

NULL GROUP

Python Fundamentals

Fahmi Hamzah
Aditya Hermawan
Muh Aksa Hadim
Fara Rizkhi Karunia
Salsabilla Atasyaputri Setyawan
Nafiatul Risa

Use Case

USE CASE SUMMARY

- Objective Statement :
 - 1.basic knowledge of Python Data Type Conversion
 - 2.basic knowledge about Implicit Type Conversion
 - 3.basic knowledge of Explicit Type Conversion
 - 4.basic knowledge of Formatting String
 - 5.basic knowledge of Text align right/left/center
 - 6.basic knowledge of String Literals
 - 7.basic knowledge of Python Assignment Operators
 - 8.basic knowledge of Conditional Expression



USE CASE SUMMARY

- Challenge

- 1.Syntax writing error

- 2.Differences in the use of each Syntax

- Expected results :

- 1.Explanation of some data types

- 2.Explanation of some data types

- 3.Example of writing syntax for data types in Python

- benefit

- 1.as a basic material for learning python in recognizing data

- 2.increase knowledge in python programming language



1. PYTHON TYPE DATA CONVERSION

THE PROCESS OF CONVERTING THE VALUE OF ONE DATA TYPE (INTEGER, STRING, FLOAT, ETC.) TO ANOTHER DATA TYPE IS CALLED TYPE CONVERSION. PYTHON HAS TWO TYPES OF TYPE CONVERSION.

- IMPLICIT TYPE CONVERSION
- EXPLICIT TYPE CONVERSION



Implicit Type Conversion

In Implicit type conversion, Python automatically converts one data type to another data type. This process doesn't need any user involvement.

```
[ ] ex_integer = 350
    ex_float = 4.35

    new = ex_integer + ex_float

    print("Value ex_integer:", ex_integer)
    print("Value ex_float:", ex_float)
    print("Value new:", new)

    print("Data type of ex_int:", type(ex_integer))
    print("Data type of ex_flo:", type(ex_float))
    print("Data type of new:", type(new))

Data type of ex_int: <class 'int'>
Data type of ex_flo: <class 'float'>
Data type of new: <class 'float'>
```

- WE ADD TWO VARIABLES EX_INTEGER AND EX_FLOAT, STORING THE VALUE IN NEW.
- WE WILL LOOK AT THE DATA TYPE OF ALL THREE OBJECTS RESPECTIVELY.
- IN THE OUTPUT, WE CAN SEE THE DATA TYPE OF EX_INTEGER IS AN INTEGER WHILE THE DATA TYPE OF EX_FLOAT IS A FLOAT.
- ALSO, WE CAN SEE THE NEW HAS A FLOAT DATA TYPE BECAUSE PYTHON ALWAYS CONVERTS SMALLER DATA TYPES TO LARGER DATA TYPES TO AVOID THE LOSS OF DATA.

This is what call implicit type conversion.
But it is useless to string data type.

Explicit Type Conversion

In Explicit Type Conversion, users convert the data type of an object to required data type. We use the predefined functions like `int()`, `float()`, `str()`, etc to perform explicit type conversion.

This type of conversion is also called typecasting because the user casts (changes) the data type of the objects.

SYNTAX :

`<required_datatype>(expression)`

Typecasting can be done by assigning the required data type function to the expression.

FOR EXAMPLE :

```
[ ] ex_int = 350
    ex_flo = 4.35
    ex_str = "50.5"

    print("Value ex_int:", ex_int)
    print("Value ex_flo:", ex_flo)
    print("Value ex_str:", ex_str)

    print("Data type of ex_int:", type(ex_int))
    print("Data type of ex_flo:", type(ex_flo))
    print("Data type of ex_str:", type(ex_str))

    print("*****")

    ex_flo = int(ex_flo)
    ex_int = str(ex_int)
    ex_str = float(ex_str)

    print("Data type of ex_flo after Type Casting:", type(ex_flo))
    print("Data type of ex_int after Type Casting:", type(ex_int))
    print("Data type of ex_str after Type Casting:", type(ex_str))

    print("Value ex_int:", ex_int)
    print("Value ex_flo:", ex_flo)
    print("Value ex_str:", ex_str)
```

