MyEduSolve

Kampus Merdeka
INDONESIA JAYA

Beginner Programing for Simple Program - Python -

Python

Python is an interpreted, high-level general purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with significant use of whitespace.

Replace String

The replace() method returns a new string with some or all matches of a pattern replaced by a replacement. The pattern can be a string or a RegExp, and the replacement can be a string or a function to be called for each match. If pattern is a string, only the first occurrence will be replaced.

Python String Method

startswith()

The startswith() method returns True if a string starts with the specified prefix(string). If not, it returns False.

endwith()

The endswith() method returns True if a string ends with the specified suffix. If not, it returns False.

String Check

isupper()

The string isupper() method returns whether or not all characters in a string are uppercased or not.

islower()

The islower() method returns True if all alphabets in a string are lowercase alphabets. If the string contains at least one uppercase alphabet, it returns False.

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

isalpha()

The isalpha() method returns True if all characters in the string are alphabets. If not, it 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.

isdecimal()

The isdecimal() method returns True if all characters in a string are decimal characters. If not, it returns False.

isspace()

The isspace() method returns True if there are only whitespace characters in the string. If not, it return False.

istitle()

The istitle() returns True if the string is a titlecased string. If not, it returns False.

formating string

zfill()

zfill() is an method to adds zeros (0) at the beginning of the string, until it reaches the specified length

ljust()

ljust() has the function of creating a left-aligned string and making the rest of the padding to the right according to the entered length and can include a fillchar char (default is a space)

center()

center() method will center align the string, using a specified character (space is default) as the fill character

rjust()

The rjust() method will right align the string, using a specified character (space is default) as the fill character

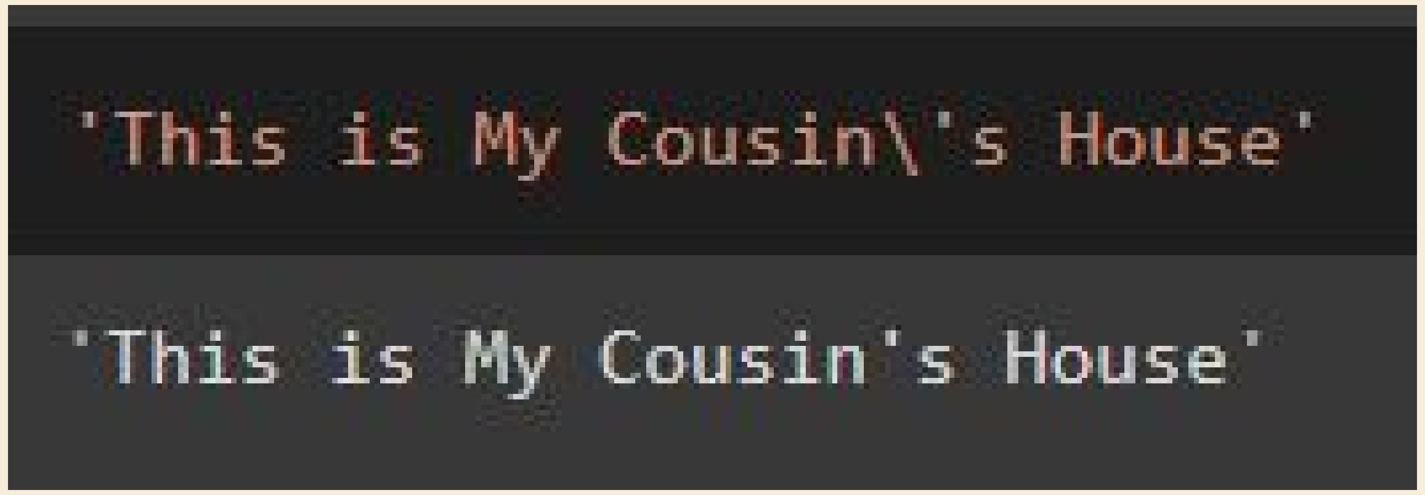
backslash character

\

We can use the backslash character () to escape characters that can break the string and make the syntax error.

raw

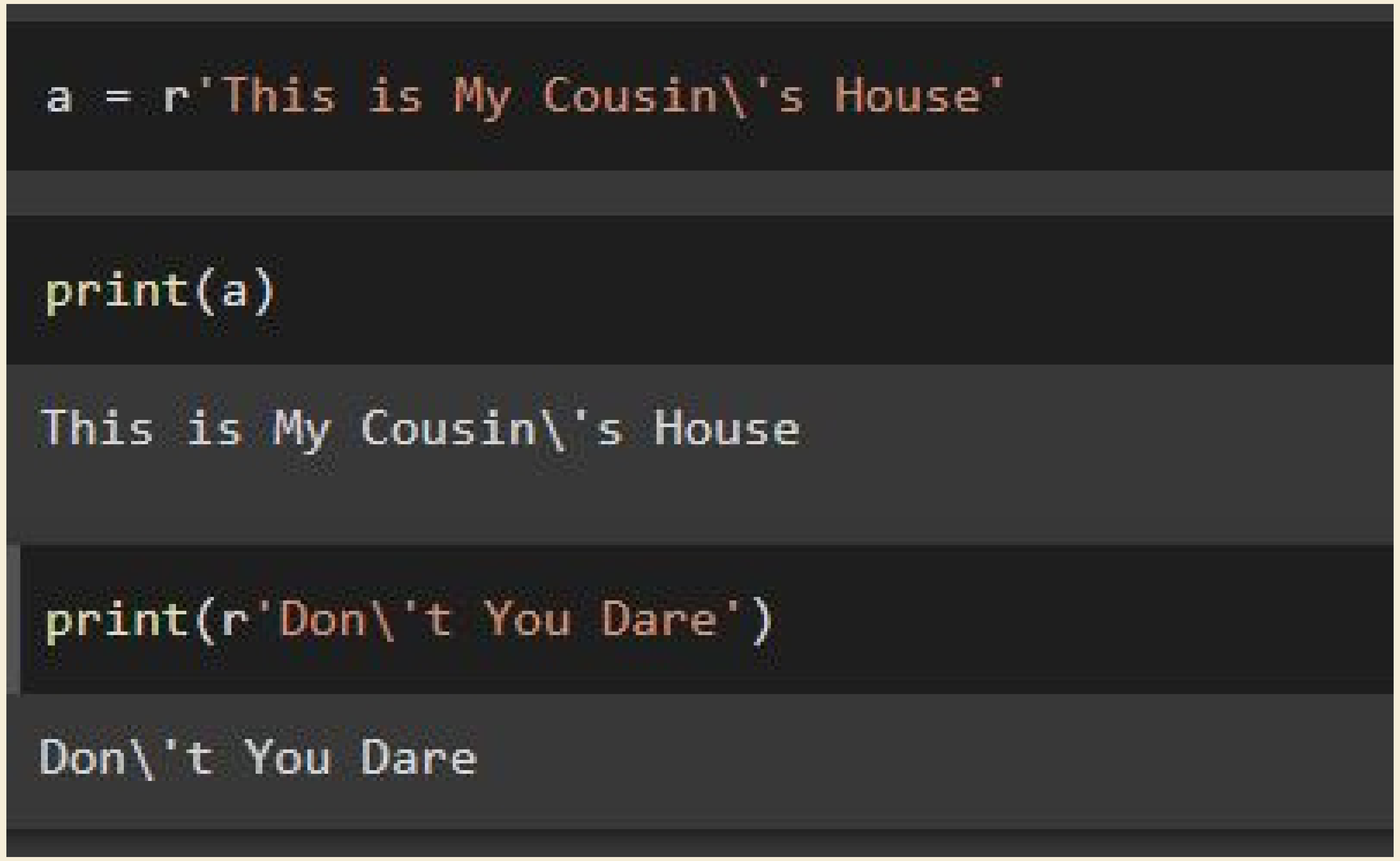
raw string will display the string according to what is input



```
'This is My Cousin\'s House'
```

```
'This is My Cousin's House'
```

backslash example



```
a = r'This is My Cousin\'s House'
```

```
print(a)
```

```
This is My Cousin\'s House
```

```
print(r'Don\'t You Dare')
```

```
Don\'t You Dare
```

raw example

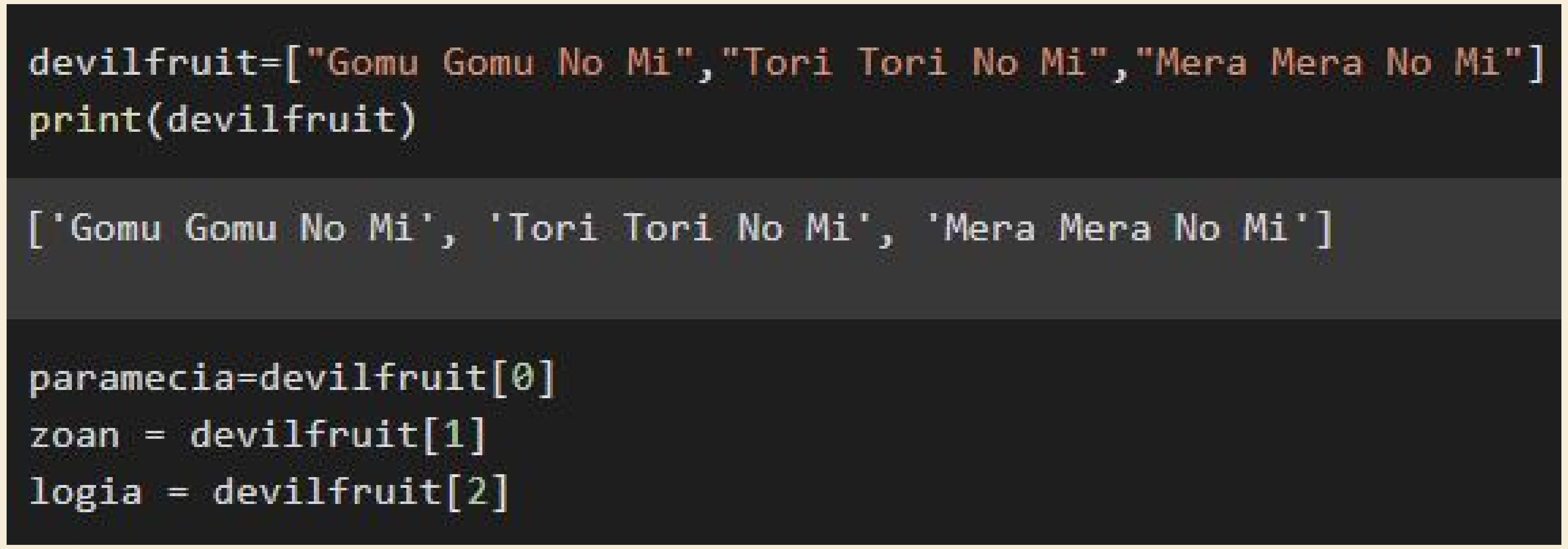
assign value to multiple variable

first way

assign value inside iterable data type to variable using index

second way

assign variable to iterable data type. note: need same number variable as number element iterable data type



