# Lecture 1-2
## Introduction to python

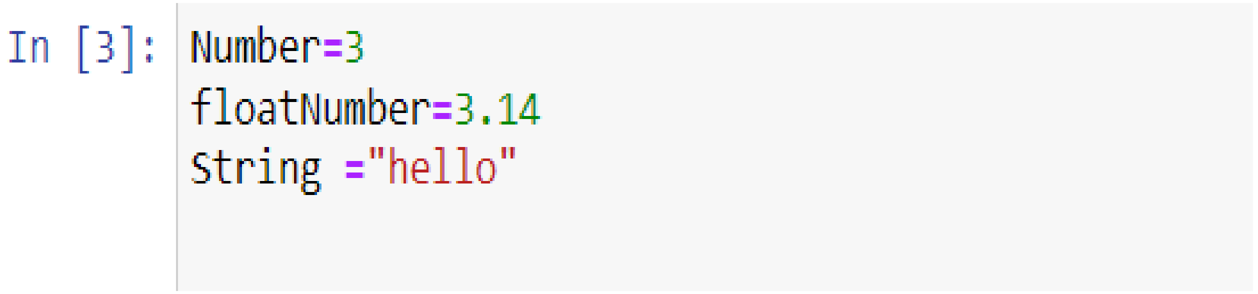
Credit: Khola Naseem

## Define a variable

- A **variable** is a named storage location that can store a value of a particular data type.
- In other words, a variable has a name, a type and stores a value.
- In order to use a variable in python we don't need to declare and specify its data type.
- The syntax to defining a new variable is to write the name of the variable and pass it a value
  - nameOfTheVariable=value

## Define a variable

- Examples:
  - Number=3
  - floatNumber=3.14
  - String =“hello”



```
In [3]: Number=3
        floatNumber=3.14
        String ="hello"
```

Credit: Khola Naseem

## Define a variable

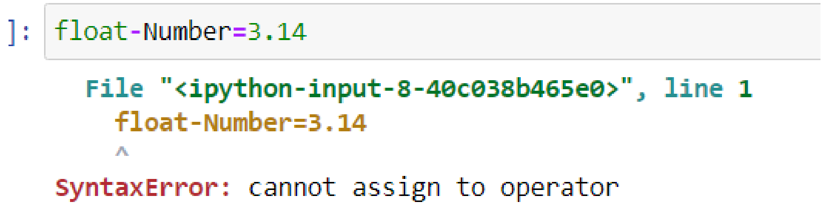
- Boolean:
    - X=True

## Name of the variable

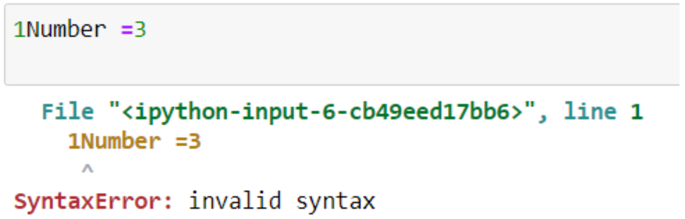
- An identifier is needed to name a variable python imposes the following rules on identifiers:
    - An identifier is a sequence of characters, comprising uppercase and lowercase letters (a-z, A-Z), digits (0-9), and underscore "_".
    - White space (blank, tab, new-line) and other special characters (such as +, -, *, /, @, &, commas, etc.) are not allowed.

## Name of the variable

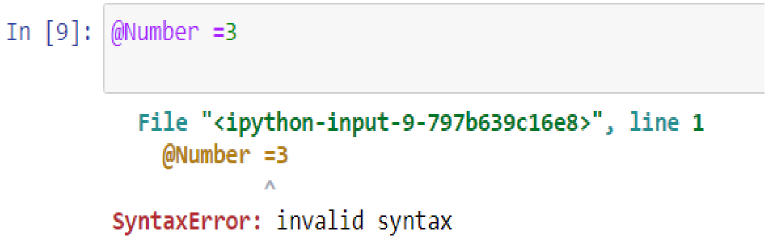
- White space (blank, tab, new-line) and other special characters (such as +, -, *, /, @, &, commas, etc.) are not allowed.
- Examples



```
]: float-Number=3.14

  File "<ipython-input-8-40c038b465e0>", line 1
    float-Number=3.14
    ^
SyntaxError: cannot assign to operator
```



```
1Number =3

  File "<ipython-input-6-cb49eed17bb6>", line 1
    1Number =3
     ^
SyntaxError: invalid syntax
```



```
In [9]: @Number =3

  File "<ipython-input-9-797b639c16e8>", line 1
    @Number =3
            ^
SyntaxError: invalid syntax
```