Output :

```
Value ex_int: 350
Value ex_flo: 4.35
Value ex_str: 50.5
Data type of ex_int: <class 'int'>
Data type of ex_flo: <class 'float'>
Data type of ex_str: <class 'str'>
*****
Data type of ex_flo after Type Casting: <class 'int'>
Data type of ex_int after Type Casting: <class 'str'>
Data type of ex_str after Type Casting: <class 'float'>
Value ex_int: 350
Value ex_flo: 4
Value ex_str: 50.5
```


In the program of Explicit Type Conversion, we can explain that :



- We add three variables `ex_int`, `ex_flo`, and `ex_str`
- We will look at the data type of all three objects respectively.
- In the output, we can see the data type of `ex_int` is an integer, `ex_str` is a string, and the data type of `num_flo` is a float.
- We converted `ex_flo` from float(higher) to integer(lower) type using `int()` function to perform the addition. `ex_int` from integer to string type using `str()` function. `ex_str` from string to float type using `float()`

WE CAN ALSO DOING CONVERSION WITH DATA LIST, TUPLE, AND SET

```
[ ] data_list=["null", "Science", 10, 6.0, "MEMBER"]  
    data_set=("null", "data", 2, 10.0, "6 Member")  
  
    print("Data list:",data_list)  
    print("Data set:",data_set)  
  
    print("List to Set:", set(data_list))  
    print("Set to Tuple:", tuple(data_set))
```

OUTPUT :

```
Data list: ['null', 'Science', 10, 6.0, 'MEMBER']  
Data set: ('null', 'data', 2, 10.0, '6 Member')  
List to Set: {'MEMBER', 6.0, 10, 'Science', 'null'}  
Set to Tuple: ('null', 'data', 2, 10.0, '6 Member')
```

Conversion to a dictionary

FOR CONVERSION TO A DICTIONARY, THE DATA MUST MEET THE KEY-VALUE REQUIREMENTS. HERE ARE TWO EXAMPLES OF CONVERSIONS:

- List of several lists whose contents are value pairs into a dictionary.
- As well as converting a list of tuples containing value pairs into a dictionary

```
[ ] [[["Name","Null Science"],["Member",6]]  
    print("Dictionary conversion:",dict([[["Name","Null Science"],["Member",6]]]))
```

```
Dictionary conversion: {'Name': 'Null Science', 'Member': 6}
```



2. HOW TO TAKE INPUT AND DISPLAY OUTPUT

- INPUT IS THE INPUT WE GIVE TO THE PROGRAM.
- THE PROGRAM WILL PROCESS IT AND DISPLAY THE OUTPUT.
- INPUT AND OUTPUT ARE THE CORE OF ALL COMPUTER PROGRAMS.

How to Take Input from Keyboard

PYTHON ALREADY PROVIDES INPUT() FUNCTIONS TO TAKE INPUT FROM THE KEYBOARD.

```
[ ] variable_name = input(" ")
```

The meaning is, the text that we input from the keyboard will be saved into variable_name.

```
# Take Input
name = input("What's your name : ")
age = input("How old are you : ")

# Display Output
print("Hi...",name,"Your age is ",age,"years old")
```

What is the input() functions?

- The input() function is used to retrieve numeric data and retrieve text.
- In Python, it's enough to just use the input() function, because the raw_input() function is already combined there.

HOW TO DISPLAY OUTPUT

TO DISPLAY TEXT OUTPUT, WE USE THE PRINT() FUNCTION

1

```
[ ] print("Hello World!")  
    print(variable_name)  
    print("Text", variable_name)
```

Displaying Variables
and Text

2

```
[ ] name = "Sara"  
    print("Hi...",name)
```

Between the words Hi and Sara
there is a space as a separator,
because use a comma.
Should not be enclosed when
using a comma.

3

```
[ ] name = "Sara"  
    print("Hi... " + name)
```

If you want to use brackets,
then we must combine the
text and variables with a plus
sign (+).