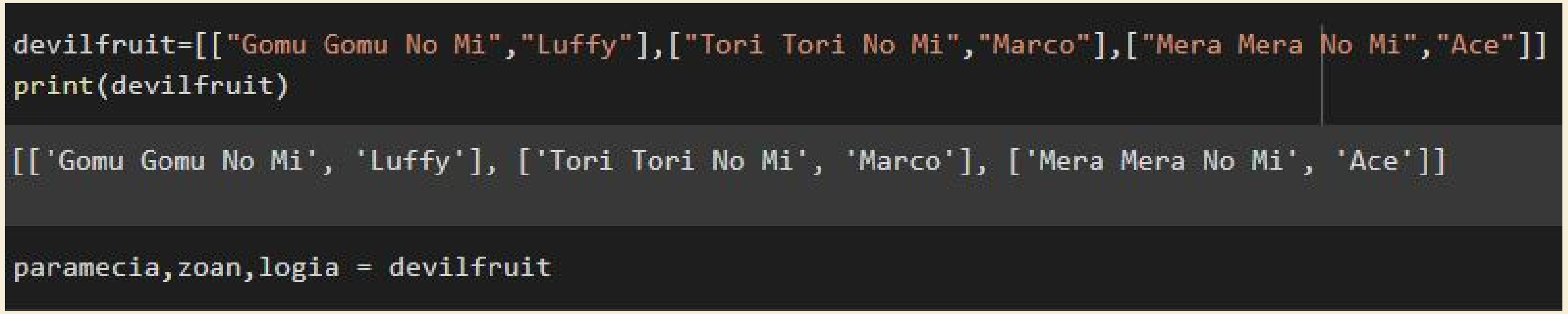
```
devilfruit=["Gomu Gomu No Mi","Tori Tori No Mi","Mera Mera No Mi"]
print(devilfruit)
```

```
['Gomu Gomu No Mi', 'Tori Tori No Mi', 'Mera Mera No Mi']
```

```
paramecia=devilfruit[0]
zoan = devilfruit[1]
logia = devilfruit[2]
```

first method

second method



```
devilfruit=[["Gomu Gomu No Mi","Luffy"],["Tori Tori No Mi","Marco"],["Mera Mera No Mi","Ace"]]
print(devilfruit)
```

```
[['Gomu Gomu No Mi', 'Luffy'], ['Tori Tori No Mi', 'Marco'], ['Mera Mera No Mi', 'Ace']]
```

```
paramecia,zoan,logia = devilfruit
```

input and output

input and output

input

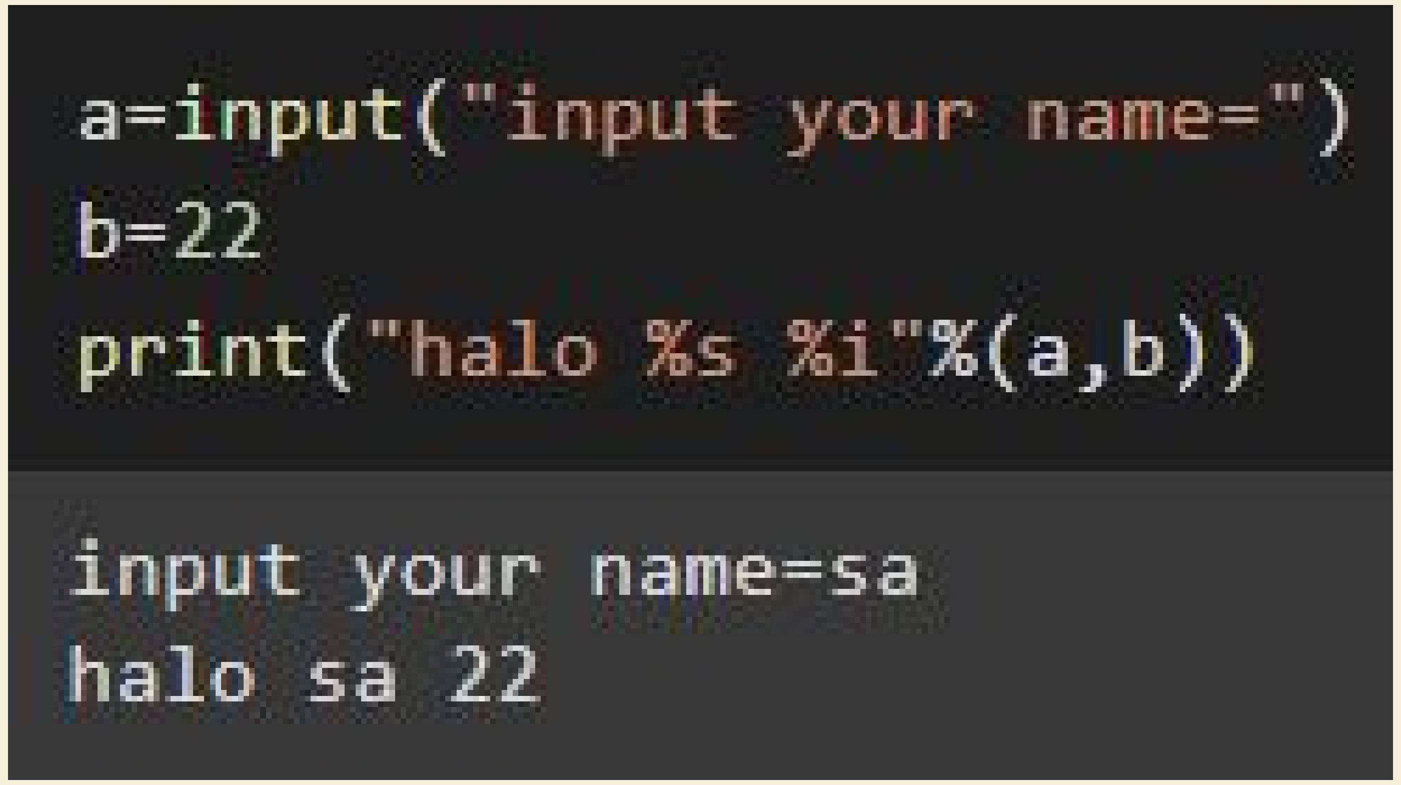
input() is one python build in function to reads a line from the input (usually from the user), converts the line into a string by removing the trailing newline, and returns it

output

the output function print() is used to display the result of the program on the screen after execution

4 ways to print data

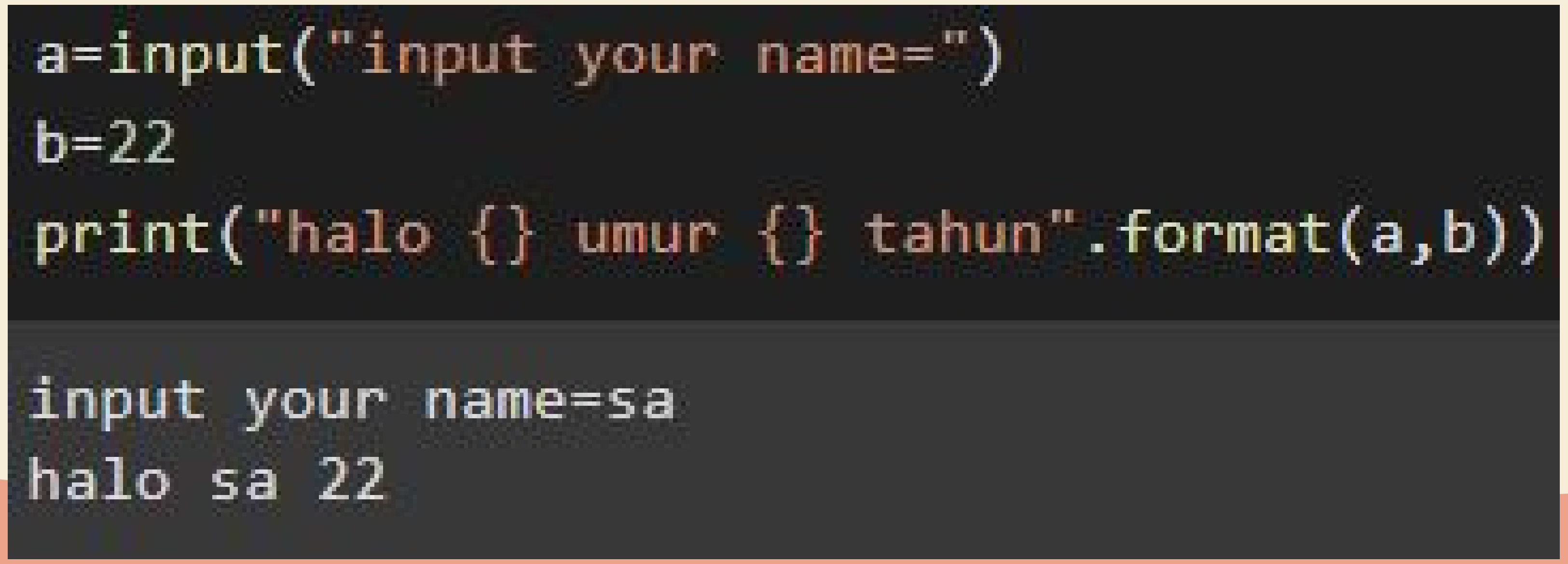
01



```
a=input("input your name=")
b=22
print("halo %s %i"%(a,b))
```

```
input your name=sa
halo sa 22
```

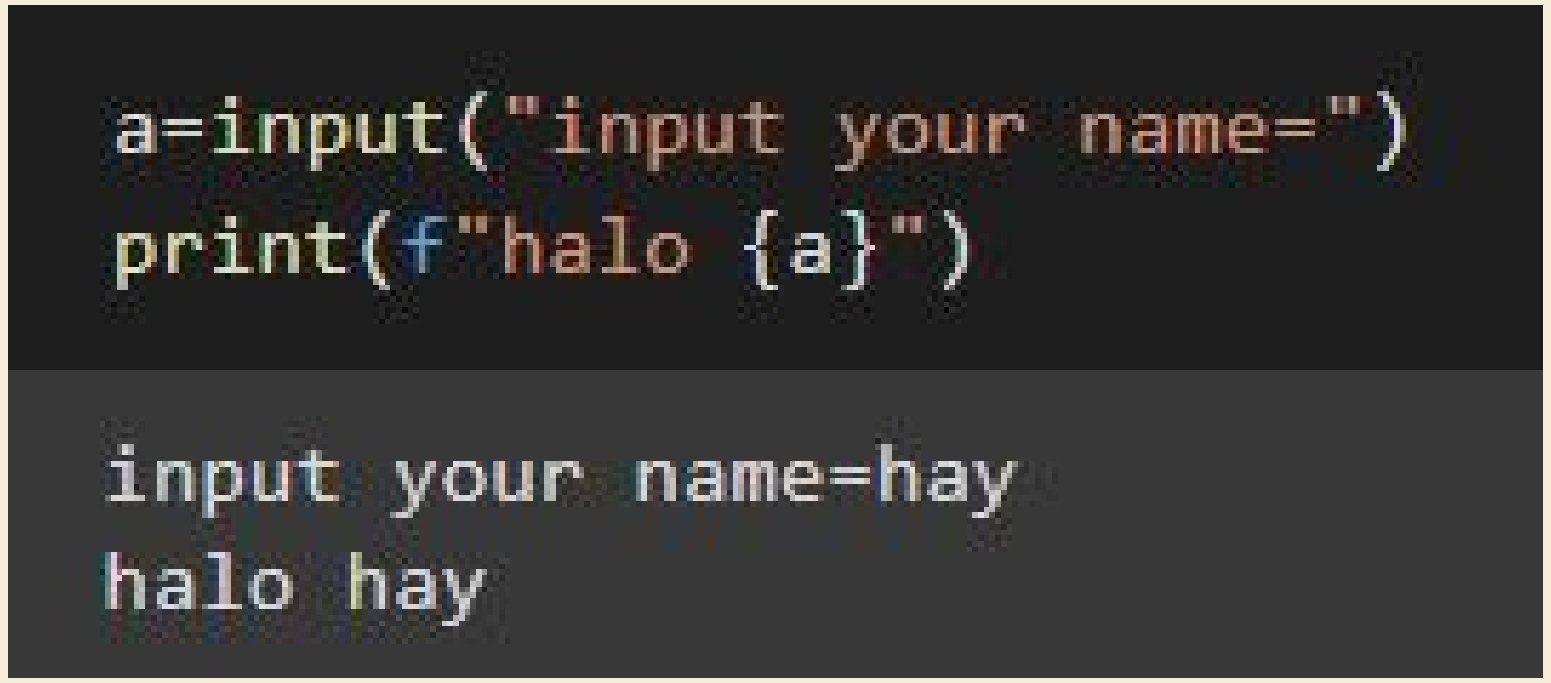
02



```
a=input("input your name=")
b=22
print("halo {} umur {} tahun".format(a,b))
```

```
input your name=sa
halo sa 22
```