Credit: Khola Naseem

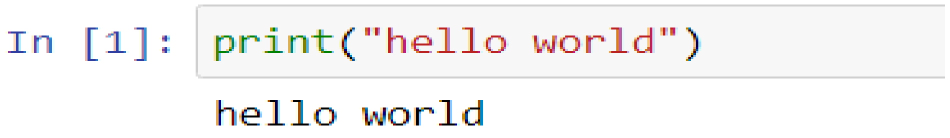
## Declaration of variables

python imposes the following rules on identifiers:

- An identifier must begin with a letter or underscore. It cannot begin with a digit.
- Identifiers are case-sensitive. A rose is NOT a Rose, and is NOT a ROSE.

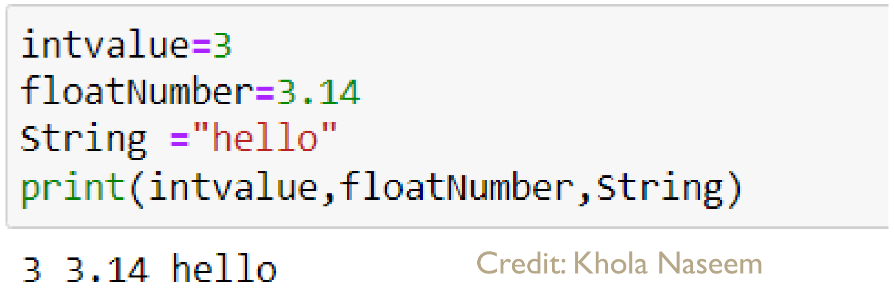
## Print data on the screen

- print is used to print data on the screen.
- Syntax:
    - print()
- Example 1:



```
In [1]: print("hello world")

        hello world
```

- Example 2:

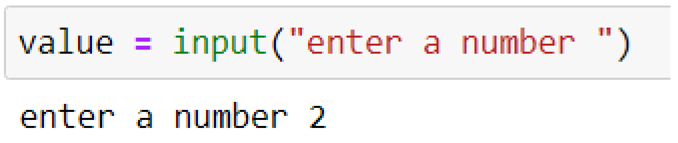


```
intvalue=3
floatNumber=3.14
String ="hello"
print(intvalue,floatNumber,String)

3 3.14 hello
```

Credit: Khola Naseem

# Take input

- input is used to take the input from user.
- syntax
    - input()
- Example:



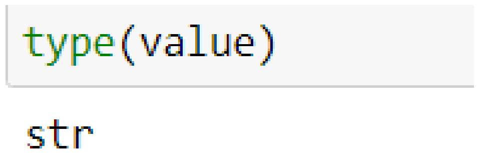
```
value = input("enter a number ")
```

```
enter a number 2
```

Credit: Khola Naseem

# Check data type

- To determine a variable's type in Python you can use the type() function. The value of some objects can be changed. Objects whose value can be changed are called mutable and objects whose value is unchangeable (once they are created) are called immutable. Here are the details of Python datatypes
- Data type:

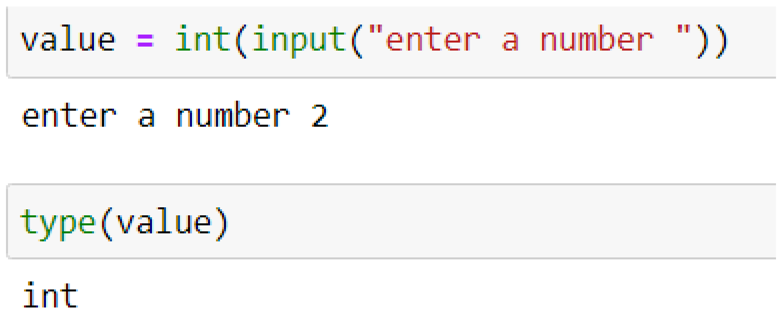


```
type(value)
```

```
str
```

## Change data type of the input

- for integer value use the built-in-function
    - int()
- Example:

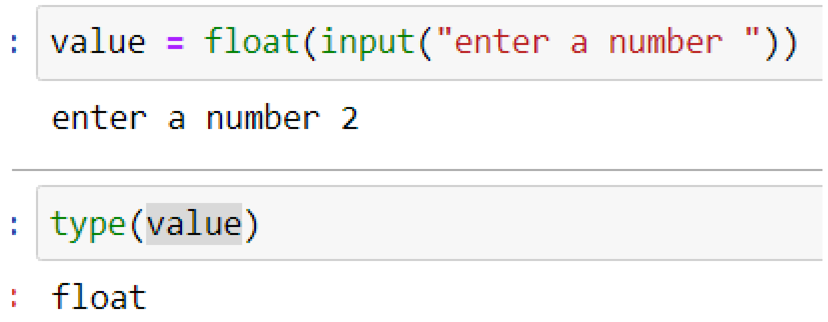


```
value = int(input("enter a number "))
enter a number 2
type(value)
int
```

Credit: Khola Naseem

## Change data type of the input

- for integer value use the built-in-function
  - float()
- Example:

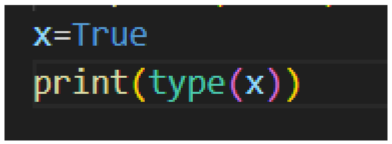


```
: value = float(input("enter a number "))

  enter a number 2

: type(value)

: float
```

Credit: Khola Naseem

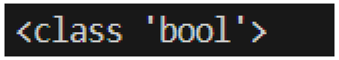
## Define a variable

- Boolean:



```
x=True
print(type(x))
```

- Examples:

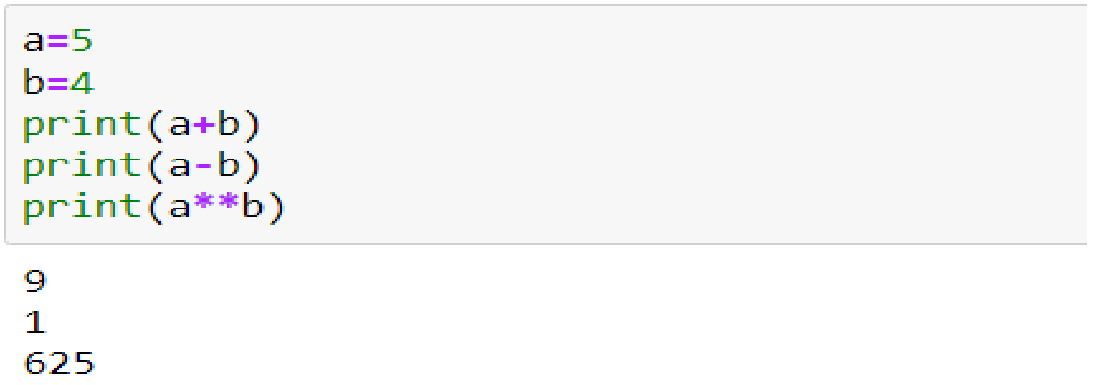


```
<class 'bool'>
```

Credit: Khola Naseem

# Boolean operator

- +, -, *, /,**



```
a=5
b=4
print(a+b)
print(a-b)
print(a**b)
```

```
9
1
625
```

Credit: Khola Naseem

# Python Reserve words:

- The following identifiers are used as reserved words of the language, and cannot be used as ordinary identifiers.

| | | | | |
|---|---|---|---|---|
| False | class | finally | is | return |
| None | continue | for | lambda | try |
| True | def | from | nonlocal | while |
| and | del | global | not | with |
| as | el | if | or | yield |
| assert | else | import | pass | |
| break | except | in | raise | |

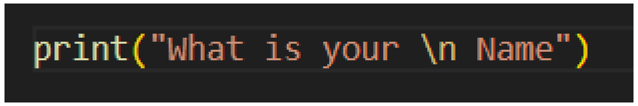
Credit: Khola Naseem

# Special characters in strings

- The backslash (\) character is used to introduce a special character. See the following table.

