

Day 3: Operators (Arithmetic & Assignment Operator)

Few additional points moving to operators

- You cannot start a variable name by any numerical value
- You cannot name your variable like "var-name"
- You cannot use spaces in variable names

Legal Variable Names

```
myvar = "John"
my_var = "John"
_my_var = "John"
myVar = "John"
MYVAR = "John"
myvar2 = "John"
```

Illegal Variable Names

```
2myvar = "John"
my-var = "John"
my var = "John"
```

Operators

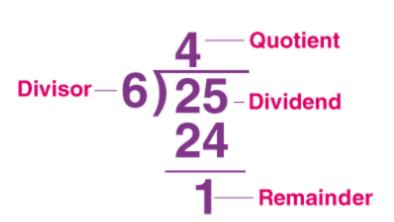
Types of operators:

Arithmetic operators (+, -, *, /, %, **, //)

- Assignment operators(=, +=, -=, *=, /=)
- Comparison operators(==, ! =, >, <, > =, < =)
- Logical operators(and, or, not)
- Identity operators
- Membership operators
- Bitwise operators

Arithmetic Operators

```
x = 5
y = 3
# +
print(x + y)
# -
print(x - y)
# *
print(x * y)
# / (Returns Quotient in Float)
print(x / y)
# % (Modulus) (Returns Remainder)
print(x % y)
# ** (Exponentiation) (Power of)
print(x ** y)
# // (Floor Division) (Returns Quotient in absolute)
print(x // y)
```



Data Type Operations

- INT (operator) INT returns INT
- INT (operator) FLOAT returns FLOAT
- INT (operator) COMPLEX returns COMPLEX
- INT (operator) STRING returns ERROR

```
a = 6
b = 2.0
c = 3j
d = '6'
e = 3

# INT (operator) INT returns INT
print(a / e)

# INT (operator) FLOAT returns FLOAT
print(a / b)

# INT (operator) COMPLEX returns COMPLEX
print(a / c)

# INT (operator) STRING returns ERROR
print(a / d)
```

Few exceptions for the case are:

```
# / (Returns Quotient in Float)
print(x / y)

# Reason: Type Casting happens inside Python Interpreter and content
```

Test

Assignment Operators

These operators are used to assign values to the variables.

```
# =
x = 5
# +=
x += 5
# -=
x -= 5
# *=
x *= 5
# /=
x /= 5
# %=
x %= 5
# //=
x // = 5
```