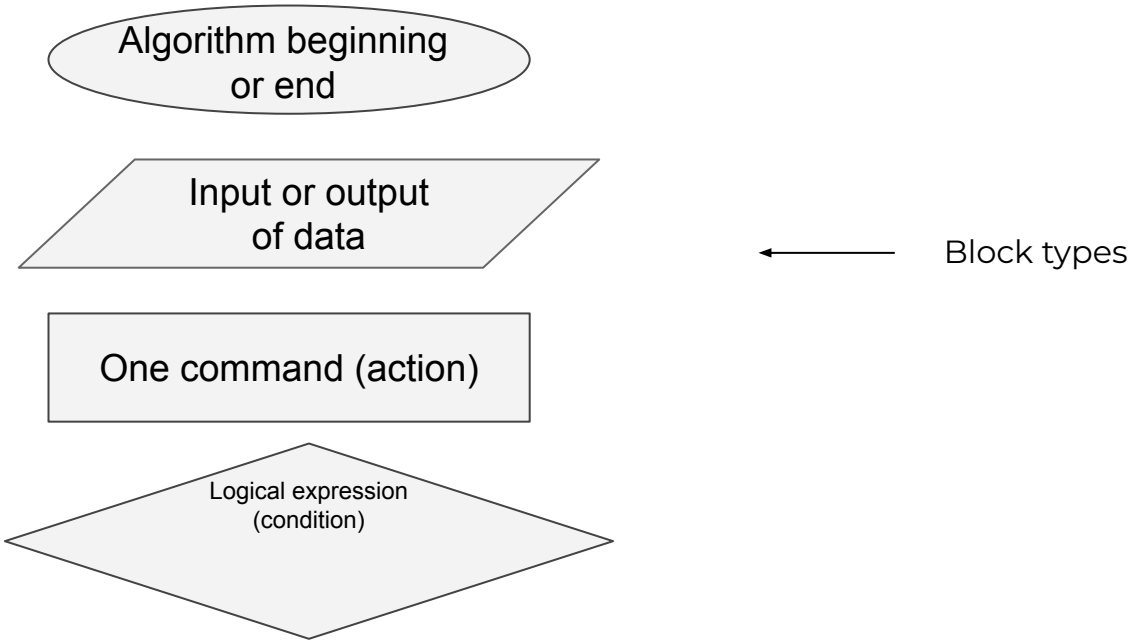


Brainstorm

Writing an algorithm as a flowchart

From now on, when we analyze algorithmic constructs, we will use flowcharts.

This is a universal method of writing an algorithm that every programmer knows.

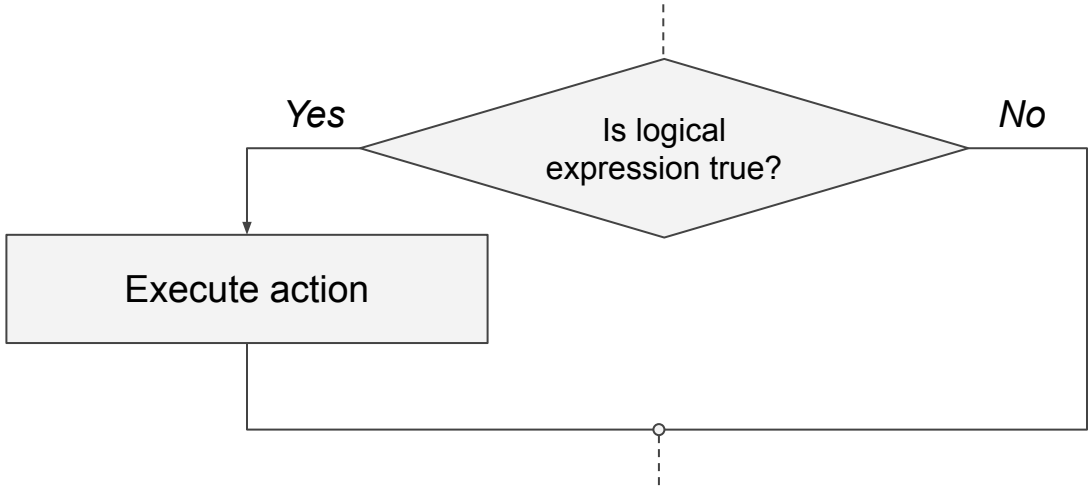


Brainstorm

A conditional statement

is a command that executes or does not execute an action depending on the value of a logical expression.

