Conversion, Transformation, Operation and Conditional Expression in Python

Seaborn Kusto

Official Teams

Malka
Rusyd
Abdussalam

Trianto
Haryo
Nugroho

Fauzi
Wardah
Ali

Rizki
Afrinal

Basic Data Types Conversion

We can convert the default data types by using the standard type conversion function : float(), int(), and str().

- Convert to float using float()

```
[1]  # Input an integer number
     3
     3

[2]  # Check the data type
     type(3)
     int
```

```
[3]  # Convert integer to float
     float(3)
     3.0

[4]  # Print the data type for integer to float conversion
     print(type(float(3)))
     <class 'float'>
```

Basic Data Types Conversion

- Convert to integer using int()

```
[5]  # Input a float number

     1.5

     1.5
```

```
[6]  # Check the data type

     type(1.5)

     float
```

```
[7]  # Convert float to integer
     # (Converting float to int will be floor/truncating or omitting the value after the comma)

     int(1.5)

     1
```

```
[8]  # Print the data type for float to integer conversion

     print(type(int(1.5)))

     <class 'int'>
```

Basic Data Types Conversion

- **Convert from-and-to string using str()**
  From-and-to-string conversions are tested and validated

String to float conversion

```
[ ]  # Input the string '1.5'
     '1.5'
     '1.5'

[ ]  # Check the data type
     type('1.5')
     str

[ ]  # Convert to float
     float('1.5')
     1.5

[ ]  # Check the data type for string to float conversion
     type(float('1.5'))
     float
```

Integer to string conversion

```
[ ]  # Input the integer number 10
     10
     10

[ ]  # Check the data type
     type(10)
     int

[ ]  # Convert to string
     str(10)
     '10'

[ ]  # Check the data type for integer to string conversion
     type(str(10))
     str
```

Basic Data Types Conversion

- Convert string using mathematical expression

Convert string containing mathematical expression to integer

```
# Input string contains mathematical expression '10+20'

'10+20'
```

```
'10+20'
```

```
[ ] # Check the data type

type('10+20')
```

```
str
```

```
[ ] # Convert to integer

int('10+20')

# The result will contain an error because invalid literal for int()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-192-cf2023c6a739> in <module>()
      1 # Convert to integer
      2
----> 3 int('10+20')

ValueError: invalid literal for int() with base 10: '10+20'
```

SEARCHSTACK OVERFLOW

```
[ ] # To convert string containing mathematical expression we can use eval()

eval('10+20')
```

```
30
```

```
[ ] # Check the data type for string containing mathematical expression to integer conversion

type(eval('10+20'))
```

```
int
```

Basic Data Types Conversion

- **Convert from-and-to string using str()**
  From-and-to-string conversions are tested and validated

String to integer with a combination of number and letter

```
[ ]  # Input the combination string '2a'

     '2a'
```

```
     '2a'
```

```
[ ]  # Check the data type

     type("2a")
```

```
     str
```

```
[ ]  # Convert to integer

     int("2a")

     # When the conversion is tested and validated, the result will be an error because invalid literal (2a is not an integer)
```

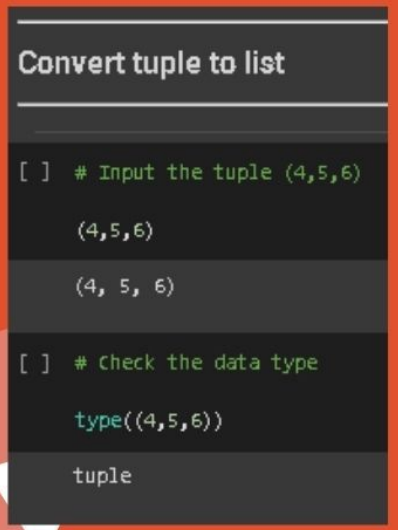
```
     ---------------------------------------------------------------------------
     ValueError                                Traceback (most recent call last)
     <ipython-input-51-3d1b272dba64> in <module>()
           1 # convert to integer
           2
     ----> 3 int("2a")

     ValueError: invalid literal for int() with base 10: '2a'
```

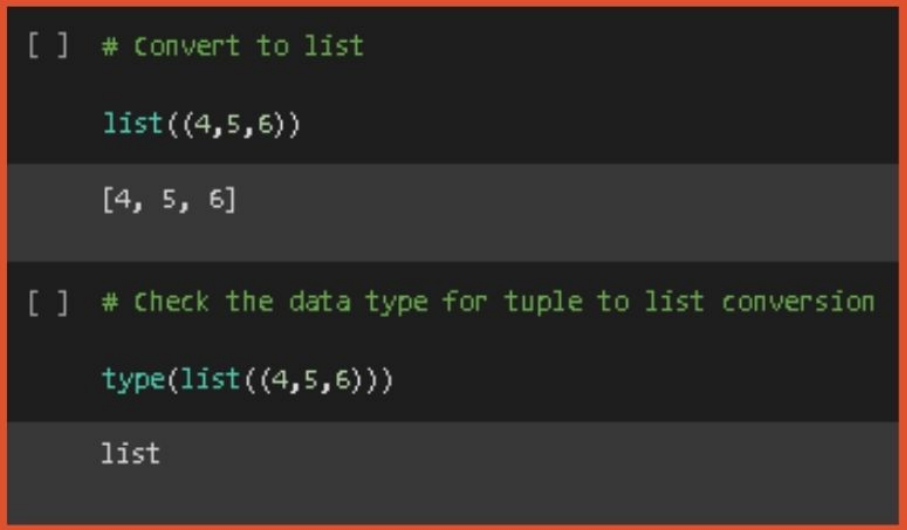
SEARCHSTACK OVERFLOW

Advanced Data Types Conversion

We can also convert any data types to advanced data types (list, tuple, set, dictionary) using list(), tuple(), set(), and dict().

