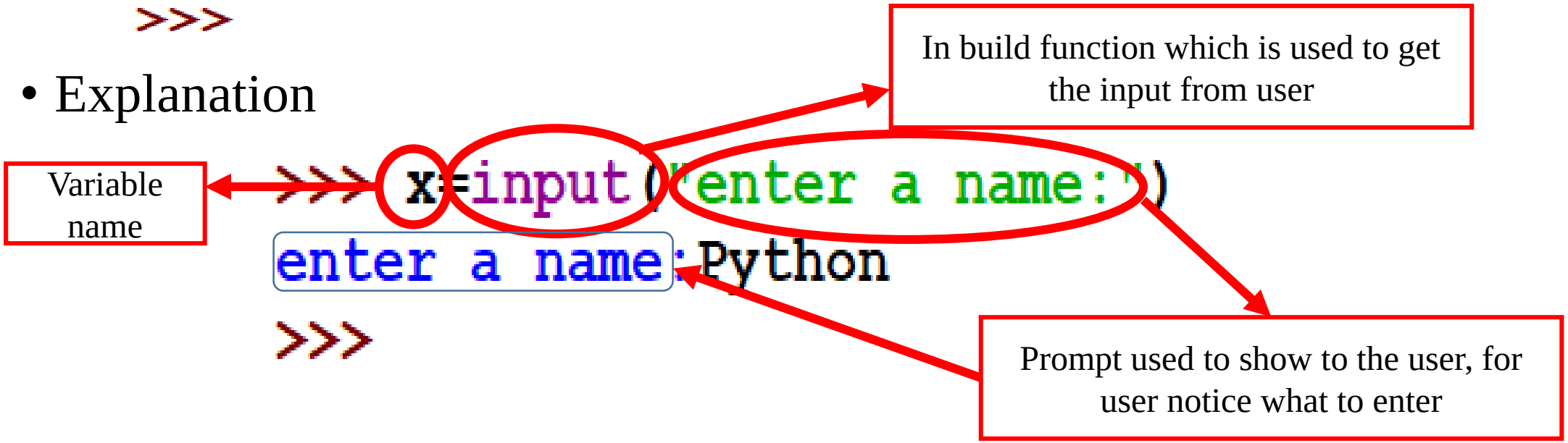
Input and Operator

Input Function

- input is a inbuilt function to get input from user, Example as follow

```
>>> x=input("enter a name:")
enter a name:Python
>>>
```

- Explanation



Input is a default string

- User defined input will be assigned **as string by default,**

Example:

```
>>> x=input("enter a name:")
enter a name:python
>>> type(x)
<class 'str'>
>>> x=input("enter a number:")
enter a number:12
>>> type(x)
<class 'str'>
>>> x=input("enter a number:")
enter a number:0.12
>>> type(x)
<class 'str'>
>>> x=input("enter a number:")
enter a number:True
>>> type(x)
<class 'str'>
>>>
```

Type Conversion

- We can able to convert variables to desired datatype

```
>>> x=input("enter a number:")
enter a number:12
>>> x=int(x)
>>> type(x)
<class 'int'>
>>> x=input("enter a number:")
enter a number:True
>>> x=bool(x)
>>> type(x)
<class 'bool'>
>>>
```

Input in desired data type

- Else while getting a input itself we can able to get as a desired input
- To get this follow the following:

```
>>> x=int(input("enter a number:"))
enter a number:12
>>> type(x)
<class 'int'>
>>> x=int(input("enter a number:"))
enter a number:0.12
Traceback (most recent call last):
  File "<pyshell#45>", line 1, in <module>
    x=int(input("enter a number:"))
ValueError: invalid literal for int() with base 10: '0.12'
>>>
```

- In this example it accept only integer datatype not others, so we are getting an error here
- We can give any data type like float, bool instead of int