USING THE FORMAT() FUNCTION

The format() function will combine the contents of the variable with the text

```
[ ] name = input("Name : ")  
    print("Hi {} How are you ?".format(name))
```

The {} sign will be automatically replaced according to the value we input to the variable name.

```
[ ] hobby = input("Your hobby : ")  
    reason = input("Because : ")  
  
    print ("My hobby is {}, i like my hobby because {} :)".format(hobby, reason))
```


USING THE STRING FORMATTING

Combining text and variables using the percent symbol (%).

```
[ ] say = input("What do you want to say : ")  
    print ("I want to say %s" % say)
```

The %s sign will be automatically replaced with the value we input to the variable say.

```
name = input("My name : ")  
age = 20  
height = 165  
  
print ("Hi %s, your age is %d years old and your height is %d cm" % (name,age,height))
```

Another note :

- %s for text data types
- %d for numbers (decimal)
- %f for fractional numbers

3. NUMBER, CHARACTER, & STRING TRANSFORM IN PYTHON

CHANGING UPPERCASE AND LOWERCASE

- Upper()

Convert character/String from lowercase to uppercase. If the letter first big, then nothing will change.

```
[1] name = 'sara'  
    name = name.upper()  
    print(name)
```

SARA

- Lower()

Convert character/String from uppercase to lowercase. If the letter first big, then nothing will change.

```
name = 'SARA'  
name = name.lower()  
print(name)
```

sara

BEGINNING AND AN END

1

```
print('Sing '.rstrip())
```

```
Sing
```

rstrip()

This method will remove the whitespace to the right of the string or the end strings.

2

```
print(' Sing'.lstrip())
```

```
Sing
```

lstrip()

Useful for removing whitespace on the left or the beginning of a string.

3

```
print(' Sing '.strip())
```

```
Sing
```

strip()

Will remove whitespace at the beginning or end of the string.

4

```
say = 'Sing&Dance'  
print(say.strip('Dance'))
```

```
Sing&
```

You can also specify which characters or sections you want to remove.

Splitting and Concatenating String



- `join()` The method used to concatenate a number of string. The string with the `join()` operation will be inserted between the string in the parameter.

```
print(' '.join(['Purple', 'Heart', '?']))
```

```
Purple Heart ?
```

```
print('007'.join(['Purple', 'Heart', '?']))
```

```
Purple007Heart007?
```

Splitting and Concatenating String



- `split()` Method that splits substrings based on a specified delimiter (default is whitespace, tab, or newline).

```
print('Purple Heart'.split())  
['Purple', 'Heart']  
  
# You can change the split(delimiter) parameter, as in the following example :  
print('Purple007Heart007:').split('007'))
```

You can also use this `split()` method to split each line in a string multi-line.

```
print('''Hi  
i like purple heart  
but i also like white heart  
i don't like black hearts.''' .split('\n'))  
['Hi', 'i like purple heart', 'but i also like white heart', "i don't like black hearts."]
```

REPLACING STRING ELEMENTS

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

```
word = "Let's learn data science track"  
print(word.replace("data science track", "python"))
```

```
Let's learn python
```

```
word = "Let's learn data science, because it is the main material in data science"  
print(word.replace("data science", "python", 1))
```

```
Let's learn python, because it is the main material in data science
```

Note : The `replace` function is case sensitive, meaning that all letters and spaces must be the same.

Optional : The third parameter in `replace` can be specified the number of substrings to be replaced.

STRING CHECK

```
[14] word = 'DATA SCIENCE'  
word.isupper()
```

True

Example that returns False :

```
word = 'Data Science'  
word.isupper()
```

False

ISUPPER()

This function will return True if all the letters in the string is uppercase, and will return False if there is only one lowercase letter in in that string.

```
term = 'data science'  
term.islower()
```

True

```
term = 'Data Science'  
term.islower()
```

False

ISLOWER()

This function is opposite of isupper() method, this method will return value True if all letters in the string are lowercase, and will return the value False if there is only one uppercase letter in the string.

```
'cat'.isalpha()
```

True

ISALPHA()

This method will return True if all characters in string are letters of the alphabet. Otherwise it will return False.

```
'rabbit007'.isalnum()
```

True

ISALNUM()

This method will return True if the character in the string is alphanumeric, only letters or only numbers or both. If not then will returns False value.

STRING CHECK

```
'040102'.isdecimal()
```

```
True
```

ISDECIMAL()

This method will return True if the characters in the string are only contains numeric, otherwise it returns False.

```
' '.isspace()
```

```
True
```

ISSPACE()

This method will return True if the string contains only whitespace, such as space, tab, newline or other whitespace characters. If not then will return False.

```
'My Cat'.istitle()
```

```
True
```

ISTITLE()

This method will return True if the string contains capital letters on each word and then followed by a lowercase letter, otherwise it will return the value false.

