algorithmics

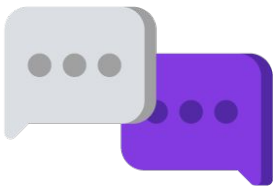
Module 2. Lesson 2.

Nested conditional statements

Link to the methodological guidelines

Discussion:

Programming smart recommendations

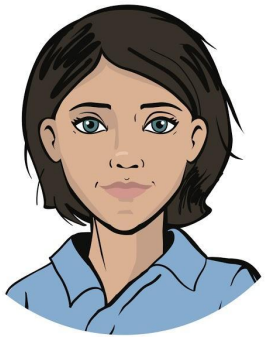


Congratulations on your professional success!

The news about the Sweet Stories software has spread quickly through the professional community.

Now, the ProTeam specialists have been approached by the Longevity health food store with a request to set up recommendations by product categories **and** customer wishes.

This task seems to be similar to the previous one. Are you ready to try it?



Emily,
Project Manager

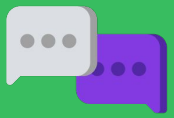
Discussion of the tasks

Multilevel smart recommendations

Task. The program asks the customer for a product category and preferences. After processing the answers, the program prints the recommended product brand. A table of recommendations was provided for this job:

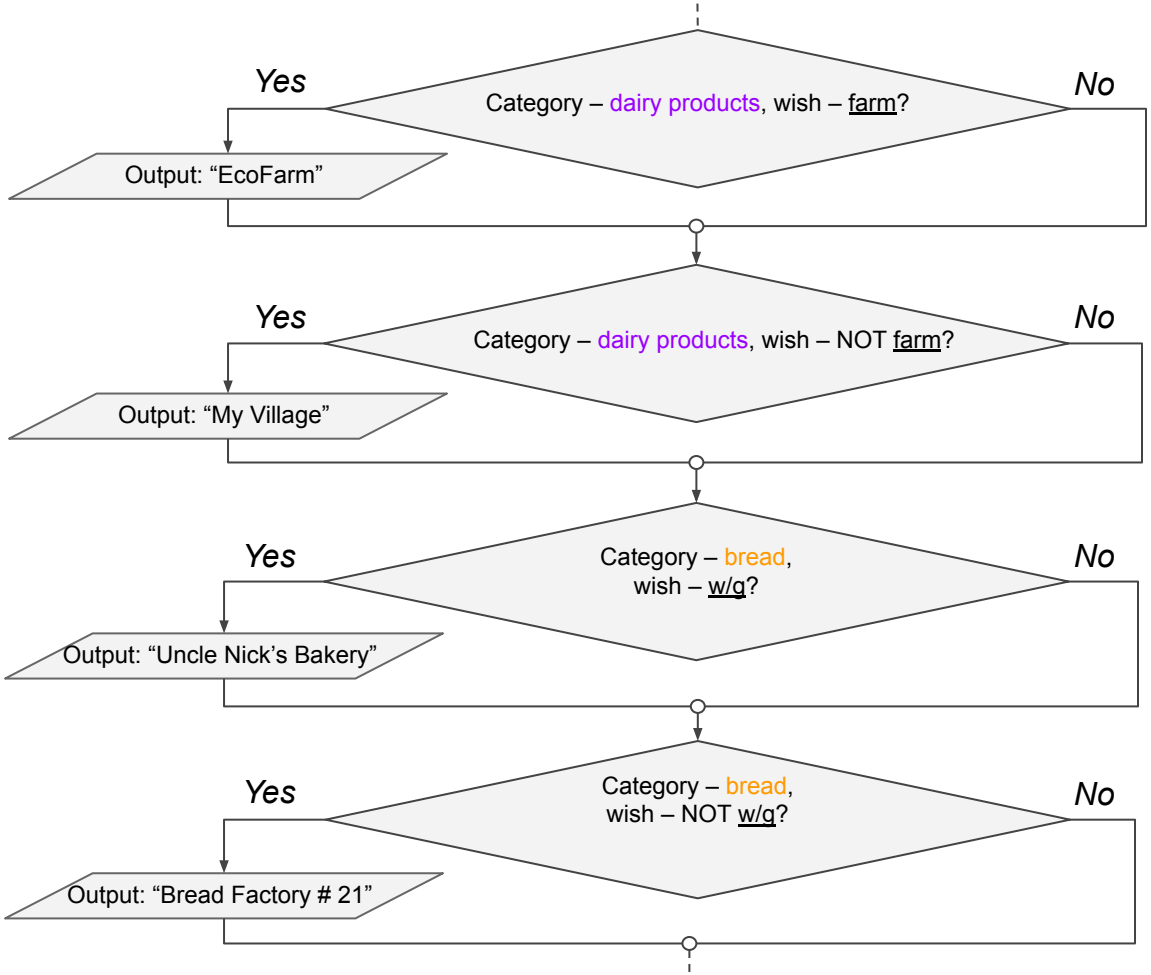
Category	*Wish*	*Recommendation*
Dairy products	Farm	“EcoFarm”
Dairy products	Other preferences	“My Village”
Bread	Whole grain	“Uncle Nick’s Bakery”
Bread	Other preferences	“Bread Factory # 21”

How do we write such a program? Let’s start with a flowchart.



Discussion of the tasks

The program flowchart might look like this:



Discussion of the tasks

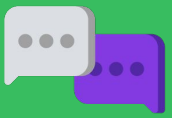
Multilevel smart recommendations

Task. The program asks the customer for a product category and preferences. After processing the answers, the program prints the recommended product brand based on the recommendations table:

```
category = input('Product category:')
wish = input('Wish:')
```

?

How do we program smart recommendations?

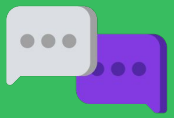


Discussion of the tasks

Multilevel smart recommendations

Task. The program asks the customer for a product category and preferences. After processing the answers, the program prints the recommended product brand based on the recommendations table:

```
category = input('Product category:')
wish = input('Wish:')
if category == 'dairy products' and wish == 'farm':
   print('EcoFarm')
if category == 'dairy products' and wish != 'farm':
   print('My Village')
if category == 'bread' and wish == 'whole grain':
  print('Uncle Nick’s Bakery')
if category == 'bread' and wish != 'whole grain':
  print('Bread Factory # 21')
```



Discussion of the tasks

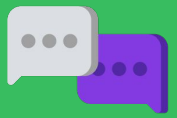
Multilevel smart recommendations

Task. The program asks the customer for a product category and preferences. After processing the answers, the program prints the recommended product brand based on the recommendations table:

```
category = input('Product category:')
wish = input('Wish:')
if category == 'dairy products' and wish == 'farm':
    print('EcoFarm')
if category == 'dairy products' and wish != 'farm':
    print('My Village')
if category == 'bread' and wish == 'whole grain':
   print('Uncle Nick’s Bakery')
if category == 'bread' and wish != 'whole grain':
   print('Bread Factory # 21')
```

The program will work correctly.

Can we **optimize it**?



Discussion of the tasks

Multilevel smart recommendations

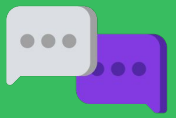
Task. The program asks the customer for a product category and preferences. After processing the answers, the program prints the recommended product brand based on the recommendations table:

```
category = input('Product category:')
wish = input('Wish:')
if category == 'dairy products' and wish == 'farm':
    print('EcoFarm')
if category == 'dairy products' and wish != 'farm':
    print('My Village')
if category == 'bread' and wish == 'whole grain':
  print('Uncle Nick's Bakery')
if category == 'bread' and wish != 'whole grain':
  print('Bread Factory # 21')
```

The category check is repeated.