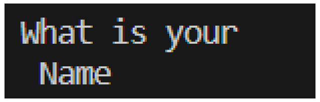
| Escape sequence | Meaning |
|---|---|
| \n | Newline |
| \t | Horizontal Tab |
| \\ | Backslash |

- \n:



```
print("What is your \n Name")
```

- Output



```
What is your
 Name
```

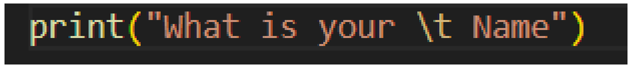
Credit: Khola Naseem

## Special characters in strings

- The backslash (\) character is used to introduce a special character. See the following table.

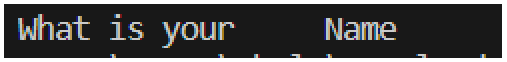
| Escape sequence | Meaning |
|---|---|
| \n | Newline |
| \t | Horizontal Tab |
| \\ | Backslash |

- \t:



```
print("What is your \t Name")
```

- Output



```
What is your 	 Name
```

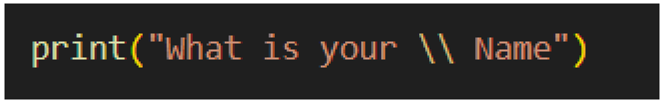
Credit: Khola Naseem

# Special characters in strings

- The backslash (\) character is used to introduce a special character. See the following table.

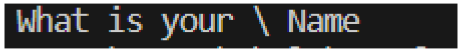
| Escape sequence | Meaning |
|---|---|
| \n | Newline |
| \t | Horizontal Tab |
| \\ | Backslash |

- \\:



```
print("What is your \\ Name")
```

- Output

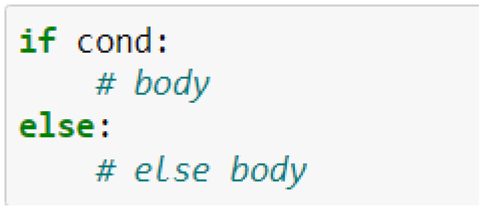


```
What is your \ Name
```

Credit: Khola Naseem

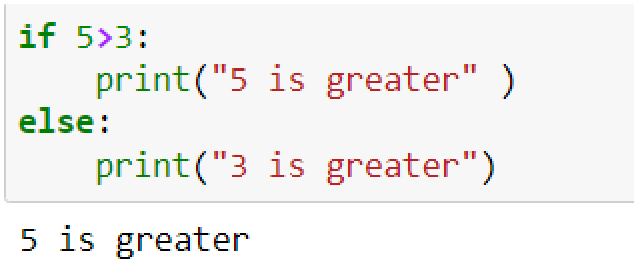
## Conditional statements

- if-else
- Syntax:



```
if cond:
    # body
else:
    # else body
```

- Example:



```
if 5>3:
    print("5 is greater" )
else:
    print("3 is greater")
```

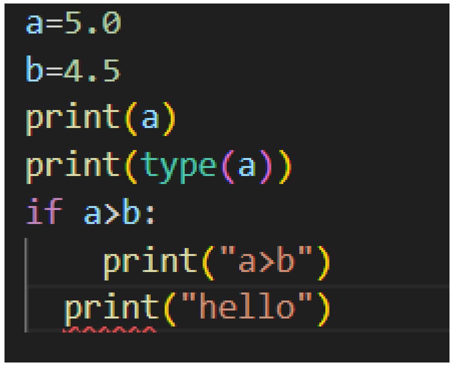
```
5 is greater
```

## Indentation

- Python uses whitespace (spaces and tabs) to define program blocks whereas other languages like C, C++ use braces ({}) to indicate blocks of codes for class, functions or flow control.
- The number of whitespaces (spaces and tabs) in the indentation is not fixed, but all statements within the block must be the indented same amount.