Text Align Right/Left/Center



RJUST() / LJUST()

Used to make the text right and left.

```
'Purple'.rjust(7)
```

```
'  Purple'
```

```
'Heart'.ljust(7)
```

```
'Heart  '
```

Text Align Right/Left/Center



In addition to spaces we can also add other parameters to the `rjust()` or `ljust()` functions such as following :

```
'White'.ljust(13, '?')  
'White????????'
```

CENTER()

This method will make the text centered.

```
'Heart'.center(7)  
  
#same as rjust() / ljust() we can add characters to function parameters center()  
  
' Heart '
```

Formatting String



ZFILL()

- A METHOD THAT CAN ADD A NUMERIC VALUE OF 0 TO THE LEFT OF A NUMBER OR STRING USING THE ZFILL() METHOD
- THE USE OF THE 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.

```
[ ] #In this example, wi will make id with zfill
id=input("Input 2 Number ID: ")
print("Your Passcode: ",str(id).zfill(7))
```

```
Input 2 Number ID: 14
Your Passcode:  0000014
```

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)

+ Code + Text

```
[ ] text="""Data are individual facts, statistics, or items of information,
often numeric.[1] In a more technical sense, data are a set of values of qualitative
or quantitative variables about one or more persons or objects,[1] while a datum
(singular of data) is a single value of a single variable."""
len(text)
```

```
290
```

```
[ ] print("Align Right :")
text.rjust(295)
```

```
Align Right :
```

```
'      Data are individual facts, statistics, or items of information, \noften numeric.[1] In a more technical sense, data are a
quantitative variables about one or more persons or objects,[1] while a datum \n(singular of data) is a single value of a single
```


ljust()

- Used for left-aligned text
- 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)

```
text="""Data are individual facts, statistics, or items of information,
often numeric.[1] In a more technical sense, data are a set of values of qualitative
or quantitative variables about one or more persons or objects,[1] while a datum
(singular of data) is a single value of a single variable."""
len(text)
```

290

```
[ ] print("Align Left :")
text.ljust(295)
```

```
Align Left :
'Data are individual facts, statistics, or items of information, \noften numeric.
itative variables about one or more persons or objects,[1] while a datum \n(singu
```

center()

- The center method will make the text centered
- 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)

```
[ ] print("Align Center :")
text.center(295)
```

```
Align Center :
'    Data are individual facts, statistics,
antitative variables about one or more per
```

String Literals

GENERALLY STRINGS ARE WRITTEN EASILY IN PYTHON, ENCLOSED IN SINGLE QUOTES. HOWEVER, UNDER CERTAIN CONDITIONS, IT TAKES A SINGLE QUOTE IN THE MIDDLE OF THE STRING.

THE ESCAPE CHARACTER I.E. BACKSLASH ALLOWS YOU TO ENTER THE 'AND' CHARACTER IN THE STRING PART.

```
[ ] "In other words, wisdom refers to the practical application of a person's knowledge in those circumstances where good may result. Thus wisdom complements and completes the series "data", "information" and "knowledge" of increasingly abstract concepts.'
```

Raw string prints the string corresponding to any given input or text. for raw string implementation, insert the letter r

```
[ ] print("""\tIn other words, wisdom refers to the practical application of a person's knowledge in those circumstances where good may result. Thus wisdom complements and completes the series \"data\", \"information\" and \"knowledge\" of increasingly abstract concepts.""")
```

```
\tIn other words, wisdom refers to the practical application of a person's knowledge in those circumstances where good may result. Thus wisdom complements and completes the series "data", "information" and "knowledge" of increasingly abstract concepts.
```

```
[ ] print(r"""\tIn other words, wisdom refers to the practical application of a person's knowledge in those circumstances where good may result. Thus wisdom complements and completes the series \"data\", \"information\" and \"knowledge\" of increasingly abstract concepts.""")
```

```
\tIn other words, wisdom refers to the practical application of a person's knowledge in those circumstances where good may result. Thus wisdom complements and completes the series \"data\", \"information\" and \"knowledge\" of increasingly abstract concepts.
```

Python Operators

OPERATORS ARE USED TO PERFORM OPERATIONS ON VARIABLES AND VALUES.