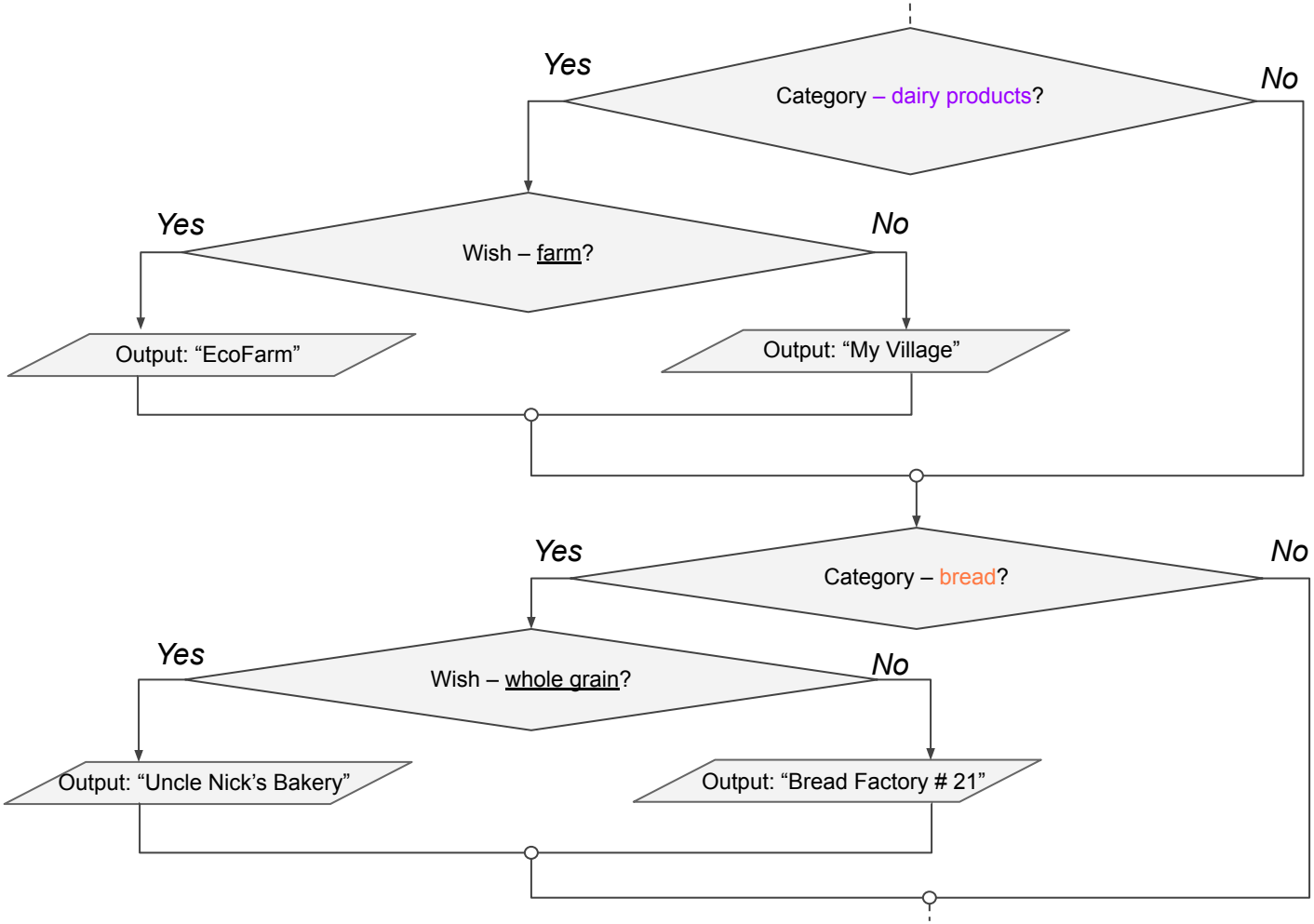
Can we **do the check once**, not twice?

If so, how will the flowchart change?



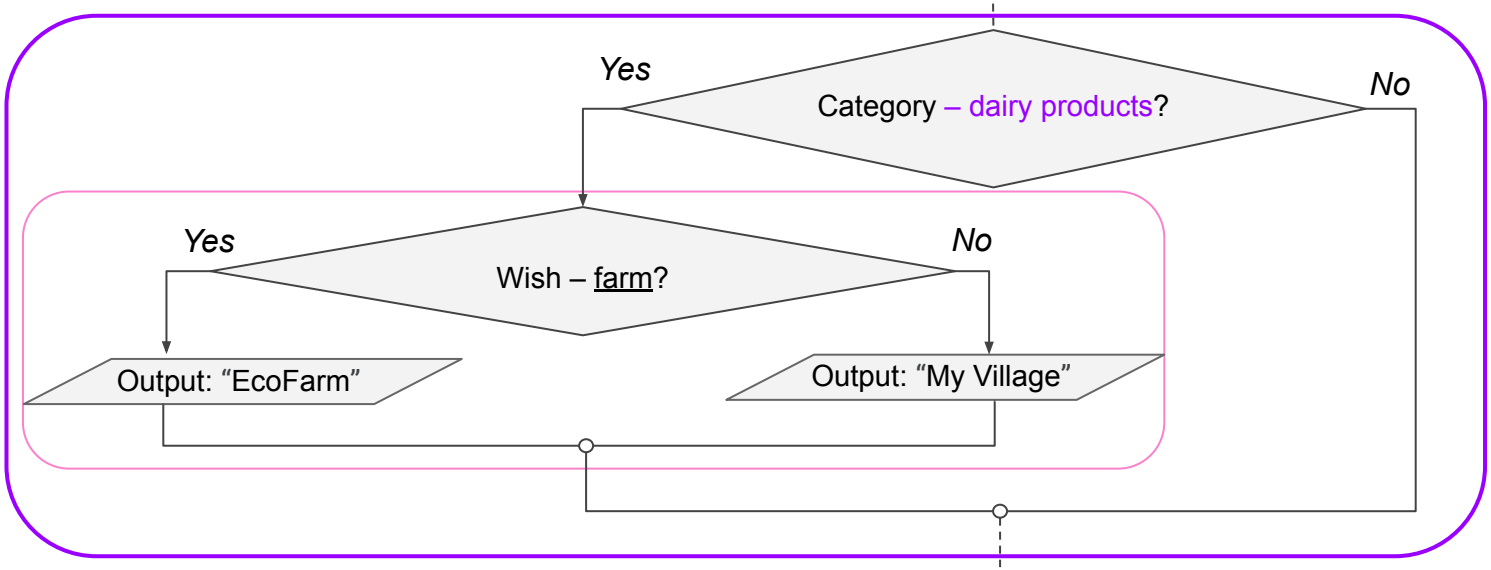
Discussion of the tasks

The program flowchart might look as follows:



Discussion of the tasks

The program flowchart might look like this:



It resembles a nested construct...

How do we program the nesting of one conditional statement in another?

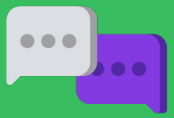
Discussion of the tasks

Multilevel smart recommendations

Task. The program asks the customer for a product category and preferences. After processing the answers, the program prints the recommended product brand based on the recommendations table:

```
category = input('Product category:')
wish = input('Wish:')
```

?



Discussion of the tasks

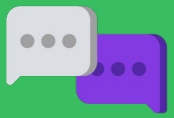
Multilevel smart recommendations

Task. The program asks the customer for a product category and preferences. After processing the answers, the program prints the recommended product brand based on the recommendations table:

```
category = input('Product category:')
wish = input('Wish:')
if category == 'dairy products':
    if wish == 'farm':
        print('EcoFarm')
    else:
        print('My Village')
if category == 'bread':
    if wish == 'whole grain':
        print('Uncle Nick’s Bakery')
    else:
        print('Bread Factory # 21')
```

It turns out that nesting works for the conditional statement too!

By using it, we avoid overly complex condition checking.



Discussion of the tasks

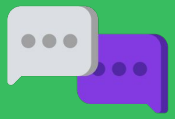
The goal of the workday is to

set up smart recommendations *for a store.*

Our client wants the program to offer products and brands based on customer wishes.

Today you:

- learn that a nested conditional statement is a programming tool for complex conditions;
- learn and program a conditional statement with multiple branches;
- program multilevel smart recommendations.



