# Python

→ Easy to use
→ Fewer lines of code
→ Interpreted language
→ Free of cost
→ Cross platform language

} Advantages

→ Not the fastest language as it is an interpreted
→ Not easy to convert in some other language
→ Lesser libraries than C, Java, etc.

} Disadvantages

## Token

→ Smallest unit of any programming language.
→ Also known as Lexical unit.

## Types of Token

1) Identifiers
2) Keyword
3) Literals
4) Punctuators
5) Operators

## Keywords

- Reserved words
- Provides special meaning to interpreter
- Cannot used as identifiers
- Total 36 keywords

True, False, None, for, while, break, continue, and, or, is, in, not, if, else, elif, return, del, non-local, global, async, await, try, finally, raise, except, assert, with, import, from, as, def, class, lambda, pass, yield

## Identifiers

- Names given to different parts of program

Naming rules:

- Must be a non-Key word
- No spaces in between
- Must be made up of only letters, numbers and underscore.
- Can't begin with a no., although they can contain numbers.

| Valid | Invalid |
| --- | --- |
| Myfile | Data-rec (Contains special character) |
| _chk | |
| Date_7_2 | 2qCLCT (Starting with digit) |
| z2TOZ9 | break (keyword) |

## Literals

I) String literals - "Pankaj"
II) Numeric literals - 10, 13, 3+5j, 17.0
III) Boolean literals - True or False
IV) Special literal - None

## Operators

- Performs some computation/action

- Arithmetic operators (+, -, *, /, //, **, %)
Bitwise operators (&, ^, |)
Shift operators (<<, >>)
Identity operators (is, isnot)
Relational operators (>, <, >=, <=, ==, !=)
Assignment operators (=)
Membership operators (in, not in)

## Punctuators

Symbols that are used in programming languages to organize sentence structures.

' " # \ ( ) { } [ ] @ , : . ' =

## Assigning values to Variables

```
Name = 'Swati'      # String data type
var = None          # variable without value
a1 = 23             # Interger
a2 = 6.2            # Float
```

## Dynamic Typing

```
X = 10                      # Output
print(X)                    10
X = "Hello-world"           Hello World
print(X)
```

## Multiple assignments

→ a = b = c = 10

It will assign value 10 to all three variables a, b, c

→ x, y, z = 10, 20, 30

# Python Type conversion

Type conversion is the process of converting the data of one type to another type

1) Implicit - Automatically converts one data to another

```
i=12
f = 4.5
a = i+f
print(a)
```

11) Explicit - Users convert the data type of an object to required data type.
Like - int(), float(), str()

## input()

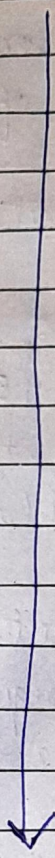
Returns a value of string type.

## print()

Used to display a string on the console

## Table operator precedence

()
**
~x
+x, -x
*, /, //, %
+, -
&
^
|
<, <=, >, >=, <>, !=, ==
is, is not
not x
and
or



Example

a) a, b = 3, 6
c = b/a ('/' always gives floating as result)
c = 6/3 = 2.0

b) a, b = 3, 6
c = b//a ('//' gives integer)
c = 6//3 = 2

c) a, b = 3, 6.0

## Relational expressions

Returns Boolean values ie. True or False.

| | | |
|---|---|---|
| $17 > 4$ | $4 < 1$ | $5 == 5$ |
| True | False | True |

```
x = [3, 3]
y = [3, 3]
x == y
```

Output:
True

```
x[0] > y[0]
```

Output:
False

```
p = [5]
q = (5)
p == q   # Output:
           False
p[0] == q[0]   # output:
                 True
```

```
a = 'India'
b = 'india'

a < b
'I' < 'i'
```

```
a = 'indiaP'
b = 'india'
a < b
```

# False

→ p = [2,3]
q = [3,2]
p == q
# False

→ 7 // -4
# -2

→ 0.1 + 0.1 + 0.1 == 0.3
# False

# Bitwise operator

& → and [Ampersion symbol]
| → Or [Pipeline symbol]
^ → XOR [Carret symbol]
~ → Complement [Tilde symbol]