Eval

- Eval accepts the value
- Evaluate used to understand the data type
- It wont understand the string without quotes, it will consider as a variable name
- Eval even understand the comma separated values as well

```
>>> x=eval(input("Enter a number"))
Enter a number23
>>> type(x)
<class 'int'>
>>> x=eval(input("Enter a number"))
Enter a number0.23
>>> type(x)
<class 'float'>
>>> x=eval(input("Enter a number"))
Enter a numberTrue
>>> x=eval(input("Enter a number"))
Enter a numbercomplex(1,2)
>>> type(x)
<class 'complex'>
>>> x=eval(input("Enter a number"))
Enter a numberpython
Traceback (most recent call last):
  File "<pyshell#16>", line 1, in <module>
    x=eval(input("Enter a number"))
  File "<string>", line 1, in <module>
NameError: name 'python' is not defined
>>> x=eval(input("Enter a number"))
Enter a number"python"
>>> x=eval(input("Enter a number"))
Enter a number12+23
>>> print(x)
35
>>> score1,score2=eval(input("enter the values of Score 1 and Score 2: "))
enter the values of Score 1 and Score 2: 12,23
>>> score1
12
>>> score2
23
>>> Avg=(score1+score2)/2
>>> Avg
17.5
>>>
```

Operator

Operators are the symbols which tells the Python interpreter to do some mathematical or logical operation.

Operators

- Following are the operators available, this is common operator name which is available

ASSIGNMENT OPERATOR	ARITHMETIC OPERATOR	RELATIONAL OPERATORS	LOGICAL OPERATORS	BITWISE OPERATORS	INCREMENT & DECREMENT OPERATOR
=	+ - * / % += -+ *= /= %=	< <= > >= == !=	! && ||	& | ~ ^ << >> <<= >>= &= = ^+	++ --

Arithmetic Operators

- Following are arithmetic Operator

Operator	Description
+ Addition	Adds values on either side of the operator.
- Subtraction	Subtracts right hand operand from left hand operand.
* Multiplication	Multiplies values on either side of the operator
/ Division	Divides left hand operand by right hand operand
% Modulus	Divides left hand operand by right hand operand and returns remainder
** Exponent	Performs exponential (power) calculation on operators
//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative, the result is floored, i.e., rounded away from zero (towards negative infinity) −

```
>>> a=12
>>> b=34
>>> a+b
46
>>> a-b
-22
>>> a*b
408
>>> a/b
0.35294117647058826
>>> a//b
0
>>> round(0.564)
1
>>> round(0.3)
0
>>> 32//23
1
>>> 32/23
1.391304347826087
>>> 32/45
0.7111111111111111
>>> 32//45
0
>>>
```

Order of Operations

- Python follows **PEMDAS**

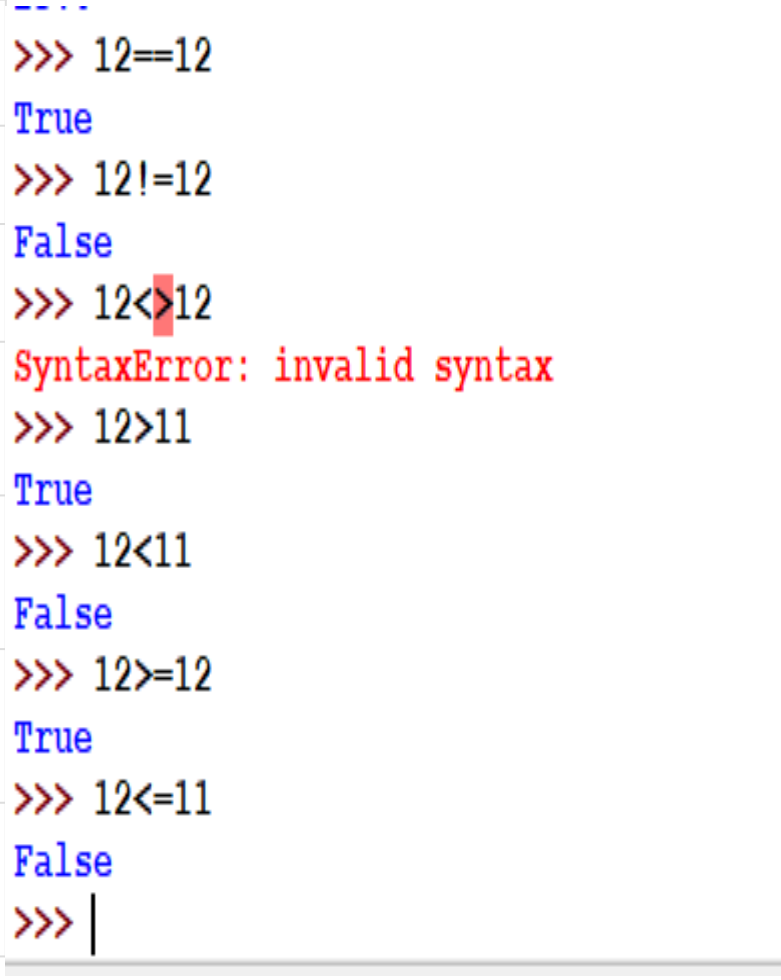
Name	Syntax
Parentheses	(...)
Exponents	**
Multiplication and Division	* / // %
Addition and Subtraction	+ -