Discussion of the tasks

Confirmation of qualifications

How can we create a simple logical expression ?

What values can it take?

Confirmation of qualifications

Simple logical expression

A logical expression accepts only the value `True` or `False`.

Comparison operators can be used to make up logical expressions.

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

Confirmation of qualifications

Name the values of the expressions

```
1 == 2
```

```
2 == 1 + 1
```

```
a == 5
```

```
15 == '15'
```

```
3.14 > 3
```

```
'Hello' != 'hello'
```

```
(3 + 2) * 0.1 == 0.5
```

Confirmation of qualifications

Name the values of the expressions

`1 == 2`	**False**	
`2 == 1 + 1`	**True**	
`a == 5`	**True**	If the value of the variable **a** is 5, otherwise False.
`15 == '15'`	**False**	The string is not equal to the number.
`3.14 > 3`	**True**	
`'Hello' != 'hello'`	**True**	Two strings are equal only if all characters in them are exactly the same.
`(3 + 2) * 0.1 == 0.5`	**True**	

Confirmation of qualifications

How can we create a compound logical expression ?

Confirmation of qualifications

Compound logical expression

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

Operator	*Name*	*Used when we need to:*
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.*

Confirmation of qualifications

Name the values of the expressions

```
'Yes' != 'Yes' and 3 > 2
```

```
1 > 2 and 3 > 2
```

```
1 > 2 or 3 > 2
```

```
ans == 'Yes' and 2 == '2'
```

```
5 > 3 and 6 > 3
```

```
ans == 'Yes' or ans != 'Yes'
```

Confirmation of qualifications

Name the values of the expressions

`'Yes' != 'Yes' and 3 > 2`	**True**	
`1 > 2 and 3 > 2`	**False**	Expression 1 is false (the first ***and*** second must be true).
`1 > 2 or 3 > 2`	**True**	Expression 2 is true (the first ***or*** second must be true).
`ans == 'Yes' and 2 == '2'`	**False**	Expression 2 is false.
`5 > 3 and 6 > 3`	**True**	
`ans == 'Yes' or ans != 'Yes'`	**True**	Indeed, the value of a variable is either equal to some value or not.

Confirmation of qualifications

What is a conditional statement ?

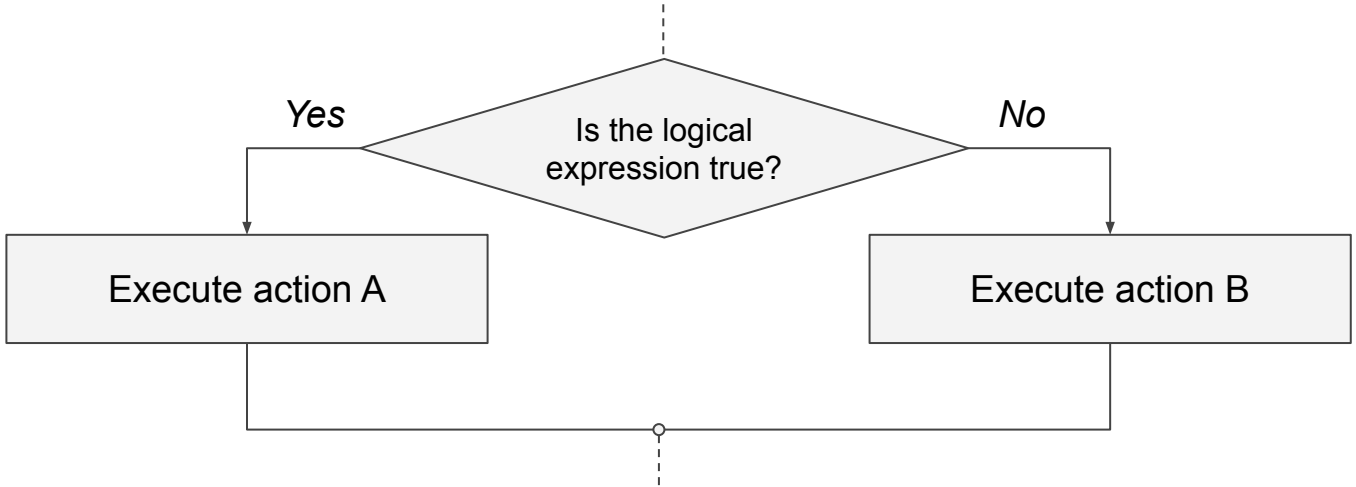
What is it used for?

Confirmation of qualifications

A conditional statement

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

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



Confirmation of qualifications

How do we write a conditional statement ?

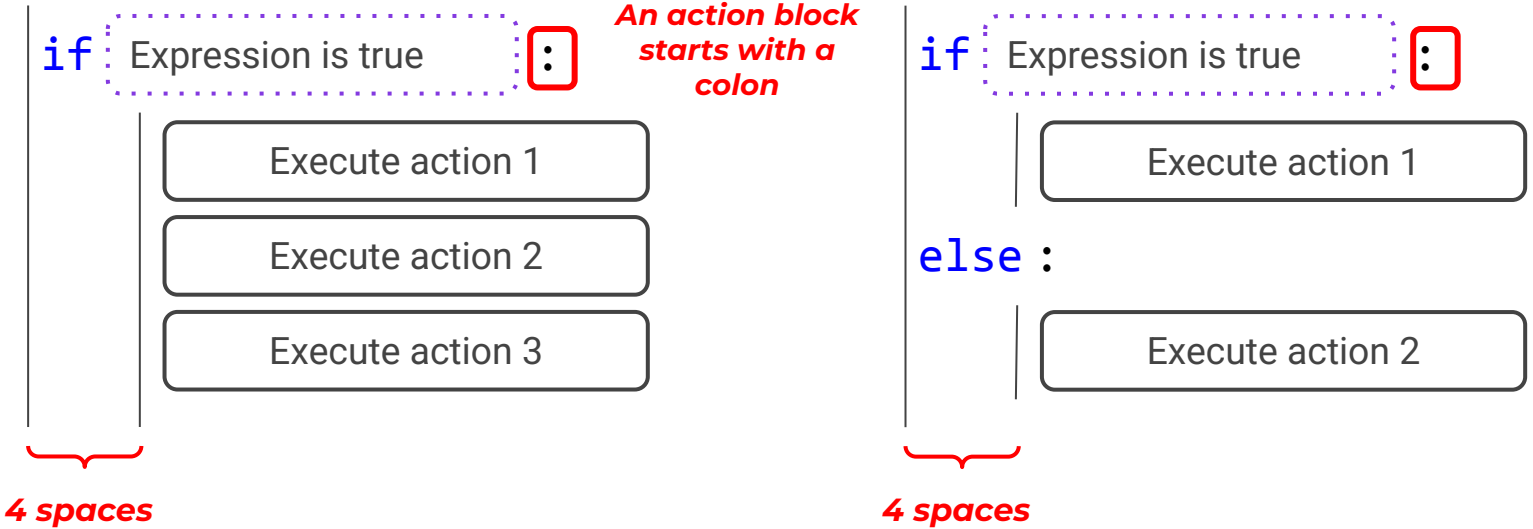
What service words do we use?

Confirmation of qualifications

Conditional statement

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

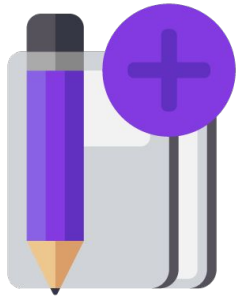
```
if
else
```



Confirmation of qualifications

Brainstorm:

Nested conditional statement

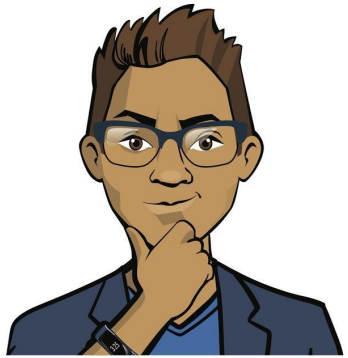


What is the special feature of a nested conditional statement?