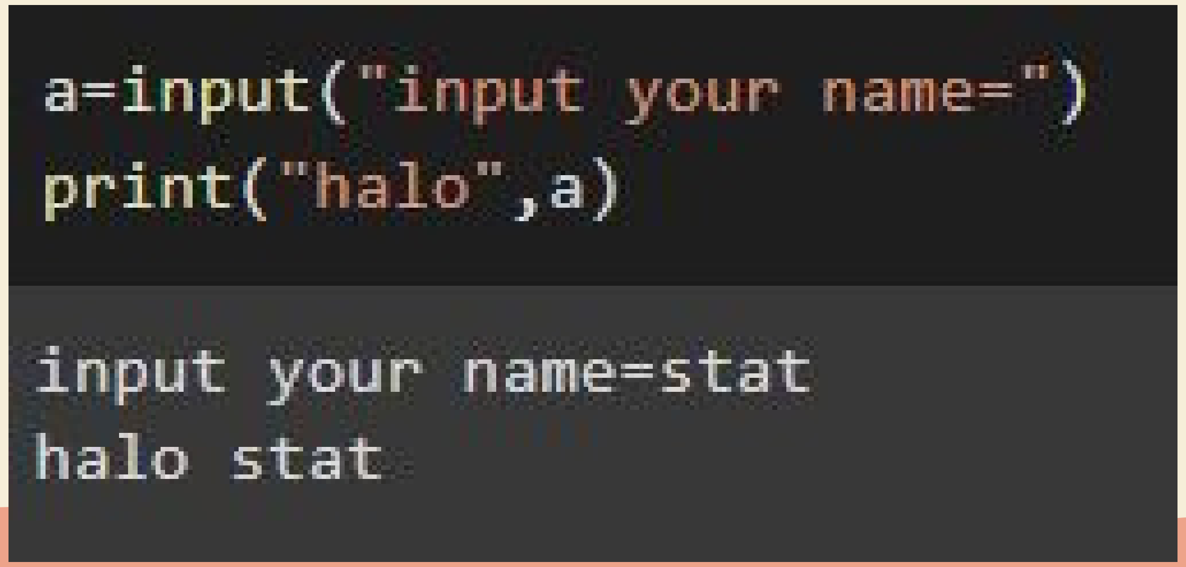
03



```
a=input("input your name=")
print(f"halo {a}")
```

```
input your name=hay
halo hay
```

04

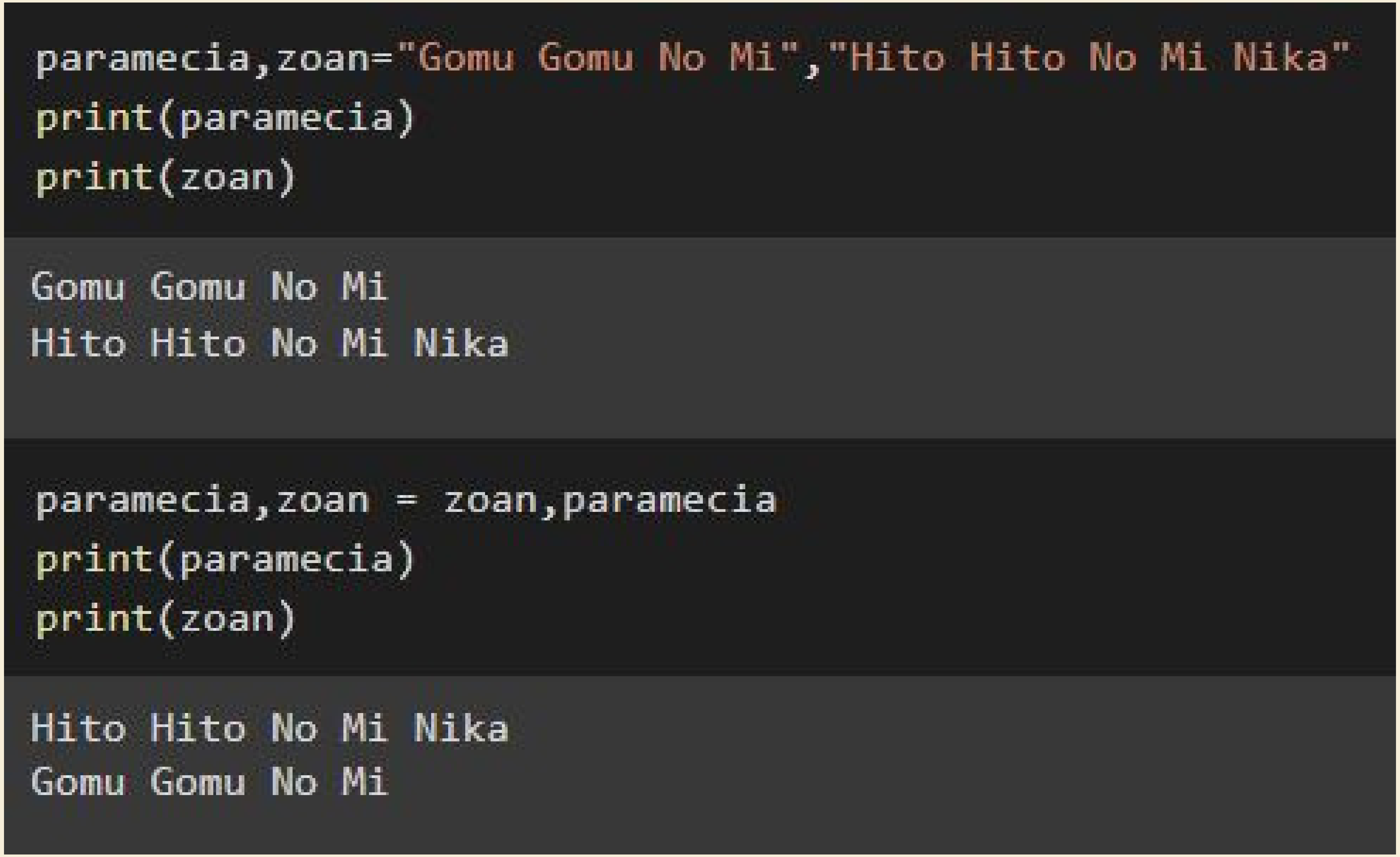


```
a=input("input your name=")
print("halo",a)
```

```
input your name=stat
halo stat
```

Swap Variable

swap variable is method to swap value between variable.



```python
paramecia,zoan="Gomu Gomu No Mi","Hito Hito No Mi Nika"
print(paramecia)
print(zoan)
```

```
Gomu Gomu No Mi
Hito Hito No Mi Nika
```

```python
paramecia,zoan = zoan,paramecia
print(paramecia)
print(zoan)
```

```
Hito Hito No Mi Nika
Gomu Gomu No Mi
```

conversion data type

- str
- int
- float
- complex
- bool

- bool
- list
- tuple
- dict
- set

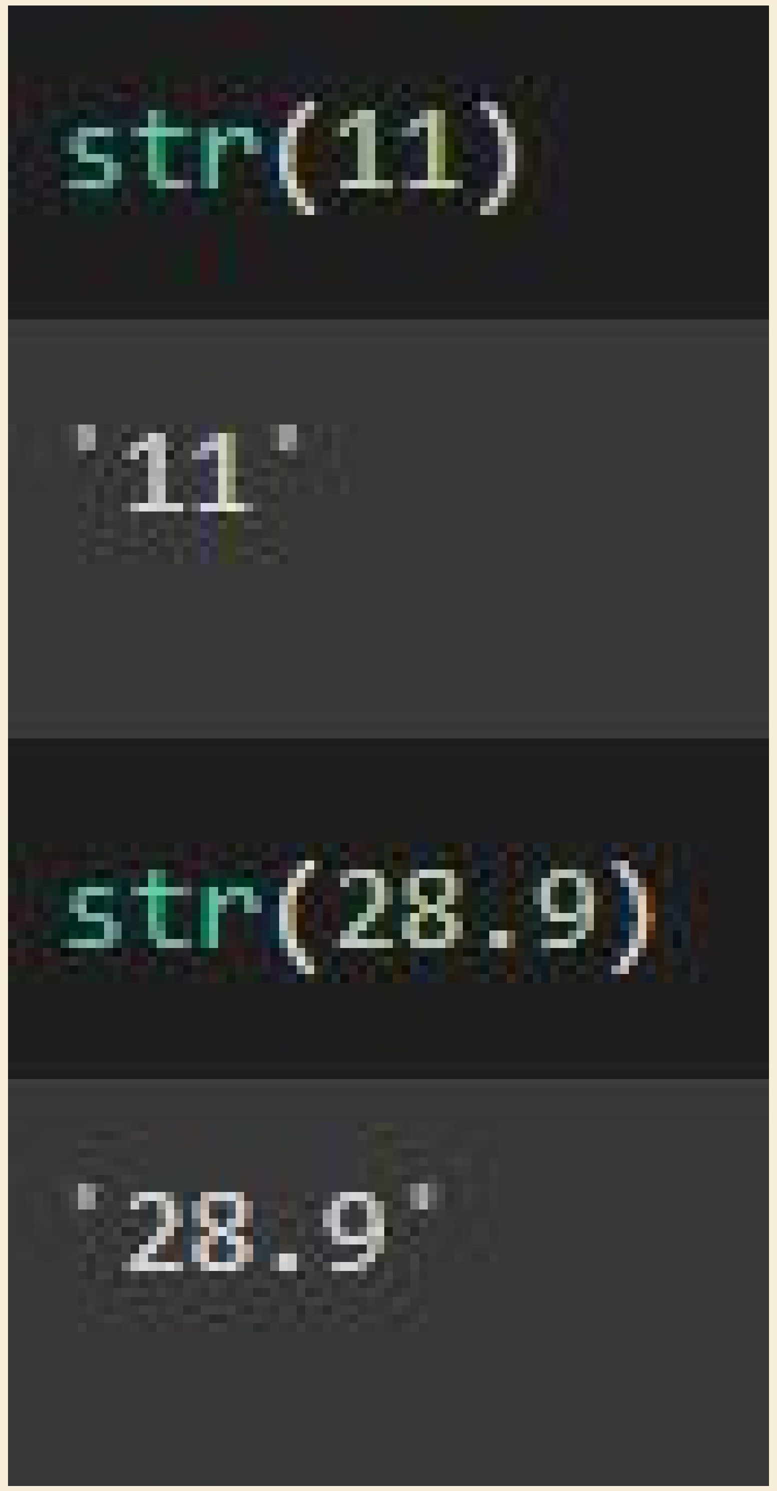
Conversion data type

It possible to change data type from one data type to another data type as long it fulfill destination data type requirment.

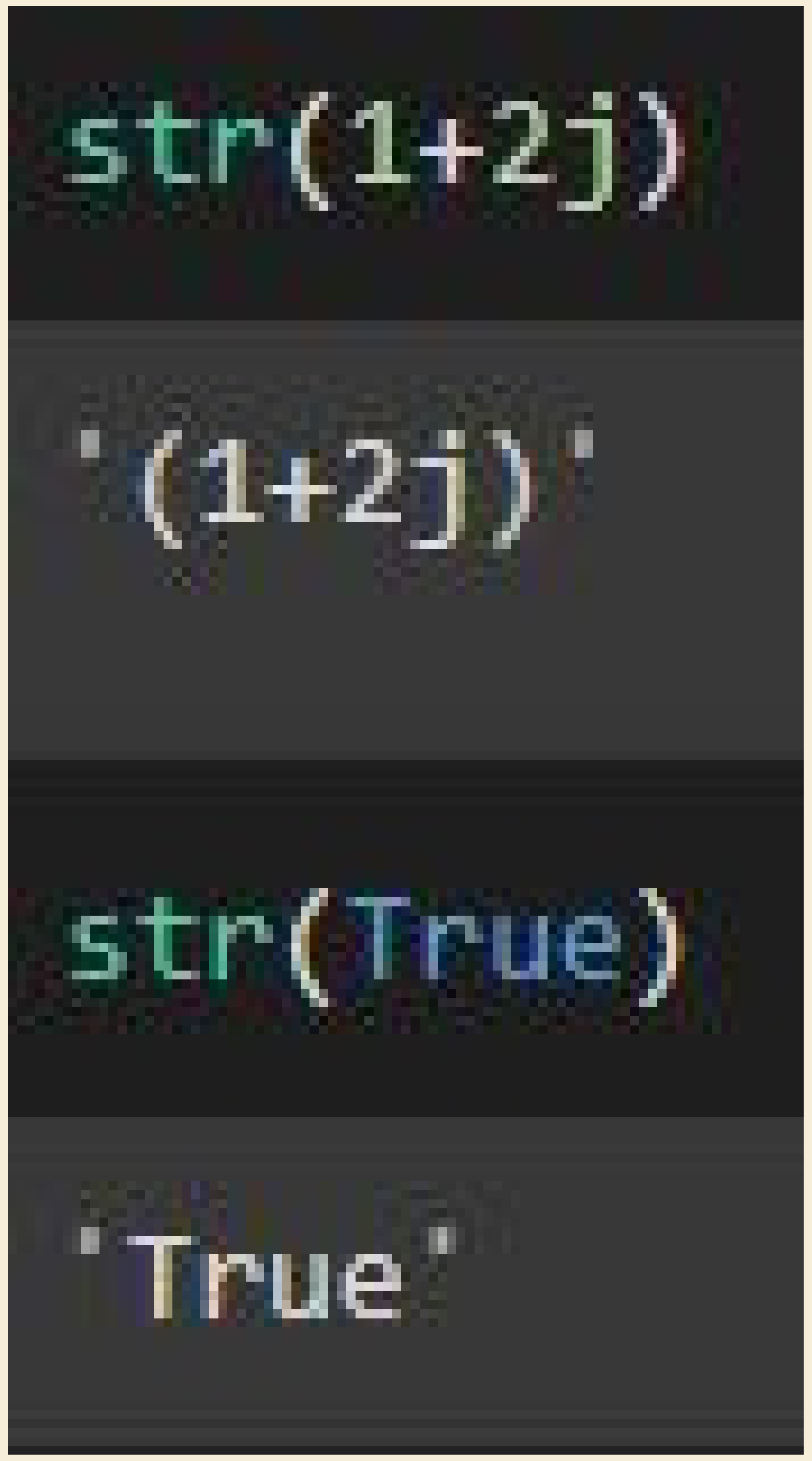
string conversion

it possible to change any kinds of data type to string. any character will categorize to string if got ' or " in the beginning and end characters. we can aslo use python build in function str() to change data type to string