Usage example:
executing some action only if the expression is true.

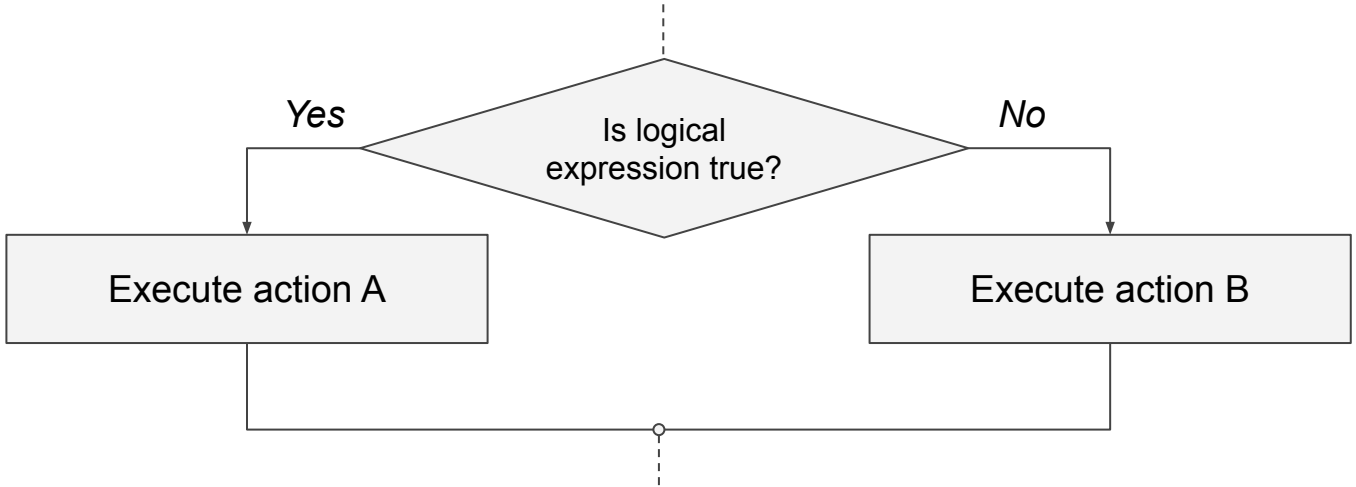


Brainstorm

A conditional statement,

is a command that executes or does not execute an action depending on the value of a logical expression.

Usage example:
executing action A if the expression is true and action B is false.



Brainstorm

Conditional statement

Task 1. Create an algorithm that checks if it is possible to make a purchase with a card.

