algorithmics

Module 2. Lesson 3.

Loops

Link to the methodological guidelines

Module 2. Lesson 3. Loops

Discussion:

Programming promo code entry

New task

Longevity’s CEO was satisfied with the technical solution we provided. He now needs to expand the site’s functionality by adding the ability to read promotional codes.

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

Emily,
Project Manager

Discussion of the tasks

Promo code entry

Sample task.

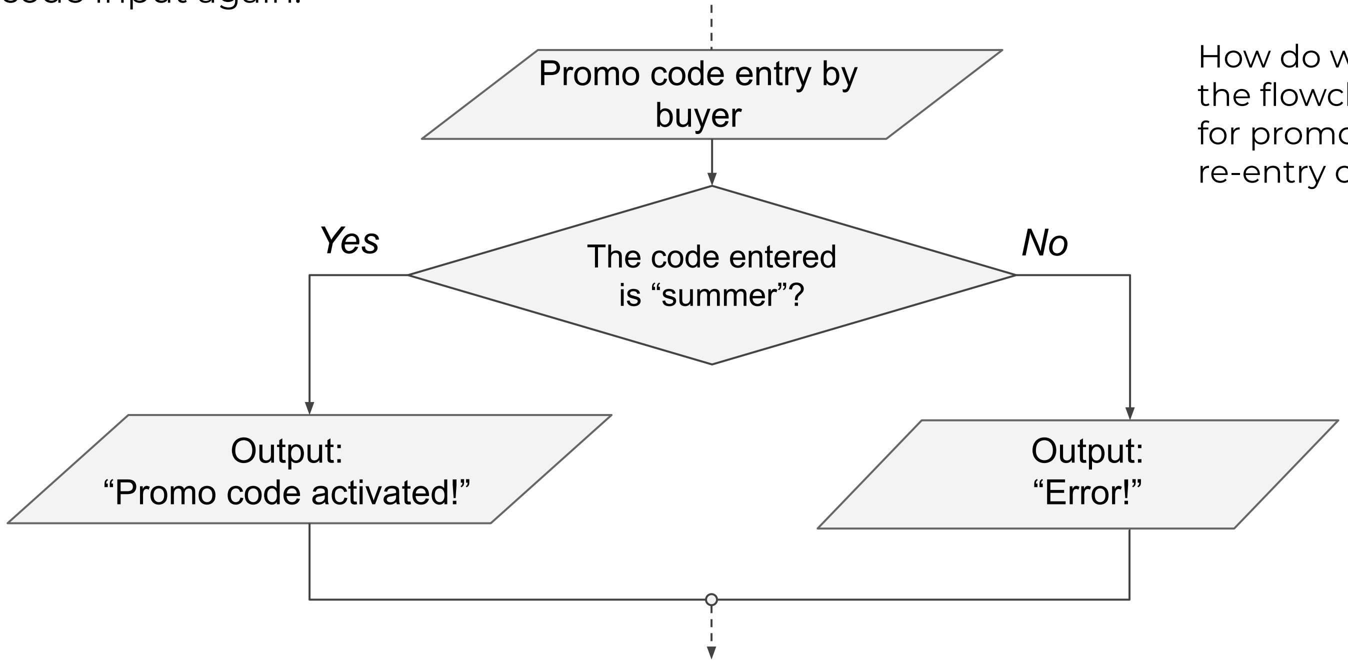
There is a discount available upon entry of the “summer” promo code. The task is to write a program asking users to enter a promo code. If the user enters “summer”, then output “Promo code activated!” Otherwise, output “Error!” and ask for promo code input again.

How can we determine if the promo code entered is the correct one? Make a flowchart for this.

Promo code entry

Sample task.

There is a discount available upon entry of the “summer” promo code. The task is to write a program asking users to enter a promo code. If the user enters “summer”, then output “Promo code activated!” Otherwise, output “Error!” and ask for promo code input again.

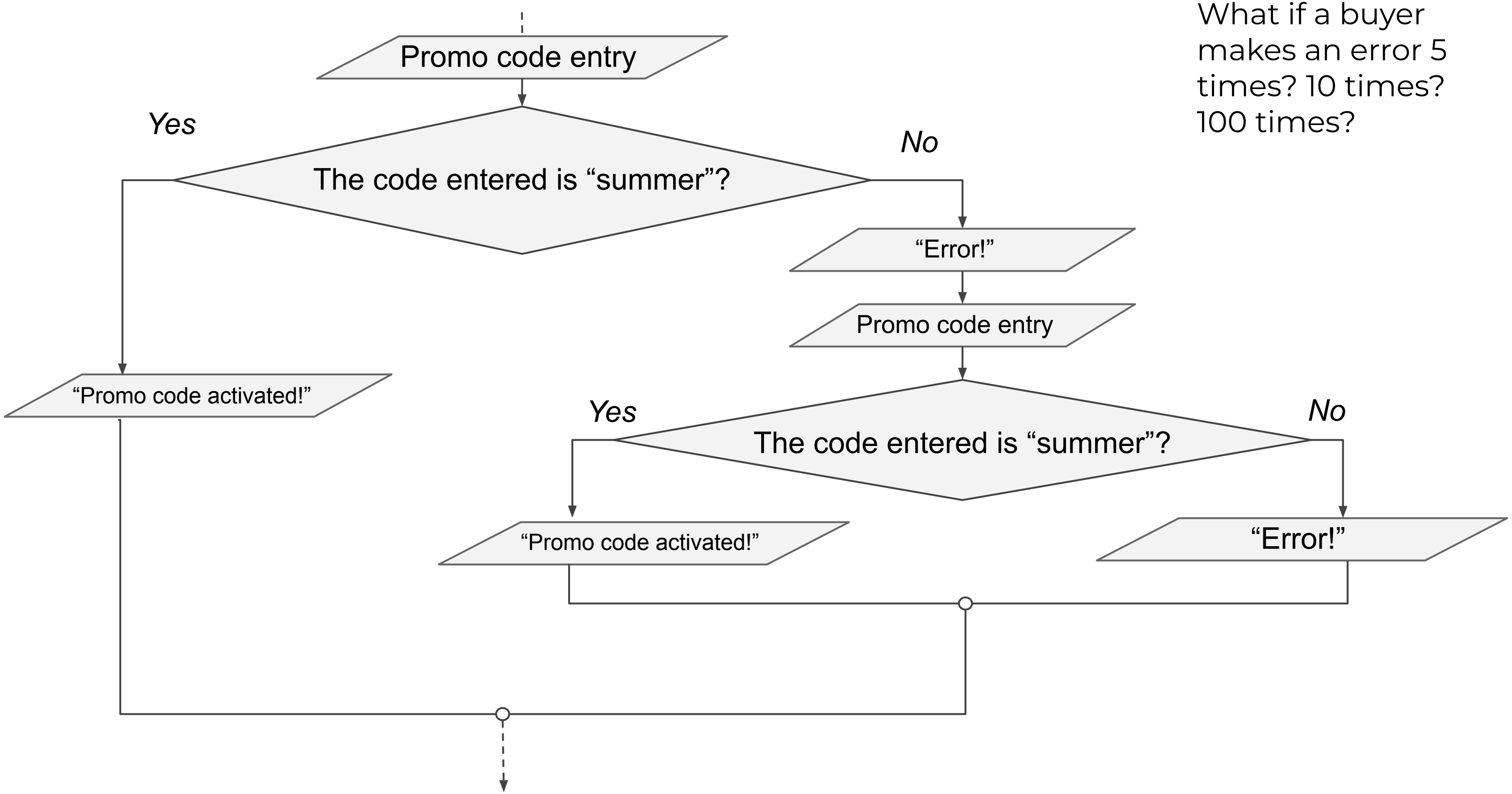

Promo code entry by buyer
The code entered is “summer”?
Yes
No
Output: “Promo code activated!”
Output: “Error!”

