
Group 3
IT SPECIALIST:
PYTHON
Part 1
A B C D E F
G H I J K L M
O P Q R S T
U V W X Y Z

OUR TEAM

1. Agung Prayogi
2. Dian Maulida
3. Dwi Fitriawati Fajrin
4. Axel Eldrian Hadiwibowo
5. Yesaya Arya Danar Kristuadji

TABLE OF CONTENTS

-PYTHON DATA TYPES-

- Introduction to Python & data types
- Number, string, & boolean
- List
- Slicing
- Set
- Tuple
- Dictionary
- Number, character, & string transformations
- Data type conversion
- Input output in Python
- Replacing string elements
- String checking
- String Formatting
- String literals

PYTHON

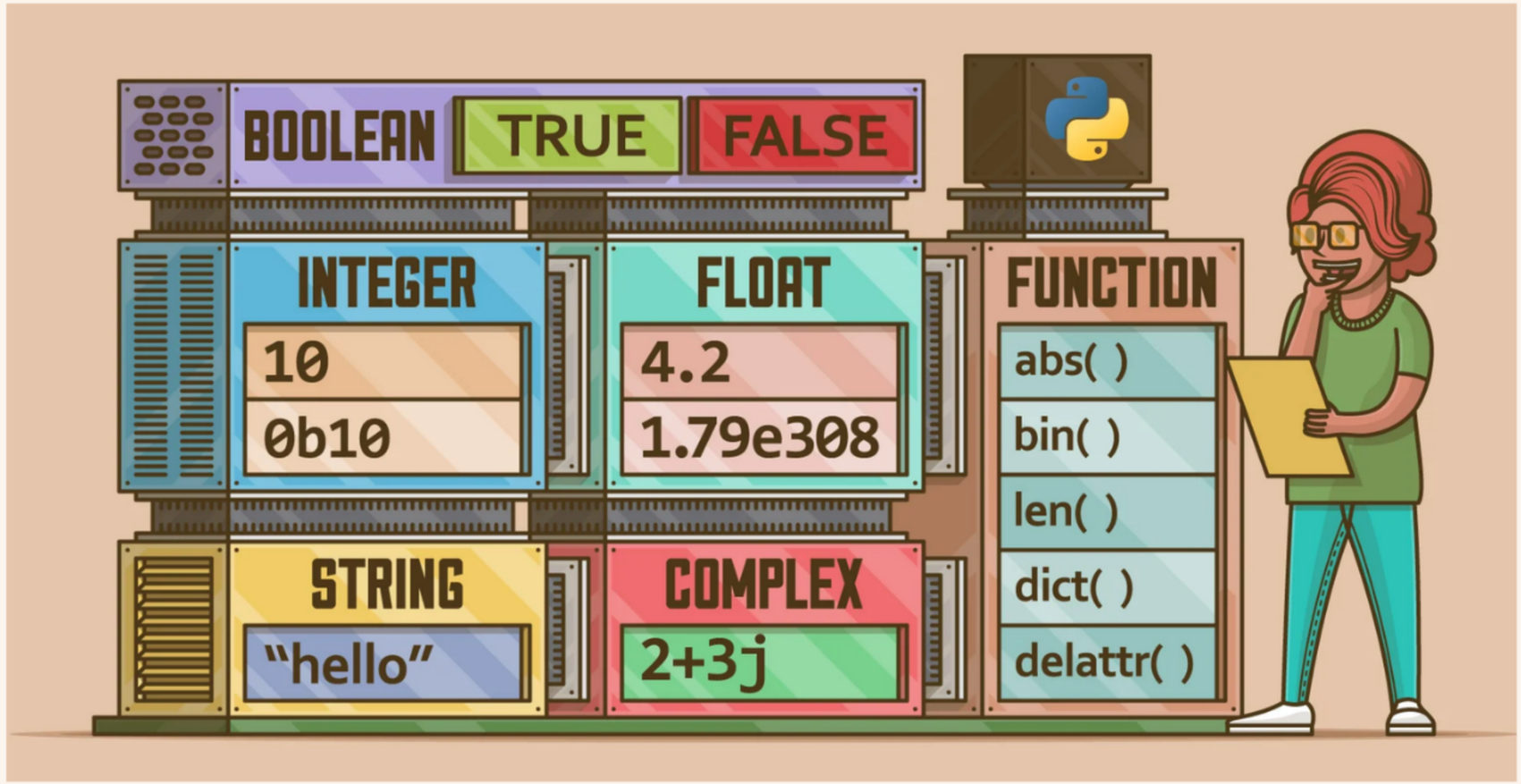
Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation.

It was created by Guido van Rossum, and released in 1991.It is used for: web development (server-side), software development, mathematics, & system scripting.

WHY PYTHON?

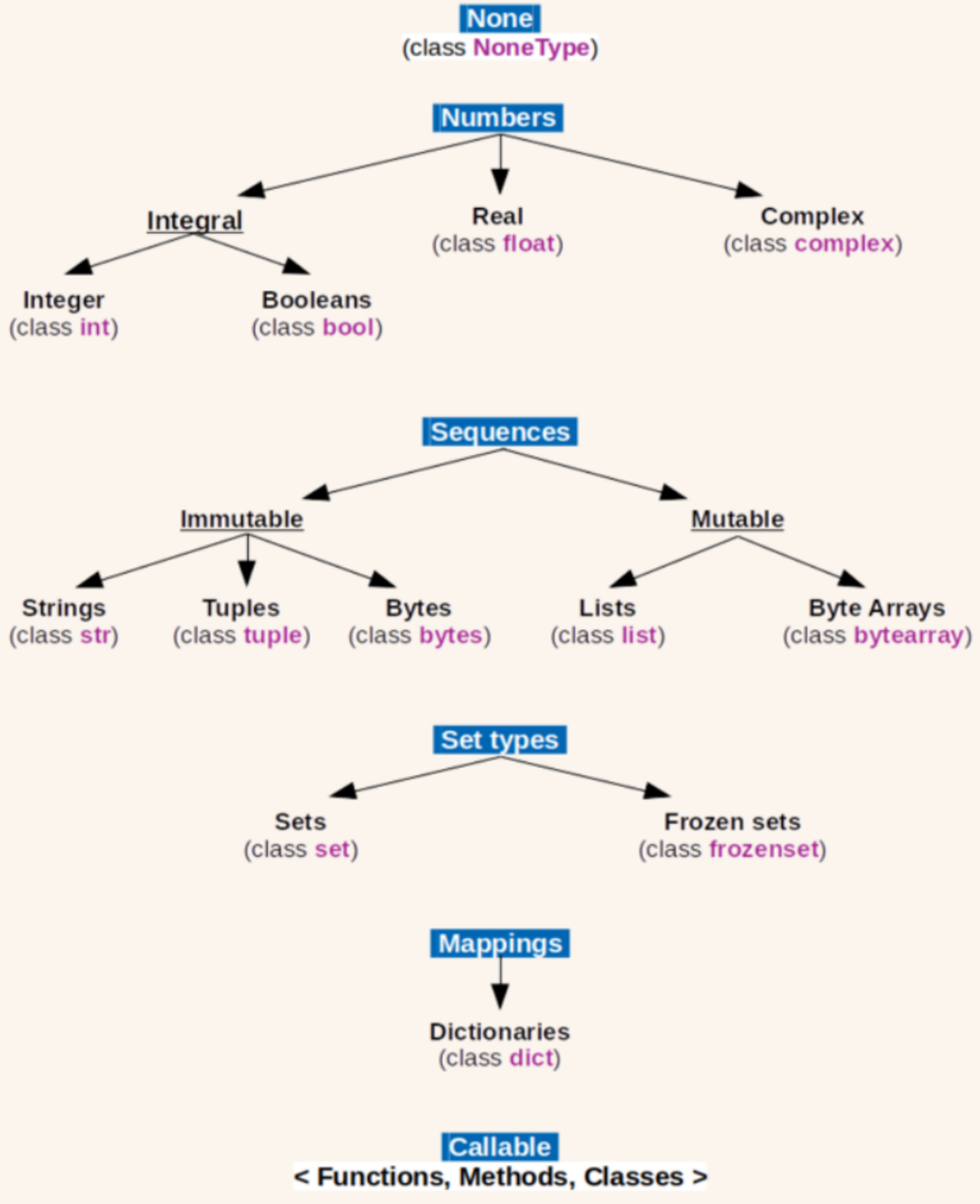
- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-oriented way or a functional way.