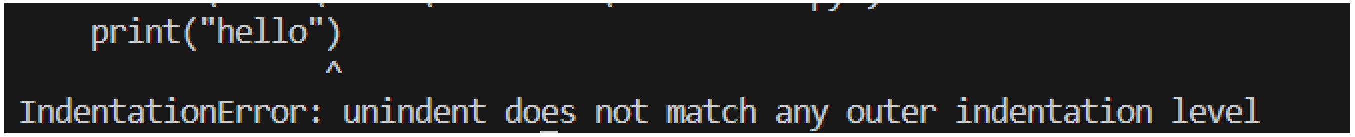
# Indentation

- Example:



```
a=5.0
b=4.5
print(a)
print(type(a))
if a>b:
    print("a>b")
  print("hello")
```

- Error:



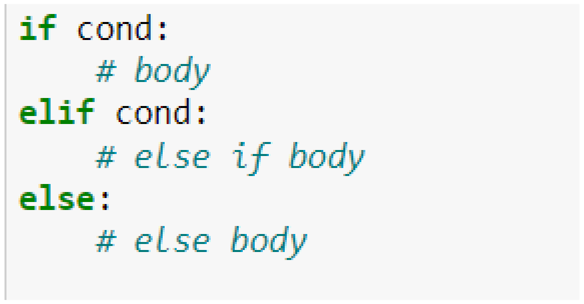
```
    print("hello")
                 ^
IndentationError: unindent does not match any outer indentation level
```

Credit: Khola Naseem

# Conditional statements

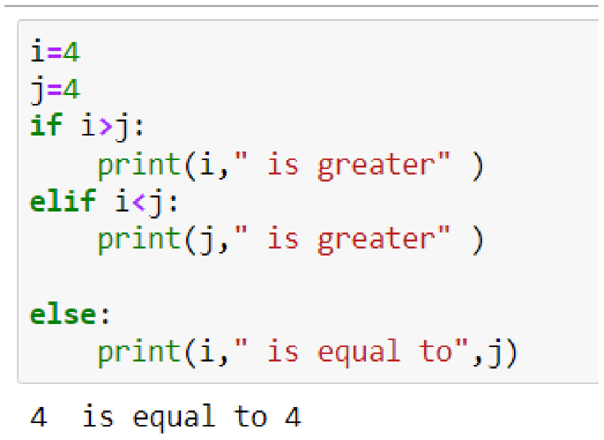
- if-elif-else

- Syntax:



```
if cond:
    # body
elif cond:
    # else if body
else:
    # else body
```

- Example:



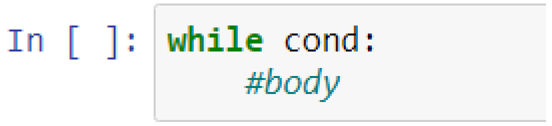
```
i=4
j=4
if i>j:
    print(i," is greater" )
elif i<j:
    print(j," is greater" )

else:
    print(i," is equal to",j)
```

```
4  is equal to 4
```

Credit: Khola Naseem

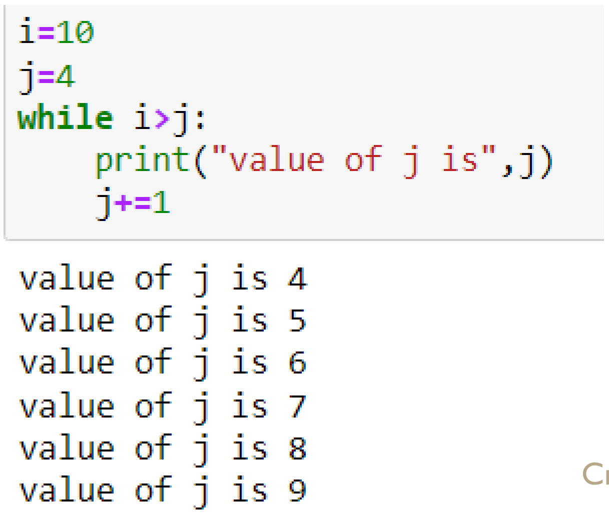
## Repetitive structures

- while and for loop
- Syntax:



```
In [ ]: while cond:
            #body
```

- Example:



```
i=10
j=4
while i>j:
    print("value of j is",j)
    j+=1
```

```
value of j is 4
value of j is 5
value of j is 6
value of j is 7
value of j is 8
value of j is 9
```