Example:



```
str(11)
'11'
str(28.9)
'28.9'
```



```
str(1+2j)
'(1+2j)'
str(True)
'True'
```

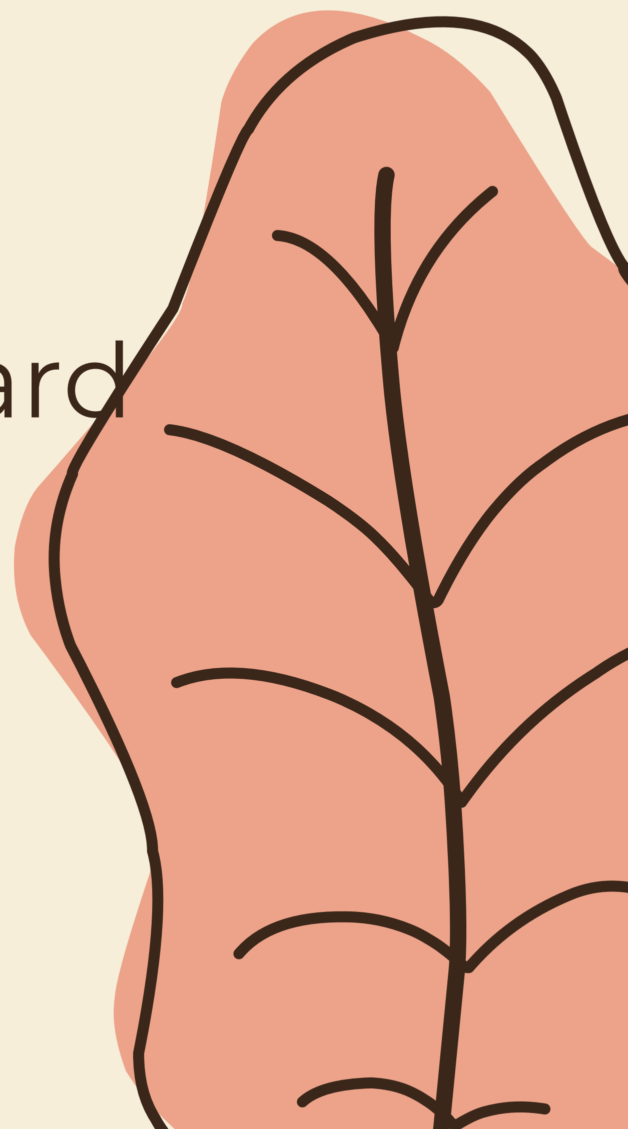
integer conversion

so, we can convert some main type data but not all can be integer using function int(). type data that eligble to convert is

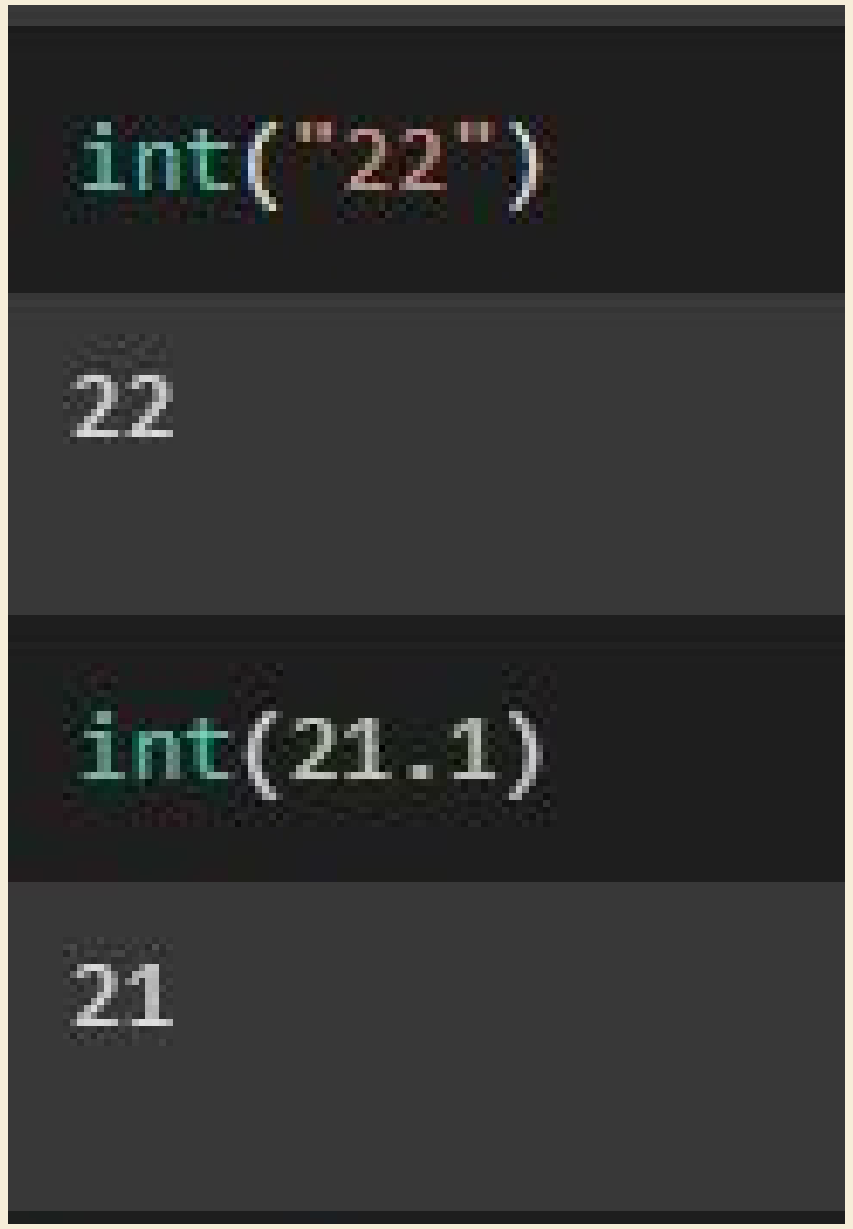
*   string if contain only integer part

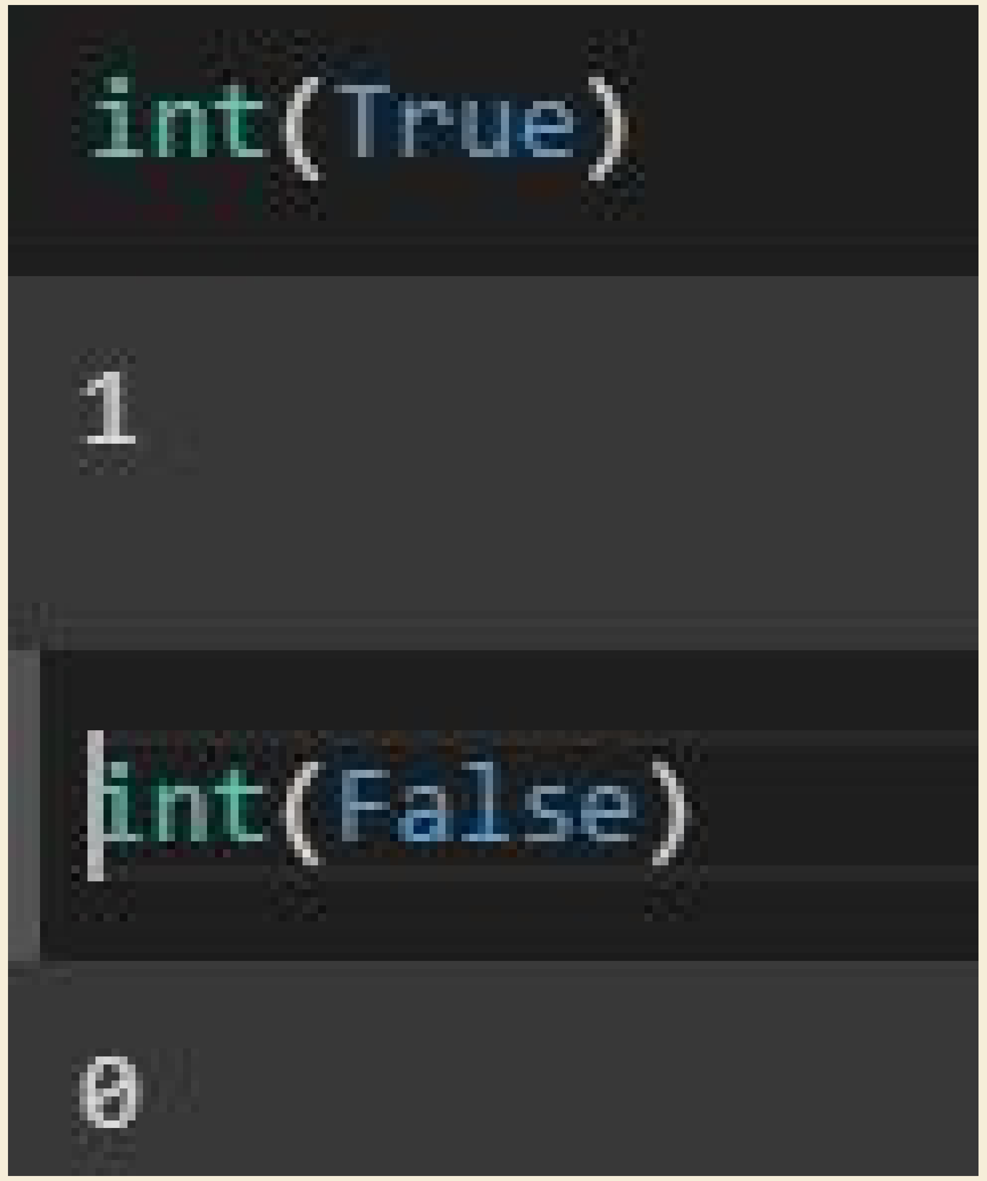
*   float with note number behind decimal will be discard not rounded