DATA TYPES



Python has several basic numeric, string, and Boolean types that are built into Python.

source: realpython.com


None
(class NoneType)
Numbers
Integral
Real
(class float)
Complex
(class complex)
Integer
(class int)
Booleans
(class bool)
Sequences
Immutable
Mutable
Strings
(class str)
Tuples
(class tuple)
Bytes
(class bytes)
Lists
(class list)
Byte Arrays
(class bytearray)
Set types
Sets
(class set)
Frozen sets
(class frozenset)
Mappings
Dictionaries
(class dict)
Callable
< Functions, Methods, Classes >
Modules

NUMBERS

INTEGER

- The integer data type in Python can be of any length but is limited by the amount of available memory
- In Python the integer type is indicated by the result int

FLOAT

- Floats are fractional numbers or numbers written using decimals.
- In python, the delimiter used for decimal markers is the dot (.)
- The float data type in python has an accuracy of up to 17 digits after dots and indicated by the result float

COMPLEX

- Complex numbers are numbers that have real and imaginary parts
- Imaginary part in python using j character
- We cannot use the print() function for complex numbers
- In Python, complex data types are represented by complex results

EXAMPLE

INTEGER

```
In [4]: a = 10

In [5]: type(a)
Out[5]: int
```

FLOAT

```
In [6]: b = 1.7

In [7]: type(b)
Out[7]: float
```

COMPLEX

```
In [17]: c = 1+2j

In [18]: type(c)
Out[18]: complex
```

STRING

A string is one or a series of characters placed between quotes, either single quotes (') or double quotes (")

String is a data type whose members are ordered and have an index.

```
In [23]: a = "Hello World"
         print(a)

         Hello World

In [24]: b = """Bootcamp Data Science
         Kampus Merdeka X MyEduSolve"""
         print(b)

         Bootcamp Data Science
         Kampus Merdeka X MyEduSolve

In [26]: c = "Pathway Data Science.\nKelas B."
         print(c)

         Pathway Data Science.
         Kelas B.
```

BOOLEAN

The boolean data type can only be filled with one of 2 values, namely TRUE or FALSE.

The boolean data type is mostly used to decide what to do when a condition occurs.

```
In [42]: 2 > 5
Out[42]: False

In [16]: 2 == 5
Out[16]: False

In [17]: 2 <= 2
Out[17]: True

In [18]: 2 != 3
Out[18]: True
```

4 BUILT-IN DATA TYPES IN PYTHON

4 built-in data types in Python used to store collections of data, there are List, Tuple, Set, and Dictionary, all with different qualities and usage.

LIST

TUPLE

SET

DICTONARY


LIST
A B C D E F
G H I J K L M
O P Q R S T
U V W X Y Z

LIST

- Lists are used to store multiple items in a single variable.
- To declare a list, brackets [ ] are used and each member is separated by a comma.
- List items are indexed, the index start form 0
- List items are ordered, changeable, and allow duplicate values
- Lists can contain members of the same or different types



The data in the list starts at index 0. Index -1 is the first order read from the back.

```
In [1]: a = [5, 6.7, 'delapan']

In [2]: a[0]
Out[2]: 5

In [3]: type(a)
Out[3]: list

In [4]: type(a[0])
Out[4]: int
```

```
In [21]: a[-1]
Out[21]: 'delapan'

In [10]: a[-2]
Out[10]: 6.7

In [11]: a[0:2]
Out[11]: [5, 6.7]
```

```
In [28]: b = [6,7,8,9,10,11,12,13,14,15]

In [29]: b[0:8]
Out[29]: [6, 7, 8, 9, 10, 11, 12, 13]

In [30]: b[0:8:2]
Out[30]: [6, 8, 10, 12]

In [31]: b[0:8:3]
Out[31]: [6, 9, 12]
```

PYTHON LIST METHODS

DEL

Definition

The del keyword is used to delete an object or an element in an object

```
number = [
    3,23,1,4,2,6
]

del number[1]
print(number)
```

✓ 0.4s

```
[3, 1, 4, 2, 6]
```

Example

Element 23 is removed because it deleted. index of value 23 is 1

LEN()

Definition

the len() method will return the sum of the total items in an object. on a string, this function will return the number of characters

```
colour = [
    'merah','kuning',
    'hijau','biru',
    'coklat','ungu',
    'hitam','putih',
    'cokelat','perak',
    ]
```

✓ 0.3s

```
len(colour)
```

✓ 0.4s

```
10
```

Example

The len() method return number 10 because total items in colour variable is 10

APPEND()

Definition

The append() method is used to add an item at the end of an object

```
colour = [
    'merah','kuning',
    'hijau','biru',
    'coklat','ungu',
    'hitam','putih',
    'cokelat','perak',
    ]
```

✓ 0.3s

Example

The append() method will appends element "emas" to the end of the list.

```
colour.append('emas')
print(colour)
```

✓ 0.1s

```
['merah', 'kuning', 'hijau', 'biru', 'coklat', 'ungu', 'hitam', 'putih', 'cokelat', 'perak', 'emas']
```

EXTEND()

Definition

The extend() method adds all elements of the iterable to the end of the list.

```
number = [1,2,3]
number.extend([4,5])
print(number)
```

✓ 0.8s

```
[1, 2, 3, 4, 5]
```

Example

The extend() method adds all element in extend method ([4,5]) to number variable

INSERT()

Definition

The insert() method inserts an element into the object according to the index specified in the parameter

```
number = ['satu', 'tiga']
number.insert(1, 'dua')
print(number)
```

✓ 0.3s

```
['satu', 'dua', 'tiga']
```

Example

This method add 'dua' at second order list (in index 1 means 2 because indexing start from 0)

REMOVE()

Definition

The remove() method removes the first element whose content is equal to the parameter value

```
number = ['satu','dua', 'tiga']
number.remove('tiga')
print(number)
```

✓ 0.2s

```
['satu', 'dua']
```

Example

Element 'tiga' is removed from number value

POP()

Definition

The pop() method removes the item according to the index in the parameter and returns the deleted value

```
number = ['satu','dua', 'tiga']
number.pop(0)
print(number)
```

✓ 0.3s

```
['dua', 'tiga']
```

Example

Pop remove value that have index 0, thats why element 'satu' is removed.

INDEX()

Definition

The index() method returns the index according to the value written in the parameter

```
number = ['satu','dua', 'tiga']
number.index('tiga')
```

✓ 0.2s

```
2
```

Example

The method return 2 because tiga in index 2

COUNT()

Definition

The count() method returns the number of elements in the object that have the same value in the parameter .

```
colour=[
    'black','yellow',
    'black','black',
    'red','purple'
]

colour.count('black')
```

✓ 0.3s

```
3
```

Example

the method return 3 because element 'black' had 3 items in colour variable

SORT()

Definition

The sort() method sorts the list items in ascending order by default

```
number = [
    3,23,1,4,2,6
]

number.sort()
print(number)
```

✓ 0.3s

```
[1, 2, 3, 4, 6, 23]
```

Example

As you can see, list are sorted in ascending order

REVERSE()

Definition

The reverse() method will return list from tail index to head

```
number = [
    3,23,1,4,2,6
]

number.reverse()
print(number)
```

✓ 0.3s

```
[6, 2, 4, 1, 23, 3]
```

Example

reverse is not sort method, reverse will return object from tail index to head

SLICING STRING

The string is mutable, in other words, the mutable object can be changed after it is created, you can return a range of characters by using the slice syntax.

Syntax = slice[start : stop : step]

STRING SLICING

```
In [2]: a = "kampus Merdeka"

In [3]: print(a[1])
a

In [4]: print(a[-3:-1])
ek

In [5]: print(a[-4:-1:1])
dek
```

display the first index character

display the character starts from the third last to the last index character

display the character starts from the fourth last to the last index character and 1 step

MODIFY STRING

It can be modified just for the word, it can't be specified in just one character.