How do we change the flowchart to ask for promo code re-entry on error?

Discussion of the tasks

Promo code entry

Sample flowchart:



What if a buyer makes an error 5 times? 10 times? 100 times?

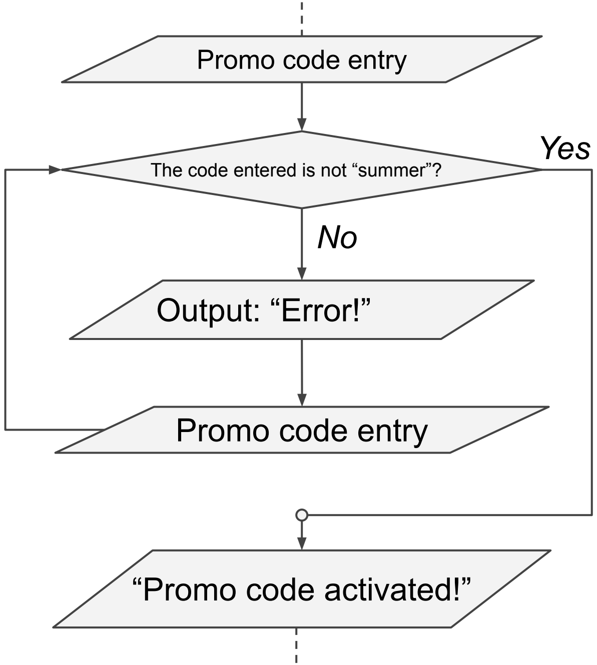
Discussion of the tasks

Promo code entry

Sample task.

There is a discount available upon entry of the “summer” promo code. The task is to write a program asking users to enter a promo code. If the user enters “summer”, then output “Promo code activated!” Otherwise, output “Error!” and ask for promo code input again.

The algorithm can be optimized:



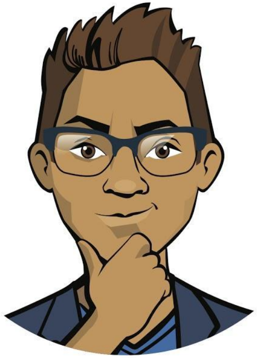
What is the corresponding code?

Discussion of the tasks

Necessary tools

To complete this task, we will need:

Tools	*Functions and operators*
Data input and output	`print()` and `input()`
Logical expression	`promo != 'summer'`
Repetition of actions as long as a certain logical expression is true	?



Cole
Senior Developer

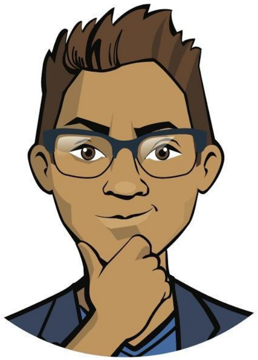
Discussion of the tasks

Necessary tools

To complete this task, we will need:

Tools	*Functions and operators*
Data input and output	`print()` and `input()`
Logical expression	`promo != 'summer'`
Repetition of actions as long as a certain logical expression is true	?

This as yet unknown but important tool is a loop.



Cole,
Senior Developer

Discussion of the tasks

The goal of the workday is to

expand the store's functionality with promo-code reading.

Users must be asked to enter their code until it is correct.

Today you will:

- learn that a loop is a tool to program actions that are repeated as long as a certain logical expression remains true;
- learn about and program several loop types;
- implement promo code entry and other discount mechanics within the Longevity Store.

Discussion of the tasks

Module 2. Lesson 3. Loops

Confirmation of qualifications

How do we construct a simple logical expression? A compound one?

What values can it take?

Confirmation of qualifications

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

Simple logical expressions can be constructed using the comparison operators:

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

Confirmation of qualifications

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

Simple logical expressions can be constructed using the comparison operators:

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

A compound logical expression can be made up from 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

Confirmation of qualifications

Name the values of the expressions:

```
‘Off‘ != ‘Off‘ and 3 == 3
```

```
10.5 > 2.0 and 5.5 > 6.5
```

```
‘John‘ == ‘John‘ or 4 > 10
```

```
ans == ‘Yes‘ and 2 == 20
```

```
2 > 3 or 6 > 3
```

```
ans == ‘No‘ or ans != ‘No‘
```

Confirmation of qualifications

Name the values of the expressions:

`‘Off‘ != ‘Off‘ and 3 == 3`	**False**	Expression 1 is false (the first ***and*** the second must be true).
`10.5 > 2.0 and 5.5 > 6.5`	**False**	Expression 2 is false (the first ***and*** the second must be true).
`‘John‘ == ‘John‘ or 4 > 10`	**True**	Expression 1 is true (the first ***or*** the second must be true).
`ans == ‘Yes‘ and 2 == 20`	**False**	Expression 2 is false.
`2 > 3 or 6 > 3`	**True**	Expression 2 is true.
`ans == ‘No‘ or ans != ‘No‘`	**True**	One of the expressions must be true.

Confirmation of qualifications

What is a conditional statement ?

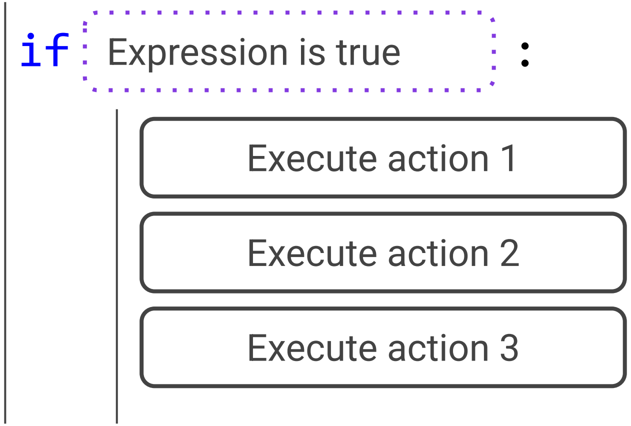
What types of conditional statements do you know?

Confirmation of qualifications

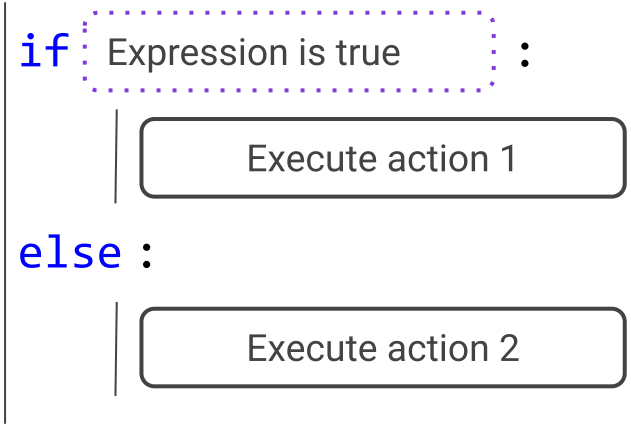
Conditional statement

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

The “classic” conditional statement:



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



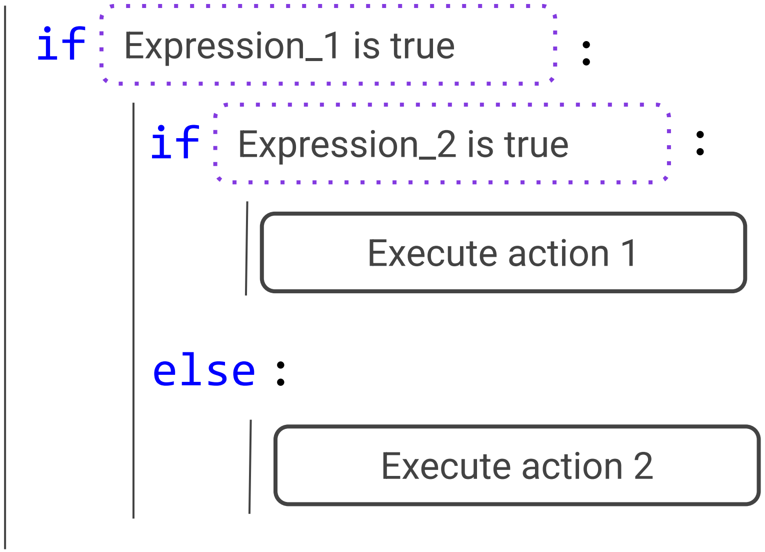
```
if Expression is true :
    Execute action 1
else :
    Execute action 2
```

Confirmation of qualifications

Conditional statement

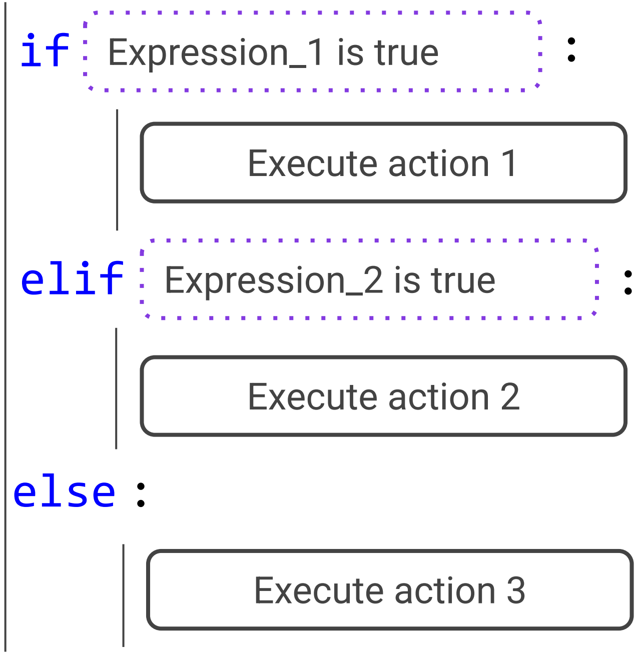
– a command that executes or does not execute an action depending on the value of the logical expression.

Nested conditional statement



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

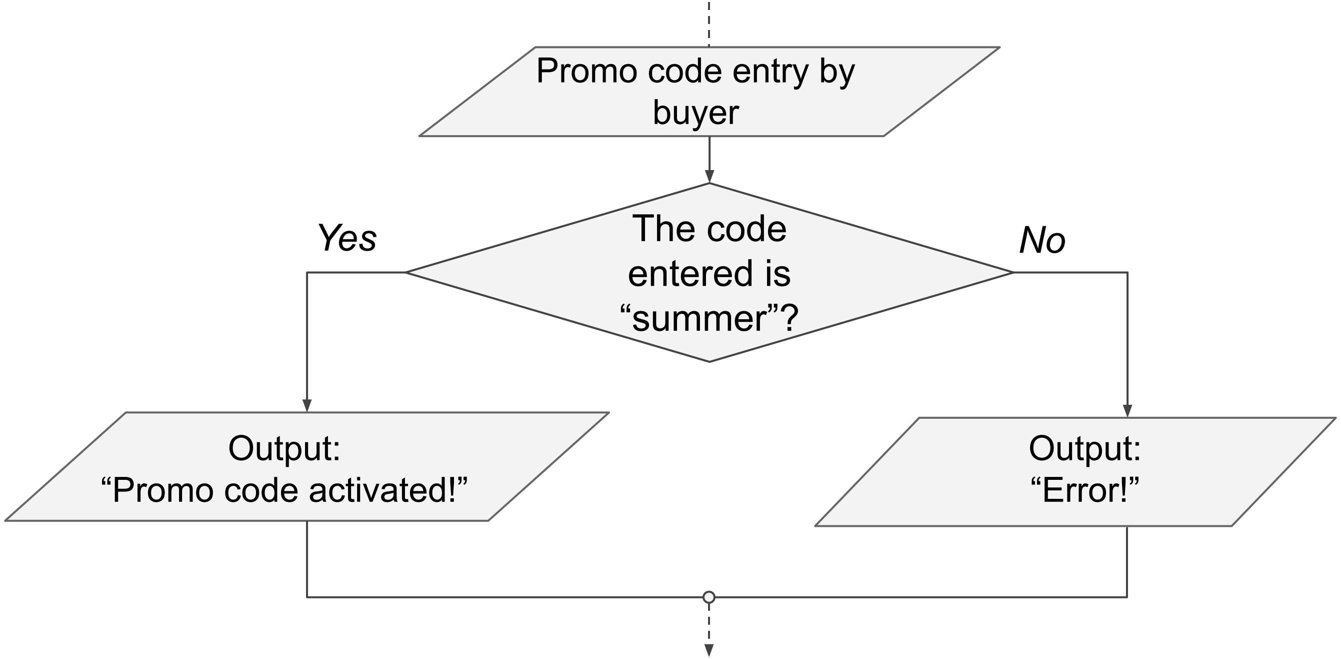
Multiple branch conditional statement:



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

Confirmation of qualifications

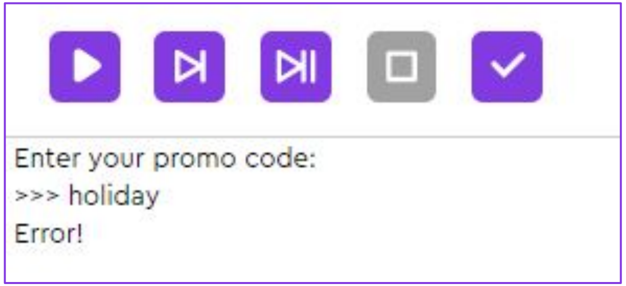
Which one corresponds to the flowchart?


Promo code entry by buyer
The code entered is "summer"?
Yes
No
Output: "Promo code activated!"
Output: "Error!"

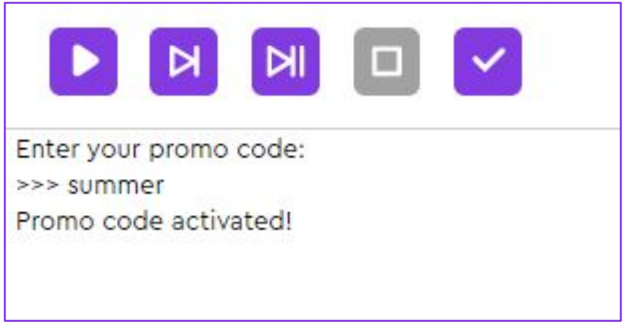
Confirmation of qualifications

Sample code:

```python
promo = input('Enter your promo code:')
if promo == 'summer':
    print('Promo code activated!')
else:
    print('Error!')
```



```
Enter your promo code:
>>> holiday
Error!
```



```
Enter your promo code:
>>> summer
Promo code activated!
```

Confirmation of qualifications

Qualifications confirmed!

Great, you are ready to brainstorm and complete your work task!