You have made sure that a nested conditional statement can be replaced with a compound logical expression.

On the other hand, nesting:

- reduces condition checking;
- optimizes the program code;
- makes the code more logical and readable.

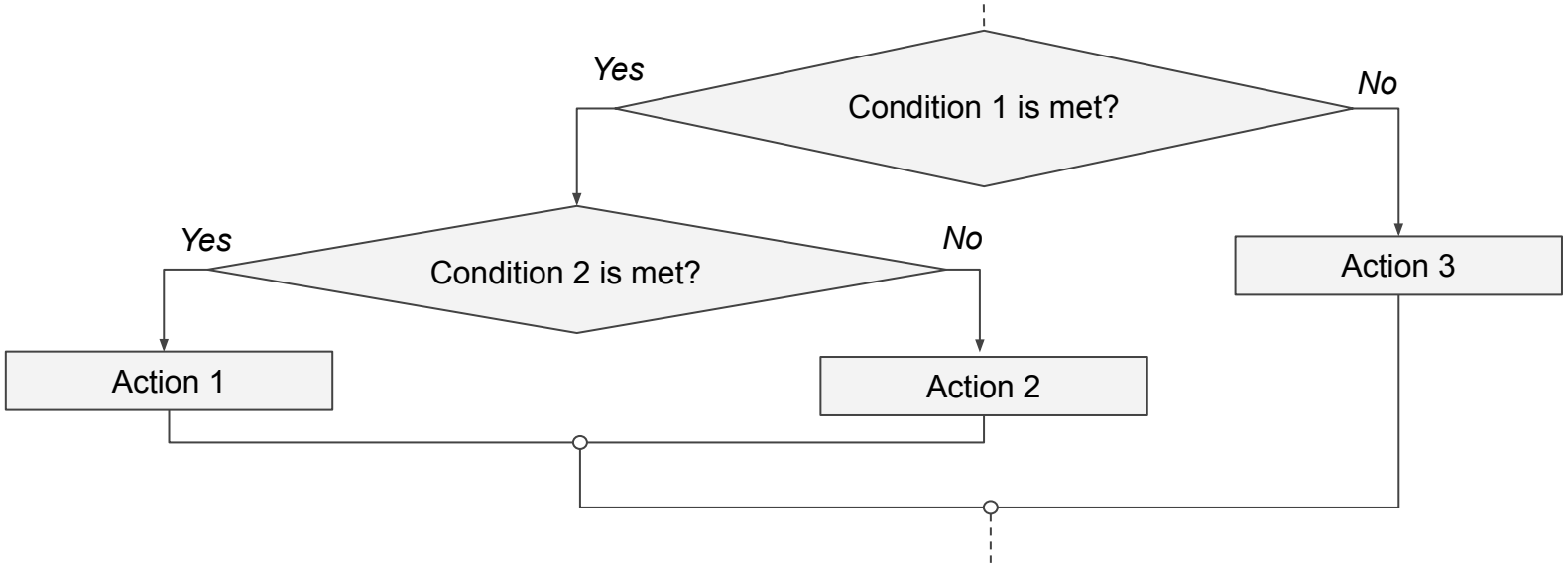


Brainstorm

Nested conditional statement

To use a nested conditional statement, you need to:

- determine the order in which the conditions will be checked;
- know how to write a conditional statement.



Brainstorm

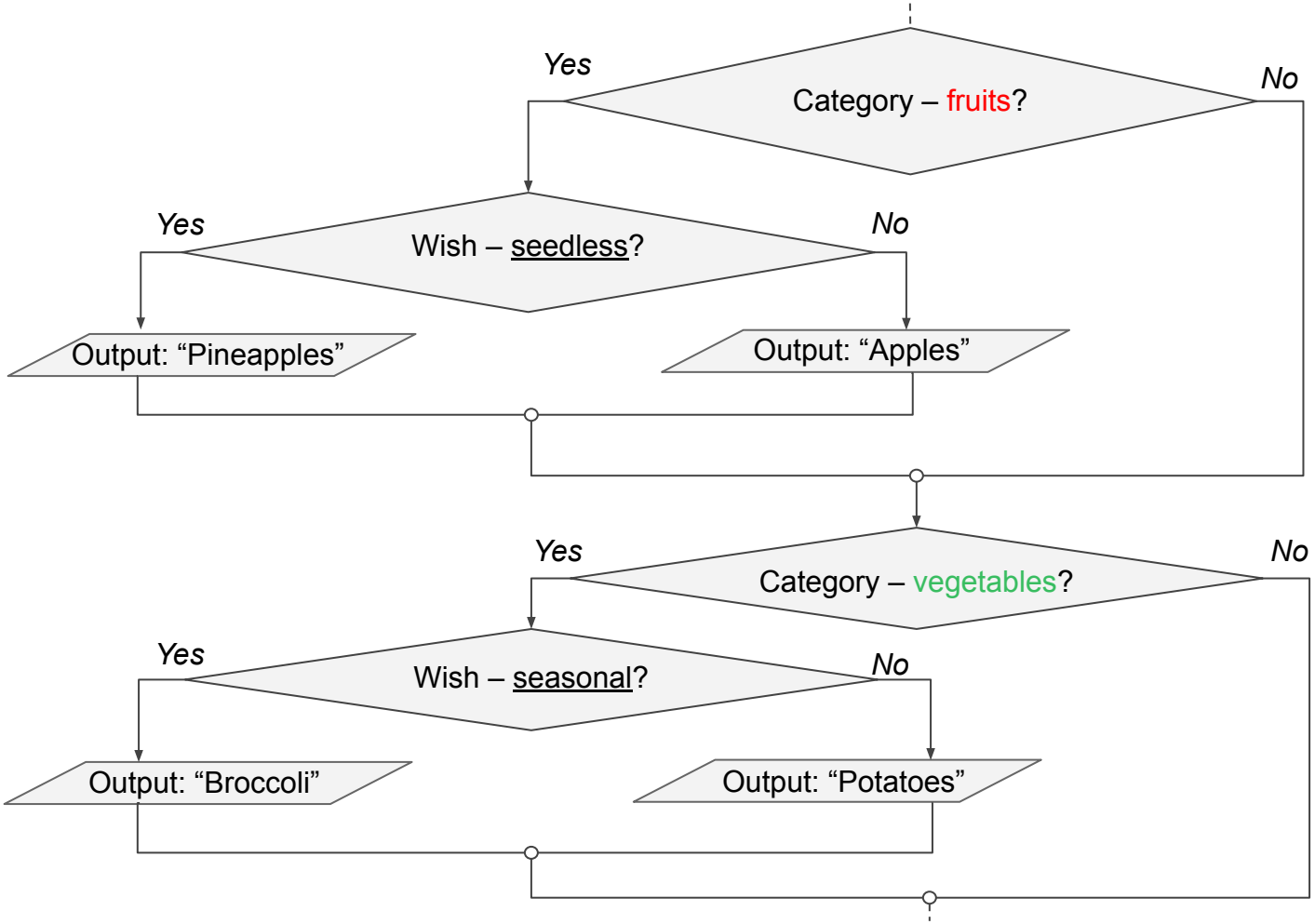
Order of checking conditions

Task. Create an algorithm for printing recommendations for fruits and vegetables. For those who want seedless fruits, print “Pineapples”. For others, print “Apples”. For those who want seasonal vegetables, print “Broccoli”. For others, print “Potatoes”.

What conditions should we specify? Which condition would it make sense to check first?