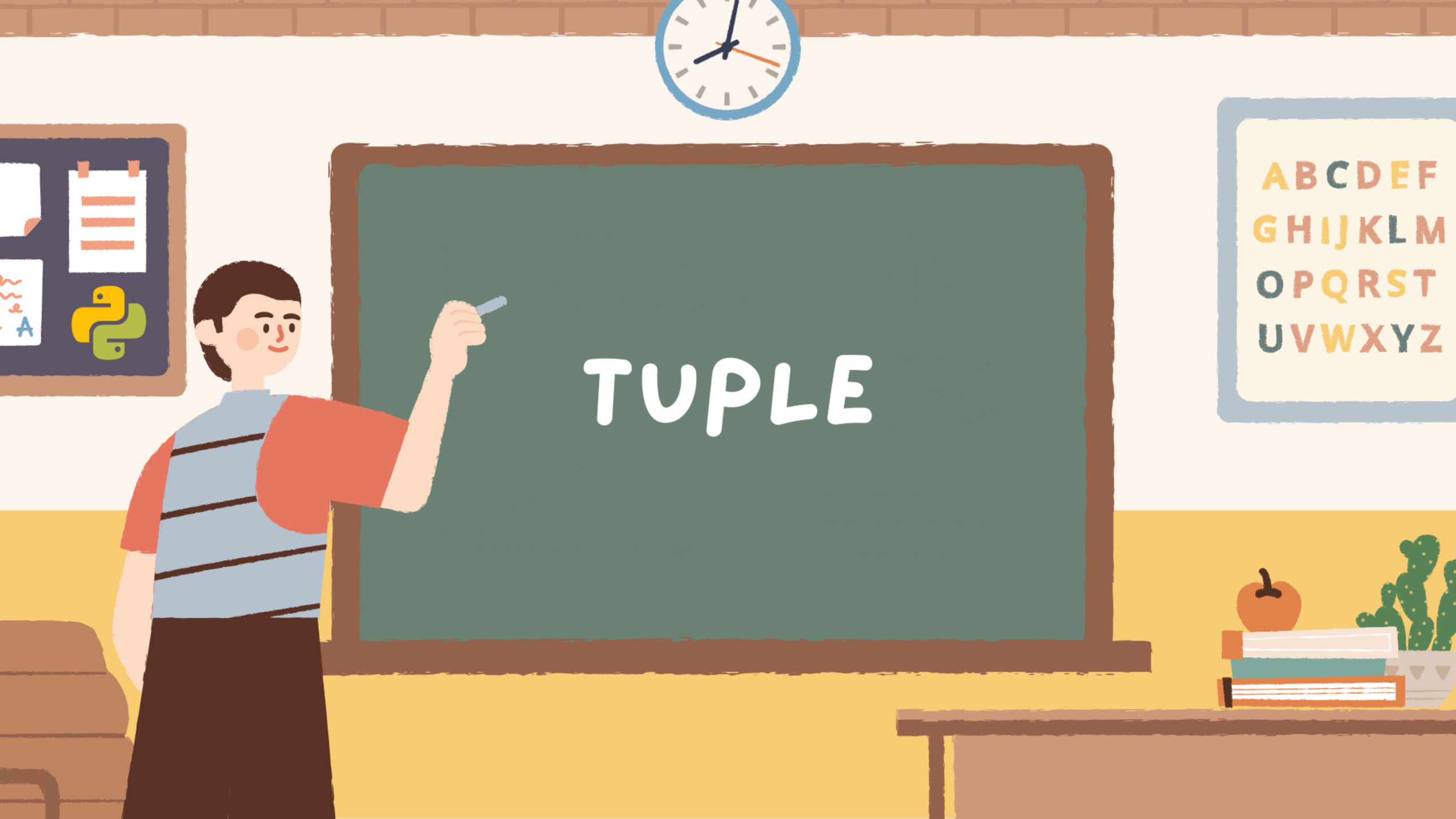
MODIFY STRING

```
a = "kampus Merdeka"
```

```
a[0] = "c"
```

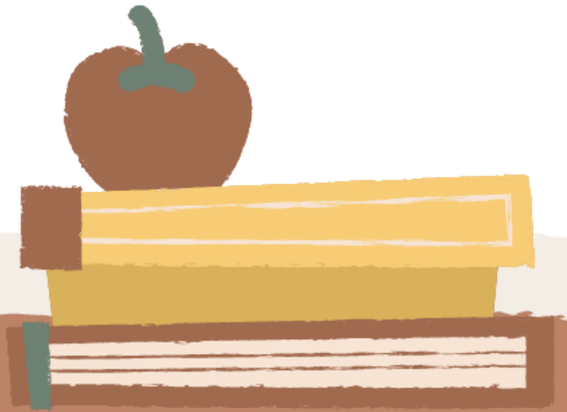
```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [5], in <cell line: 1>()
----> 1 a[0] = "c"

TypeError: 'str' object does not support item assignment
```

It would appear error because you can't change the "k" character to "c"


TUPLE
A B C D E F
G H I J K L M
O P Q R S T
U V W X Y Z

TUPLE

- A tuple is a collection which is ordered, allow duplicate values and unchangeable.
- Tuples are used to store multiple items in a single variable
- Tuples are written with round brackets ( )
- Tuple items can be of any data type

EXAMPLE

```
a = ('one', 'two', 'three', 'four', 'five', 6, 7, 8, 9, 10)
```

this variable contain string and integer

```
print(a)
```

```
('one', 'two', 'three', 'four', 'five', 6, 7, 8, 9, 10)
```

TUPLE SLICING

```
a = ('one', 'two', 'three', 'four', 'five', 6, 7, 8, 9, 10)
```

```
print(a)
```

```
('one', 'two', 'three', 'four', 'five', 6, 7, 8, 9, 10)
```

- Slicing

```
print(a[3])
```

```
four
```

display element in the third index

```
print(a[-3:-1])
```

```
(8, 9)
```

display element start from third from the last, to the last index

```
print(a[0:8:2])
```

```
('one', 'three', 'five', 7)
```

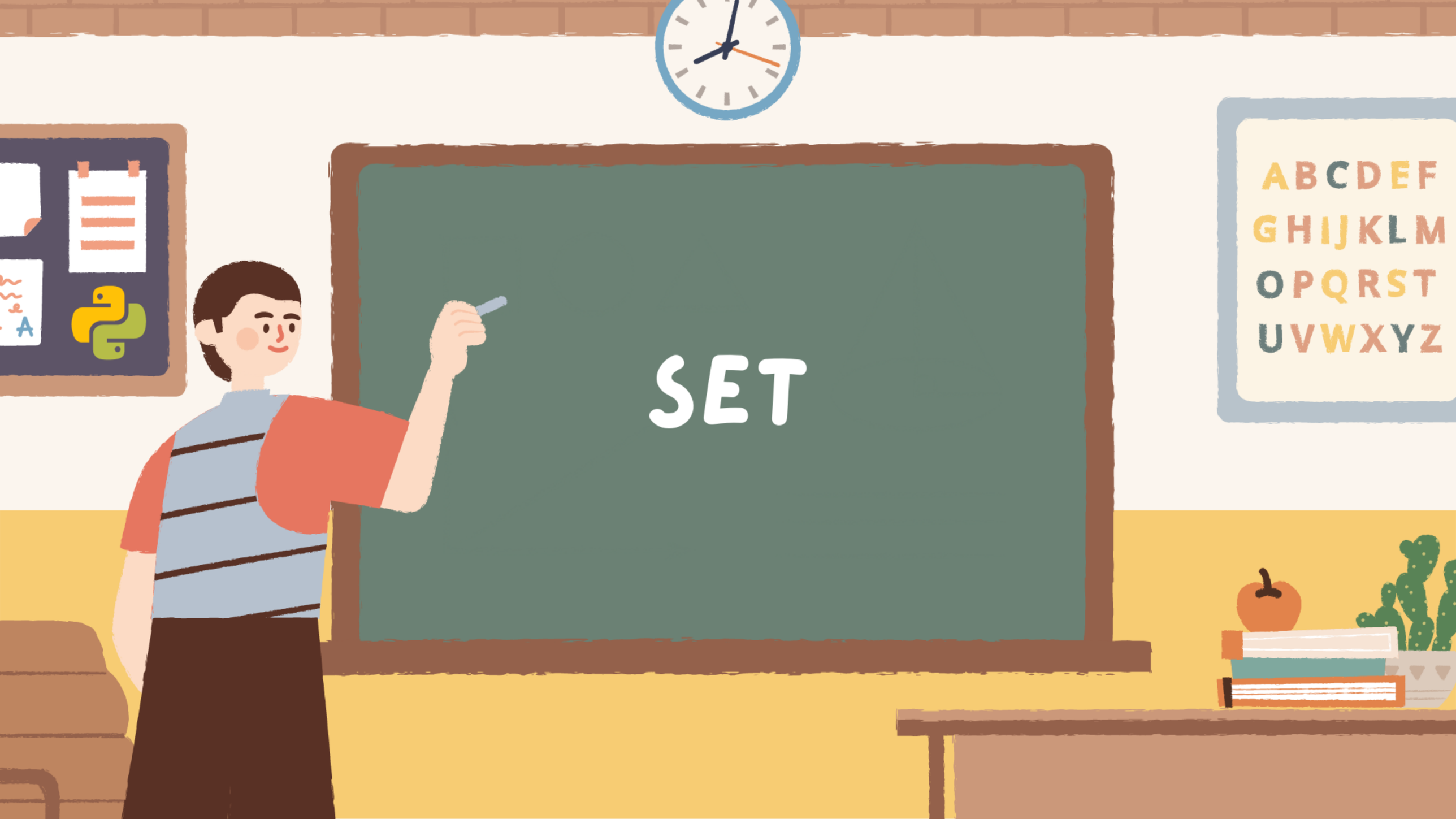
display element start 0 index to the eightht and 2 step