Python Arithmetic Operators

ARITHMETIC OPERATORS ARE USED WITH NUMERIC VALUES TO PERFORM COMMON MATHEMATICAL OPERATIONS.

```
[1] x = 10
    y = 5
    z = 7

[2] # Addition (+)
    print(x+y)

15

[3] # Subtraction (-)
    print(x-y)

5
```

```
[4] # Multiplication (*)
    print(x*y)

50

[5] # Division (/)
    print(x/y)

2.0

[6] # Modulus (%) --- remaining quotient
    print(z%y)

2

[7] # Exponentiation (**)
    print(x**y)

100000
```

```
# Floor Division (//) --- rounds the result down to the nearest whole number
print(z//x)

0
```

PYTHON ASSIGNMENT OPERATORS

ASSIGNMENT OPERATORS ARE USED TO ASSIGN VALUES TO VARIABLES. ALSO WE CAN USING OUR VARIABLE AS AN OPERAND.

```
[9] as_operators = 7

generally we will write our syntax like below

[10] as_operators = as_operators + 3
     print(as_operators)

10

you know what, we can use this method. And this method can work in every Arithmetic Operators

[11] as_operators += 3
     print(as_operators)

13
```

```
[12] as_operators *= 3
     print(as_operators)

39
```

PYTHON COMPARISON OPERATORS

COMPARISON OPERATORS ARE USED TO COMPARE TWO VALUES. THIS OPERATORS WILL RETURN TRUE OR FALSE

- EQUAL (==)

- NOT EQUAL (!=)

- LESS THAN (<) AND
LESS THAN OR
EQUAL TO (<=)

- GREATER THAN (>)
AND GREATER
THAN OR EQUAL TO
(>=)



```
[32] com_operators = 7
    com_operators2 = 3

equal (==)

[33] com_operators2 == 7
False

not equal (!=)

[34] com_operators2 != 7
True
```

```
less than (<) and less than or equal to (<= >)

[35] com_operators2 < com_operators
True

[36] com_operators <= com_operators2
False

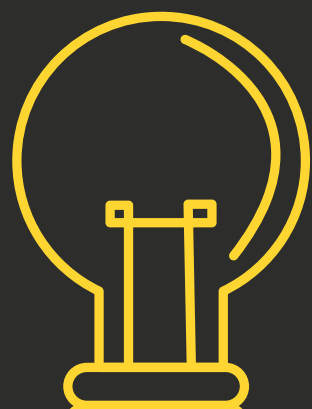
greater than (>) and greater than or equal to (>= >)

[37] com_operators > com_operators2
True

[38] com_operators2 >= com_operators
False
```

```
we can also use more than 2 operand.

[39] com_operators > com_operators < 10
False
```



Python Logical Operators

Logical operators are used to combine conditional statements. also we known as BOOLEAN OPERATORS. So the return will be True or False

- and (Return True if both statements are true)
- or (Return True if one of statements is true)
- not (Reverse the result)

✓
0s [21] x_log = 3
y_log = 7

✓
0s [22] x_log < 10 and y_log < 10

True

✓
0s [23] x_log < 5 or x < 10

True

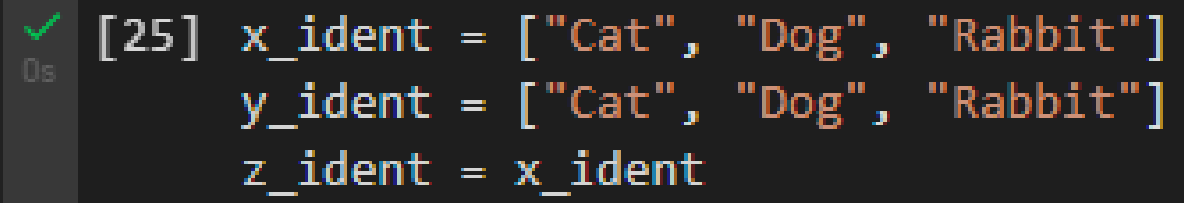
✓
0s [24] not(x_log < 10 and y_log < 10)

False

Python Identity Operators

IDENTITY OPERATORS ARE USED TO COMPARE THE OBJECTS, NOT IF THEY ARE EQUAL, BUT IF THEY ARE ACTUALLY THE SAME OBJECT.