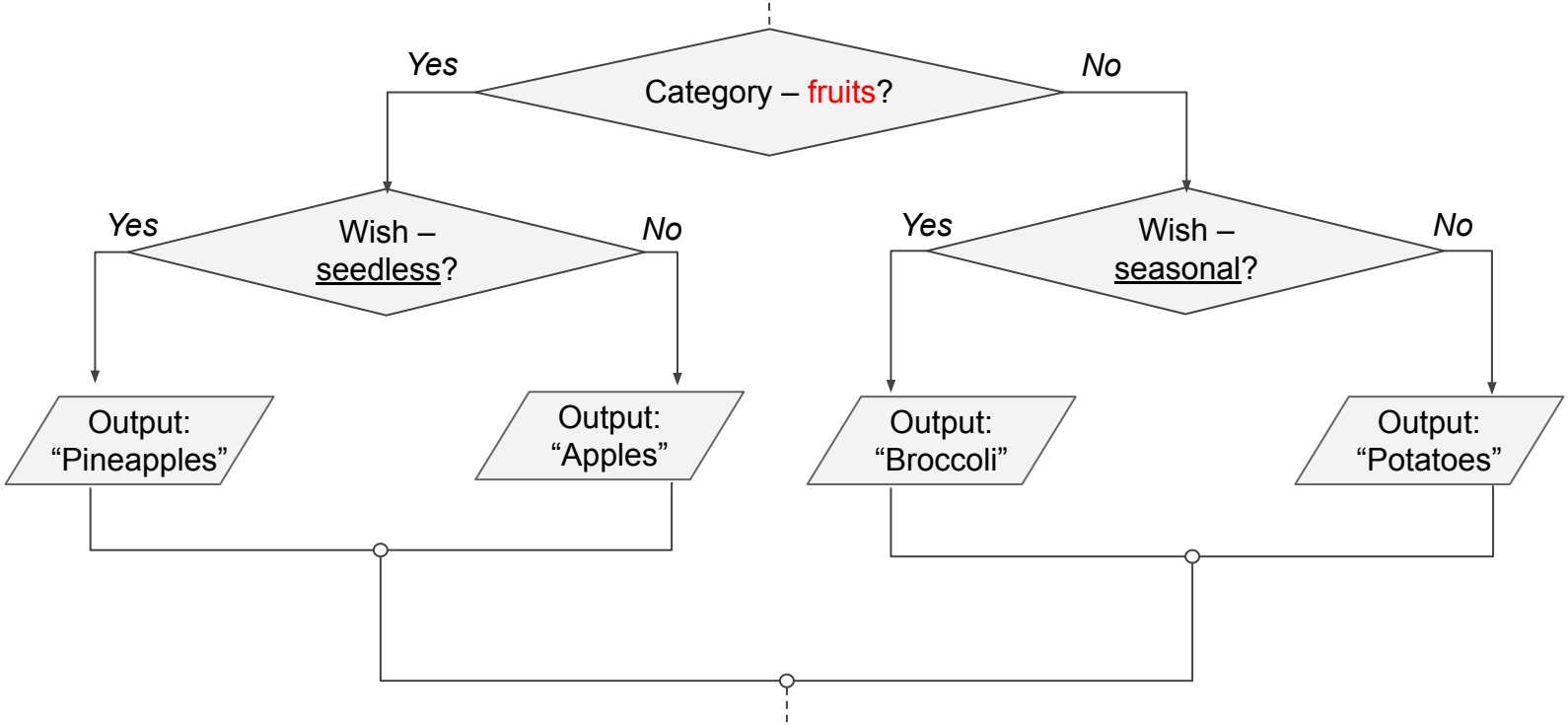
Brainstorm

Possible algorithm of the program:



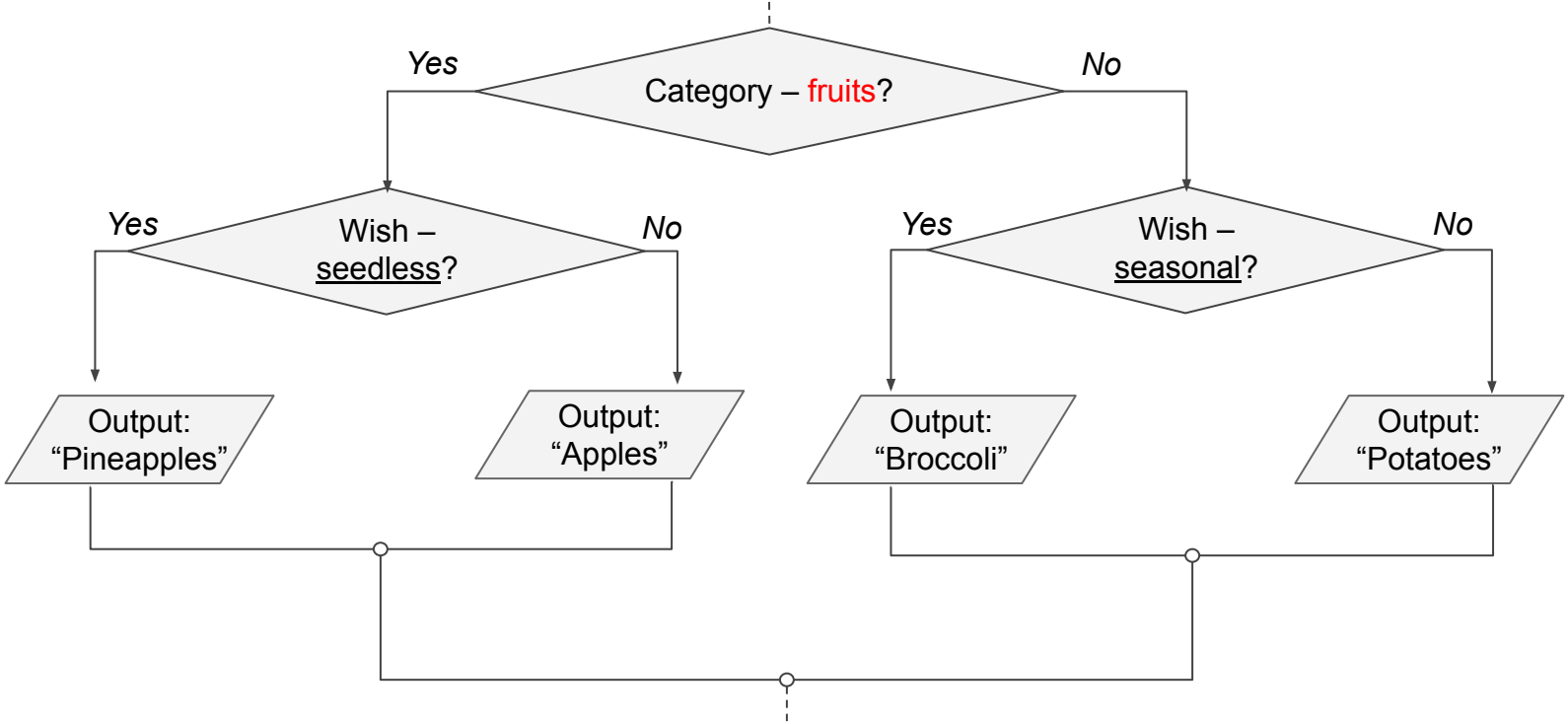
Brainstorm

Will such an algorithm work?



Brainstorm

Will such an algorithm work?



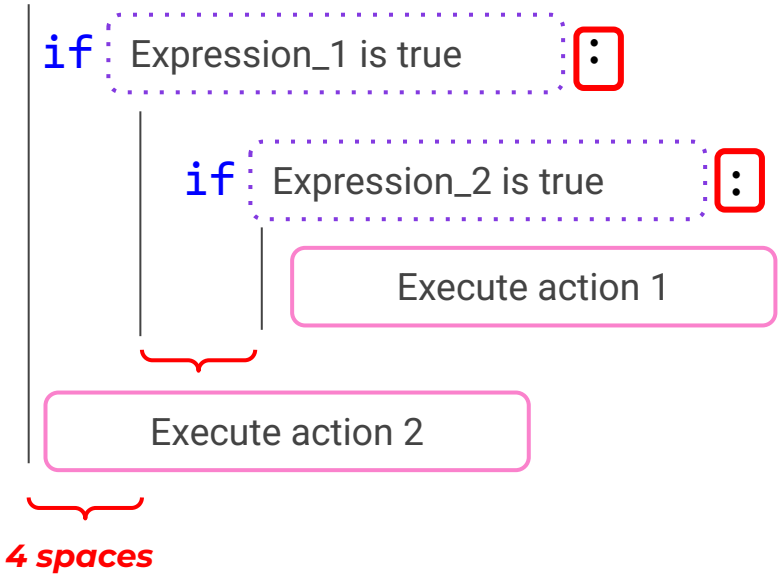
Yes, but not always correctly.