MODIFY TUPLE

```
print(q[0]=17)
```

```
  Input In [8]
    print(q[0]=17)
          ^
SyntaxError: expression cannot contain assignment, perhaps you meant "=="?
```

Tuple is unchangeable, so you can't change the element in tuple


SET
A B C D E F
G H I J K L M
O P Q R S T
U V W X Y Z

SET

- A set is a collection which is unordered, unchangeable, and unindexed.
- Sets are used to store multiple items in a single variable and must be unique
- Sets are written with curly brackets { }
- Set items can be of any data type




SET EXAMPLE

```
a = {'one', 'two', 'three', 'four', 'five',6, 7, 8, 9, 10}

print(a)
```

```
{'five', 'two', 6, 'three', 7, 8, 9, 10, 'one', 'four'}
```

This variable contain string and integer and the output is unorder

UNION AND INTERSECTION

```
a = {'one', 'two', 'three', 'four', 'five',6, 7, 8, 9, 10, 10, 9, 8, 7}

print(a)
```

```
{'five', 'two', 6, 'three', 7, 8, 9, 10, 'one', 'four'}
```

```
b = {'six', 'seven', 'eight', 8, 9, 10}

print(b)
```

```
{'seven', 8, 9, 10, 'six', 'eight'}
```

- Union

Returns a set that contains all items from the original set, and all items from the specified set(s)

- Intersection

Returned set contains only items that exist in both sets, or in all sets if the comparison is done with more than two sets

UNION AND INTERSECTION

```
print(a.union(b))
```

```
{'five', 'two', 6, 'three', 7, 8, 9, 10, 'seven', 'one', 'four', 'six', 'eight'}
```

The output of union by variable a and b

*Intersection

the returned set contains only items that exist in both sets

or in all sets if the comparison is done with more than two sets

```
print(a.intersection(b))
```

```
{8, 9, 10}
```

The output of intersection by variable a and b


DICTIONARY
A B C D E F
G H I J K L M
O P Q R S T
U V W X Y Z

DICTIONARY

- A dictionary is a collection that is unordered, changeable, and do not allow duplicates
- Dictionary items are presented in key:value pairs, and can be referred to by using the key name
- Dictionaries are written with curly brackets { }

DICTIONARY EXAMPLE

```
data = {"nama": "Ana", "kelas": "data science B", "umur": 9, "nilai" : {"ujian 1" : 10}}
```

```
print(data)
```

```
{'nama': 'Ana', 'kelas': 'data science B', 'umur': 9, 'nilai': {'ujian 1': 10}}
```

```
type(data)
```

```
dict
```

This variable contain string and integer and the data type is dict as dictionary

DICTIONARY EXAMPLE

```
print('ujian :', data['nilai']['ujian 1'])
```

```
ujian : 10
```

```
data["nama"]
```

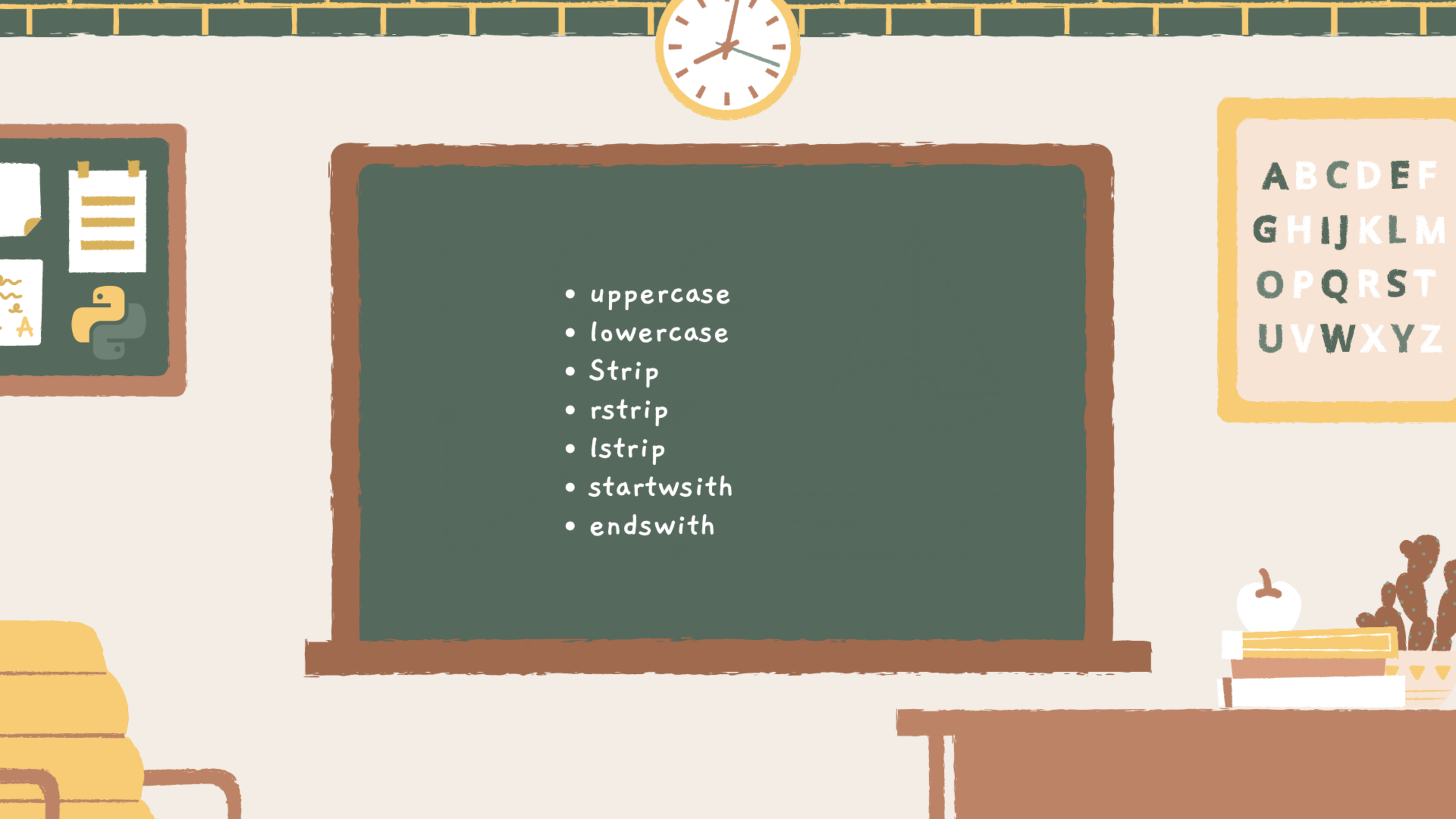
```
'Ana'
```

```
data["kelas"]
```

```
'data science B'
```

If you want to display the value from the variable, you need to call the key


NUMBER, CHARACTER, & STRING TRANSFORMATION
A B C D E F
G H I J K L M
O P Q R S T
U V W X Y Z


A B C D E F
G H I J K L M
O P Q R S T
U V W X Y Z
• uppercase
• lowercase
• Strip
• rstrip
• lstrip
• startwsith
• endswith

UPPERCASE AND LOWERCASE EXAMPLE

```
data = "kampus merdeka"
```

```
data.upper()
```

```
'KAMPUS MERDEKA'
```

