Confirmation of qualifications

To get started on the work tasks, demonstrate your knowledge level.

Prove that you are ready for the brainstorm!

Which data types do you know?

Confirmation of qualifications

Which **data types** do you know?

We know three data types:

- **integer** numbers,
- **decimal** fractions,
- **strings** .

Numbers		*Strings*
144	Integer number (int)	`'John'` (str)
48.3	Decimal fraction (float)	`'256'`(str)
(2*11)	Integer number (int)	`'15.05.2007'` (str)
(4*8.2)	Decimal fraction (float)	`'Data received'` (str)

Confirmation of qualifications

Which operations can you perform on strings ?

Confirmation of qualifications

Which operations can you perform on strings ?

Determine the length of a string

Cut one or more characters out of a string

Search for a word or phrase in a string

Convert all letters in a string to lowercase

Confirmation of qualifications

Which **commands** match those operations?

Determine the length of a string	?
Cut one or more characters out of a string	?
Search for a word or phrase in a string	?
Convert all letters in a string to lowercase	?

Confirmation of qualifications

Which **commands** match those operations?

Confirmation of qualifications

Determine the length of a string	`length = len(string)`

Cut one or more characters out of a string	`symbol = feedback[5]`
	`word = feedback[0:14]`

Search for a word or phrase in a string	`pos = feedback.find(‘word’)`
Convert all letters in a string to lowercase	`string = string.lower()`

What is an **interpreter** ?
What is it meant to do?

Confirmation of qualifications

An interpreter

is a special program that <u>recognizes</u> and <u>executes</u> commands.

The programmer enters a command **in a programming language**

↓

The programmer clicks the “Run the program” button

↓

The command is **translated** into the language of signals (1’s and 0’s)

↓

The command **is executed**

$\left.\begin{matrix} \text{The command is translated into the language of signals (1's and 0's)} \\ \text{The command is executed} \end{matrix}\right\}$ *interpreter*

Confirmation of qualifications

Which functions for switching from one data type to another do you know?

Confirmation of qualifications

int() and str() are functions for switching from one data type to another.

The interpreter can be told explicitly which data type it is dealing with.

```
point1 = input('Rate the hotel’s convenience from 1 to 5:')
point1 = int(point1)
point2 = input('Rate the hotel’s convenience from 1 to 5:')
point2 = int(point2)
total_rating = point1 + point2
```

Recognition: there is an operator between the numbers.

Command: add the two numbers.

Confirmation of qualifications

Is it true that each operator in Python only has a single meaning?

Confirmation of qualifications

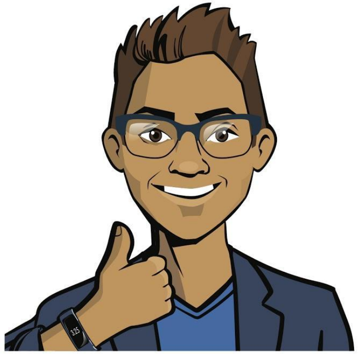
No. Thanks to the smart interpreter, some operators handle different types differently .

Operator	*Meaning for strings*	*Meaning for numbers*
+	Concatenation of strings	The sum of the numbers
*	Multiple repetitions of a string	Multiplication of numbers

'Great' + 'place'	Great place	Concatenation of two strings
3 * 'Cool! '	Cool! Cool! Cool!	Repetition of a string 3 times
'Great' * 'place'	can't multiply sequence by non-int of type 'str'	Interpreter does not understand how many times to repeat the string

Qualifications confirmed!

Great! You are ready to brainstorm and improve your coding skills!



What do we need to know to create a nested construct ?

We need to:

1) understand **which** Constructions may be nested inside one other;
2) find out **how** to **form** a nested Construction correctly.

```
mass = int(input('Weight of the bag'))
```

We know two types of commands:

Commands handling numbers and strings

Upon completion **return <u>no</u>** value

Return a value upon completion

```
print('15.05.2007')
```

```
date = input('Date:')

sum = price1 + price2

length = len(feedback)
```

Which type can be used to create nested Constructions?

Brainstorm

We know two types of commands:

Commands handling numbers and strings

Upon completion **return no** value

```
print('15.05.2007')
```

✅

Return a value upon completion

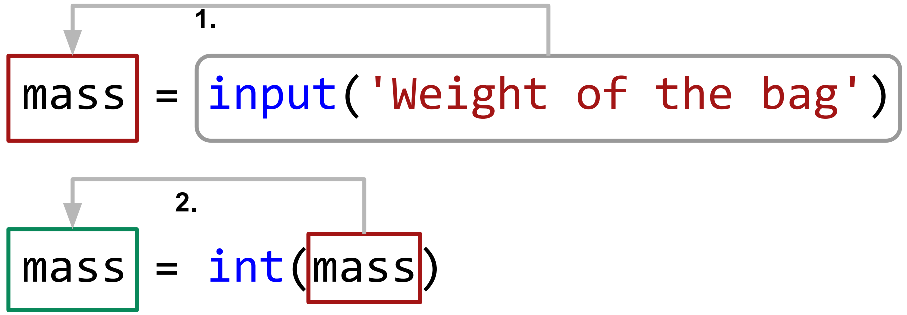
```
date = input('Date:')

sum = price1 + price2

length = len(feedback)
```