*   boolean wuth True is 1 ad False is 0



Example:


```
int("22")
```
```
22
```
```
int(21.1)
```
```
21
```


```
int(True)
```
```
1
```
```
int(False)
```
```
0
```

float conversion

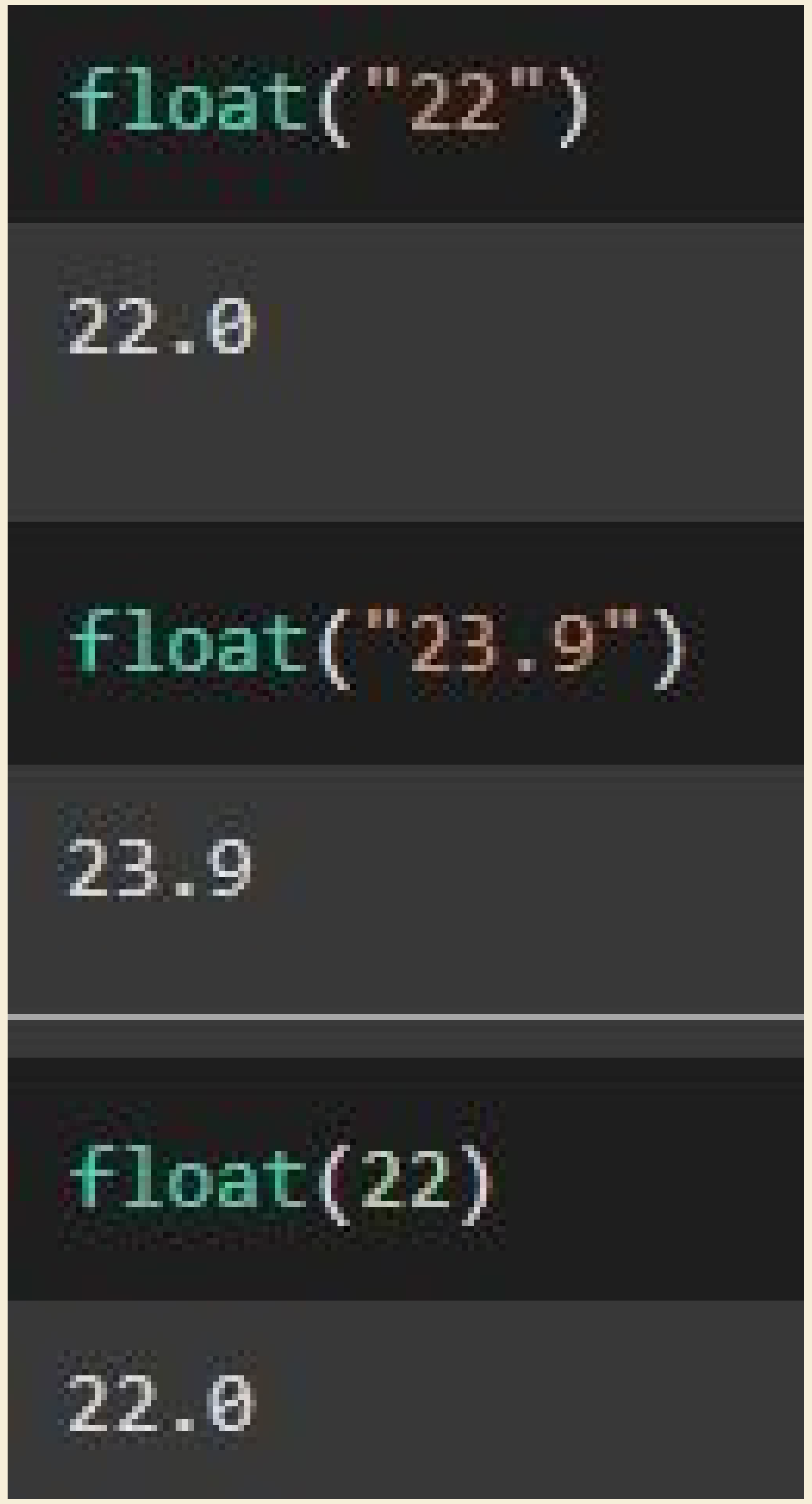
so, not all type data is eligble to convert as float. using function float().

* string can be convert if string contain only integer or float

* any integer can be convert to float

* boolean can be convert to float with True as 1.0 and False as 0.0

Example:



```
float("22")
```

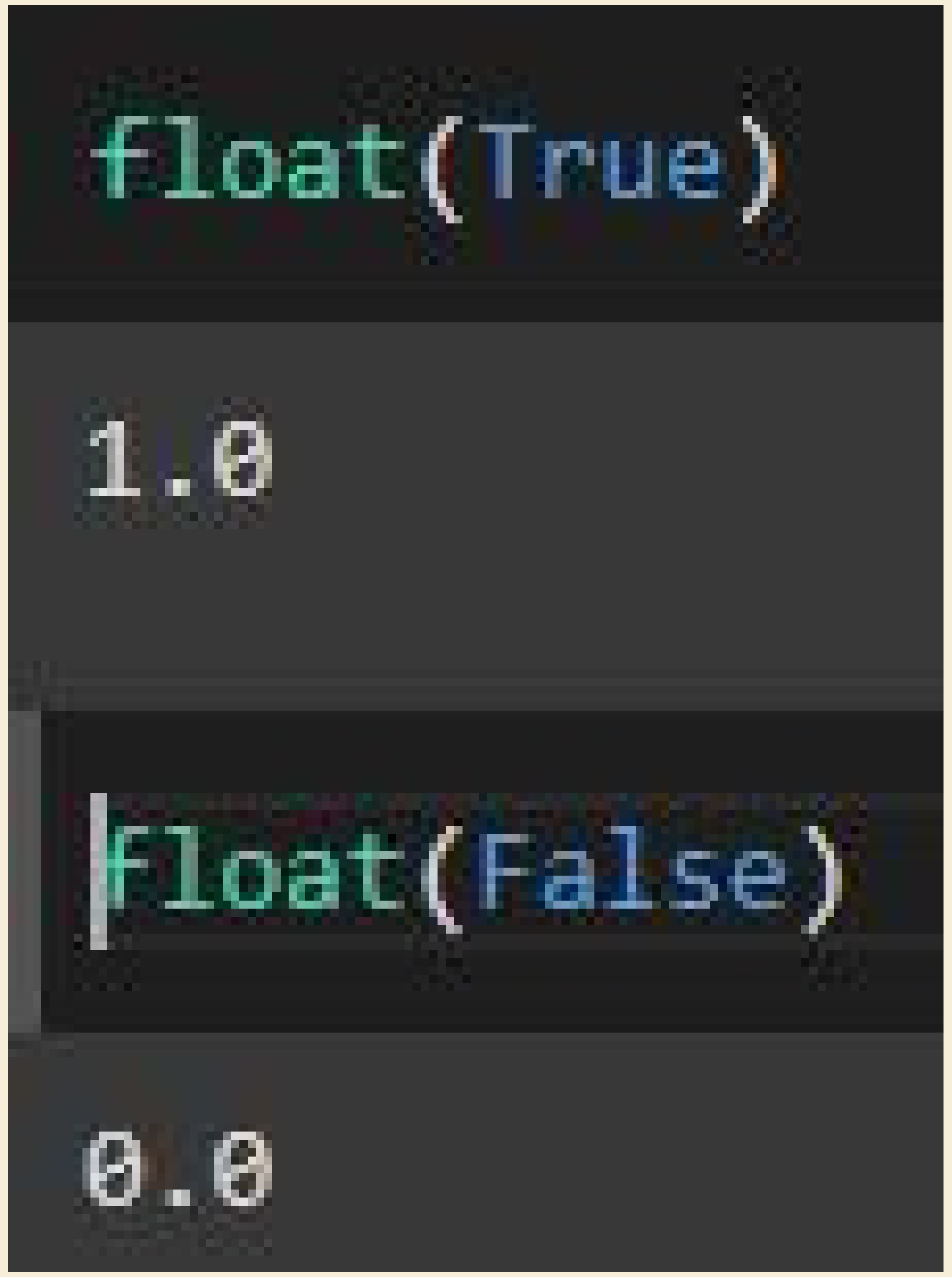
```
22.0
```

```
float("23.9")
```

```
23.9
```

```
float(22)
```

```
22.0
```



```
float(True)
```

```
1.0
```

```
float(False)
```

```
0.0
```

complex conversion

so, type data eligble to convert to complex:

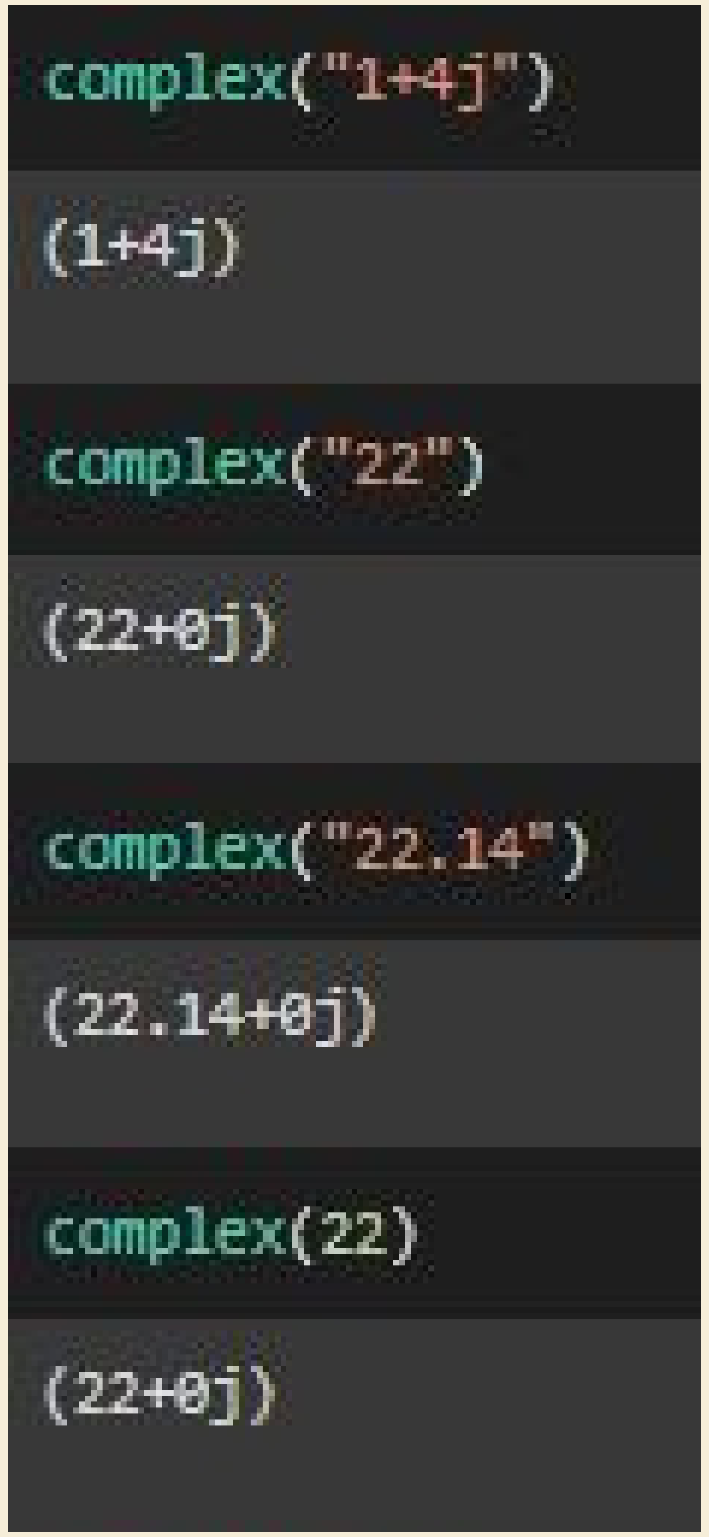
* string that contain only numeric type data

* float wiith imaginer part is 0j

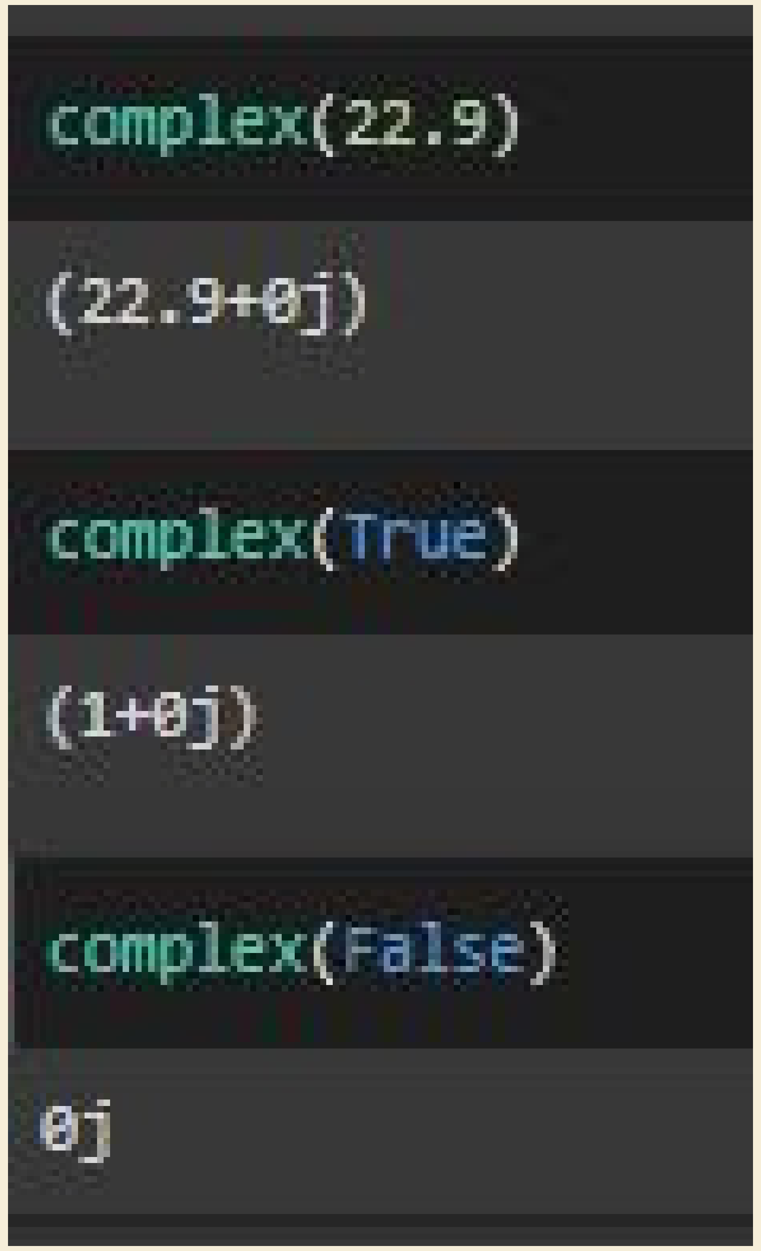
* integer with imaginer part is 0j

* boolean with True equal to 1 for real component and 0j for imaginer component and False equal to 0 for real component and 0j for imaginer component