Return the string to upper case

```
data = "kampus merdeka 2022"
```

```
data.upper()
```

```
'KAMPUS MERDEKA 2022'
```

```
data_1 = "KAMPUS MERDEKA"
```

```
data_1.lower()
```

```
'kampus merdeka'
```

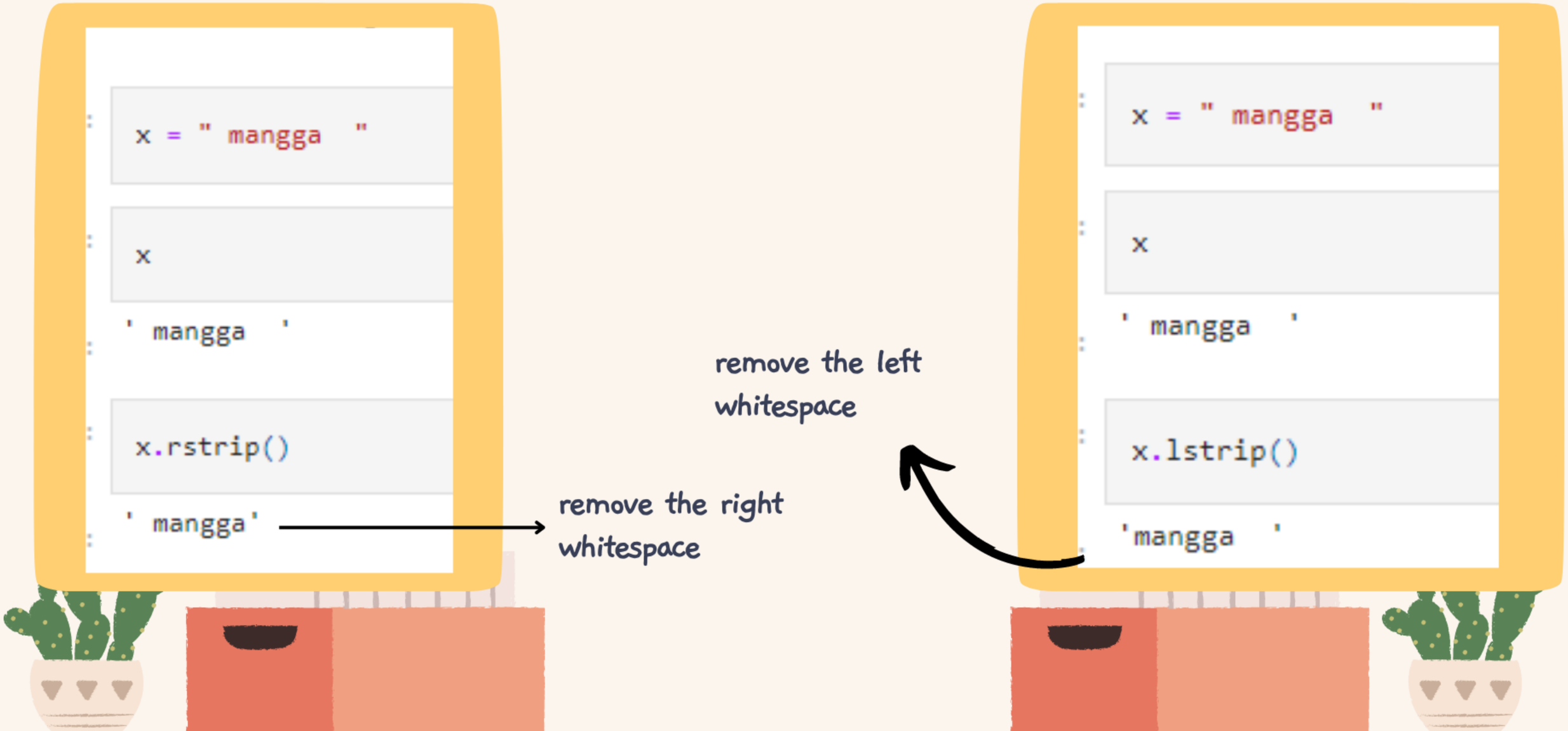
```
data_1 = "KAMPUS MERDEKA, ANGKATAN KE-3"
```

```
data_1.lower()
```

```
'kampus merdeka, angkatan ke-3'
```

Return the string to lower case

RSTIRP AND LSTRIP EXAMPLE



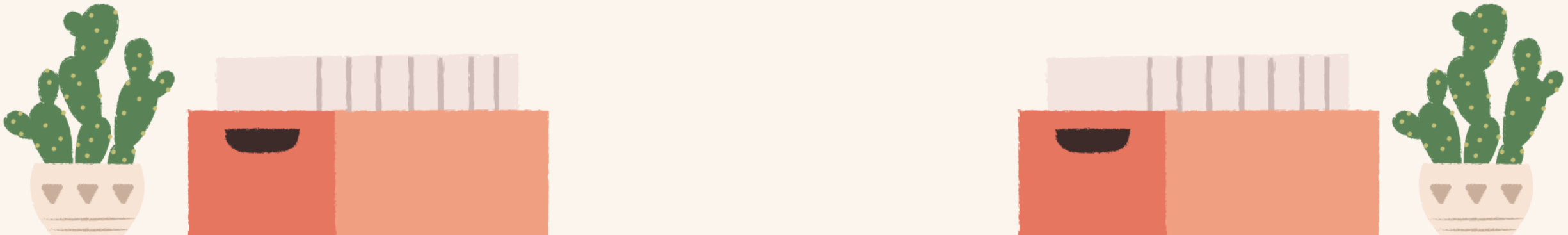
STRIP EXAMPLE

```
c = "Kampus Merdeka Batch3"
```

```
c.strip("Batch3")
```

```
'Kampus Merdeka '
```

is used to remove all the leading and trailing spaces from a string



STARTSWITH AND ENDSWITH EXAMPLE

```
r = "Kampus Merdeka Angkatan Tahun Ketiga"
```

```
r.startswith("Kampus")
```

```
True
```

```
r.startswith("Kampus Merdeka")
```

```
True
```

```
r.startswith("Angkatan")
```

```
False
```

The startswith() method returns True if the string starts with the specified value, otherwise False

The endswith() method returns True if the string end with the specified value, otherwise False

```
r = "Kampus Merdeka Angkatan Tahun Ketiga"
```

```
r.endswith("Ketiga")
```

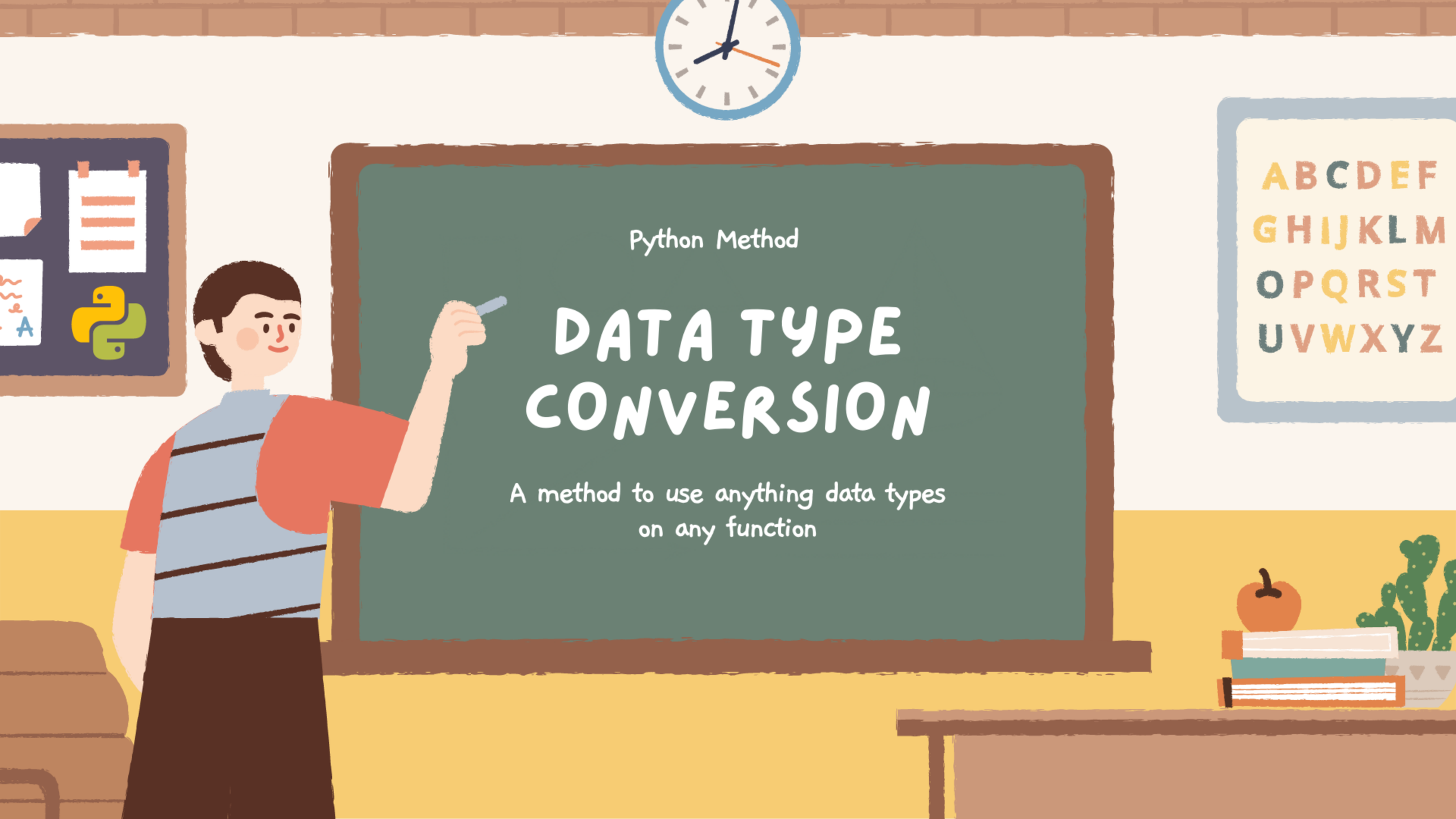
```
True
```

```
r.endswith("Tahun Ketiga")
```

```
True
```

```
r.endswith("Angkatan")
```

```
False
```