## & operator

'A' & 'B'

## | Or operator

4 | 12

(4 → 100)
(12 → 1100)
or → (100 or 1100)

= 0100
  1100
  ----
  1100

Output = 12

Q. 24 & 2

```
  11 000
  00 010
 -------
  00000
 -------
```

XOR Bitwise operator (^)

24 ^ 2 = 11000
   00010
   -----
   11010 = 26

Bitwise complement operator (~)

~ 12
00001100
= $(11110011)_2$
= ~~$(243)_{10}$~~

11110011
2's comp. → 00001101 = −13

## Logical expressions (and, or, not)

~~25/5 or 20/10~~

### And

F and F → F
F and T → F
T and F → .F
T and T → T

eg: a = 5
b = 3
e = 7

a < b and b > c  # Output
False and False  False

### Or

F or F → False
F or T → True
T or F → True
T or T → True

## Not operator

```
p = 10
q = 20                    # Output
not (p>q)                 True
```

Each value in python having a truth value in form of true or false.

**False :-**

None

0

Empty set " " ( ) [ ] { }

Rest entities have their truth value true

* For and operator if it is in form of p and q, if truth value of p is false, then p will be the final output.

Eg. i) [] and 5
[]

ii) 12 and 4
4

iii) () and (5)
()

IV) {3,43 and False
False

* For or operator, if it is in the form of p or q, if p is false then q will be output.

I) {3 or 5
5

II) 5 or []
[] 5

III) 10 < 45 and 45 < 5
True False
False

Identity operators

is and is not are the two identity operators.

```
a = 5
b = 5          # Output
a is b         True
```

```
a = 5                                  # Output
b = int(input('Enter b'))              True
a is b
```

```
a = 3.5
b = float(input('Enter a value')) # 3.5 is entered
a == b      # True
a is b      # False
```

## Casting

int ( )
float ( )
complex ( )
str ( )

int ( 17.2 )
Output : 17

int ( 19.9 )
Output : 19

float ( 17 )
Output = 17.0

complex ( 0 )
Output = 0 + 0j

string ( 2 )
Output = '2'

## Functions from math module:

import math

ceil ( )
floor ( )
sqrt ( )
fabs ( )
exp ( )
log ( )
log10 ( )
pow ( )
sin ( )
cos ( )
tan ( )
degrees ( )
radians ( )

Constants.

pi

e

* sqrt

Q. The python equivalent statement for $\sqrt{b^2 - 4ac}$

Ans
```
math.sqrt(b*b - 4*a*c)
math.sqrt(b**2 - 4*a*c)
(b*b - 4**a*c)**0.5
```

* fabs() → finds the absolute value of any floating number

Q.
```
math.fabs(-4)
```

Output : 4.0

* pow

It finds the exponent value of $x^y$, where x and y can be integer or float.

→
```
math.pow(2,4)
```
Output: 16

→
```
math.pow(4,0.5)
```
Output : 2.0

$\log_{10} 3 = 4$

Date
Page

$(2^2)^{1/2}$

- math.pow(-4, 0.5) ⟶ Error
- math.sqrt(-4) ⟶ Error
- exp( ) ⟶ $e^x$

example

math.exp(2)

math.exp(0) output 1.0

- log

It is used to find logarithm

$x^y = a$

$\log_x a = y$

$\log_{10} 100 = 2$

output : 2.0

$\log_2 16 = 4$

output : 4.0

log (1) = 0.0

* sin ( ) function.

It is trigonometry function. It find the sin value given in radian.

sin (math.pi / 2)

Output : 1.0

sin (0) = 0.0

* math.degrees ( )

It converts angle x from radian to degree.

(math.pi)

≈ 180.0

* math.rad ( )

It converts an angle x from degree to radian

math.radian (180) = 3.141....

* math.ceil ( ).

It returns the smallest integer not less than the given number.

math.ceil (786.786)

~ 787

math.ceil(-786.786)

Output : -786

math.floor

floor function returns the largest no. not greater than the given number

math.floor(786.786) ~ 786

math.floor(-786.786) ~ -787