- Convert tuple to list





Advanced Data Types Conversion

- Convert set to list

Convert set to list

```
[ ]  # Input the set {4,5,6}
     {4,5,6}
     {4, 5, 6}

[ ]  # Check the data type
     type({4,5,6})
     set

[ ]  # Convert to list
     list({4,5,6})
     [4, 5, 6]

[ ]  # Check the data type for set to list conversion
     type(list({4,5,6}))
     list
```

- Convert dictionary to list

Convert dictionary to list

```
[ ]  # input the dictionary {4: 5, 6: 7}
     {4: 5}
     {4: 5}

[ ]  # Check the data type
     type({4: 5})
     dict

[ ]  # Convert to list
     list({4: 5})
     # Dictionary contain of key-value (only the key converted to list)
     [4]
```

Advanced Data Types Conversion

- Convert integer to list

Convert integer to list

```
[ ]  # input the integer number 4

     4
```

```
[ ]  # Check the data type

     type(4)

     int
```

```
[ ]  # Convert to list

     list(4)

     # The result will be an error because integer object is not iterable
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-119-b4ee47a5a500> in <module>()
      1 # Convert to list
      2
----> 3 list(4)

TypeError: 'int' object is not iterable
```

SEARCHSTACK OVERFLOW

```
[ ]  # To convert integer to list we can input the integer to the brackets []

     [4]

     [4]
```

```
[ ]  # Check the data type

     type([4])

     list
```

Advanced Data Types Conversion

- Convert float to list

Convert float to list

```
[ ]  # Input the float number 1.5

     1.5
```

```
[ ]  # Check the data type

     type(1.5)

     float
```

```
[ ]  # Convert to list

     list(1.5)

     # The result will be an error because float object is not iterable
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-122-969a9fb14327> in <module>()
      1 # Convert to list
      2
----> 3 list(1.5)

TypeError: 'float' object is not iterable
```

SEARCHSTACK OVERFLOW

```
[ ]  # To convert integer to list we can input the float to the brackets []

     [1.5]

     [1.5]
```

```
[ ]  # Check the data type

     type([1.5])

     list
```

Advanced Data Types Conversion

- Convert string to list

Convert string to list

```
[ ]  # input the string 'Data Science'
     'Data Science'
```

```
     'Data Science'
```

```
[ ]  # Check the data type
     type('Data Science')
```

```
     str
```

```
[ ]  # Convert to list
     list('Data Science')
     # String elements will be broken down into per character
```

```
     ['D', 'a', 't', 'a', ' ', 'S', 'c', 'i', 'e', 'n', 'c', 'e']
```

Advanced Data Types Conversion

- Convert list to tuple

Convert list to tuple (immutable)

```
[ ]  # Input the list [4,5,6]
     [4,5,6]
     [4, 5, 6]

[ ]  # Check the data type
     type([4,5,6])
     list

[ ]  # Convert to tuple
     tuple([4,5,6])
     (4, 5, 6)

[ ]  # Check the data type for list to tuple conversion
     type(tuple([4,56]))
     tuple
```

- Convert set to tuple

Convert set to tuple (immutable)

```
[ ]  # input the set {4,5,6}
     {4,5,6}

[ ]  # Check the data type
     type({4,5,6})
     set

[ ]  # Convert to tuple
     tuple({4,5,6})
     (4, 5, 6)

[ ]  # Check the data type for set to tuple conversion
     type(tuple({4,5,6}))
     tuple
```

Advanced Data Types Conversion

- Convert dictionary to tuple

Convert dictionary to tuple (immutable)

```
[ ]  # input the dictionary {'Job': "Data Scientist"}

     {'Job': "Data Scientist"}
```

```
     {'Job': 'Data Scientist'}
```

```
[ ]  # Check the data type

     type({'Job': "Data Scientist"})
```

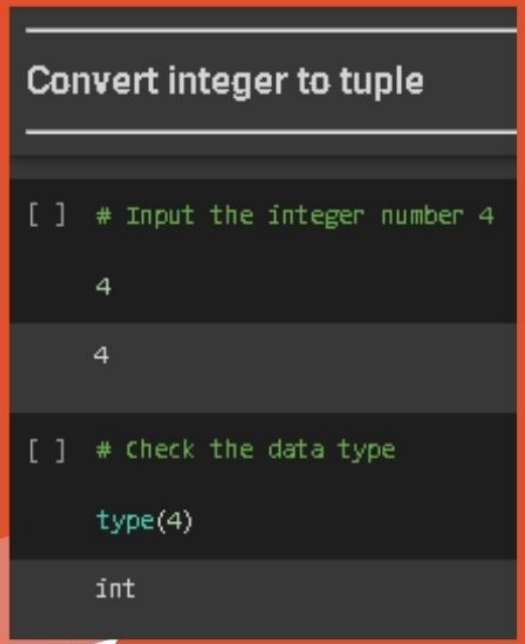
```
     dict
```

```
[ ]  # Convert to tuple

     tuple({'Job': "Data Scientist"})

     # Dictionary contain key-value (only the key converted to tuple, the value will be empty)
```

```
     ('Job',)
```

Advanced Data Types Conversion

- Convert integer to tuple




```
[ ]  # Convert to tuple

     tuple(4)

     # The result will be an error because integer object is not iterable

     ---------------------------------------------------------------------------
     TypeError                                 Traceback (most recent call last)
     <ipython-input-141-0665188eb38a> in <module>()
           1 # Convert to tuple
           2
     ----> 3 tuple(4)
           4
           5 # The result will be an error because float object is not iterable

     TypeError: 'int' object is not iterable

     SEARCH STACK OVERFLOW

[ ]  # To convert integer to tuple we can insert the integer into parentheses (), but at least consist of 2 integer

     (4,5)

     (4, 5)

[ ]  # Check the data type

     type((4,5))

     tuple
```