Python Method
DATA TYPE CONVERSION
A method to use anything data types on any function
A B C D E F
G H I J K L M
O P Q R S T
U V W X Y Z

DATA TYPE CONVERSION

Many Python functions are sensitive to the type of data. For example, you cannot concatenate a string with an integer:

```
In [6]: age = 6

In [7]: sign = 'You must be ' + age + '-years-old to enter this pla
        ---------------------------------------------------------------
        TypeError                                 Traceback (most
        recent call last)
        Input In [7], in <cell line: 1>()
        ----> 1 sign = 'You must be ' + age + '-years-old to enter
        this place'

        TypeError: can only concatenate str (not "int") to str
```

DATA TYPE CONVERSION

You will find yourself needing to convert one data to another. conversion functions are easy to remember: the type names double up as a conversion function. For example : str() is a function to convert an integer,a list, etc to a string

```
In [1]: age = 6

In [3]: sign = 'You must be ' + str(age) + '-years-old to enter
        sign

Out[3]: 'You must be 6-years-old to enter this place'
```


Build-in Function
PYTHON I/O
A Function to perform input and output task on python
A B C D E F
G H I J K L M
O P Q R S T
U V W X Y Z

OUTPUT USING PRINT() FUNCTION

We use print() function to output data to standard output device(screen). An example of this function is given below :

```
print("This sentense to the screen")
```

```
This sentense to the screen
```

Another example of this function is given below :

```
In [5]: a = 21

In [7]: print("The value of a is", a)
        The value of a is 21
```

OUTPUT FORMATTING

Sometimes we would like to format our output to make it look attractive. this can be done by using str.format() method. This method is visible to any string object. This is example of that :

```
a = 21
b = 20

print("The value of a is {} and b is {}".format(a,b))
```

```
The value of a is 21 and b is 20
```

ANOTHER OUTPUT FORMATTING

We can also format a string like the old sprintf() style used in C langunge. We use the % sign operator to accomplish that

```
a = 21
b = 20

print("The value of a is %i and b is %i"%(a,b))
```

```
The value of a is 21 and b is 20
```

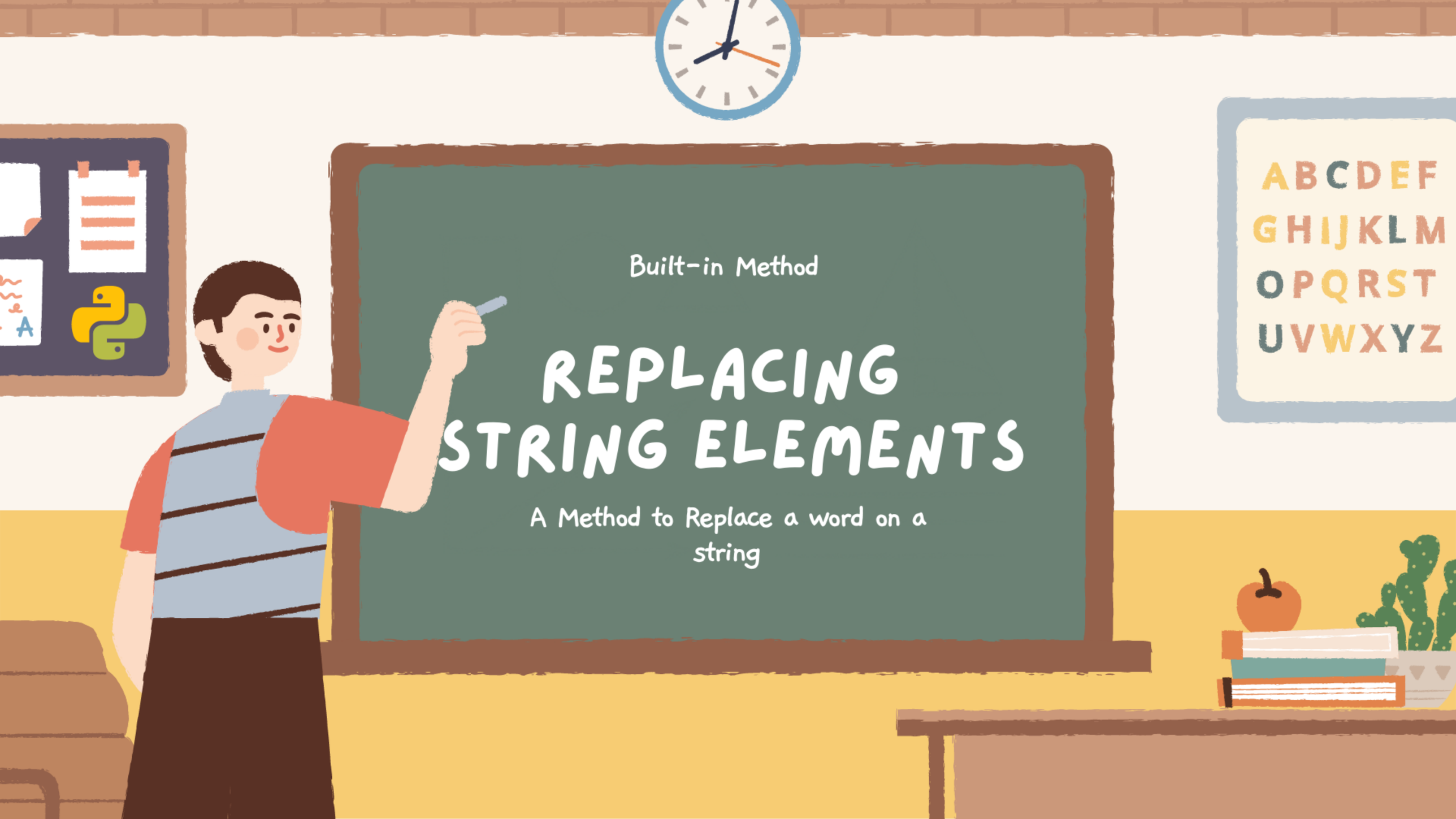
INPUT USING INPUT() FUNCTION

To allow flexibility. We might want to take input from the user. In python, we have the input() function to allow this. this example of input :

```
a = input("What is your name ? ")
```

```
What is your name ? John Harris
```

P.S : Everything what you input on input function, it will be a string. Make sure your data type before perform an operation on it.