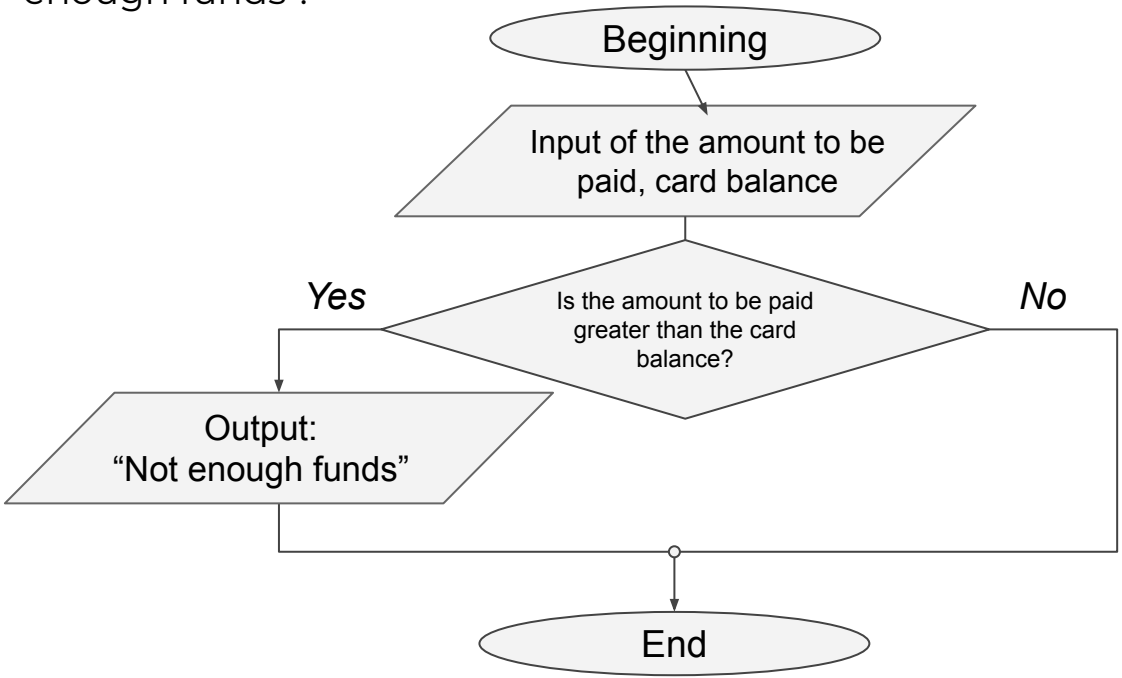
If the cost of goods is more than the amount of money on the card, output: “Not enough funds”.

Brainstorm

Conditional statement

Task 1. Create an algorithm that checks if it is possible to make a purchase with a card.

If the cost of goods is more than the amount of money on the card, output: “Not enough funds”.



Brainstorm

Conditional statement

Task 2. Create an algorithm that checks if it is possible to make a purchase with a card.

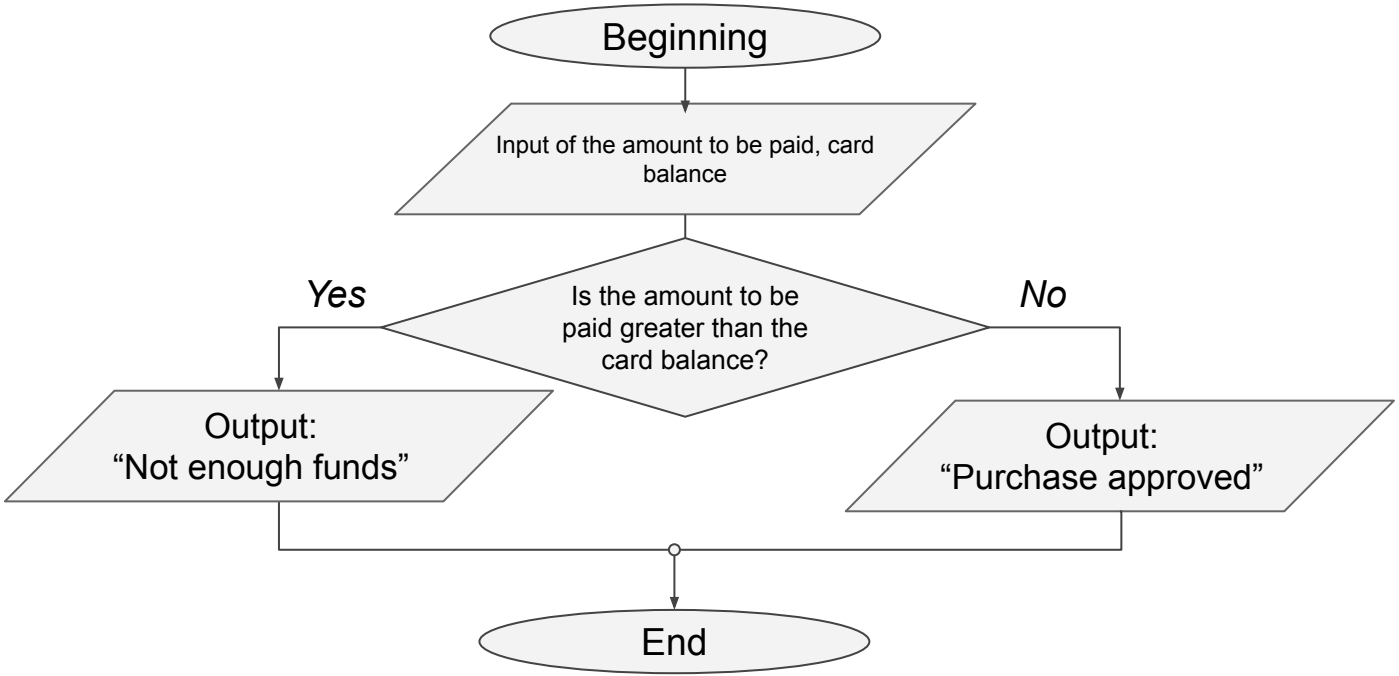
If the cost of goods is more than the amount of money on the card, output: “Not enough funds”. Otherwise, output “Purchase approved”