If the customer types the category "Bread" and the wish "With seeds", they will be recommended potatoes.

Brainstorm

Nesting design

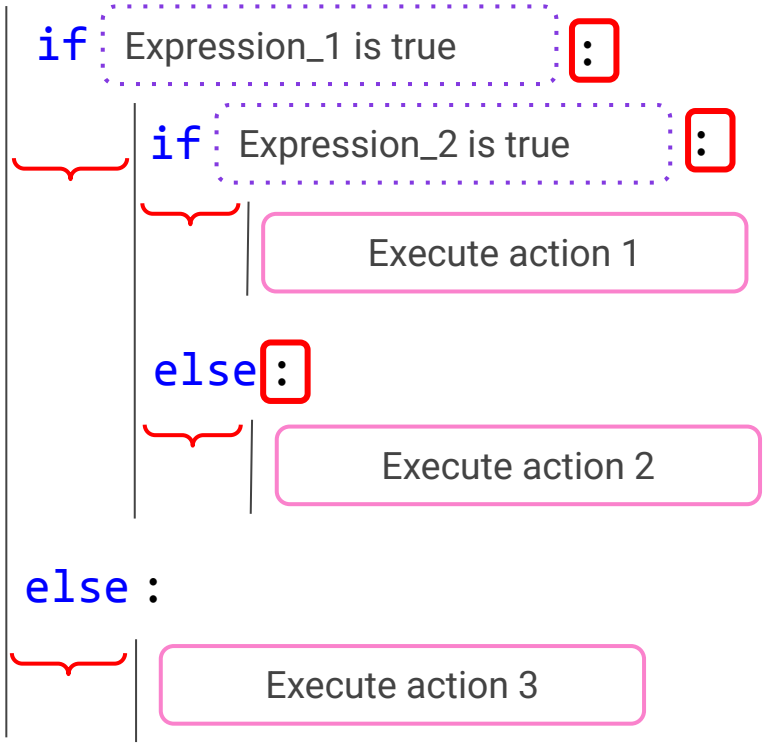
There are no new design rules. You must adhere very carefully to the rules you already know.



Brainstorm

Nesting design

There are no new design rules. You must adhere very carefully to the rules you already know.



Brainstorm

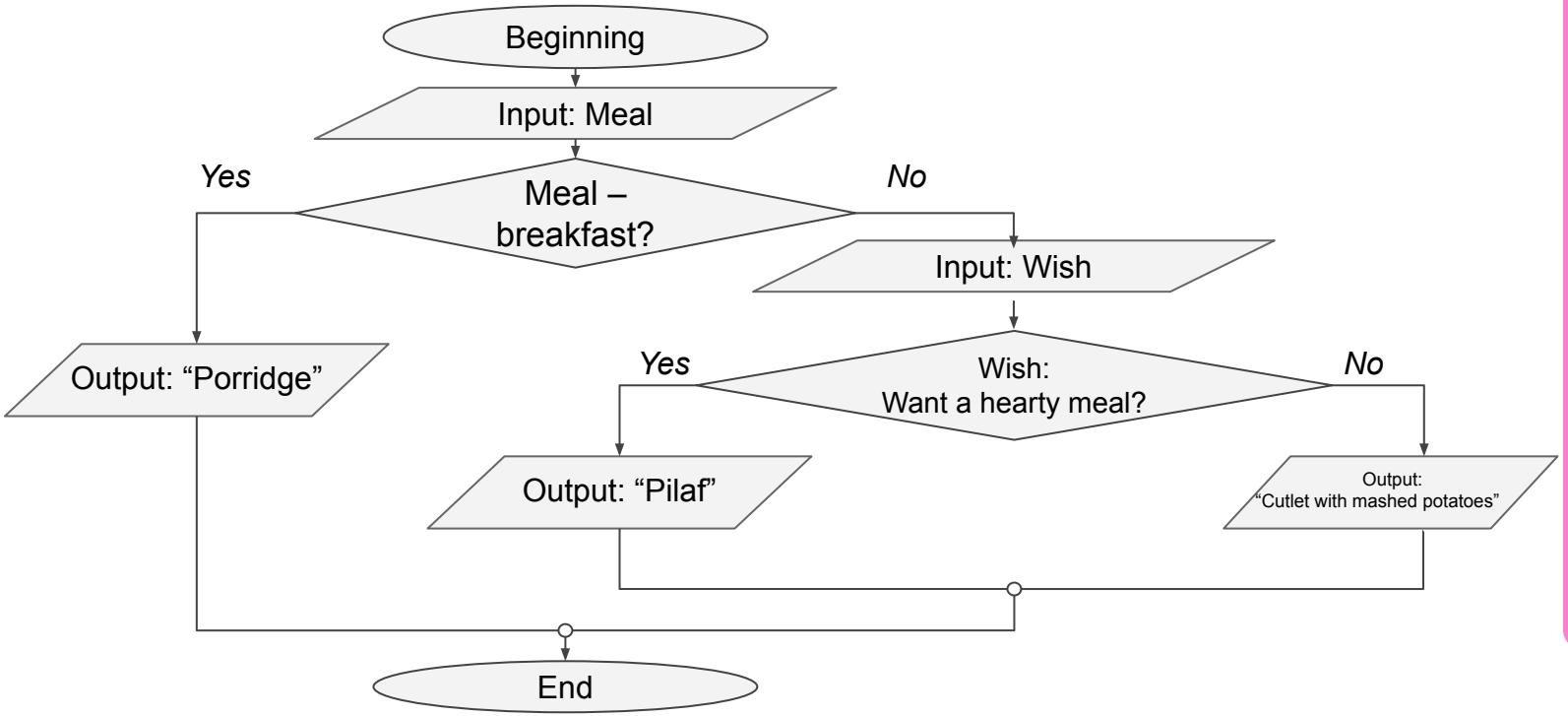
Nested conditional statement

Task. Write a program that offers ready-to-eat dishes for meals.
If the customer enters “Breakfast”, then recommend porridge. For other meals, if the customer wants to eat a hearty meal, recommend pilaf; in other cases, recommend a cutlet with mashed potatoes.

Brainstorm

Nested conditional statement

Task. Write a program that offers ready-to-eat dishes for meals.
If the customer enters “Breakfast”, then recommend porridge. For other meals, if the customer wants to eat a hearty meal, recommend pilaf; in other cases, recommend a cutlet with mashed potatoes.