Advanced Data Types Conversion

- Convert float to tuple

Convert float to tuple

```
[ ]  # Input the float number 1.5

     1.5
```

```
[ ]  # Check the data type

     type(1.5)

     float
```

```
     # Convert to tuple

     tuple(1.5)

     # The result will be an error because float object is not iterable
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-143-97652d50f38f> in <module>()
      1 # Convert to tuple
      2
----> 3 tuple(1.5)

TypeError: 'float' object is not iterable
```

SEARCHSTACK OVERFLOW

```
[ ]  # To convert float to tuple we can insert the float into parentheses (), but at least consist of 2 floats

     (1.5, 2.5)

     (1.5, 2.5)
```

```
[ ]  # Check the data type

     type((1.5, 2.5))

     tuple
```

Advanced Data Types Conversion

- Convert string to tuple

Convert string to tuple

```
[ ]  # Input the string 'Data Science'

     'Data Science'
```

```
     'Data Science'
```

```
[ ]  # Check the data type

     type('Data Science')
```

```
     str
```

```
[ ]  # Convert to tuple

     tuple('Data Science')

     # String elements will be broken down into per character
```

```
     ('D', 'a', 't', 'a', ' ', 'S', 'c', 'i', 'e', 'n', 'c', 'e')
```

- Convert list to set

Convert list to set

```
[ ]  # Input the list [4,5,6]

     [4,5,6]
```

```
     [4, 5, 6]
```

```
[ ]  # Check the data type

     type([4,5,6])
```

```
     list
```

```
[ ]  # Convert to set

     set([4,5,6])
```

```
     {4, 5, 6}
```

```
[ ]  # Check the data type for list to set conversion

     type(set([4,5,6]))
```

```
     set
```

Advanced Data Types Conversion

- Convert dictionary to set

Convert dictionary to set

```
[ ]  # Input the dictionary {'Job': "Data Scientist"}
     {'Job': "Data Scientist"}
     {'Job': 'Data Scientist'}

[ ]  # Check the data type
     type({'Job': "Data Scientist"})
     dict

[ ]  # Convert to set
     set({'Job': "Data Scientist"})
     # Dictionary contains key-value (only the key converted to list)
     {'Job'}
```

- Convert list to dictionary

Convert list to dictionary

```
[ ]  # Input the list cointain value pairs [[4,5],[6,7]]
     [[4,5],[6,7]]
     [[4, 5], [6, 7]]

[ ]  # Check the data type
     type([[4,5],[6,7]])
     list

[ ]  # Convert to dictionary
     dict([[4,5],[6,7]])
     {4: 5, 6: 7}

[ ]  # Check the data type for list to dictionary conversion
     type(dict([[4,5],[6,7]]))
     dict
```

Advanced Data Types Conversion

- Convert tuple to dictionary

Convert tuple to dictionary

```
[ ]  # Input the tuple (('Job','Data Scientist'),('Salary','$ 10,0
     ((4,5),(6,7))
```

```
[ ]  # Check the data type
     type(((4,5),(6,7))
     )
     tuple
```

```
[ ]  # Convert to dictionary
     dict((('Job','Data Scientist'),('Salary','$ 10,000')))
     {'Job': 'Data Scientist', 'Salary': '$ 10,000'}
```

```
[ ]  # Check the data type for tuple to dictionary conversion
     type(dict((('Job','Data Scientist'),('Salary','$ 10,000'))))
     dict
```

- Convert set to dictionary

Convert set to dictionary

```
[ ]  # Input the set {('Job','Data Scientist'),('Salary','$ 10,000
     {('Job','Data Scientist'),('Salary','$ 10,000')}
     {('Job', 'Data Scientist'), ('Salary', '$ 10,000')}
```

```
[ ]  # Check the data type
     type({('Job','Data Scientist'),('Salary','$ 10,000')})
     set
```

```
[ ]  # Convert to dictionary
     dict({('Job','Data Scientist'),('Salary','$ 10,000')})
     {'Job': 'Data Scientist', 'Salary': '$ 10,000'}
```

```
[ ]  # Check the data type for set to dictionary conversion
     type(dict({('Job','Data Scientist'),('Salary','$ 10,000')}))
     dict
```

Input and Output

Variable

A variable is a place (in computer memory) to store certain data type values. To assign a value to a variable, we use the "=" operator between the variable name and the value we want to store.

```
[ ]  # Store the integer data 3 to the variable a

     a = 3

[ ]  # Store the float data 2.75 to the variable float

     float = 2.75

[ ]  # Store the string 'Data Science' to the variable b

     b = 'Data Science'

[ ]  # Store the list data [1,3,5,7,9] to the variable odd

     odd = [1,3,5,7,9]
```

```
[ ]  # Store the tuple data ('Data Engineer', 'Data Scientist', 'Data Analyst') to the variable job

     job = ('Data Engineer', 'Data Scientist', 'Data Analyst')

[ ]  # Store the set data {2,4,6,8,10} to the variable even

     even = {2,4,6,8,10}

[ ]  # Store the dictionary data {'Name': 'Trianto Haryo Nugroho', 'City': 'Bogor', 'Major': 'Actuarial Science'} to the variable profile

     profile = {'Name': 'Trianto Haryo Nugroho', 'City': 'Bogor', 'Major': 'Actuarial Science'}
```

Input and Output

Print()

The print() function is the direct way of output to the console/screen.

```
# Print the integer output 99

print(99)
```

```
99
```

```
[ ] # Print the float output 2.75

print(2.75)
```

```
2.75
```

```
[ ] # Print the string output 'How are you?'

print('How are you?')
```

```
How are you?
```

```
[ ] # Print the list output [2,4,6,8,10]

print([2,4,6,8,10])
```

```
[2, 4, 6, 8, 10]
```

```
[ ] # Print the tuple output (1,3,5,7,9)

print((1,3,5,7,9))
```

```
(1, 3, 5, 7, 9)
```

```
[ ] # Print the set output {'apple', 'banana', 'orange'}

print({'apple', 'banana', 'orange'})
```

```
{'banana', 'orange', 'apple'}
```

```
[ ] # print the dictionary output {'Name': 'Trianto Haryo Nugroho', 'Grade': 'A', 'Score': 90}

print({'Name': 'Trianto Haryo Nugroho', 'Grade': 'A+', 'Score': 95})
```

```
{'Name': 'Trianto Haryo Nugroho', 'Grade': 'A+', 'Score': 95}
```

Input and Output

Inserting variable values in strings

Python has several ways to enter a variable value in a string.