Brainstorm

Conditional statement

Task 2. Create an algorithm that checks if it is possible to make a purchase with a card.

If the cost of goods is more than the amount of money on the card, output: “Not enough funds”. Otherwise, output “Purchase approved”



Brainstorm

Conditional statement

To program a conditional statement, the following commands are used:

```
if
```

```
else
```

Brainstorm

Conditional statement

To program a conditional statement, the following commands are used:

```
if
else
```

```
if Expression is true :
    Execute action 1
    Execute action 2
    Execute action 3
```

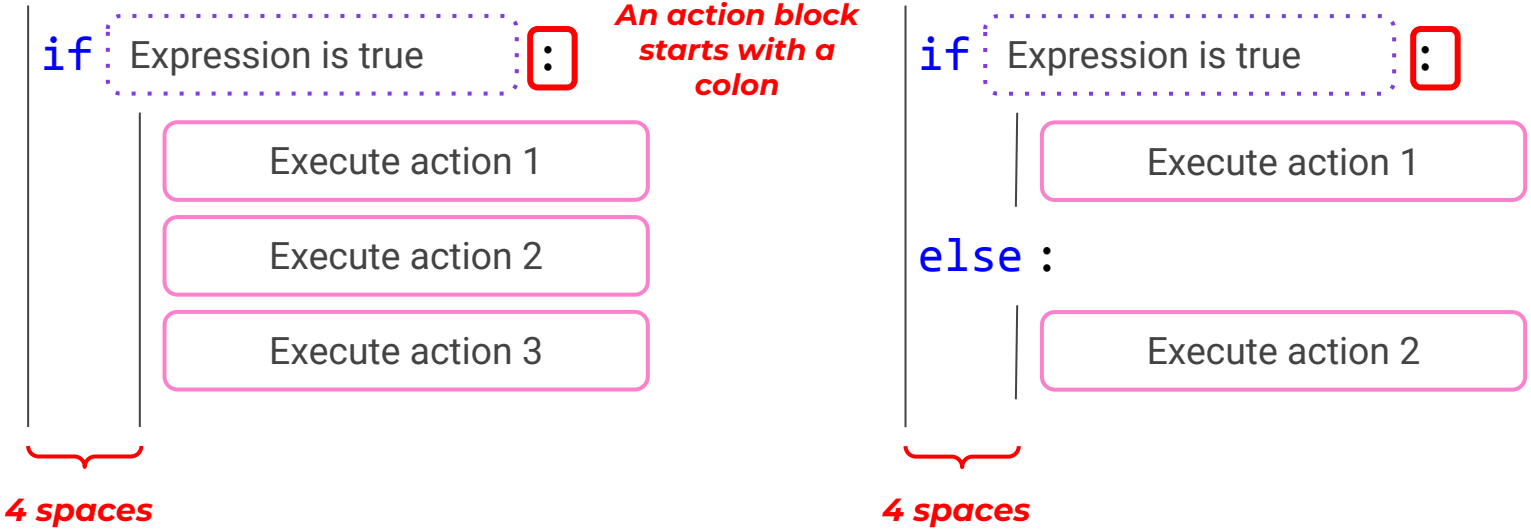
```
if Expression is true :
    Execute action 1
else :
    Execute action 2
```