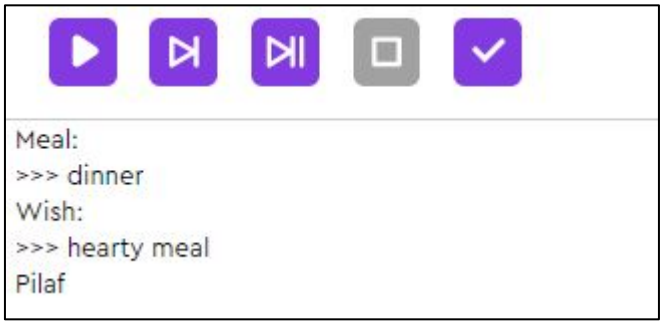


Brainstorm

Nested conditional statement

Task. Write a program that offers ready-to-eat dishes for meals.
If the customer enters “Breakfast”, then recommend porridge. For other meals, if the customer wants to eat a hearty meal, recommend pilaf; in other cases, recommend a cutlet with mashed potatoes.

```
meal = input('Meal:')
meal = meal.lower()
if meal == 'breakfast':
    print('Porridge')
else:
    wish = input('Wish:')
    wish = wish.lower()
    if wish == 'hearty meal':
        print('Pilaf')
    else:
        print('Cutlet with mashed potatoes')
```



Brainstorm

Nested conditional statement

Task. Write a program that offers ready-to-eat dishes for meals.
If the customer enters “Breakfast”, then recommend porridge. For other meals, if the customer wants to eat a hearty meal, recommend pilaf; in other cases, recommend a cutlet with mashed potatoes.

```
meal = input('Meal:')
meal = meal.lower()
if meal == 'breakfast':
    print('Porridge')
else:
    wish = input('Wish:')
    wish = wish.lower()
    if wish == 'hearty meal':
        print('Pilaf')
    else:
        print('Cutlet with mashed potatoes')
```

What the program will print on sequential data input:

- Breakfast
- Dinner – Japanese cuisine
- Dinner – hearty meal
- Supper – no wishes

Brainstorm

Nested conditional statement

Task. Write a program that offers ready-to-eat dishes for meals.
If the customer enters “Breakfast”, then recommend porridge. For other meals, if the customer wants to eat a hearty meal, recommend pilaf; in other cases, recommend a cutlet with mashed potatoes.

```
meal = input('Meal:')
meal = meal.lower()
if meal == 'breakfast':
    print('Porridge')
else:
    wish = input('Wish:')
    wish = wish.lower()
    if wish == 'hearty meal':
        print('Pilaf')
    else:
        print('Cutlet with mashed potatoes')
```

What the program will print on sequential data input:

- Breakfast
- Dinner – Japanese cuisine
- Dinner – hearty meal
- Supper – no wishes

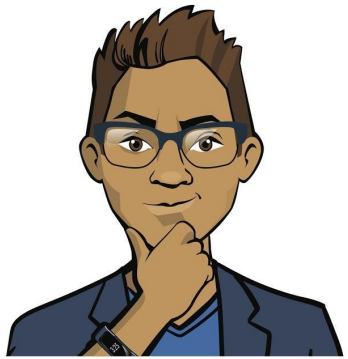
Give an example of input data so that the program prints:

- Cutlet with mashed potatoes
- Porridge
- Pilaf

Brainstorm

Conclusions:

1. **The idea of nested constructs** applies to the conditional statement as well. This helps avoid the unnecessary complexity of condition checks.
2. To **program** a nested conditional statement, you need to:
   - determine the order in which the conditions will be checked;
   - know how to down a conditional construct.



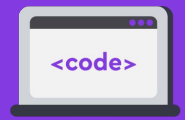
Brainstorm

Store:

Product recommendation

Do the task on the platform

“Store: Recommendations”

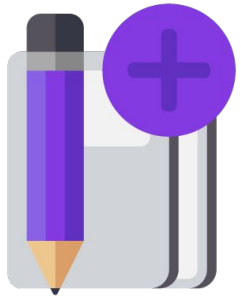


Store:
Product
recommendation

Break

Brainstorm:

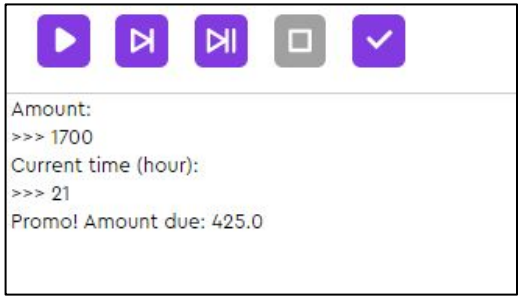
Conditional statement with multiple branches



Let’s go over a task

“Happy Hours” Task . Write a program that requests input of the amount due and the current time. If goods are bought from 10 a.m. to 12 a.m., the amount is reduced 2 times. From 8 p.m. to 10 p.m., reduce it 4 times. The Longevity Store works from 8 a.m. to 10 p.m.

?



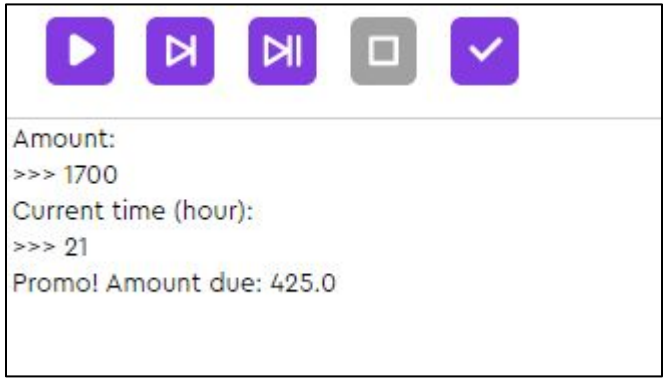
How do we write such a program?

Brainstorm

Let’s go over a task

"Happy Hours" Task . Write a program that requests input of the amount due and the current time. If goods are bought from 10 a.m. to 12 a.m., the amount is reduced 2 times. From 8 p.m. to 10 p.m., reduce it 4 times. The Longevity Store works from 8 a.m. to 10 p.m.

```
total = int(input('Amount:'))
time = int(input('Current time (hour):'))
if time >= 10 and time <= 12:
    total = total/2
    print('Promo! Amount due:', total)
if time >= 20 and time <= 22:
    total = total/4
    print('Promo! Amount due:', total)
if time > 8 and time < 10 or time > 12 and time < 20:
    print('Amount due:', total)
```



Brainstorm

Let’s go over a task

"Happy Hours" Task . Write a program that requests input of the amount due and the current time. If goods are bought from 10 a.m. to 12 a.m., the amount is reduced 2 times. From 8 p.m. to 10 p.m., reduce it 4 times. The Longevity Store works from 8 a.m. to 10 p.m.

```
total = int(input('Amount:'))
time = int(input('Current time (hour):'))
if time >= 10 and time <= 12:
    total = total/2
    print('Promo! Amount due:', total)
if time >= 20 and time <= 22:
    total = total/4
    print('Promo! Amount due:', total)
if time > 8 and time < 10 or time > 12 and time < 20:
    print('Amount due:', total)
```

Sometimes it becomes difficult to combine parts of a condition into an expression.
Is there an easier way to describe all the remaining cases?

Brainstorm

Multiple branch conditional statement

As in the case of a nested conditional statement, this construct can be replaced with compound expressions, but using it can make your code simpler and more efficient.

```
if Expression_1 is true :
    Execute action 1
elif Expression_2 is true :
    Execute action 2
else :
    Execute action 3
```

If Expression_1 is true,

then execute action 1.

Else if Expression_2 is true,

then execute action 2.

In all other cases,

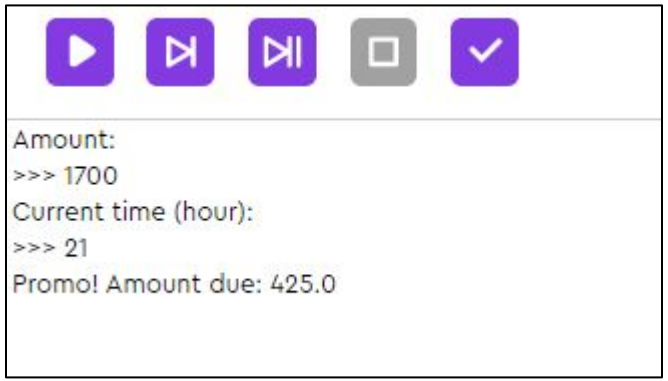
execute action 3.