Credit: Khola Naseem

## Repetitive structures

- for loop
- Syntax:

```
for cond:
    #body
```

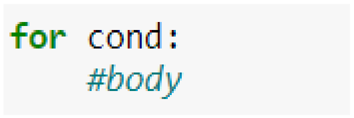
- Example:

```
for i in range(9):
    print(i)
```

```
0
1
2
3
4
5
6
7
8
```

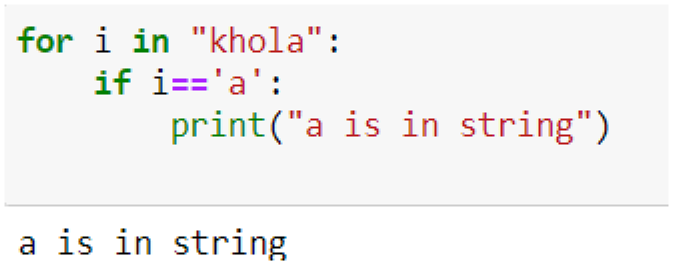
## Repetitive structures

- for loop
- Syntax:



```
for cond:
    #body
```

- Example:



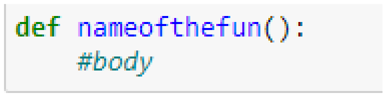
```
for i in "khola":
    if i=='a':
        print("a is in string")
```

```
a is in string
```

Credit: Khola Naseem

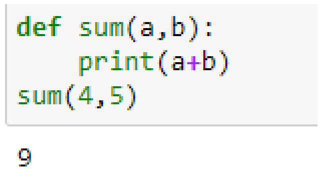
## Functions

- def
- Syntax:



```python
def nameofthefun():
    #body
```

Example:

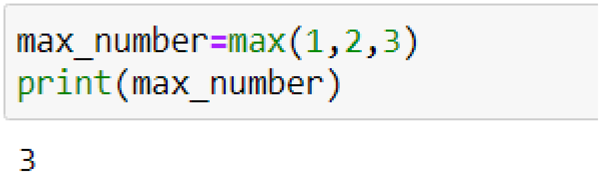


```python
def sum(a,b):
    print(a+b)
sum(4,5)
```

```
9
```

# Built in functions

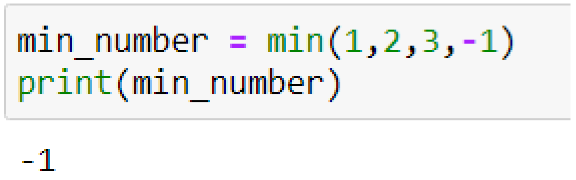
- max():



```
max_number=max(1,2,3)
print(max_number)
```

```
3
```

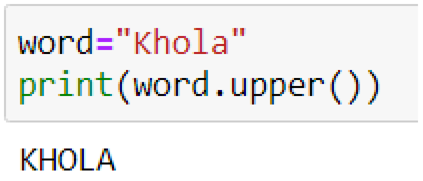
- min()



```
min_number = min(1,2,3,-1)
print(min_number)
```

```
-1
```

- upper()

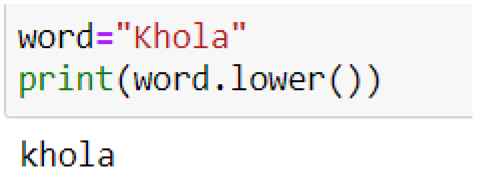


```
word="Khola"
print(word.upper())
```

```
KHOLA
```

## Built in functions

- lower:



```
word="Khola"
print(word.lower())
```

```
khola
```

## Built in functions

- input()
- type()
- int()
- float()
- range()
- Max()
- Min()
- Upper()
- Lower()