Brainstorm

Conditional statement

To program a conditional statement, the following commands are used:

```
if
else
```



Brainstorm

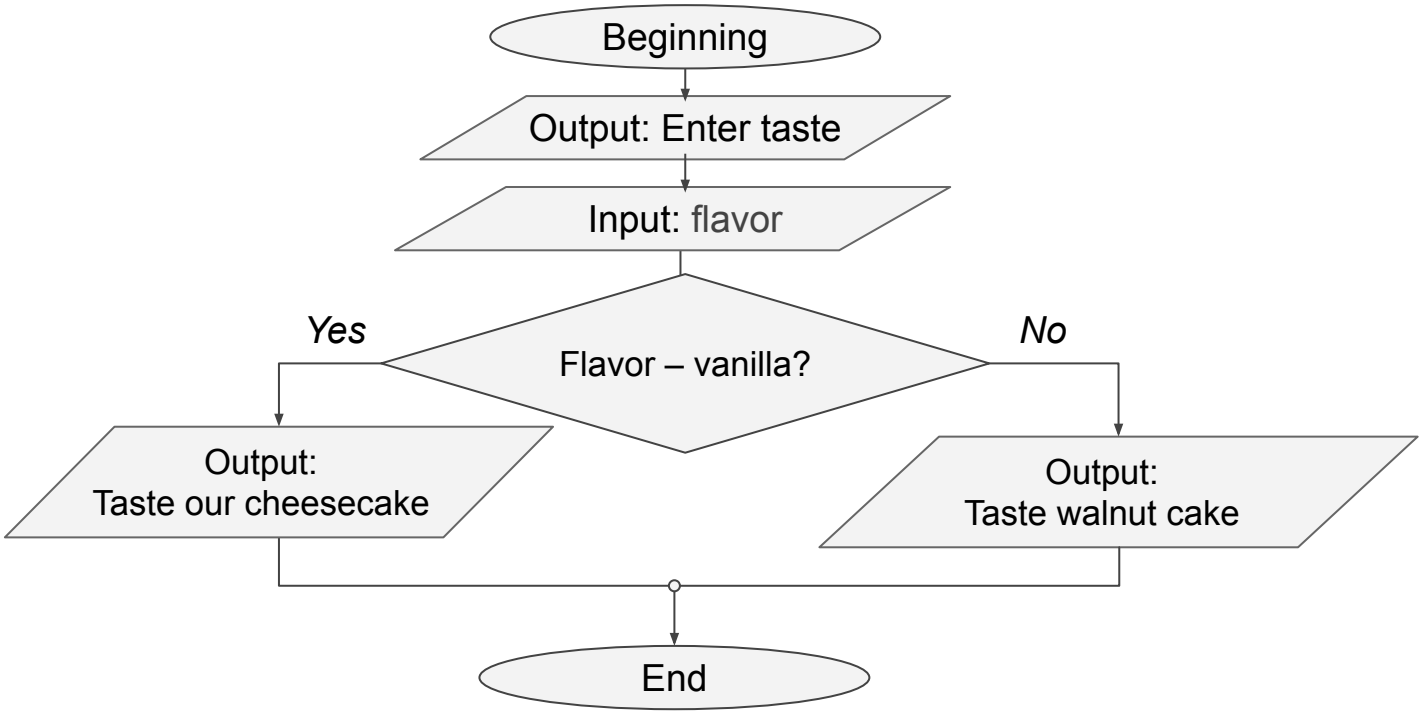
Conditional statement

Task. Write a program that offers a product according to taste preferences. The program asks what flavor the user likes. If it's vanilla, recommend cheesecake. Else – walnut cake.

Brainstorm

Conditional statement

Task. Write a program that offers a product according to taste preferences. The program asks what flavor the user likes. If it's vanilla, recommend cheesecake. Else – walnut cake.



Brainstorm

Conditional statement

Task. Write a program that offers a product according to taste preferences. The program asks what flavor the user likes. If it's vanilla, recommend cheesecake. Else – walnut cake.

```
taste = input('Enter your favorite taste:')
taste = taste.lower()
```

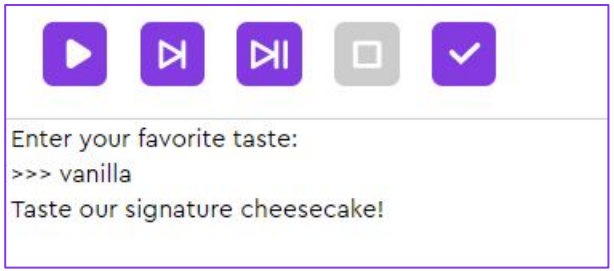
?

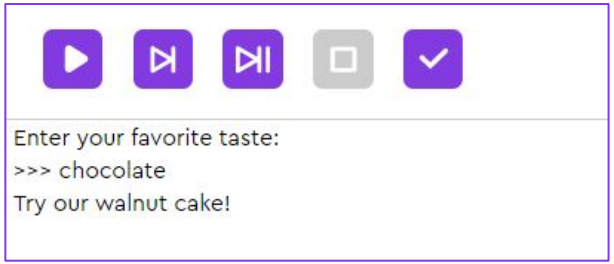
Brainstorm

Conditional statement

Task. Write a program that offers a product according to taste preferences. The program asks what flavor the user likes. If it's vanilla, recommend cheesecake. Else – walnut cake.