Inside another construct, a command which returns a value **should be used.**

Brainstorm

Format of nested constructions

Let’s compare code without a nested Construction to code containing one:



```
mass = input('Weight of the bag')

mass = int(mass)
```

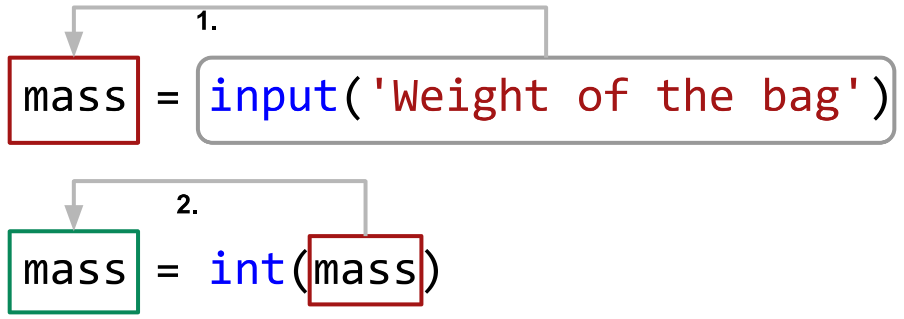
1. The result of the operation is a string.

2. The result is an integer.

Brainstorm

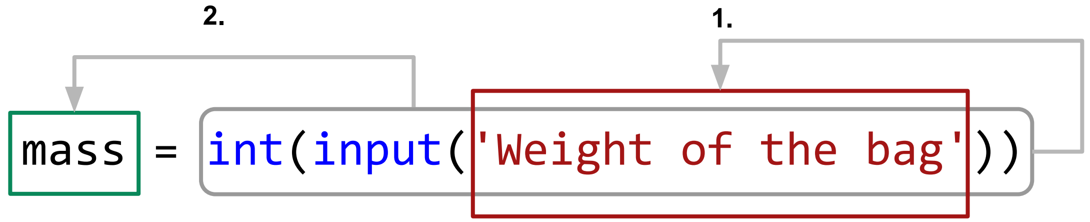
Format of nested Constructions

Let’s compare code without a nested Construction to code containing one:



```
mass = input('Weight of the bag')
mass = int(mass)
```

1. The result of operation is a string.
2. The result is an integer.



```
mass = int(input('Weight of the bag'))
```

The code performs the same actions, but has become more concise.

Brainstorm

Use of nested Constructions

Task.
Write a program that calculates the cost of dinners during a hotel stay. The cost of the dinner and the duration of the stay are entered by the user.

How can we complete this task?

Brainstorm

Use of nested Constructions

Task.
Write a program that calculates the cost of dinners during a hotel stay. The cost of the dinner and the duration of the stay are entered by the user.

```
price = input('Price:')
price = int(price)
days = input('Number of days:')
days = int(days)
total_price = price*days
print(total_price)
```

How can we use nested Constructions to make the solution concise?

Brainstorm

Use of nested Constructions

Task.
Write a program that calculates the cost of dinners during a hotel stay. The cost of the dinner and the duration of the stay are entered by the user.

```
price = input('Price:')
price = int(price)
days = input('Number of days:')
days = int(days)
total_price = price*days
print(total_price)
```

```
price = int(input('Price:'))
days = int(input('Number of days:'))
total_price = price*days
print(total_price)
```

The code is shorter and clearer

Brainstorm

Use of nested Constructions

Task.
Write a program that finds the sum of the lengths of a user's answers to the survey questions:

"What did you like?", "What did you dislike?", "What should be improved?"

How do we complete this task?

Brainstorm

Use of nested Constructions

Task.
Write a program that finds the sum of the lengths of a user's answers to the survey questions:

"What did you like?", "What did you dislike?", "What should be improved?"

```
fb1 = input('What did you like?')
fb2 = input('What did you dislike?')
fb3 = input('What should be improved?')
l1 = len(fb1)
l2 = len(fb2)
l3 = len(fb3)
total_lenght = l1 + l2 + l3
print('Total characters:',
total_lenght)
```

How can we use nested Constructions to make the solution more concise?

Brainstorm

Use of nested Constructions

Task.
Write a program that finds the sum of the lengths of a user's answers to the survey questions:

"What did you like?", "What did you dislike?", "What should be improved?"

```
fb1 = input('What did you like?')
fb2 = input('What did you dislike?')
fb3 = input('What should be improved?')
l1 = len(fb1)
l2 = len(fb2)
l3 = len(fb3)
total_lenght = l1 + l2 + l3
print('Total characters:',
total_lenght)
```

```
l1 = len(input('What did you like?'))
l2 = len(input('What did you dislike?'))
l3 = len(input('What should be
improved?'))
total_lenght = l1 + l2 + l3
print('Total characters:', total_lenght)
```

You can optimize not only a type change, but also the way you handle strings.