▼ Directly concatenate variables in a print () statement with this format:

print(variable)

```
[ ]  # Store the string data 'Back Yi Jin' in the variable 'Name'

     name = 'Back Yi Jin'
     age = 25
     city = 'Seoul'
```

```
[ ]  # Print the data of variable Name one by one

     print(name)
     print(age)
     print (city)
```

```
Back Yi Jin
25
Seoul
```

▼ Print all of the data with the format:

print(string statement, variable)

```
[ ]  # Store severals string data to its variable

     name = 'Back Yi Jin'
     age = 25
     city = 'Seoul'
```

```
[ ]  # Print the data one by one with this formatting

     print('My name:',name)
     print('My age:',age)
     print('My city:',city)
```

```
My name: Back Yi Jin
My age: 25
My city: Seoul
```

Input and Output

▼ Print all of the data with the format:

print(string statement 1, variable 1, string statement 2, variable 2, string statement 3, variable 3, etc)

```
[ ]  # Store severals string data to its variable

     name = 'Back Yi Jin'
     age = 25
     city = 'Seoul'
```

```
     int
```

```
[ ]  # print all of the data with this format

     print('My name is',name,'\nMy age is',age,'\nI live in',city)
```

```
     My name is Back Yi Jin
     My age is 25
     I live in Seoul
```

Input and Output

▼ Print the data one by one with the format:

print(string statement.format(variable))

```
[ ]  # Store severals string data to its variable

     name = 'Back Yi Jin'
     age = 25
     city = 'Seoul'
```

```
[ ]  # print the data one by one with this format

     print('My name is {}'.format(name))
     print('My age is {}'.format(age))
     print('I live in {}'.format(city))
```

```
My name is Back Yi Jin
My age is 25
I live in Seoul
```

▼ Print the data one by one with the format:

print(string statement 1 %(data type), string statement 2 %(data type), and string statement 3 %(data type)' % (variable 1,variable 2,variable 3)

- %d = integer
- %f = float
- %.2f = float with 2 decimal places
- %s = string

```
[ ]  # Store severals string data to its variable

     name = 'Back Yi Jin'
     age = 25
     city = 'Seoul'
```

Input and Output

```
[ ]  # print all of the data with this format

     print('My name is %s, my age is %d, and I live in %s' % (name,age,city))
```

```
     My name is Back Yi Jin, my age is 25, and I live in Seoul
```

```
[ ]  # print all of the data down with \n

     print('My name is %s \nMy age is %d \nI live in %s' % (name,age,city))
```

```
     My name is Back Yi Jin
     My age is 25
     I live in Seoul
```

Input and Output

▼ Store the data is the variable using input() and print the data one by one with the format:

variable = input('string statement: ')

*print(variable)

Input:

To allow the user to provide input to your program, use the input() function, where the argument in brackets () is the text you want to display (the prompt) and the variable before the equal sign (=) is the resultant holder of the user's input. By default input is string.

```
[ ]  # Store the data to the variable using input()
     name = input('My name: ')
```

```
My name: Back Yi Jin
```

```
[ ]  # Store the data to the variable using input()
     age = input('My age: ')
```

```
My age: 25
```

```
[ ]  # Store the data to the variable using input()
     city = input('My city: ')
```

```
My city: Seoul
```

Input and Output

▼ Directly print the data one by one with input() with the format:

print(input('string statement: ')

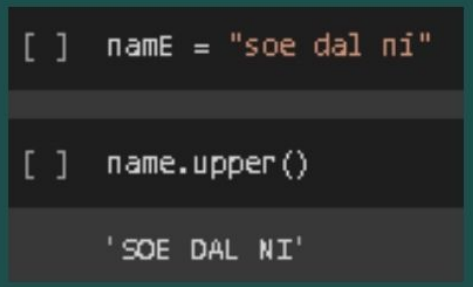
```
[ ]  # print all of the data the data directly with input()

     print(input('My name: '),input('My age: '), input('My city: '))
```

```
My name: Back Yi Jin
My age: 25
My city: Seoul
Back Yi Jin 25 Seoul
```

Number, Character, and String Transformation

upper()

- The upper() method can be used to convert a character or string from lowercase to uppercase.
- However, if the original letter is uppercase, then the letter is not changed.



Number, Character, and String Transformation

lower()

- lower() converts a character or string from uppercase to lowercase.
- However, if the original letter is lowercase, then the letter does not change.

```
[ ]  name = "SOE DAL MI"

[ ]  name.lower()

     'soe dal mi'
```

Number, Character, and String Transformation

- Both of these methods, both upper() and lower() are python's built-in methods that are used to handle string operations.
- If there are non-letter characters (such as symbols or numbers) that do not have a capital option, they are not changed

```
[ ]  name = "kin tae ri - 1"

[ ]  name.upper()

     'KIN TAE RI - 1'
```

Number, Character, and String Transformation

Prefix and Suffix

rstrip()

rstrip() will remove the whitespace to the right of the string or the end of the string

```
[ ]  a = "Data "

[ ]  a

'Data '
```