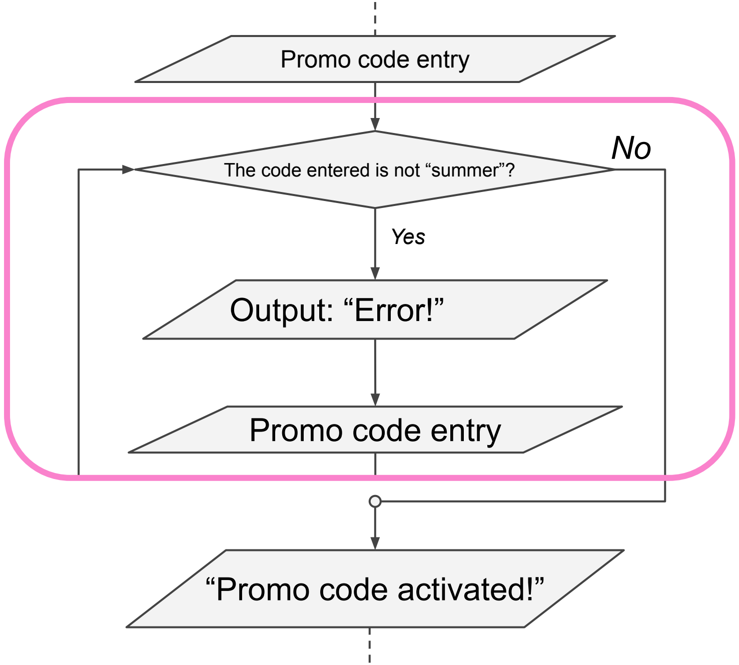
Brainstorm:

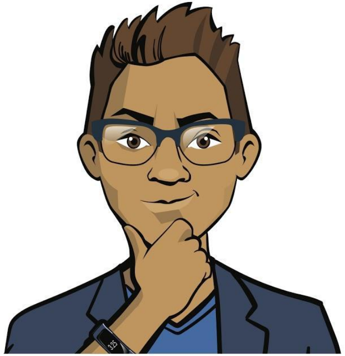
Loops

How do we program action repetition?

We know how to program a condition — a statement that can be true or false.

We will now learn about a construct that repeats as long the corresponding condition remains true.



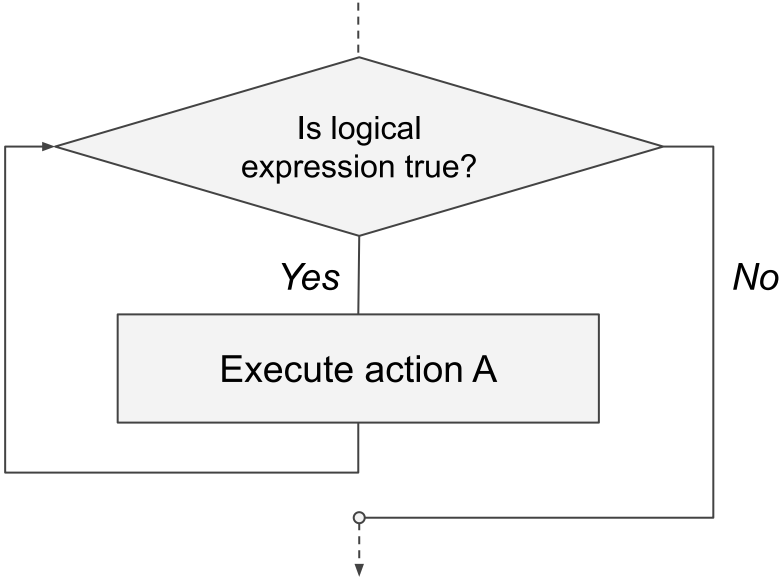


Brainstorm

Loop

– a command that executes actions given as long as a certain logical expression (condition) remains true.

Example:
The loop performs action A as long as the logical expression is true.



Brainstorm

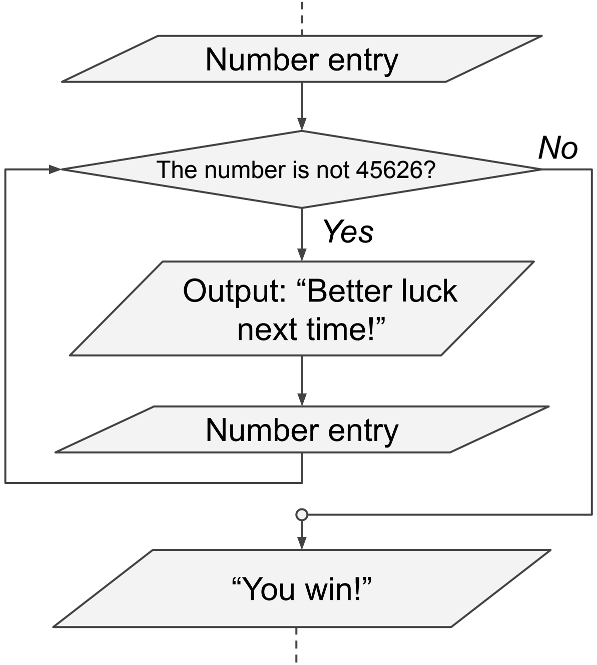
Let’s go over a task

Task 1a. Make up an algorithm to check if a card number entered is the winning number. If the number is 45626, then print “You win!” Otherwise, print “Better luck next time!” and ask for re-entry.

Brainstorm

Let’s go over a task

Task 1a. Make up an algorithm to check if a card number entered is the winning number. If the number is 45626, then print “You win!” Otherwise, print “Better luck next time!” and ask for re-entry.

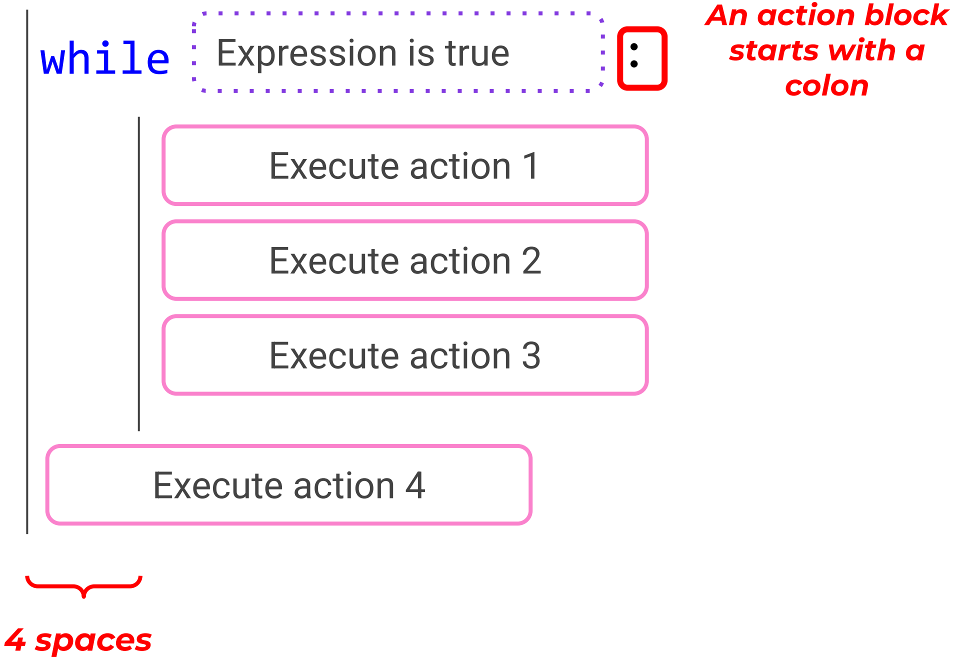


Brainstorm

Loop

The loop can be programmed using the while operator:

`while` (*similar to “as long as”*).



```
while Expression is true :
    Execute action 1
    Execute action 2
    Execute action 3
Execute action 4
```

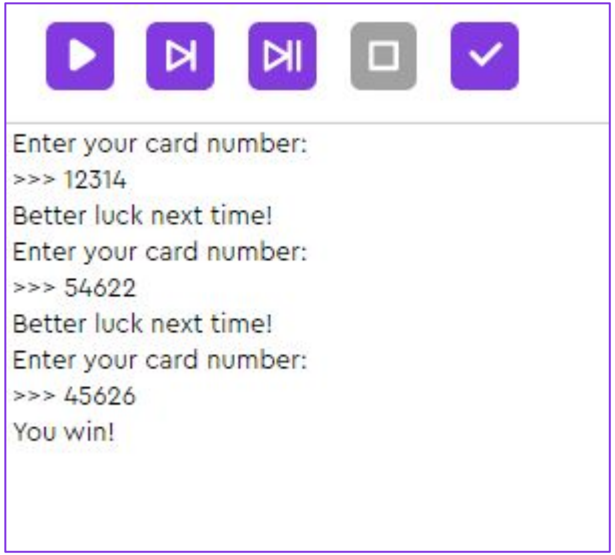
An action block starts with a colon

4 spaces

Brainstorm

Let’s go over a task

Task 1b. Make up a program to check if a card number entered is the winning number. If the number is 45626, then print “You win!” Otherwise, print “Better luck next time!” and ask for re-entry.



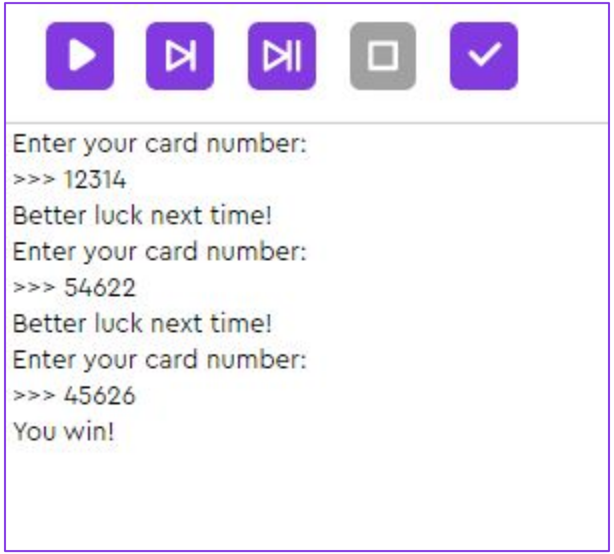
```
Enter your card number:
>>> 12314
Better luck next time!
Enter your card number:
>>> 54622
Better luck next time!
Enter your card number:
>>> 45626
You win!
```

Brainstorm

Let’s go over a task

Task 1b. Make up a program to check if a card number entered is the winning number. If the number is 45626, then print “You win!” Otherwise, print “Better luck next time!” and ask for re-entry.

```
card_number = int(input('Enter your card number:'))
while card_number != 45626:
   print('Better luck next time!')
   card_number = int(input('Enter your card number:'))
print('You win!')
```

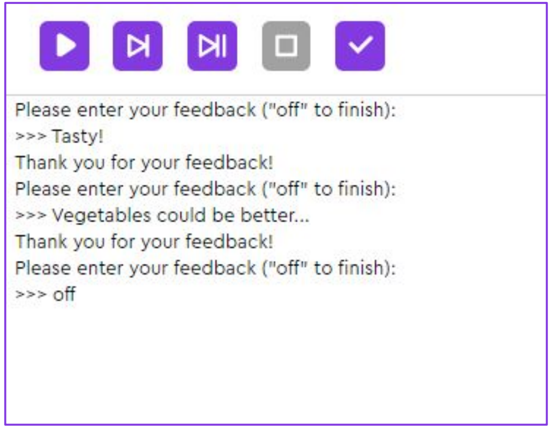


```
Enter your card number:
>>> 12314
Better luck next time!
Enter your card number:
>>> 54622
Better luck next time!
Enter your card number:
>>> 45626
You win!
```

Brainstorm

Let’s go over a task

Task 2. Make up a program asking customers for feedback. Once run, the program must ask for input until the user enters “off”. For every piece of feedback, the program prints out, “Thank you for your feedback!”



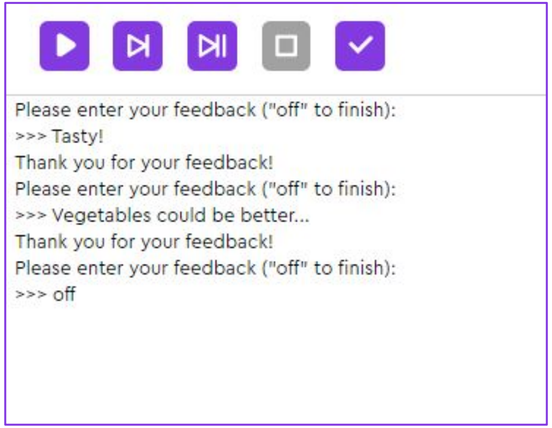
```
Please enter your feedback ("off" to finish):
>>> Tasty!
Thank you for your feedback!
Please enter your feedback ("off" to finish):
>>> Vegetables could be better...
Thank you for your feedback!
Please enter your feedback ("off" to finish):
>>> off
```

Brainstorm

Let’s go over a task

Task 2. Make up a program asking customers for feedback. Once run, the program must ask for input until the user enters “off”. For every piece of feedback, the program prints out, “Thank you for your feedback!”

```
feedback = input('Please enter your feedback ("off" to finish):')
while feedback != 'off':
   print('Thank you for your feedback!')
   feedback = input('Please enter your feedback ("off" to finish):')
```



```
Please enter your feedback ("off" to finish):
>>> Tasty!
Thank you for your feedback!
Please enter your feedback ("off" to finish):
>>> Vegetables could be better...
Thank you for your feedback!
Please enter your feedback ("off" to finish):
>>> off
```

Sample solution. Can we optimize it?

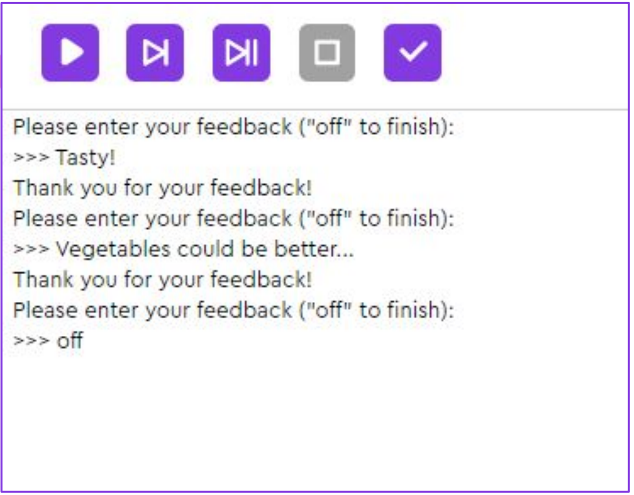
Brainstorm

Let’s go over a task

Task 2. Make up a program asking customers for feedback. Once run, the program must ask for input until the user enters “off”. For every piece of feedback, the program prints out, “Thank you for your feedback!”

```
while input('Please enter your feedback ("off" to
finish):') != 'off':
    print('Thank you for your feedback!')
```