Brainstorm

Let’s go over a task

"Happy Hours" Task . Write a program that requests input of the amount due and the current time. If goods are bought from 10 a.m. to 12 a.m., the amount is reduced 2 times. From 8 p.m. to 10 p.m., reduce it 4 times.

```
total = int(input('Amount:'))
time = int(input('Current time (hour):'))
if time >= 10 and time <= 12:
   total = total/2
   print('Promo! Amount due:', total)
elif time >= 20 and time <= 22:
   total = total/4
   print('Promo! Amount due:', total)
else:
   print('Amount due:', total)
```



Brainstorm

Let’s go over a task

"Happy Hours" Task . Write a program that requests input of the amount due and the current time. If goods are bought from 10 a.m. to 12 a.m., the amount is reduced 2 times. From 8 p.m. to 10 p.m., reduce it 4 times.

```
total = int(input('Amount:'))
time = int(input('Current time (hour):'))
if time >= 10 and time <= 12:
   total = total/2
   print('Promo! Amount due:', total)
elif time >= 20 and time <= 22:
   total = total/4
   print('Promo! Amount due:', total)
else:
   print('Amount due:', total)
```

What the program will print on sequential data input:

- 1000 — 9
- 1000 — 12
- 1000 — 15
- 1000 — 22
- 1000 — 23

Brainstorm

Let’s go over a task

"Meal" Task . Write a program that offers dishes for meals. If the user enters “Breakfast”, then offer “Porridge”. If “Dinner”, then “Meatball soup”. In all other cases, print “Pancakes with fish”.

?

How do we write such a program?

Brainstorm

Let’s go over a task

"Meal" Task . Write a program that offers dishes for meals. If the user enters “Breakfast”, then offer “Porridge”. If “Dinner”, then “Meatball soup”. In all other cases, print “Pancakes with fish”.

```
meal = input('Meal:')
if meal == 'breakfast':
   print('Porridge')
if meal == 'Dinner':
   print('Meatball soup')
if meal != 'Breakfast' and meal != 'Dinner':
   print('Pancakes with fish')
```

Option 1. Using a compound logical expression.

Brainstorm

Let’s go over a task

"Meal" Task . Write a program that offers dishes for meals. If the user enters “Breakfast”, then offer “Porridge”. If “Dinner”, then “Meatball soup”. In all other cases, print “Pancakes with fish”.

```
meal = input('Meal:')
if meal == 'breakfast':
   print('Porridge')
elif meal == 'Dinner':
   print('Meatball soup')
else:
   print('Pancakes with fish')
```

Brainstorm

Option 2. Using a conditional statement with multiple branches.

Let’s go over a task

"Meal" Task . Write a program that offers dishes for meals. If the user enters “Breakfast”, then offer “Porridge”. If “Dinner”, then “Meatball soup”. In all other cases, print “Pancakes with fish”.

```
meal = input('Meal:')
if meal == 'breakfast':
   print('Porridge')
elif meal == 'Dinner':
   print('Meatball soup')
else:
   print('Pancakes with fish')
```

What the program will print upon data input:

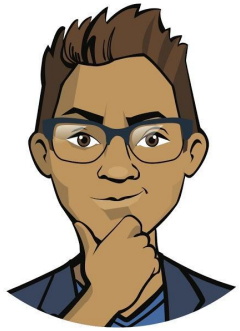
- Breakfast
- Lunch
- Dinner
- I don’t know

Brainstorm

Option 2. Using a conditional statement with multiple branches.

Conclusions:

1. There are **many ways** to program a conditional construct:
   - conditional statement;
   - nested conditional statement;
   - conditional statement with multiple branches.
2. Any task can be solved through the “standard” conditional operator. But:
   - **nesting allows us to reduce** condition checking;
   - **multiple branches make it easier** to check the remaining conditions.



Cole,
Senior Developer

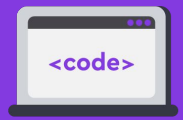
Brainstorm

Store:

Product recommendation

Do the task on the platform

“Store: Recommendations 2”

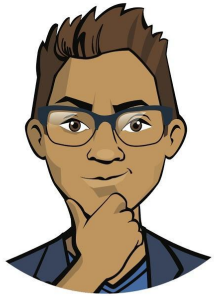


Store:
Product recommendation 2

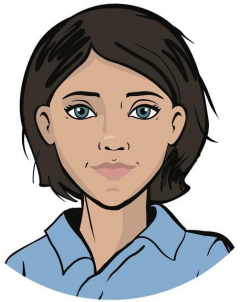
End of the workday

To complete the workday, pass a technical interview

1. What is a nested conditional statement? Where and why can it be applied?
2. What is a conditional statement with multiple branches? Where and why can it be applied?
3. Which conditional statement is better: standard, nested or multi-branch?



Cole,
Senior Developer



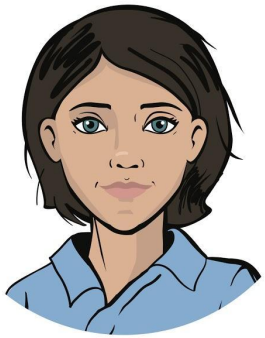
Emily,
Project Manager

Wrapping up the workday

Congratulations on completing the workday!

Today you have:

1. Learned that a nested conditional statement is a programming tool for complex conditions.
2. Learned the new if-elif-else service words used to program a conditional statement with multiple branches.
3. Solved tasks in different ways, choosing the optimal one.



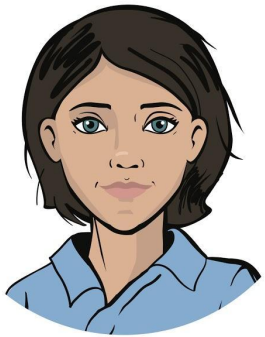
Emily,
Project Manager

Wrapping up the workday

Performance review

Share with your colleagues:

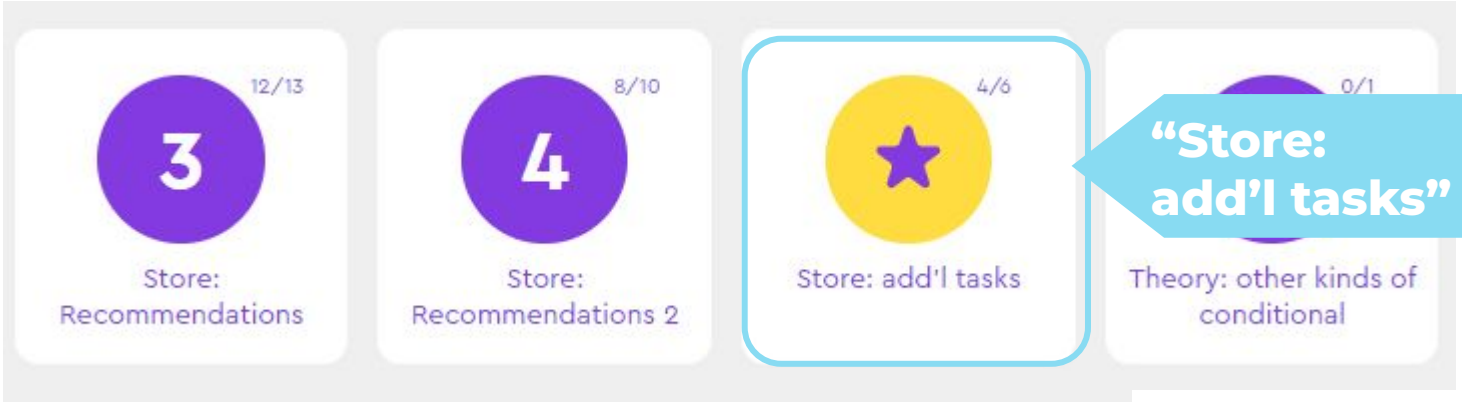
1. What was the best thing you managed to do?
2. What didn’t work out the way you wanted?
3. What should you do next time to ensure success?



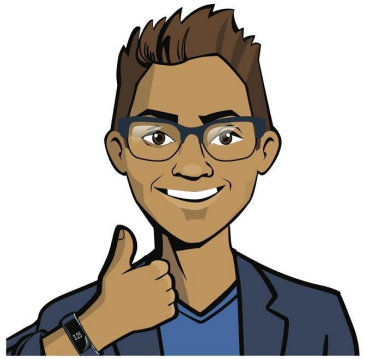
Emily,
Project Manager

Wrapping up the workday

Additional tasks to improve efficiency



"Store: add'l tasks"



Wrapping up the workday