- **The Example:**

```
>>> 12+34/2
29.0
>>> (12+34)/2
23.0
>>>
```

Comparison Operator

Operator	Description
==	If the values of two operands are equal, then the condition becomes true.
!=	If values of two operands are not equal, then condition becomes true.
<>	If values of two operands are not equal, then condition becomes true. Now its not available in previous version it was there
>	If the value of left operand is greater than the value of right operand, then condition becomes true.
<	If the value of left operand is less than the value of right operand, then condition becomes true.
>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.
<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.



Assignment Operators

Operator	Description
=	Assigns values from right side operands to left side operand
+= Add AND	It adds right operand to the left operand and assign the result to left operand
-= Subtract AND	It subtracts right operand from the left operand and assign the result to left operand
*= Multiply AND	It multiplies right operand with the left operand and assign the result to left operand
/= Divide AND	It divides left operand with the right operand and assign the result to left operand
%= Modulus AND	It takes modulus using two operands and assign the result to left operand
**= Exponent AND	Performs exponential (power) calculation on operators and assign value to the left operand
//= Floor Division	It performs floor division on operators and assign value to the left operand

```
>>> x=12
>>> y=23
>>> x+=y
>>> print(x)
35
>>> x-=y
>>> print(x)
12
>>> x*=y
>>> print(x)
276
>>> x/=y
>>> print(x)
12.0
>>> x%=y
>>> print(x)
12.0
>>> x//=y
>>> print(x)
0.0
>>>
```

Bitwise Operator

- Just understand what is bitwise is more than enough, we are not handling this in real time

Operator	Description	Example	Same as	Result	Decimal
&	AND	x = 5 & 1	0101 & 0001	0001	1
|	OR	x = 5 | 1	0101 | 0001	0101	5
~	NOT	x = ~ 5	~0101	1010	10
^	XOR	x = 5 ^ 1	0101 ^ 0001	0100	4
<<	Left shift	x = 5 << 1	0101 << 1	1010	10
>>	Right shift	x = 5 >> 1	0101 >> 1	0010	2

```
>>> a&b
0
>>> a|b
30
>>> a~b
SyntaxError: invalid syntax
>>> a^b
30
>>> a<<b
10485760
>>> b<<a
20480
>>>
```

Logical Operators

- There are following logical operators supported by Python language. Assume variable a holds 10 and variable b holds 20

Operator	Description
and Logical AND	If both the operands are true then condition becomes true.
or Logical OR	If any of the two operands are non-zero then condition becomes true.

```
>>> a=10
>>> b=20
>>> a==10 and b==10
False
>>> a==10 and b==20
True
>>> a==10 or b==20
True
>>> a==10 or b==10
True
```

Membership operators

- Python’s membership operators test for membership in a sequence, such as strings, lists, or tuples. There are two membership operators

Operator	Description	Example
in	Evaluates to true if it finds a variable in the specified sequence and false otherwise.	x in y, here in results in a 1 if x is a member of sequence y.
not in	Evaluates to true if it does not finds a variable in the specified sequence and false otherwise.	x not in y, here not in results in a 1 if x is not a member of sequence y.

```
>>> a="string"
>>> "i" in a
True
>>> "z" in a
False
>>> "i" not in a
False
>>> "z" not in a
True
>>>
```

Identity Operators

- Identity operators compare the memory locations of two objects.

Operator	Description	Example
is	Evaluates to true if the variables on either side of the operator point to the same object and false otherwise.	x is y, here is results in 1 if id(x) equals id(y).
is not	Evaluates to false if the variables on either side of the operator point to the same object and true otherwise.	x is not y, here is not results in 1 if id(x) is not equal to id(y).

```
>>> a
'string'
>>> b
20
>>> a is b
False
>>> a is not b
True
>>> id(a)
2333172311128
>>> id(b)
140704356459760
>>>
```

Good Job

Next Topic: Data Types