Asking for and returning feedback

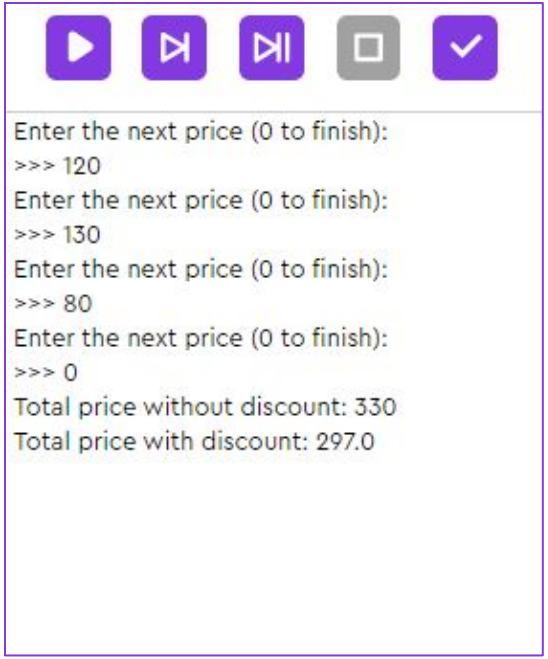


```
Please enter your feedback ("off" to finish):
>>> Tasty!
Thank you for your feedback!
Please enter your feedback ("off" to finish):
>>> Vegetables could be better...
Thank you for your feedback!
Please enter your feedback ("off" to finish):
>>> off
```

Brainstorm

Let's go over a task

Task 3. Make up a program to output the total price of all the purchases with a 10% discount. The program asks the user to enter purchases until the user enters "0". After that, it outputs the total price of all the purchases with a 10% discount.



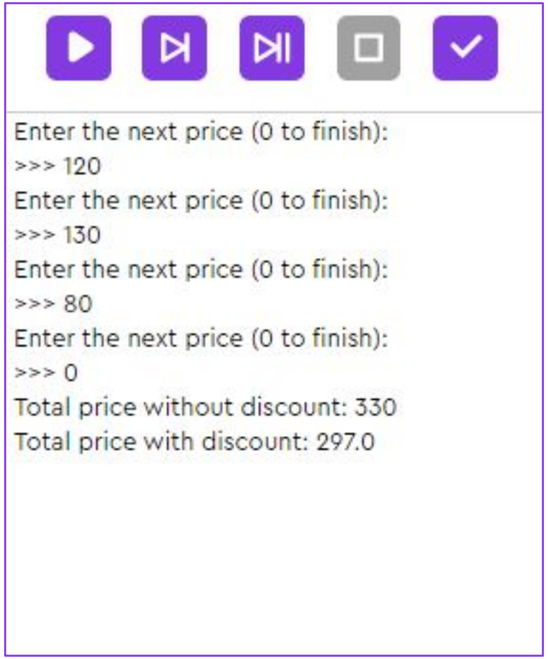
```
Enter the next price (0 to finish):
>>> 120
Enter the next price (0 to finish):
>>> 130
Enter the next price (0 to finish):
>>> 80
Enter the next price (0 to finish):
>>> 0
Total price without discount: 330
Total price with discount: 297.0
```

Brainstorm

Let’s go over a task

Task 3. Make up a program to output the total price of all the purchases with a 10% discount. The program asks the user to enter purchases until the user enters “0”. After that, it outputs the total price of all the purchases with a 10% discount.

```python
price = int(input('Enter the next price (0 to finish):'))
total_price = 0
while price != 0:
    total_price += price
    price = int(input('Enter the next price (0 to finish):'))
print('Total price without discount:', total_price)
total_price = total_price * 0.9
print('Total price with discount:', total_price)
```

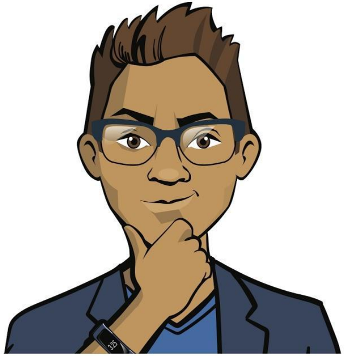


```
Enter the next price (0 to finish):
>>> 120
Enter the next price (0 to finish):
>>> 130
Enter the next price (0 to finish):
>>> 80
Enter the next price (0 to finish):
>>> 0
Total price without discount: 330
Total price with discount: 297.0
```

Brainstorm

Before we continue:

1. What will the program print if we enter 0 right away?
2. What price, without the discount, will the program print if we input, successively, “100”, “120”, “215”, and “0”?
3. What price, with the discount, will the program print if we input, successively, “50”, “100”, “50”, and “0”?
4. What will happen if the user types “250”?



Brainstorm

Platform:

“Longevity”

Do the task on the platform

“Longevity: ordering”

mars.alg.academy

Longevity:
Ordering

Break

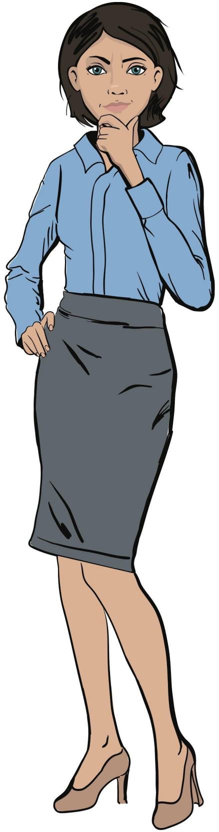
“Brainstorm”:

Loop with counter

The customer’s wish

Longevity’s CEO has asked us to program a counter for entry attempts. The site will ignore abnormal numbers of entries.

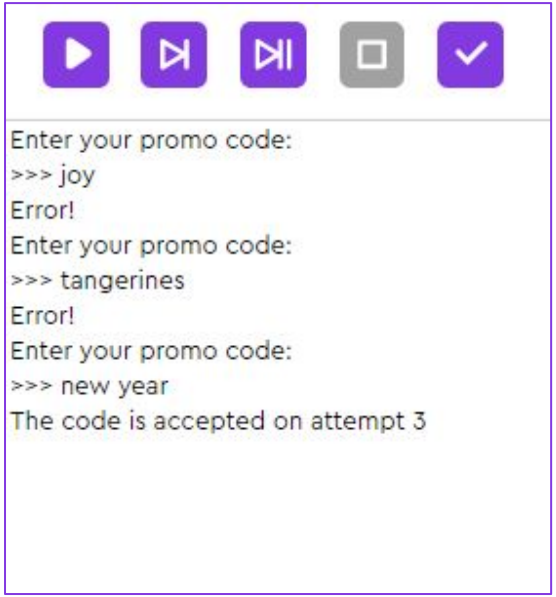
Otherwise, fraudsters will be able to hack the promo-code base!