```
taste = input('Enter your favorite taste:')
taste = taste.lower()
if taste == 'vanilla':
    print('Taste our signature cheesecake!')
else:
    print('Try our walnut cake!')
```

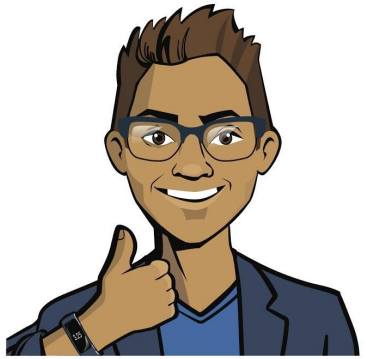




Brainstorm

Conclusions:

1. A conditional statement is a command that executes or does not execute an action depending on the value of a logical expression.
2. To program a conditional statement, the `if` and `else` statements are used.
3. Actions within a conditional statement begin with a colon and are indented by 4 spaces.

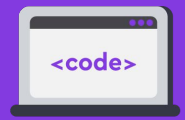


Brainstorm

Bake shop:
Recommendations

Do the task on the platform

➡ “Bake shop: recommendations”

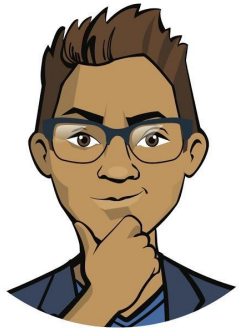


Confectionery shop: recommendations

End of the workday

To complete the workday, pass a technical interview

1. What is a logical data type? What values can a logical expression take?
2. What is a conditional statement? What operators is it set up by?
3. Where are the logical data type and conditional statement used?