Built-in Method
REPLACING STRING ELEMENTS
A Method to Replace a word on a string
A B C D E F
G H I J K L M
O P Q R S T
U V W X Y Z

REPLACE AN ELEMENT USING REPLACE() METHOD

The replace() method replaces each matching occurrence of the old character/text in the string with the new character/text. it takes maximum 3 parameter. the example to this method is given below :

```
a = "Today is Saturday"
```

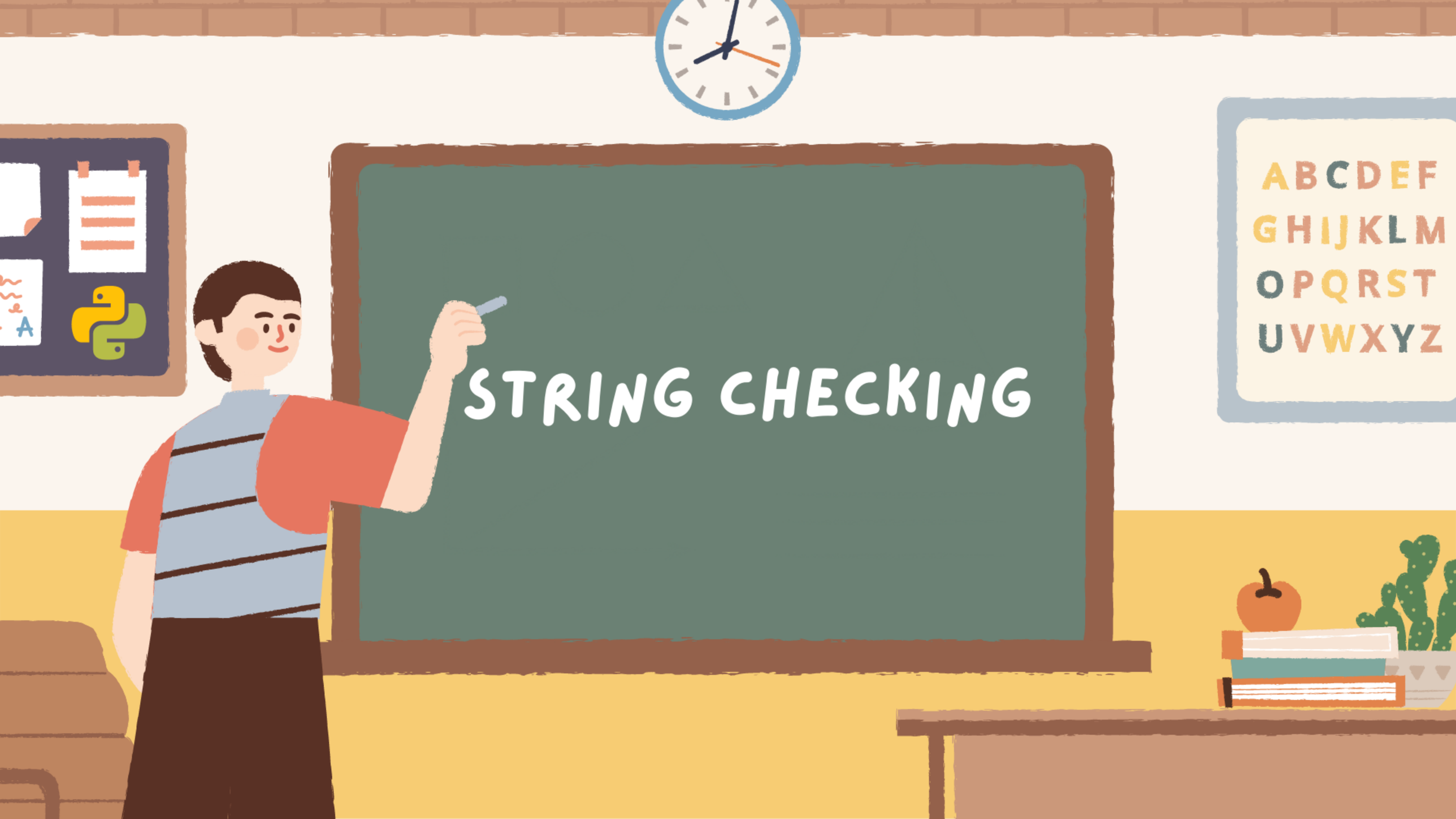
```
a.replace("Saturday", "Sunday")
```

```
'Today is Sunday'
```

```
b = "Yesterday is Saturday, Today is Saturday"
```

```
b.replace("Saturday", "Sunday", 2)
```

```
'Yesterday is Sunday, Today is Sunday'
```


STRING CHECKING
A B C D E F
G H I J K L M
O P Q R S T
U V W X Y Z


A B C D E F
G H I J K L M
O P Q R S T
U V W X Y Z
• isupper
• islower
• isalpha
• isalnum
• isdecimal
• istitle
• isspace

ISLOWER()

islower() method is checks whether all the case-based characters (letters) of the string are lowercase and it will return a boolean value

```
In [1]: a = "Team 3"

In [3]: a.islower()
Out[3]: True
```

ISUPPER()

isupper() method is checks whether all the case-based characters (letters) of the string are uppercase and it will return a boolean value

```
In [2]: b = "TEAM 3"

In [3]: b.isupper()
Out[3]: True
```

ISALPHA()

isalpha() method checks whether all characters of the string are alphabet and will return a boolean value.

```
In [1]: print("Python".isalpha())

        True
```

ISALNUM()

isalnum() method checks whether all characters of the string are alphanumeric and will return a boolean value.

```
In [5]: print("Team3".isalnum())

        True
```

ISDECIMAL()

isalnum() method checks whether all characters of the string are numeric and will return a boolean value.

```
In [7]: print("2022".isdecimal())

True
```

ISTITLE()

istitle() method checks whether all characters of the string contain a capital letter in each word followed by a lowercase letter and will return a boolean value.

```
In [9]: print("Team3 Data".istitle())

        True
```

ISSPACE()

issapce() method checks whether all characters of the string are whitespace and will return a boolean value.

```
In [11]: print(" ".isspace())

         True
```

CHALLENGE

Team 3

CREATE A SIMPLE SYNTAX FOR STRING DATA INPUT PROVIDED THAT ALL CHARACTERS ARE CAPITAL LETTERS OTHERWISE, ENTER IT AGAIN.

STRING

Python

CHECKER

I/O

CODE

```
while True:
    print('Please input your name:')
    name = input()
    if name.isupper():
        print("Hello Welcome", name)
        break
    print('Please correct your name.')
```

LET'S TRY INPUT NAME

NAME 1

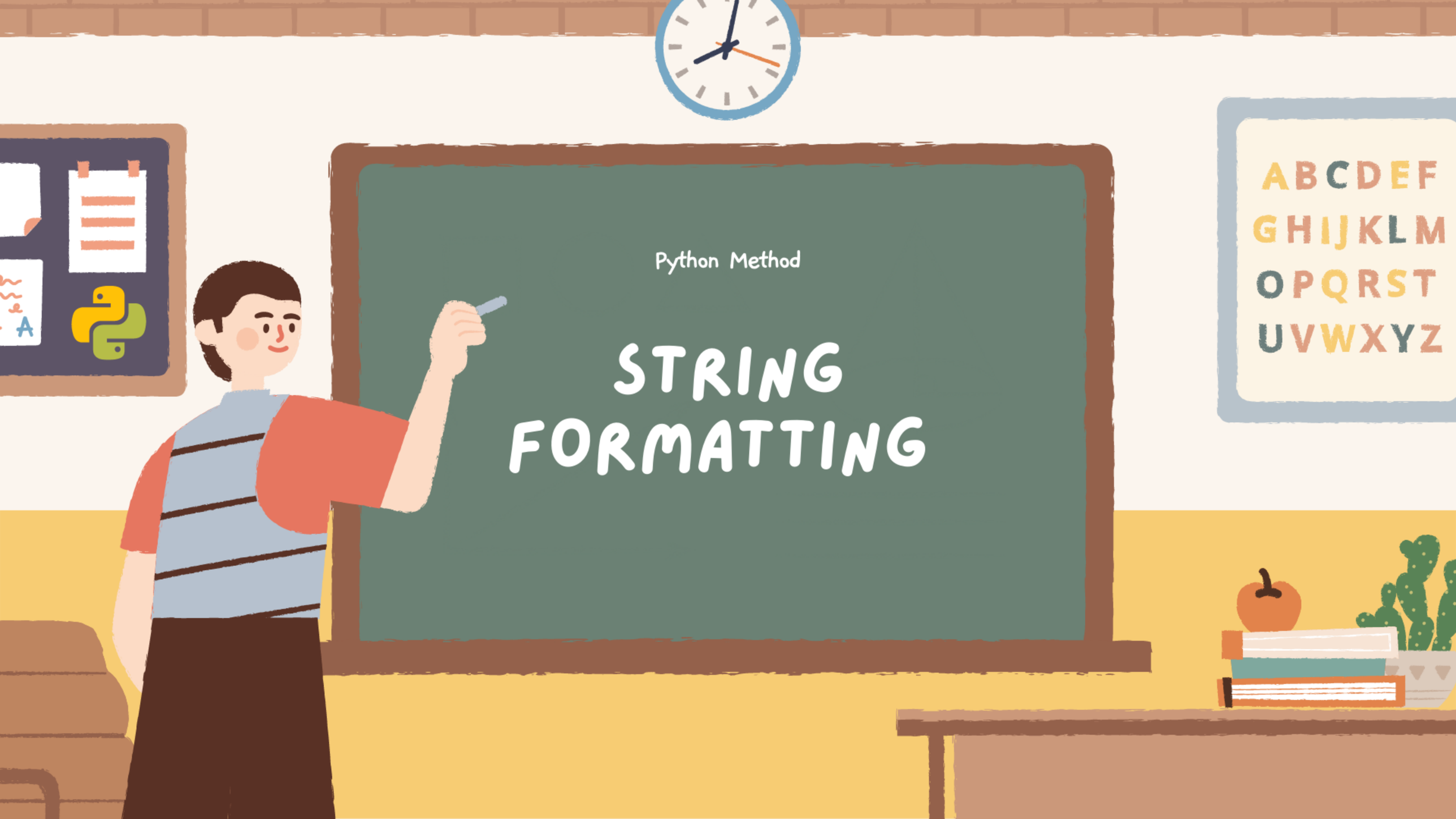
```
Please input your name:
Danar
Please correct your name.
Please input your name:
```