Brainstorm

Let’s go over a task

Task 1. Make up a program that asks for a promo code and counts the number of entry attempts. Upon entry of the correct code, “new year”, the program outputs: “The code is accepted on attempt ...” and finishes execution.



```
Enter your promo code:
>>> joy
Error!
Enter your promo code:
>>> tangerines
Error!
Enter your promo code:
>>> new year
The code is accepted on attempt 3
```

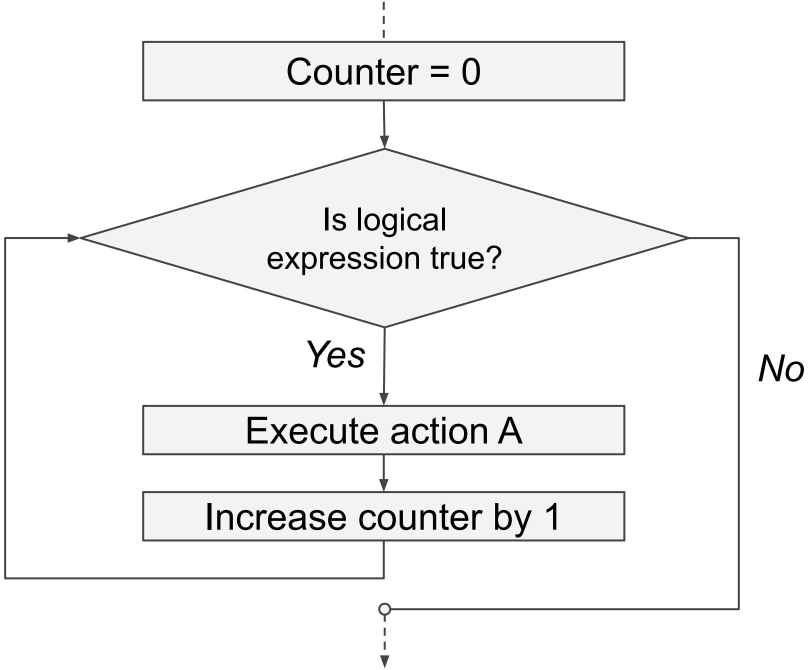
How do we count the number of attempts?

Brainstorm

Counter

– a variable storing the number of steps of a certain loop.

Example 1:
Counter storing all the loop steps



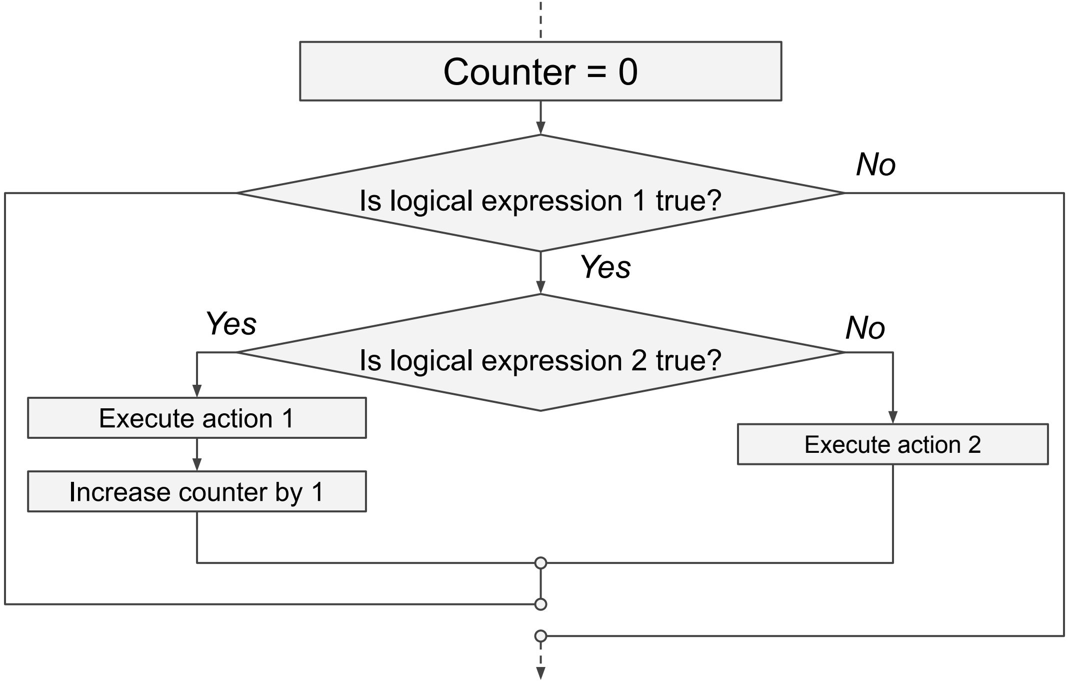
Brainstorm

Counter

– a variable storing the number of steps of a certain loop.

Example 2:

Counter storing all the loop steps where the condition was true



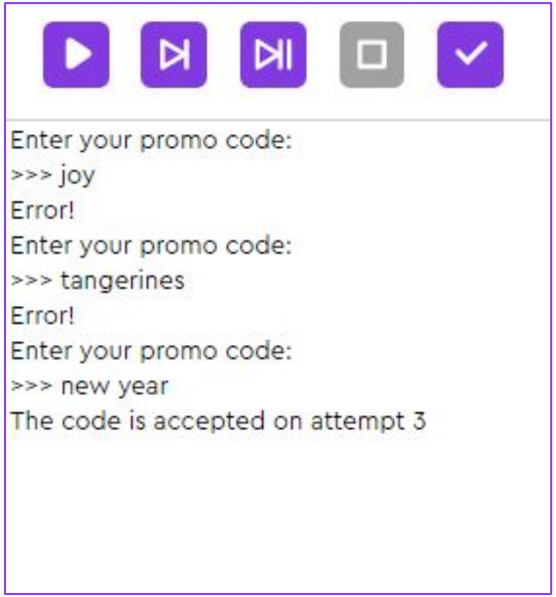
Brainstorm

Let’s go over a task

Task 1. Make up a program that asks for a promo code and counts the number of entry attempts. Upon entry of the correct code, “new year”, the program outputs: “The code is accepted on attempt …” and finishes execution.

Let’s use the first type of a counter:

```
promo = input('Enter your promo code:')
attempts = 1
while promo != 'new year':
    attempts += 1
    print('Error!')
    promo = input('Enter your promo code:')
print('The code is accepted on attempt', attempts)
```



```
Enter your promo code:
>>> joy
Error!
Enter your promo code:
>>> tangerines
Error!
Enter your promo code:
>>> new year
The code is accepted on attempt 3
```

Brainstorm

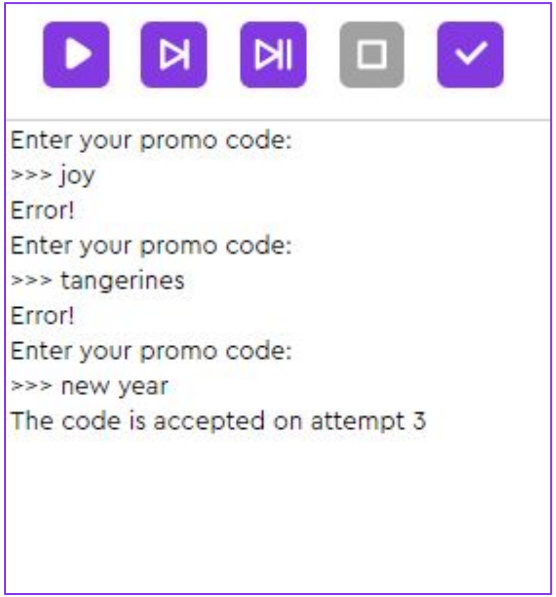
Let’s go over a task

Task 1. Make up a program that asks for a promo code and counts the number of entry attempts. Upon entry of the correct code, “new year”, the program outputs: “The code is accepted on attempt …” and finishes execution.