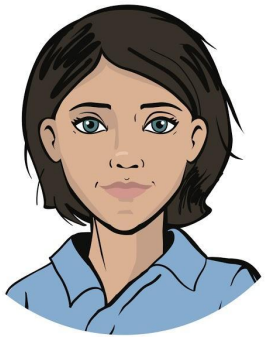
Cole
Senior Developer

Emily,
Project Manager

Congratulations on completing the workday!

Today you have:

1. Learned the data type used to program statements that take the values “true” or “false”.
2. Learned a new algorithmic construct – the conditional statement.
3. Learned to program a choice of executable command depending on whethe condition is true.



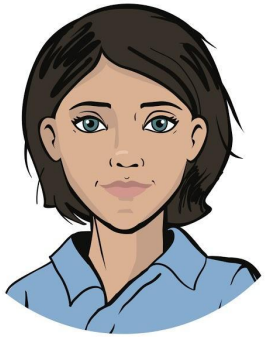
Emily,
Project Manager

Wrapping up the workday

Performance review

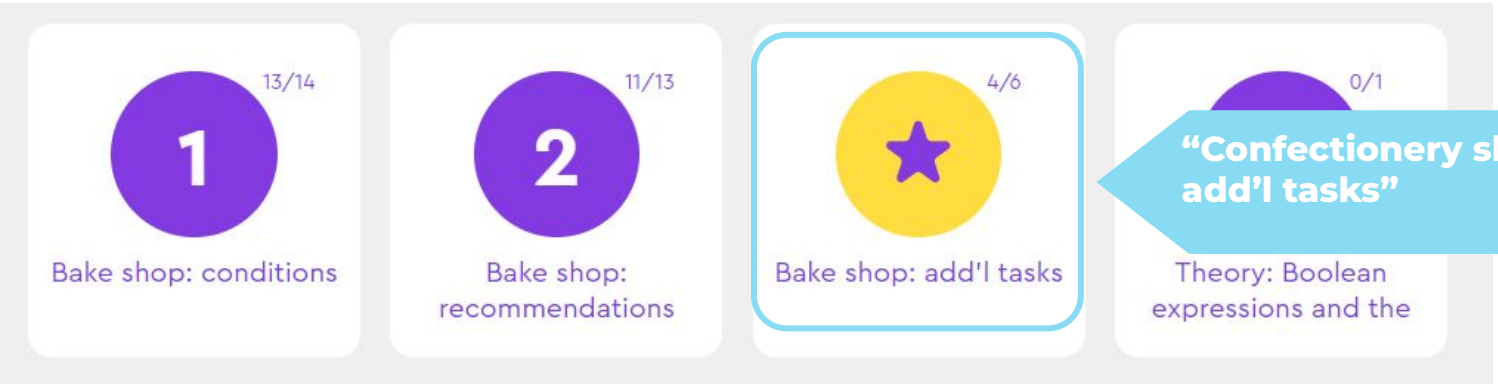
Share with your colleagues:

1. What was the best thing you managed to do?
2. What didn’t work out the way you wanted?
3. What should you do next time to ensure success?

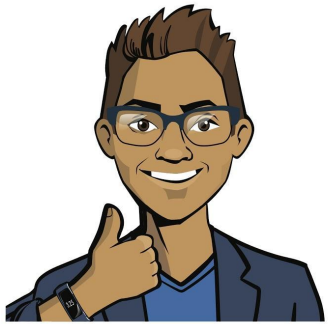


Emily,
Project Manager

Additional tasks to improve efficiency



“Confectionery shop: add’l tasks”



Wrapping up the workday