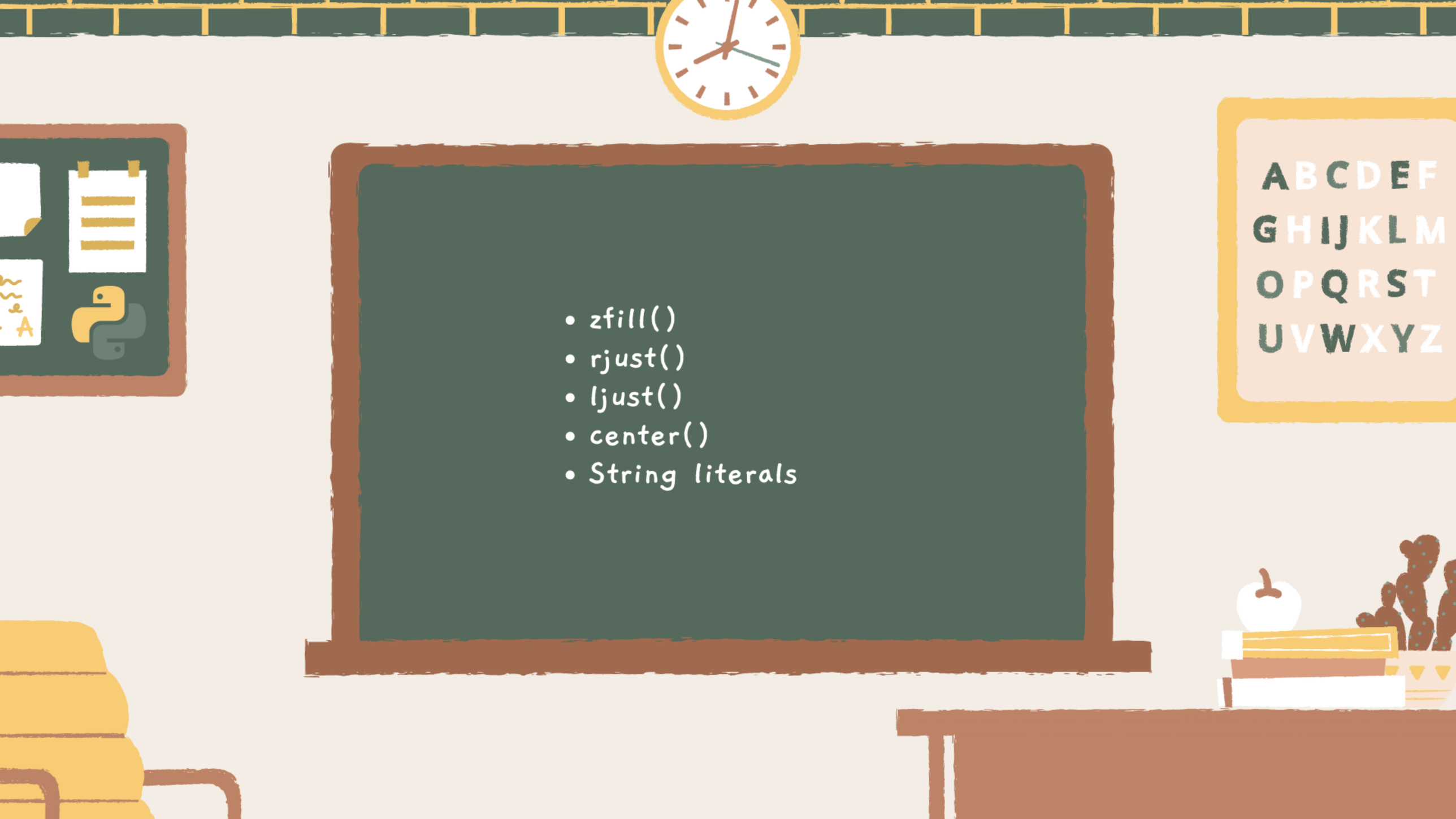
Doesn't work because 'Danar' is not capitalized

NAME 2

```
Please input your name:
AGUNG
Hello Welcome AGUNG
```

successful


Python Method
STRING FORMATTING
A B C D E F
G H I J K L M
O P Q R S T
U V W X Y Z


zfill()
rjust()
ljust()
center()
String literals
A B C D E F
G H I J K L M
O P Q R S T
U V W X Y Z

ZFILL() METHOD

The zfill() method will fill the string with a specified number of 0 values at the beginning. The type data must be string, whether not string, must be converted first. An example to this method is given below :

```
example1 = 7                # Type Data Integer

str(example1).zfill(3)  # Convert Type Data Integer to String
'007'
```

RJUST()

1. The rjust() method used to make text right-aligned.
2. This method will add a space to the string to make it match.
3. The parameter is an integer which is the overall length of the text (not the number of spaces added).
4. We can replace the whitespace with another symbol by entering the second parameter.

LJUST()

1. The rjust() method used to make text left-aligned.
2. This method will add a space to the string to make it match.
3. The parameter is an integer which is the overall length of the text (not the number of spaces added).
4. We can replace the whitespace with another symbol by entering the second parameter.

CENTER()

1. The rjust() method used to make text centered.
2. This method will add a space to the string to make it match.
3. The parameter is an integer which is the overall length of the text (not the number of spaces added).
4. We can replace the whitespace with another symbol by entering the second parameter.

EXAMPLE

```
In [5]: "Team3".rjust(10,"*")
Out[5]: '*****Team3'
```

rjust() method

```
In [7]: "Team3".ljust(10,"*")
Out[7]: 'Team3*****'
```

ljust() method

```
In [9]: "Team3".center(10,"*")
Out[9]: '**Team3***'
```

center() method

STRING LITERALS

Enscape Character
Raw Strings

ESCAPE CHARACTER

Escape Character is a subset of string literals that allows us to include the characters ' and " in the string part.

```
In [2]: print('Jum'at')

          Input In [2]
            print('Jum'at')
                      ^
        SyntaxError: invalid syntax
```

Replace to

```
In [3]: print('Jum\'at')

        Jum'at
```

RAW STRINGS

Raw strings used to print strings according to whatever input or text given. Raw Strings marked with an r before the string opener.

```
In [5]: print("Data\tScience")
        #Command 't' will create a new tab

        Data    Science
```

Replace to

```
In [6]: print(r"Data\tScience")
        #We can use raw strings

        Data\tScience
```

GITHUB LINK

https://github.com/agung15-debug/data-types-python


THANK YOU