- IS (RETURNS TRUE IF BOTH VARIABLES ARE THE SAME OBJECT)
- IS NOT (RETURNS TRUE IF BOTH VARIABLES ARE NOT THE SAME OBJECT)



```
✓ [25] x_ident = ["Cat", "Dog", "Rabbit"]  
0s y_ident = ["Cat", "Dog", "Rabbit"]  
z_ident = x_ident
```

```
✓ [26] x_ident is y_ident  
0s
```

```
False
```

```
✓ [27] x_ident is not z_ident  
0s
```

```
False
```

Python Membership Operators

MEMBERSHIP OPERATORS ARE USED TO TEST IF A SEQUENCE IS PRESENTED IN AN OBJECT.

- IN
- NOT IN

```
✓ [28] phonetic = ["Alpha", "Beta", "Charlie", "Delta"]  
0s
```

```
✓ [29] print("Charlie" in phonetic)  
0s
```

True

```
✓ [30] print("Delta" not in phonetic)  
0s
```

False

```
✓ [31] print("Gamma" not in phonetic)  
0s
```

True

OPERATION in list, set, and String

LEN()

LEN() WILL RETURNS THE LENGTH OF AN OBJECT



```
[1] # List
    a_list = [3,5,5,7,7,7]

[2] print(a_list)

    [3, 5, 5, 7, 7, 7]

[3] len(a_list)

    6
```

```
[4] # Set
    a_set = {3,5,5,7,7,7}

[5] print(a_set)

    {3, 5, 7}

[6] len(a_set)

    3

[7] a_string = "Python Operation"

[8] len(a_string)

    16
```



MIN() AND MAX()

MIN() RETURN THE LOWEST VALUE, AND MAX RETURN THE HIGHEST VALUE

```
[9] min(a_list)
3

[10] max(a_list)
7
```

COUNT()

DATA OCCURANCE

```
[11] a_list.count(7)
3

[14] a_set.count(7) #can't use count in set

-----
AttributeError                                Traceback (most recent call last)
<ipython-input-14-e195bd55cb52> in <module>()
----> 1 a_set.count(7) #can't use count in set

AttributeError: 'set' object has no attribute 'count'

SEARCH STACK OVERFLOW
```


MERGING (+) AND REPLICATION (*)



```

Merging (+) and replication (*)

[15] a_merging = [1, 9, 9, 1]
     s_merging = ['p', 'y', 't', 'h', 'o', 'n']

[16] # Merging
     s_merging + a_merging

['p', 'y', 't', 'h', 'o', 'n', 1, 9, 9, 1]

[17] # Replication
     s_merging * 2

['p', 'y', 't', 'h', 'o', 'n', 'p', 'y', 't', 'h', 'o', 'n']

```

```

we can using replication for List

[18] initi = [2,7,11]*3

[19] print(initi)

[2, 7, 11, 2, 7, 11, 2, 7, 11]

```

RANGE()
USUALLY USED FOR LOOP PROCESSES. WE CAN USE THE RANGE() IN 3 WAYS (PARAMETERS). AND IT WILL LOOK SIMILAR TO SLICING DATA ON A LIST

```

1 Parameter --- :: range(n)

from 0 to n

for i in range(3):
    print(i*2)

0
2

```

```

2 Parameters --- range(x,n)

from x (inclusive) to n (exclusive)

[33] for i in range (2, 11):
      print(i)

2
3
4
5
6
7
8

```

```

3 Parameters --- :: range(x,n,y)

from x to n with y step

for i in range(5,10,2):
    print(i)

5

```

MULTIPLE VALUE USING LIST [USING NEW VARIABLE TO ASSIGN A LIST INDEX

```
[35] ponethic = ["Alpha", "Tokyo", "Zeta"]  
[36] print(ponethic)  
      ['Alpha', 'Tokyo', 'Zeta']  
[37] new_pone = ponethic[2]  
[38] print(new_pone)  
      Zeta
```