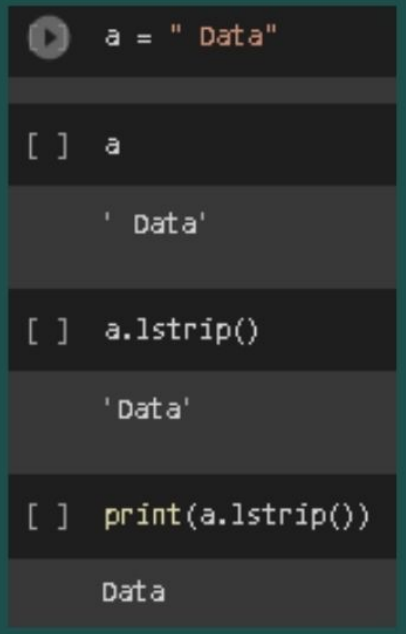
```
[ ]  print(a)

Data

[ ]  a.rstrip()

'Data'
```

```
print(a.rstrip())

Data
```

Number, Character, and String Transformation

lstrip()

lstrip() is used to remove whitespace to the left or the beginning of the string



Number, Character, and String Transformation

strip()

Strip() will remove whitespace at the beginning and end of the string.

```
[ ]  a = " Data "

[ ]  a

     ' Data '

[ ]  a.strip()

     'Data'
```

Number, Character, and String Transformation

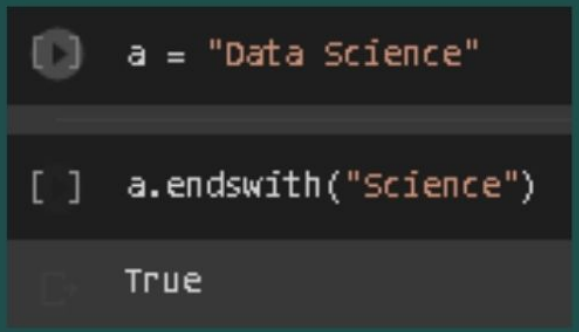
strip() can also specify which characters or sections you want to remove

```
a = "Data Science"

a.startswith("Science")
False

b = "Data Science Track"

b.startswith("Data")
True

c = "Data Science Track"

c.startswith("Data Science")
True

d = "Data Science Track"

d.startswith("Track")
```

Number, Character, and String Transformation

endswith()

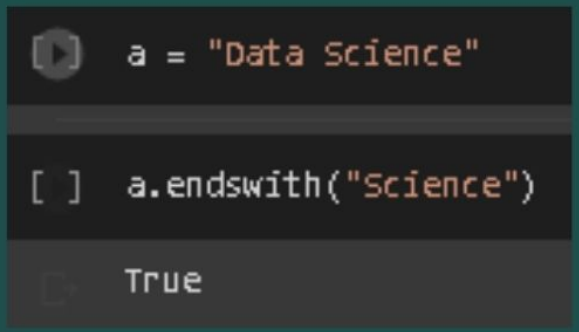
The endswith() method is the opposite of the startswith() method, this method will return True if the string ends with the specified suffix we want, otherwise it will return False.



Number, Character, and String Transformation

endswith()

The endswith() method is the opposite of the startswith() method, this method will return True if the string ends with the specified suffix we want, otherwise it will return False.



Replace String Elements

replace()

The replace() method can return a new string in the condition that the substring has been replaced with the entered parameter.

```
[ ] m = "My friend and I play tennis together"

[ ] print(m)

    My friend and I play tennis together

[ ] print(m.replace('tennis', 'football'))

    My friend and I play football together
```

replace()

The third parameter in replace can be filled with the number of substrings you want to replace.

```
a = "Hari ini hujan dan saya membawa jas hujan"
```

```
print(a.replace('hujan', 'mendung',1))
```

```
Hari ini mendung dan saya membawa jas hujan
```

```
b = "I don't understand excel so I take excel class, this week I will take an excel exam"
```

```
print(b.replace('excel', 'Python',2))
```

```
I don't understand Python so I take Python class, this week I will take an excel exam
```

String Checking

String Cheking
In the string checking category we will check the boolean of string
isupper()

iThe isupper() method returns True if all the characters are in upper case, otherwise False.
Numbers, symbols and spaces are not checked, only alphabet characters

```
[ ] a = "DATA SCIENCE"

[ ] a.isupper()

    True
```

```
[ ] b = 'data science'

[ ] b.isupper()

    False
```

islower()

iThe islower() method returns True if all the characters are in lower case, otherwise False.
Numbers, symbols and spaces are not checked, only alphabet characters.

```
[ ] a = "data science"

[ ] a.islower()

    True
```

```
[ ] b = "Data Science"

[ ] b.islower()

    False
```

We can perform operations on the result of the operation (method chain)

```
[ ] print("Science".upper().lower())

    science

[ ] print("Science".lower().upper())

    SCIENCE

[ ] print("Science".upper().lower().islower())

    True

[ ] print("Science".upper().lower().isupper())

    False
```

isalpha()

The isalpha() method returns True if all the characters are alphabet letters (a-z). Exemple of characters that are not alphabet letters: (space)!#%&? etc.

```
[ ] print("Data".isalpha())

    True

[ ] print("Ibu2".isalpha())

    False
```

Because there is a number, then the output returns False

isalnum()

The isalnum() method returns True if all characters in the string are alphanumeric (either alphabets or numbers). If not, it returns False.

```
[ ] print('Tahun2022'.isalnum())
    True

[ ] print('Data 2022'.isalnum())
    False
```

because there is a space then False

```
[ ] print('2022'.isalnum())
    True

[ ] print('Data'.isalnum())
    True
```

isdecimal()

The isdecimal() method returns True if all the characters are decimals (0-9).
This method is used on unicode objects.

```
[ ] print("Tahun2022".isdecimal())

    False

[ ] print("2022".isdecimal())

    True
```

isspace()

The isspace() method returns True if all the characters in a string are whitespaces, otherwise False.

```
[ ] print('\n'.isspace())
    True

[ ] print(' '.isspace())
    True
```

istitle()

The istitle() method returns True if all words in a text start with a upper case letter, AND the rest of the word are lower case letters, otherwise False. Symbols and numbers are ignored.