Example:



```
complex("1+4j")
(1+4j)
complex("22")
(22+0j)
complex("22.14")
(22.14+0j)
complex(22)
(22+0j)
```



```
complex(22.9)
(22.9+0j)
complex(True)
(1+0j)
complex(False)
0j
```

Boolean conversion

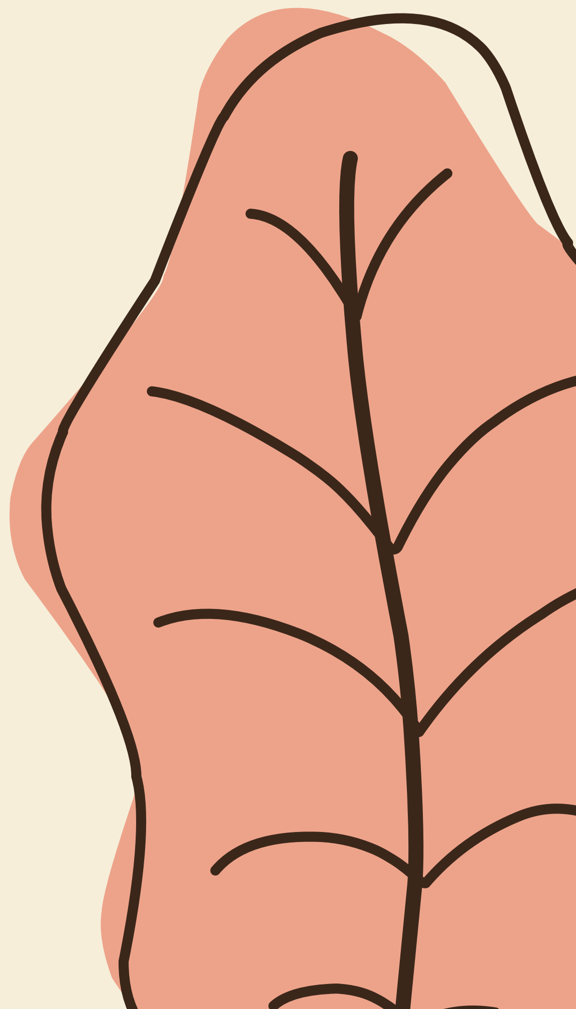
so, all data type is eligble to convert to complex type data with spesification:

* string as long there value will be convert to True value if blank value will be convert to False

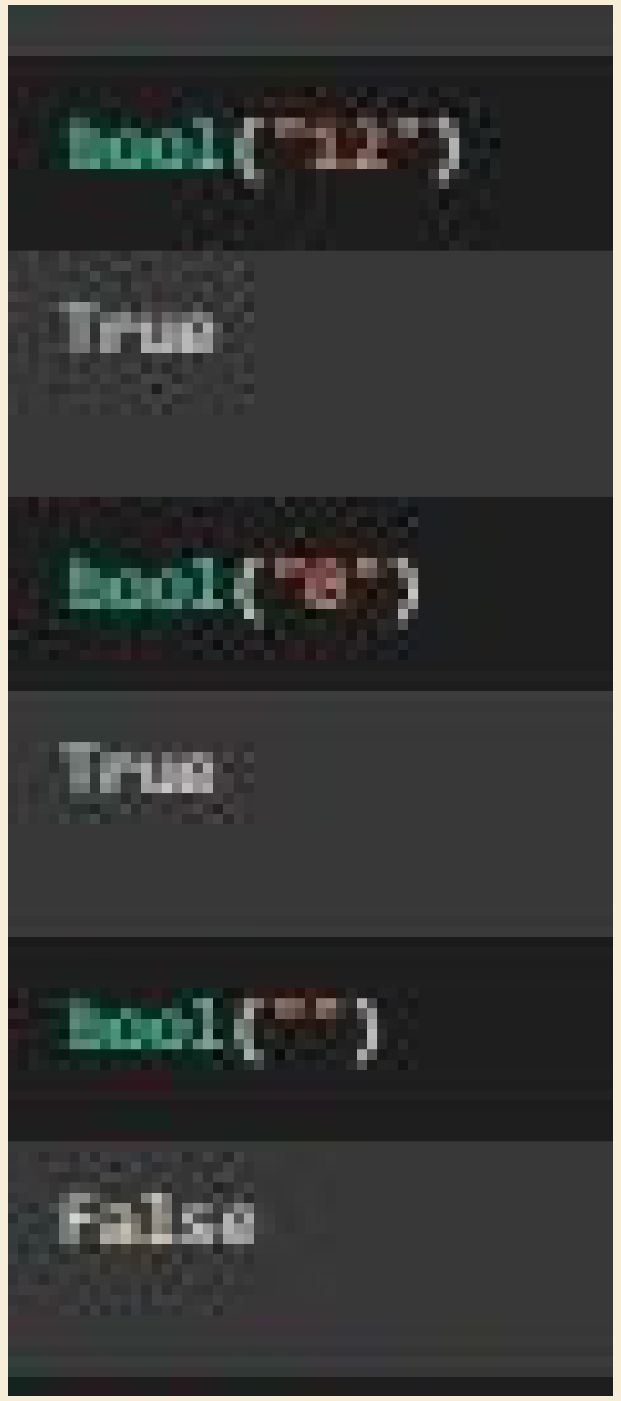
* integer will be convert as True value as long as not 0 or blank

* float will be convert as True value as long not 0.0 or blank

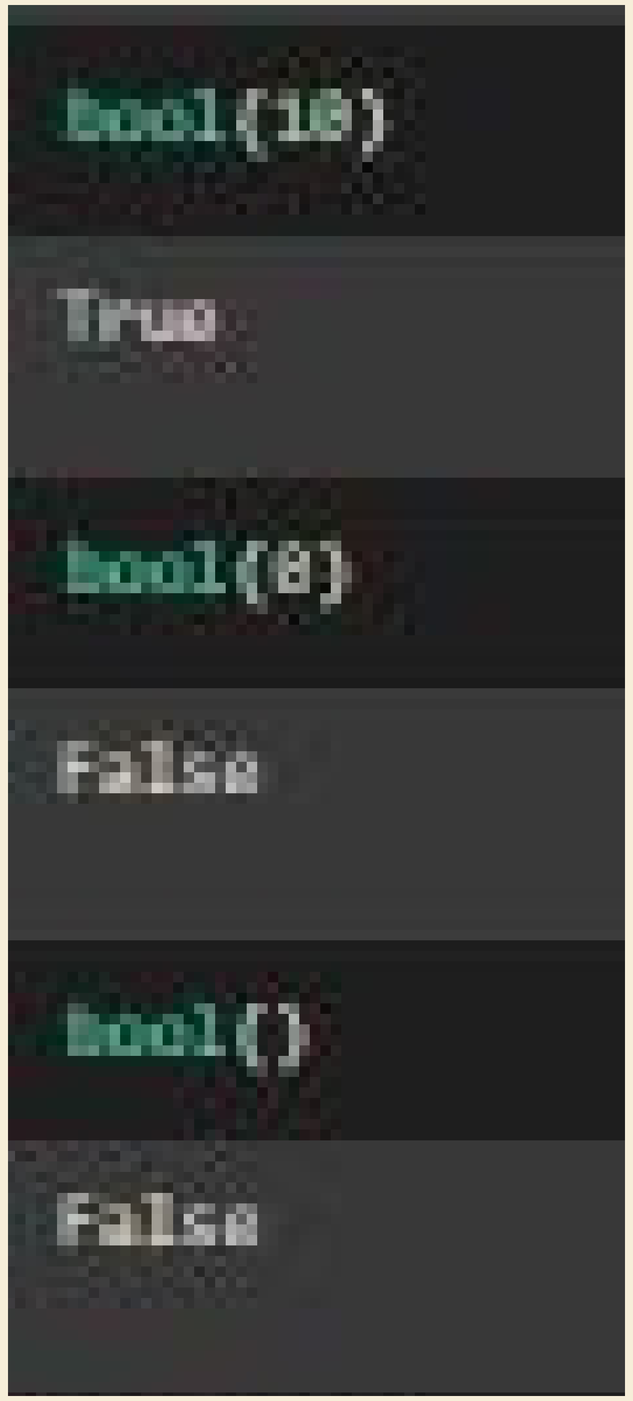
* complex will be convert as True value as ling not 0j or blank



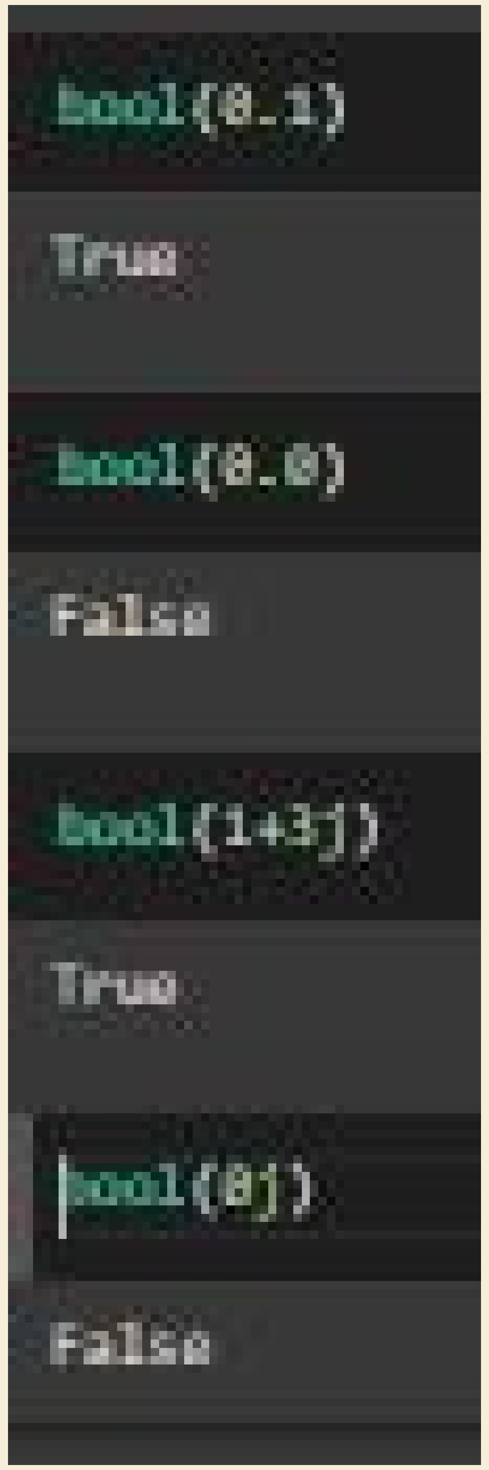
Example:



```
bool("12")
True
bool("0")
True
bool("")
False
```



```
bool(10)
True
bool(0)
False
bool()
False
```



```
bool(0.1)
True
bool(0.0)
False
bool(1+3j)
True
bool(0j)
False
```

list conversion

so, data type eligble for covert to list is:

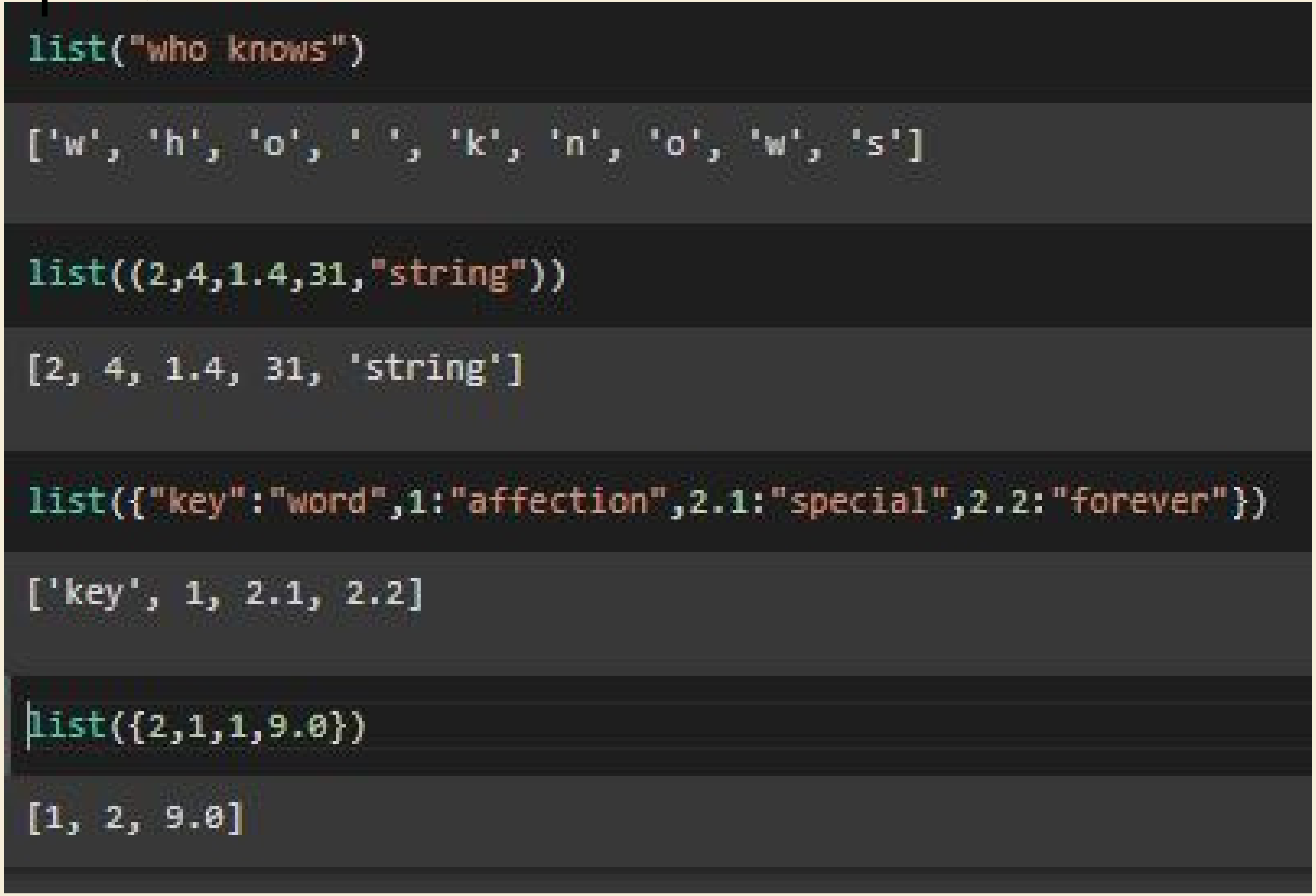
* string with each element in string is convert to each element that seperated by comma inside square bracket

* tuple change in parentheses to square bracket

* dictioary only key will be convert to element in list and value will be discard

* set duplicated number will gone first cause set unique characteristic

Example:



```
list("who knows")
```

```
['w', 'h', 'o', ' ', 'k', 'n', 'o', 'w', 's']
```

```
list((2,4,1.4,31,"string"))
```

```
[2, 4, 1.4, 31, 'string']
```

```
list({"key":"word",1:"affection",2.1:"special",2.2:"forever"})
```

```
['key', 1, 2.1, 2.2]
```

```
list({2,1,1,9.0})
```

```
[1, 2, 9.0]
```

tuple conversion

so, data type eligble for covert to tuple is:

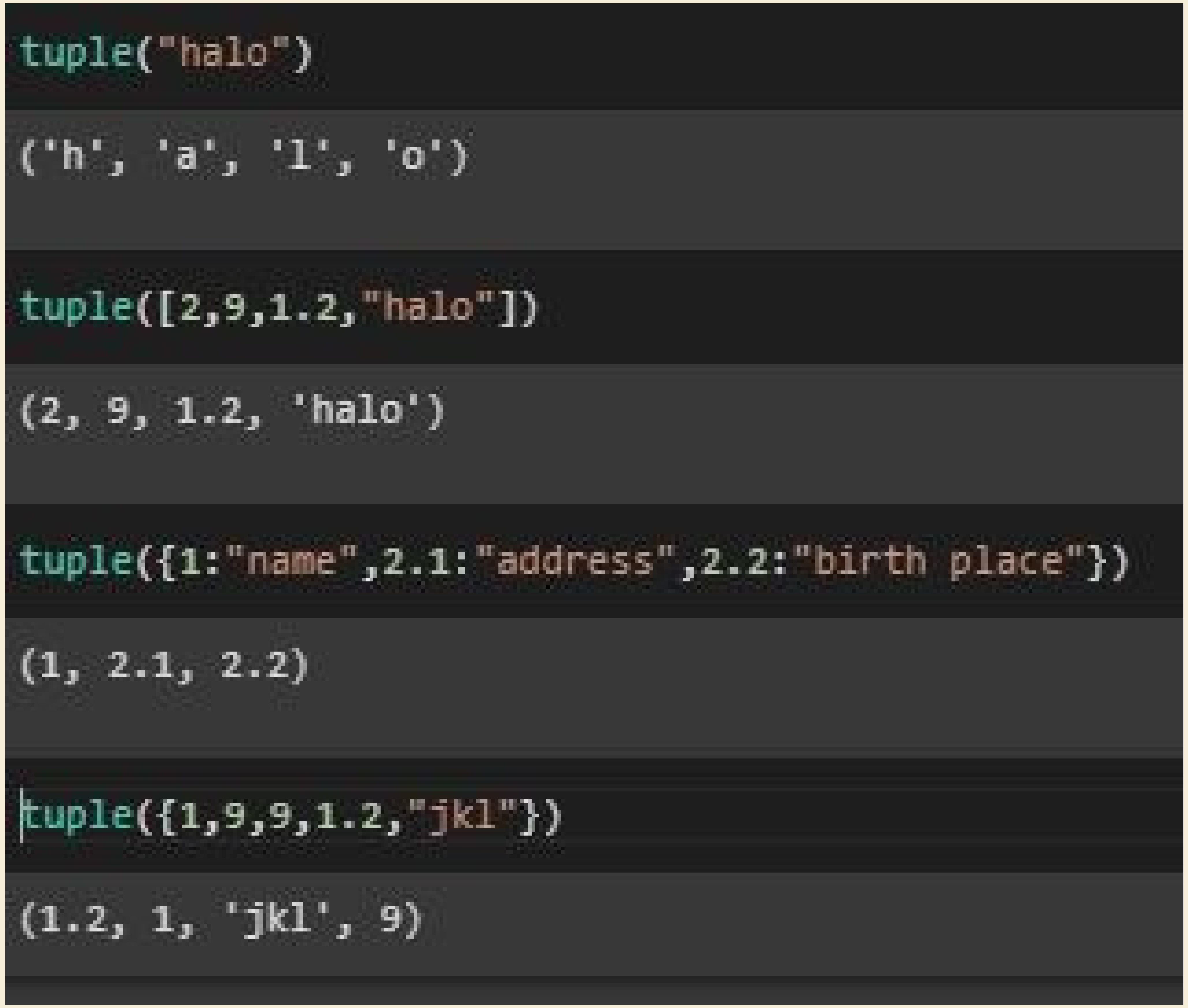
* string with each element in string is convert to each element that seperated by comma inside parentheses

* list change in square bracket to paretheses

* dictioary only key will be convert to element in tuple and value will be discard

* set duplicated number will gone first cause set unique characteristic

Example:



```
tuple("halo")
```

```
('h', 'a', 'l', 'o')
```

```
tuple([2,9,1.2,"halo"])
```

```
(2, 9, 1.2, 'halo')
```

```
tuple({1:"name",2.1:"address",2.2:"birth place"})
```

```
(1, 2.1, 2.2)
```

```
tuple({1,9,9,1.2,"jkl"})
```

```
(1.2, 1, 'jkl', 9)
```

dictionary conversion

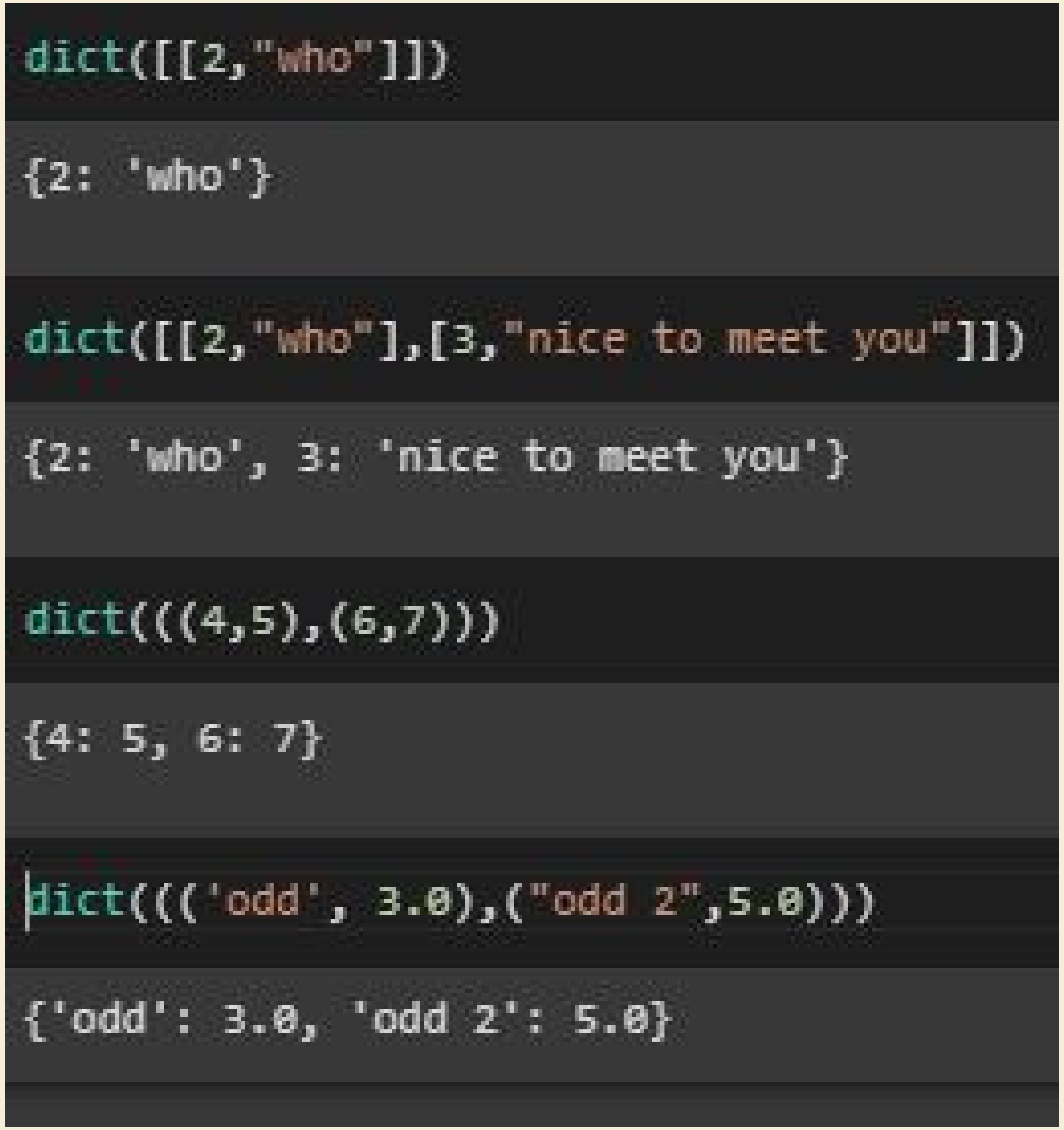
type data eligble to convert to dictionary:

* list

* tuple

note: to convert sequence data need to be in pair and can't be convert if above or below 2 element

Example:



```
dict([[2,"who"]])
```

```
{2: 'who'}
```

```
dict([[2,"who"],[3,"nice to meet you"]])
```

```
{2: 'who', 3: 'nice to meet you'}
```

```
dict(((4,5),(6,7)))
```

```
{4: 5, 6: 7}
```

```
dict((('odd', 3.0),("odd 2",5.0)))
```

```
{'odd': 3.0, 'odd 2': 5.0}
```

set conversion

so, data type eligble for covert to set is:

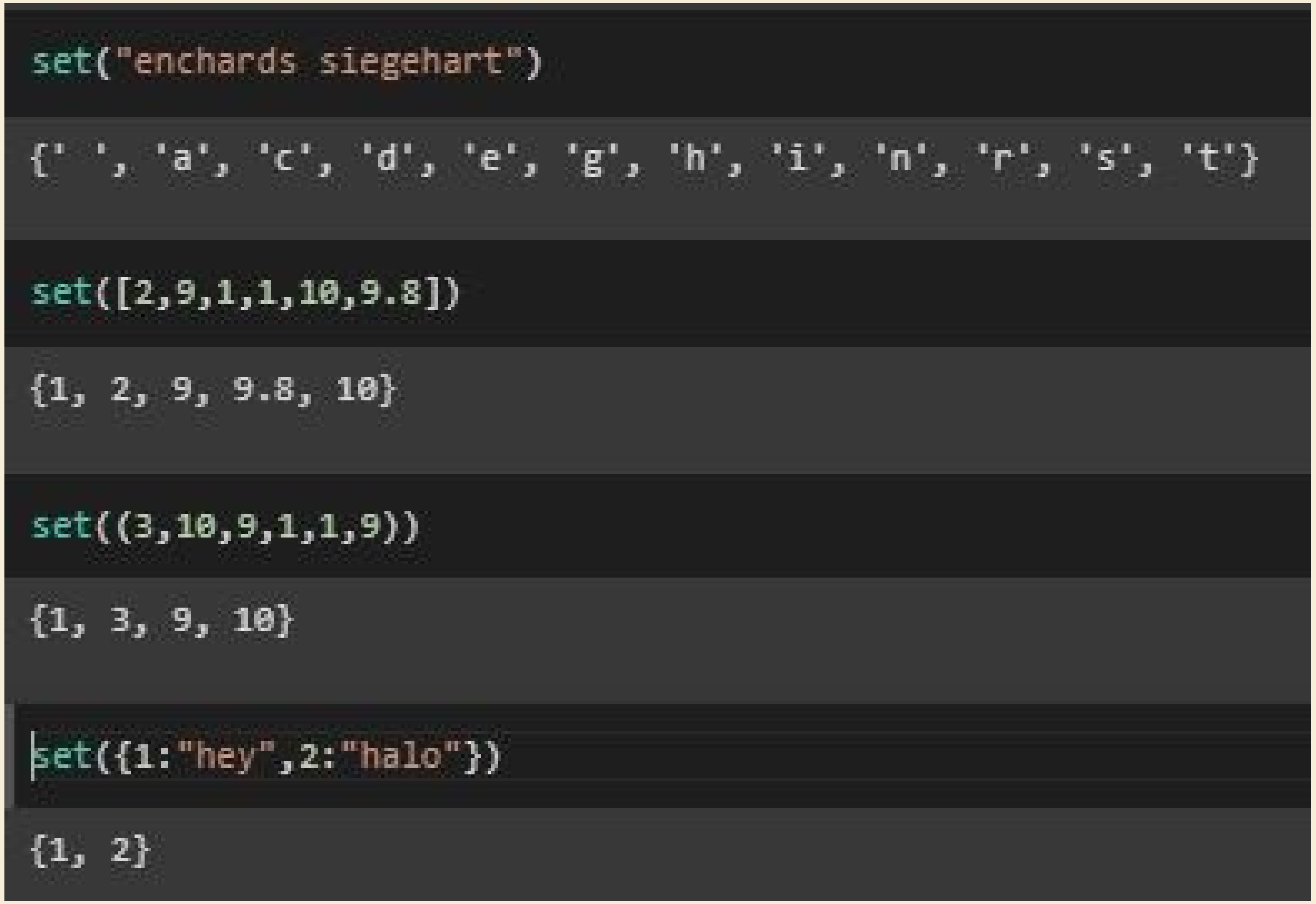
* string with each element in string is convert to each element that seperated by comma inside curl bracket with duplicated value discard so only unique value remain

* list convert to set and its square bracket change to curl bracket with its duplicated value discard so only unique value remain

* tuple convert to set and its parentheses change to curl bracketwith its duplicated value discard so only unique value remain

* dictioary only key will be convert to element in set and value will be discard.

Example:



```
set("enchards siegehart")
```

```
{' ', 'a', 'c', 'd', 'e', 'g', 'h', 'i', 'n', 'r', 's', 't'}
```

```
set([2,9,1,1,10,9.8])
```

```
{1, 2, 9, 9.8, 10}
```

```
set((3,10,9,1,1,9))
```

```
{1, 3, 9, 10}
```

```
set({1:"hey",2:"halo"})
```

```
{1, 2}
```

Boolean Operation

AND, OR, NOT

AND

AND— Returns True if both statements are true otherwise false

- **x AND y will return false if x is false or a function that returns y false**

Example :

```
Naik_motor = True
Pakai_helm = True
Tidak_membawa_surat =False
Membawa_muatan_berlebih = False
```

if it has the same variable value then the output that comes out will be the same, like the following:

print(Naik_motor and Pakai_helm)
Output : True

if it has a different variable value then the output that comes out will be opposite to the first variable, as follows :

print(Naik_motor and Tidak_membawa_surat)
Output : False

OR

OR—either term (or both) will be in the returned document

- x or y, then the value will return true.

Example :

```
Throwing_garbage_in its place = True
Protect_environment = True
Burn_garbage = False
Throwing_garbage_arbitrarily = False
```

Input Code :

print(Throwing_garbage_in its place or Protect_environment)

Output : True

Input Code:

print(Burn_garbage or Throwing_garbage_arbitrarily)

Output : False

NOT

Not operator - allows you to invert the truth value of Boolean expressions and objects.

- IF X TRUE, THE FUNCTION WILL RETURN FALSE.
- IF Y FALSE, IT WILL RETURN TRUE

```
Example :
x = 5
not x < 10
False
not callable(x)
True
```

Output : True

Arithmetic operations

Addition, Subtraction, Multiplication, Division, Exponentiation

- operator are special symbols in Python that carry out arithmetic or logical computation.
- operands is The value that the operator operates on.
- expression is a combination of operators and operands that is interpreted to produce some other value.

Addition

- Addition is the addition operator. Used to add 2 or more values

Example :
```
100+20
```
Output : 80

Example :
```
12.1+12
```
Output : 24.1

Example :
```
"kata"+"kamu"
```
Output : katakamu

Example :
```
[3,90]+[3,17]
```
Output : [3, 90, 3, 17]

Example :
```
(3,90)+(9,2)
```
Output : (3, 90, 9, 2)

Example :
```
{"key":"value"}+{"simple":"desire"}
```
Output : Error

From little experiment above, we know addition can work for a lot different data type except dictionary and set. addition works different way in different data type

- addition in string mean merge between string like we see above
- addition in numeric mean addition in matematic
- addition in list mean adding another list to other list to become one list
- addition in tuple mean adding another tuple to other tuple to become one tuple

Note: addition only possible between same data type and for numeric case as long same numeric data type it possible to use addition function so it possible to use addition between float,integer and complex.

Subtraction

- Subtraction is the subtraction operator. It is used to subtract the second value from the first value.

Example :

```
"kata"-"ka"
```

Output : TypeError

Example :

```
122-9-11
```

Output : 102

Example :

```
[2,19]-[7,10]
```

Output : TypeError

substraction only possible for numeric data type

Multiplication

- **Multiplication (*)**

is the multiplication operator. It is used to find the product of 2 values.

```
Example :
540*50
```

Output : 27000

```
Example :
"kata"*"kata"
```

Output : TypeError

```
Example :
{2,8}*3
```

Output : TypeError

```
Example :
(2,5)*3
```

Output : (2, 5, 2, 5, 2, 5)

```
Example :
a=1+8j
b=3.0
a*b
```

Output : (3+24j)

Multiplication only possible for some data type and must to be multiplication with integer data type for non numeric combination. if between numeric data type no problem occur. different combination make different output

- string with integer = repeat string based on integer number
- numeric data type multiplication = same as matematic multiplication
- list with integer = repeating sequence based on integer number and make it in one list
- tuple with integer = repeating sequance based on integer number and make it in one tuple

Division and Exponentiation

- **Division (/)**

is the division operator. It is used to find the quotient when first operand is divided by the second.

Example :
```
100/20
```
Output : 5

- **Exponentiation (**)**

is the exponentiation operator. It is used to raise the first operand to power of second.

Example :
```
20**8
```
Output : 25600000000

only possible for numeric data type

Floor Division (//)

divides two numbers and rounds the result down to the nearest integer.

Example :

```
10//3
```

Output : 3

Example :

```
(1+2j)//3
```

Output : TypeError

Modulo

The modulus operation is an operation that returns the remainder of the division of one number by another number. In programming languages this operation is generally denoted by the symbol %, mod or modulo, depending on the programming language used

Example:

```
Example :
  10%3
```

Output : 1

less than

Example:

```
Example :
  10%3
```

Output : 1

Sort mathematically in evaluating

if question :

- 1+1 *0
- 50+8 * 20+10-8/3*7+5^8
- Tips: **use parentheses for easy readability and ensure proper order**

- **1+1 *0**

True :
1+(1*0)
Output : 1

False :
(1+1)*0
Output : 0

- **50+8 * 20+10-8/3*7+5^8**

True :
50+(8*20)+10-((8/3)*7)+(5**8)
Output : 390826.333

False :
50+(8*20)+10-8/(3*7)+(5^8)
Output : 232.61904761904762

Conditional Expression And Loop

IF, ELSE, ELIF

IF

- The expression is placed after the if, and the decision is based on the truth value of the expression.
- if the expression declares as true, then the statement block inside the if statement will be executed.
- by convention, this block chooses an indent after the colon (;) if the magic expression as false, then the next block (after the statement) will be executed.

Example :

```
i=15
if i > 11 :
    print("i is greater than 11")
```

Output : i is greater than 11

ELIF

- elif is short for else if
- elif is an alternative to tiered
- An if statement can be followed by one or more elif statements.

Example :

```
tamu_hotel = int(input("berapa tamu hotel saat ini: "))

if tamu_hotel >= 1000:
    print("banyak tamu pengunjung")
elif nilai >= 500:
    print("normal")
elif nilai <100:
    print("kurang baik")
```

Output :

```
berapa tamu hotel saat ini: 1001
banyak tamu pengunjung
```

ELSE

- else statement, can be combined with if statement, as a way out when the condition/evaluation result is false.
- else is singular or optional.

Example :

```
certification_exam_requirements= int(input("exam status : "))

if certification_exam_requirements >= 700:
    print("Pass")
else:
    print("Fail")
```

Output :

```
exam status : 850
Pass
```

For

The for loop structure is usually used to repeata process with a known number of iterations.This looping statement is the most used :

for ...(1) in ...(2):

Example :

```
dary=["wake up","eat","repeat","code"]
for item in dary:
  print(dary)
for x in range (4):
  print(x)
```

Output : exam status : ['wake up', 'eat', 'repeat', 'code'] ['wake up', 'eat', 'repeat', 'code'] ['wake up', 'eat', 'repeat', 'code'] ['wake up', 'eat', 'repeat', 'code'] 0 1 2 3

Thank You

Link Github :

Klik dibawah ini:

Github