Let’s use the first type of a counter:

```
promo = input('Enter your promo code:')
attempts = 1
while promo != 'new year':
    attempts += 1
    print('Error!')
    promo = input('Enter your promo code:')
print('The code is accepted on attempt', attempts)
```

Loop with counter

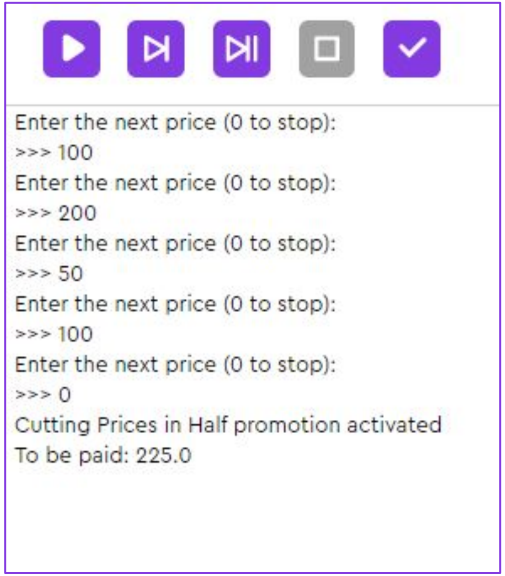


```
Enter your promo code:
>>> joy
Error!
Enter your promo code:
>>> tangerines
Error!
Enter your promo code:
>>> new year
The code is accepted on attempt 3
```

Brainstorm

Let’s go over a task

Task 2. Make up a program that asks for prices of goods until the user enters “0” and then calculates the total sum of the purchases. If the number of goods is even, then the “Cutting Prices in Half” promotion is enabled, and the total sum is divided by two. In the end, the program outputs the total to be paid.



```
Enter the next price (0 to stop):
>>> 100
Enter the next price (0 to stop):
>>> 200
Enter the next price (0 to stop):
>>> 50
Enter the next price (0 to stop):
>>> 100
Enter the next price (0 to stop):
>>> 0
Cutting Prices in Half promotion activated
To be paid: 225.0
```

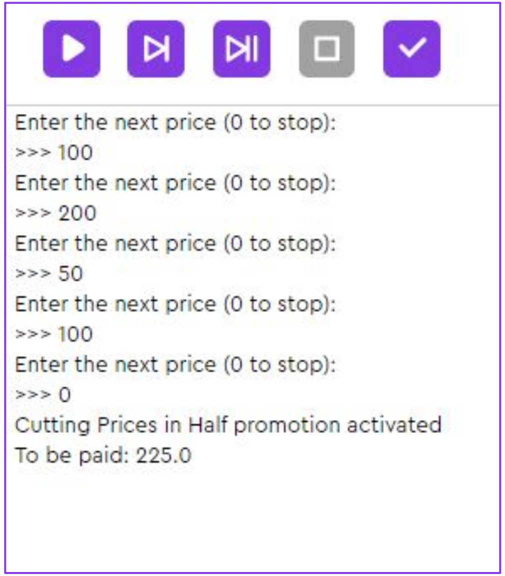
How do we count the number of purchases?

Brainstorm

Let’s go over a task

Task 2. Make up a program that asks for prices of goods until the user enters “0” and then calculates the total sum of the purchases. If the number of goods is even, then the “Cutting Prices in Half” promotion is enabled, and the total sum is divided by two. In the end, the program outputs the total to be paid.

```
price = int(input('Enter the next price (0 to stop):'))
amount = 0
total_price = 0
while price != 0:
    total_price += price
    amount += 1
    price = int(input('Enter the next price (0 to
stop):'))
if amount % 2 == 0:
    print('Cutting Prices in Half promotion activated')
    total_price = total_price/2
print('To be paid:', total_price)
```

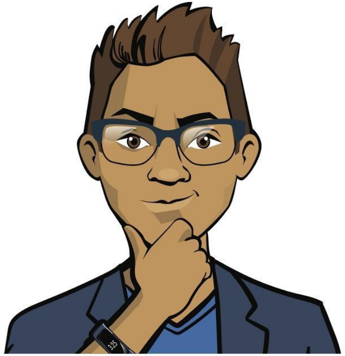


```
Enter the next price (0 to stop):
>>> 100
Enter the next price (0 to stop):
>>> 200
Enter the next price (0 to stop):
>>> 50
Enter the next price (0 to stop):
>>> 100
Enter the next price (0 to stop):
>>> 0
Cutting Prices in Half promotion activated
To be paid: 225.0
```

Brainstorm

Before we continue:

1. What will the previous program print if we input, successively, “50”, “120”, “80”, “0”?
2. How many times will the loop work if the user inputs, successively, “35”, “20”, “0”?
3. Can we create a loop that works infinitely? If so, give an example of such a loop.



Brainstorm

Platform:

“Longevity”

Do the task on the platform

“Longevity: counting actions”

mars.alg.academy

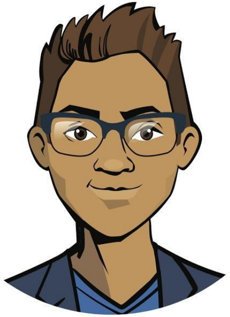
<code>

Longevity:
Counting actions

End of the workday

To complete the workday, pass a technical interview

1. What is a loop? How do we set a loop condition? What reserved word do we use to create a loop?
2. What is a counter? In a loop, what can a counter be used for?



Cole,
Senior Developer

Emily,
Project Manager

Wrapping up the workday

Congratulations on completing the workday!

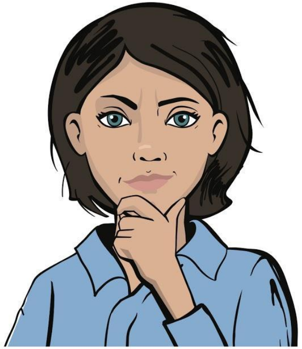
Today you:

1. Learned that a loop is a tool to program actions that are repeated as long as a certain logical expression remains true.
2. Programmed while loops with and without counters.
3. Implemented promo code entry and other discount mechanics within the Longevity Store.

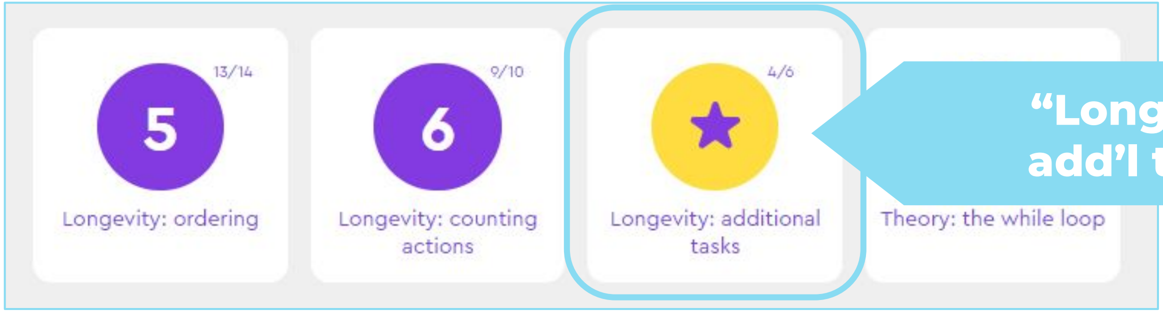
Performance review

Answer the questions together 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?



Additional tasks to improve efficiency



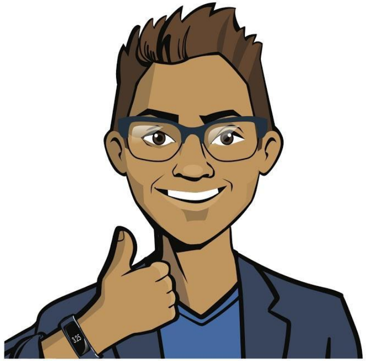
13/14
5
Longevity: ordering

9/10
6
Longevity: counting actions

4/6
★
Longevity: additional tasks

Theory: the while loop

“Longevity: add’l tasks”



Wrapping up the workday