SIMPLE MULTIPLE LIST DECLARATION

```
[41] # List  
      alph_list = ["P", "Y", "T", "H", "O", "N"]  
[42] p, y, t, h, o, n = alph_list  
[43] print(o)  
      0  
[44] # Tuple  
      alph_tuple = ("P", "Y", "T", "H", "O", "N")  
[45] pt, yt, tt, ht, ot, nt = alph_tuple  
      print(yt)  
      Y
```

SORT()

SORTING A VALUE FROM NUMERICAL OR STRING

```
✓ [47] # Numerical
0s val = [20, 10, 2, 33, 7]

✓ [48] val.sort()
0s

✓ [49] print(val)
0s
[2, 7, 10, 20, 33]

✓ [50] # String
0s val_str = ["Mike", "India", "Oscar"]

✓ [51] val_str.sort()
0s

✓ [52] print(val_str)
0s
['India', 'Mike', 'Oscar']
```

```
reverse the sorting to become a Descending

✓ [53] val_str.sort(reverse=True)
0s

✓ [54] print(val_str)
0s
['Oscar', 'Mike', 'India']

Can't sorting a numerical together with string in one List !!

sorting String — Uppercase > Lowercase

✓ [55] uplow = ["Quebec", "tango", "india", "Oscar", "alpha"]
0s

✓ [56] uplow.sort()
0s

✓ [57] print(uplow)
0s
```

```
{x} ✓ [56] uplow.sort()
0s

[57] print(uplow)
0s
['Oscar', 'Quebec', 'alpha', 'india', 'tango']

we can generalize the string by making it lowercase

✓ [58] uplow.sort(key=str.lower)
0s

✓ [59] print(uplow)
0s
['alpha', 'india', 'Oscar', 'Quebec', 'tango']
```




6. PYTHON TYPE DATA CONVERSION

A CONDITIONAL EXPRESSION IN PYTHON IS AN EXPRESSION (IN OTHER WORDS, A PIECE OF CODE THAT EVALUATES TO A RESULT) WHOSE VALUE DEPENDS ON A CONDITION.

IN THE REAL WORLD, WE COMMONLY MUST EVALUATE INFORMATION AROUND US AND THEN CHOOSE ONE COURSE OF ACTION OR ANOTHER BASED ON WHAT WE OBSERVE:

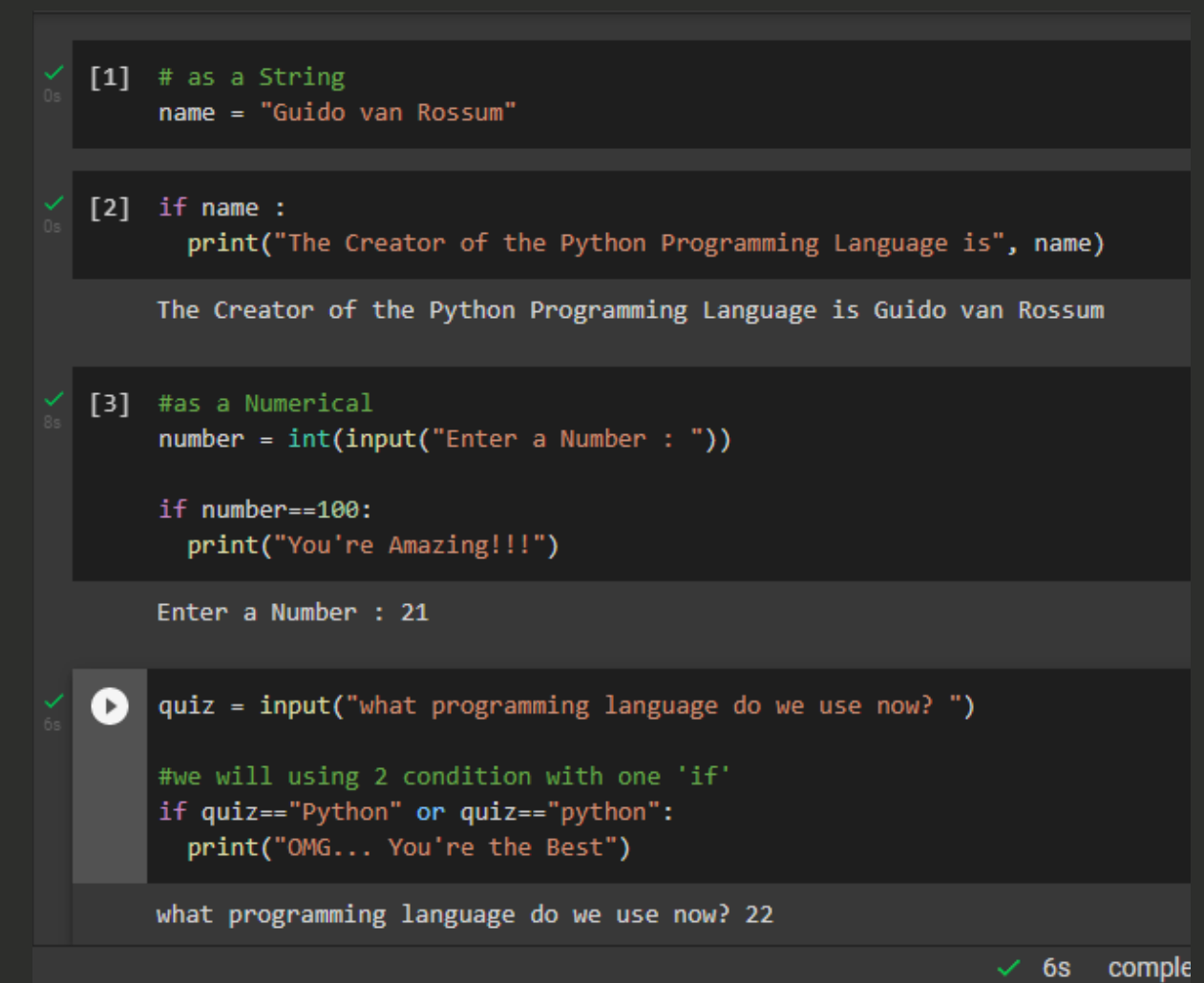
If the weather is nice, then I'll mow the lawn. (It's implied that if the weather isn't nice, then I won't mow the lawn.)