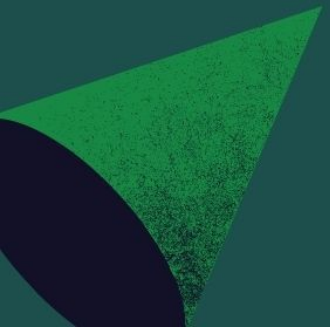
```
[ ] print("Data Science".istitle())

    True

[ ] print("Malka Rusyd".istitle())

    True

[ ] print("maulana ishaq".istitle())

    False
```

implementation String Checking

```
[5] while True:
        print('Enter Your Name:')
        name = input()
        if name.isalpha():
            print("Halo", name)
            break
        print('enter your name correctly.')
```

```
Enter Your Name:
Malka
Halo Malka
```

String Formating



zfill()

- A method that can add a numeric value of 0 to the left of a number or string using the zfill() method
- Users of this zfill() method can be applied to invoice numbers or queue numbers
- The number of characters must be less than or equal to the zfill value.

```
[ ] a = 7

[ ] str(a).zfill(3)
'007'
```

```
[2] c = -0.25

[4] str(c).zfill(5)
'-0.25'
```

```
[3] b = "why"

[5] b.zfill(6)
'000why'
```

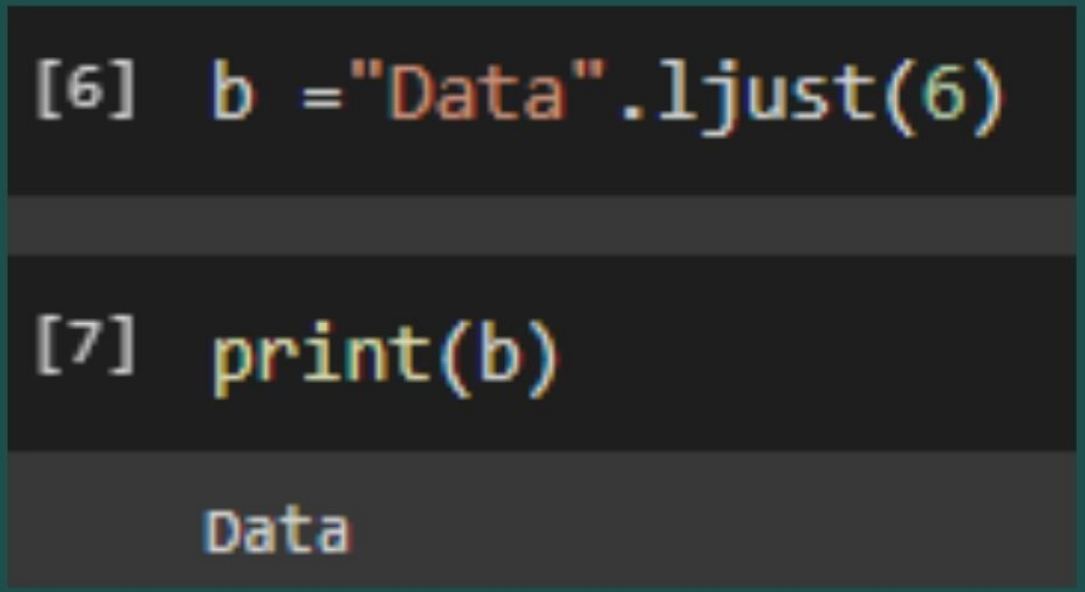
rjust()

- The rjust() method is used to make the text right-align
- This method will add a space to the string to make it match
- The parameter is an integer which is the overall length of the text (not the number of spaces added)

```
[ ] a = "Data".rjust(10)

[ ] a

    '      Data'

[ ] print(a)

          Data

[ ] len(a)

    10
```

ljust()

- The ljust() method is used to make text left-aligned
- This method will add extra padding space to make it match
- The parameter is an integer which is the total length of the text (not the number of spaces added)



center()

The center() method as the name suggests will center the text

```
[ ] c = "Data".center(10)

[ ] print(c)

       Data

[ ] 'Data'.center(10, "-")

    '---Data---'
```

String Literals

Generally, strings are written easily in Python, enclosed in single quotes, but, under certain conditions, a single quote is required in the middle of the string.

```
[8]  "we'are good"

     'we'are good'
```

if using single ' then error

```
'we're good'
  File "<ipython-input-11-1f303d8fb373>", line 1
    'we're good'
         ^
SyntaxError: invalid syntax

SEARCH STACK OVERFLOW
```

Input Escape character "backslash" allows you to enter the character 'and' in the string part.

```
[12] 'he\'s coming to me'

     'he's coming to me'

[14] 'Jumat\'at'

     'Jumat'at'

[13] print('data\\nku')

     data\nku
```

IRaw String

Python also provides a way to print strings according to any input

```
[ ] print("Data\tScience")
```

```
Data    Science
```

```
[ ] print(r"Data\tScience")
```

```
Data\tScience
```

Supposedly, the \t command will create a tab, but since we are using a raw string, the sentence is printed raw as is

Operation Tuple, List, Set, and String

LEN

Returns the length of an object (Tuple, List, and String)

```
[ ] a = 'Data Science'
    len(a)

    12

[ ] a = (2,7,1,1,2,4,8,'eleven')
    len(a)

    8

[ ] len(a[-1])

    6

[ ] a = ['eight','eleven','ten']
    len(a)

    3

[ ] a = {2,7,1,1,2,4,8,'eleven'}
    len(a)

    6

[ ] print(a)

    {1, 2, 4, 7, 8, 'eleven'}
```

the data in the set is unique so the duplicate data will only appear once

COUNT

Returns the number of elements with the specified value

```
[ ] a = "Data Science"
    a.count('a')

    2

[ ] a = (2,7,1,1,2,4,8,'eleven')
    a.count('eleven')

    1

[ ] a = [2,7,1,1,2,4,8,'eleven','ten']
    a.count(2)

    2

    a[-2].count('e')

    3

    a = {2,7,1,1,2,4,8,'eleven'}
    a.count(2)

    ---------------------------------
    ---------------------------------
    -----------
    AttributeError
```

object 'set' does not have attribute 'count' because 'set' is uniqe so each data must be one

REVERSE

reverses the sorting order of the elements of list.

```
[ ] a = [2,7,1,1,2,4,8,11]
    a.reverse()
    print(a)
```

```
[11, 8, 4, 2, 1, 1, 7, 2]
```

```
[ ] a = [3,'one',9,2,'ten']
    a.reverse()
    print(a)
```

```
['ten', 2, 9, 'one', 3]
```

```
[1] a = ['Science','Analyst','Python']
    a.reverse()
    print(a)
```

```
['Python', 'Analyst', 'Science']
```

SORT

Sorts the list ascending by default.