Brainstorm

Use of nested Constructions

Task.
Write a program that finds the sum of the lengths of a user's answers to the survey questions:

"What did you like?", "What did you dislike?", "What should be improved?"

```
l1 = len(input('What did you like?'))
l2 = len(input('What did you dislike?'))
l3 = len(input('What should be improved?'))
total_lenght = l1 + l2 + l3
print('Total characters:', total_lenght)
```

Is it possible to make the program code even shorter?

Brainstorm

Use of nested Constructions

Task.
Write a program that finds the sum of the lengths of a user's answers to the survey questions:

"What did you like?", "What did you dislike?", "What should be improved?"

```
l1 = len(input('What did you like?'))
l2 = len(input('What did you dislike?'))
l3 = len(input('What should be improved?'))
total_lenght = l1 + l2 + l3
print('Total characters:', total_lenght)
```

```
print('Total characters:', len(input('What did you like?')) +
     len(input('What did you dislike?')) + len(input('What should be
improved?')))
```

Brainstorm

Use of nested Constructions

Task.
Write a program that finds the sum of the lengths of a user’s answers to the survey questions:

“What did you like?”, “What did you dislike?”, “What should be improved?”

```
l1 = len(input('What did you like?'))
l2 = len(input('What did you dislike?'))
l3 = len(input('What should be improved?'))
total_lenght = l1 + l2 + l3
print('Total characters:', total_lenght)
```

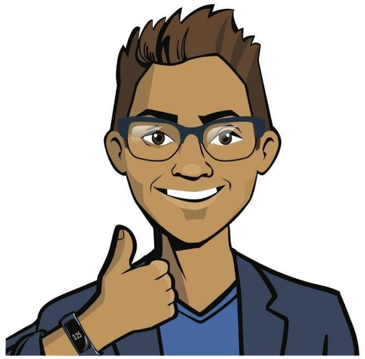
It is possible, but the program has become almost unreadable!

```
print('Total characters:', len(input('What did you like?')) +
     len(input('What did you dislike?')) + len(input('What should be
improved?')))
```

Brainstorm

Conclusions:

1. Nested Constructions allow us to make program code simpler and more concise.
2. It is possible to nest the commands which return a value.
3. It is important to strike a balance between code conciseness and readability.



Brainstorm

Nested Constructions:
Tasks

How do we make easily readable code?

There are a few ways:

- clear and meaningful variable names;
- moderate use of nested Constructions;
- use of comments in the code.

Brainstorm

How do we make easily readable code?

There are a few ways:

- clear and meaningful variable names;
- moderate use of nested Constructions;
- use of comments in the code.

$\downarrow$

Clear names and good organization are not always enough.

In these instances, programmers leave explanatory **#comments** in their code.

Brainstorm

Commenting on the code

A **comment** is a string in a program that is not analyzed by the interpreter.

When an interpreter sees a string in a special format, it does not try to recognize it as a command.

Brainstorm

Commenting on the code

A **comment** is a string in a program that is not analyzed by the interpreter.

When an interpreter sees a string in a special format, it does not try to recognize it as a command.

```
#a short explanatory comment
```

A **short**, single-line comment.

```
'''A large comment explaining
the program structure'''
```

A **long**, multi-line comment.

Brainstorm

Commenting on the code

A **comment** is a string in a program that is not analyzed by the interpreter.

When an interpreter sees a string in a special format, it does not try to recognize it as a command.

Example:

```
price = int(input('Enter the package tour price:'))

#discount - 5% of package tour price

print('Cashback earned:', price*0.05)
```

The interpreter sees # and understands that this line contains no command.

Brainstorm

Commenting on the code

A **comment** is a string in a program that is not analyzed by the interpreter.

When an interpreter sees a string in a special format, it does not try to recognize it as a command.

Example:

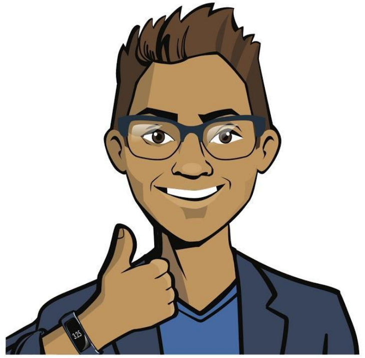
```
'''A program which prints the number of
1-dollar and 10-dollar coins to issue the entered
amount'''
change = int(input('Enter the amount:'))
dol1 = change%10
dol10 = change//10
print(dol1, '- in 1 dollar.')
print(dol10, '- in 10 dollar.')
```

The interpreter also sees the beginning and end of a comment marked with `'''`

Brainstorm

Any program should have a code that is concise and clear to other programmers.

This is why it ’s important to know the rules of the Python language and be able to write easily readable code.



Brainstorm