if Statement

There is only one condition or expression, where if the condition is True , it will return the specified statement or value or output.

In its simplest form, it looks like this:

```
if <expr>:  
    <statement>
```



```
[1] # as a String  
name = "Guido van Rossum"  
  
[2] if name :  
    print("The Creator of the Python Programming Language is", name)  
The Creator of the Python Programming Language is Guido van Rossum  
  
[3] #as a Numerical  
number = int(input("Enter a Number : "))  
  
if number==100:  
    print("You're Amazing!!!")  
Enter a Number : 21  
  
[4] quiz = input("what programming language do we use now? ")  
  
#we will using 2 condition with one 'if'  
if quiz=="Python" or quiz=="python":  
    print("OMG... You're the Best")  
what programming language do we use now? 22
```

✓ 6s comple

else Statement

There are two conditions, where if the first condition is True, it will return the specified value or output in that condition. and if the value is False it will output another value or find another alternative.

In its simplest form, it looks like this:

```
if <expr>:  
    <statement>  
else:  
    <statement2>
```

```
exam = int(input("What is the result of '100x0+1'? : "))  
  
if exam==1:  
    print("Congratulations !!, You're best of the best")  
else :  
    print("Sorry, Please Try Again :( ")
```

```
What is the result of '100x0+1'? : 1  
Congratulations !!, You're best of the best
```

else if / elif Statement

There are more than two conditions or alternatives, the statement will be executed if one of the conditions is True. If it does not get a True value then it will go to the last option which is in else.

In its simplest form, it looks like this:

```
if <expr>:  
    <statement>  
elif <expr2>:  
    <statement2>  
elif <expr3>:  
    <statement3>  
.....  
else:  
    <staement_n>
```

SIMPLE LOGIN SESSION USING IF-ELIF-ELSE STATEMENT

we can implement
conditional expressions for
the login process of an
application.

```
name = input("Enter Your Name : ")
username = input("Username : ")
password = input("Password : ")

print("\n")

if (username == "admin") and (password == "admin"):
    print("Hello...", name, "Welcome Home !")
elif (username != "admin") and (password == "admin"):
    print("Username is wrong")
elif (username == "admin") and (password != "admin"):
    print("Password is wrong")
else:
    print("Username & Password Wrong")
```

```
Enter Your Name : admin
Username : admin
Password : admin

Hello... admin Welcome Home !
```