String in list sorted alphabetically from the first character. if the first character is the same then sorted by the second character, and so on.

```
[ ] # sort descending
    a = [2,7,11,2,3,4,8,11,9,9]
    a.sort(reverse=True)
    print(a)
```

```
[11, 11, 9, 9, 8, 7, 4, 3, 2, 2]
```

```
[ ] # case sensitive
    a = ['data','Python','coding','string','List']
    a.sort()
    print(a)
```

```
['List', 'Python', 'coding', 'data', 'string']
```

```
[ ] # sort descending by ignoring case sensitive without changing the contents of the list
    a = ['data','Python','coding','string','List']
    a.sort(key=str.lower,reverse=True)
    print(a)
```

```
['string', 'Python', 'List', 'data', 'coding']
```

```
[ ] # the numbers type string in list are sorted according to the sorting rules on the string
    a = ['1','16','55','151','213','3']
    a.sort()
    print(a)
```

```
['1', '151', '16', '213', '3', '55']
```

```
[3] a = [2,7,1,1,2,4,8,11]
    a.sort()
    print(a)
```

```
[1, 1, 2, 2, 4, 7, 8, 11]
```

```
[4] # comparison not supported between
    # instances of string and number
    a = [3,'one',9,2,'ten']
    a.sort()
    print(a)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-4-6284020a10a7> in <module>()
      2 # instances of string and number
      3 a = [3,'one',9,2,'ten']
----> 4 a.sort()
```

Tuple, set, and string cannot support reverse or string

MAX

Returns the item with the highest value, or the item with the highest value in an iterable.

```
[5] a = "Data Science"
    max(a)

't'

[6] a = (2,7,1,1,2,4,8,11)
    max(a)

11

[7] a = {2,7,1,1,2,4,8}
    max(a)

8

[8] a = ['eight','eleven','ten']
    max(a)

'ten'

[9] a = [2,7,1,1,2,4,8,'eleven']
    max(a)

---------------------------------
---------------------------------
```

MIN

Returns the item with the lowest value, or the item with the lowest value in an iterable.

```
[16] a = "python"
     min(a)

'h'

[18] a = [2,7,1,1,2,4,8,11]
     min(a)

1

[17] # Case sensitive.
     a = 'Python'
     min(a)

'P'

[19] a = {2,7,1,1,2,4,8}
     min(a)

1

[20] a = (2,7,1,1,2,4,8,'eleven')
     min(a[-1])

'e'

[21] a = (2,7,1,1,2,4,8,'eleven')
     min(a)
```

sort, max and min. comparison not supported between instances of string and number (integer or float)

IN

Returns True if a sequence with the specified value is present in the object

```
[28] a = [1,2,3,4,5]
     1 in a

True

[29] a = 'Data Science'
     'c' in a

True

[30] # case sensitive
     a = 'Data Science'
     'd' in a

False
```

NOT IN

Returns True if a sequence with the specified value is not present in the object

```
[33] a = (1,2,3,4,5)
     1 not in a

False

[34] a = 'Data Science'
     'd' not in a

True
```

MULTIPLE VALUE

Create a variable name for each item in the list on condition that it must be the same length.

```
[22] a = ['string','12','arial']
     typedata , size , font = a

[23] print(typedata)

     string

[24] print(font)

     arial
```

RANGE

The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and stops before a specified number.

syntax : range(start, stop, step)

```
[25] for i in range(3):
       print(i)

     0
     1
     2

[26] for i in range (2,6):
       print(i)

     2
     3
     4
     5

[27] for i in range (2,10,2):
       print(i)

     2
     4
     6
     8
```

Operator, Operand, and Expression

Addition (+)

```
[1] # If the operands are number.
    # The result is the sum of a and b
    1 + 2
```

```
3
```

```
[2] 350 + 42.5
```

```
392.5
```

```
[3] # If the operands are string.
    # The result is a concatenation
    # of the words a and b
    'Data' + 'Science'
```

```
'DataScience'
```

```
[4] '150' + '200'
```

```
'150200'
```

Subtraction (-)

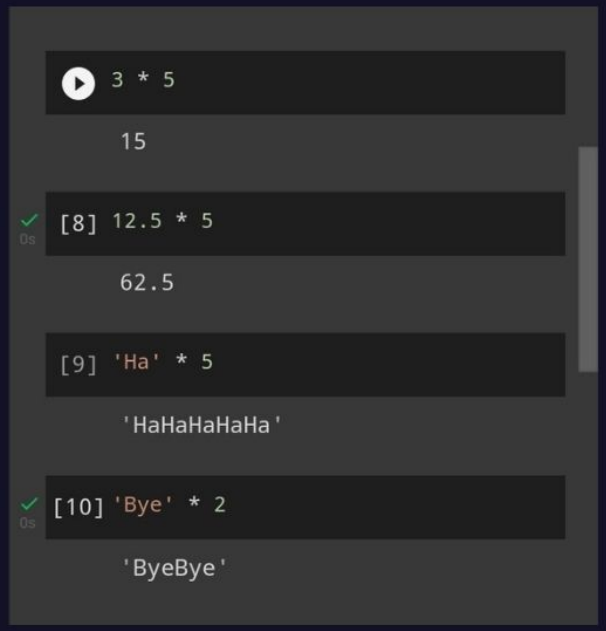
```
[5] 150 - 81
```

```
69
```

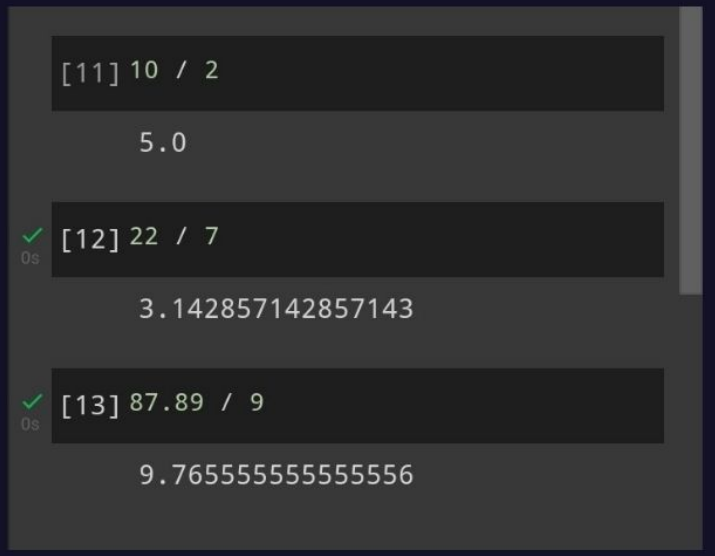
```
[6] 31 - 90.75
```

```
-59.75
```

MULTIPLICATION (*)



DIVISION (/)



EXPONENTATION (**)

```
[14] # The result is a raised to
     # the power of b
     2 ** 3
```

```
8
```

```
[15] 4 ** 0.5
```

```
2.0
```

MODULE (%)

```
[16] # The result is the remainder
     # when a is divided by b
     22 % 7
```

```
1
```

```
[17] 20 % 3.5
```

```
2.5
```

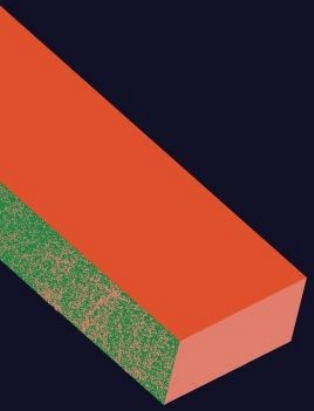
FLOOR DIVISION (//)

```
[20] # Quotient when a is divided by b,
     # rounded to the next smallest
     # whole number
     22 // 7
```

```
3
```

```
[21] 20.5 // 3.5
```

```
5.0
```



EQUAL TO (==)

The result is True if the value of a is equal to the value of b. False otherwise

```
[1] 50 == 50
```

```
True
```

```
[2] 35 == 50
```

```
False
```

```
[3] "Data" == "Data"
```

```
True
```

```
[4] # Case sensitive in python
    'Data' == 'data'
```

```
False
```

NOT EQUAL TO (!=)

True if a is not equal to b. False otherwise

```
[5] 71 != 45
```

```
True
```

```
[6] 71 != 71
```

```
False
```

```
[7] # Case sensitive in python
    'Python' != 'python'
```

```
True
```

GREATER THAN (>)

The result is True if the value of a is greater than the value of b. False otherwise

```
[21] 67 > 80
     False

[23] 91 > 67
     True

[24] 'z' > 'a'
     True

[25] 'Z' > 'a'
     False

[26] '30' > '251'
     True
```

GREATER THAN OR EQUAL TO (>=)

True if a is greather than or equal to b. False otherwise

```
[15] 80 >= 80
     True

[16] 99 >= 100
     False

[17] 'z' >= 'z'
     True

[18] # Case sensitive
     'A' >= 'a'
     False
```

If the operands are string. The results will be in alphabetical order (like a sort)

LESS THAN (<)

The result is True if the value of a is less than the value of b. False otherwise

```
[35] 100 < 510
     True

[36] 1000 < 510
     False

[37] 'data' < 'date'
     True

[39] 'Abjad' < 'abjad'
     True

[40] 'abjad' < 'Abjad'
     False

[41] '30' < '251'
     False
```

LESS THAN OR EQUAL TO (<=)

True if a is less than or equal to b. False otherwise

```
[42] 71 <= 71
     True

[43] 60 <= 80
     True

[44] 'data' <= 'data'
     True
```

If the operands are string. The results will be in alphabetical order (like a sort)

Conditional Expression

Conditional Expression in Python perform different computations or actions depending on whether a specific Boolean constraint evaluates to true or false. The syntax used to create conditional control flow in Python is if, elif, and else

Conditional Expression

if Condition

```
[1] x = 5
    y = 6
```

```
[2] if (x < y): # check condition
      print("x is less than y") # if the condition True Python will execute this code
```

```
x is less than y
```

```
[3] if (x > y):
      print("x is greater than y") # in this case the condition is False
```

Conditional Expression

else Condition

```
[ ] a = 2
    b = 6
```

```
[ ] if (a == b) : # check condition
      print("a is same with b")
    else :
      print("a is not same with b") # this code will execute if condition False
```

```
a is not same with b
```

Conditional Expression

elif Condition

```
[ ] p = 4
    q = 4
    r = 5
```

```
if (p > q) :
  print("p greater than q")
elif (p == q) :
  print("p is same with q") #this code will execute if condition False
else :
  print("p less than q")
```

```
p is same with q
```

```
[ ] if (p > r) :
      print("p greater than r")
    elif (p == r) :
      print("p is same with r") #this code will execute if condition False
    else :
      print("p less than r") #this code will execute if and elif condition False
```

```
p less than r
```

Thank You!