TOPIC 5 : Basic Operators in Python

Introduction.

Operators are the constructs which can manipulate the value of operands. Consider the expression 4 + 5 = 9. Here, 4 and 5 are called operands and + is called operator. Operators are used to perform operations on variables and values. Operators are used to perform operations on values and variables. Operators can manipulate individual items and returns a result. The data items are referred as operands or arguments. Operators are either represented by keywords or special characters.

Objectives

Objectives by the end of this topic you should be able to describe:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Identity operators
- Membership operators
- Bitwise operators

Learning activities

Learning Activity 5.1: Reading

Read further on Bitwise operators.

Learning Activity 5.2: Journal

Summarize the usage of operators from python documentation.

Learning Activity 5.3: Discussion

In groups of three write programs to demonstrate usage od python operators.

Assessment

Topic resources

1. The Python Tutorial¶. (n.d.). Retrieved from https://docs.python.org/3/tutorial/index.html
2. Mueller, J. P. (n.d.). *Beginning Programming with Python For Dummies*. S.l.: For Dummies.
3. (n.d.). Python 3.7.4 documentation. Retrieved from https://docs.python.org/3
4. (n.d.). Git Handbook. Retrieved from https://guides.github.com/introduction/git-handbook/
5. Shaw, Z. (2017). *Learn Python 3 the hard way: a very simple introduction to the terrifyingly beautiful world of computers and code*. Boston: Addison-Wesley.
6. Bader, D. (2018). *Python tricks: the book*. Vancouver, BC: Dan Bader.

7. Downey, A. B. (2015). *Think Python*. Sebastopol: OReilly.
8. Ramalho, L. (2016). *Fluent Python:*Beijing: OReilly.

URL Links

https://www.w3schools.com/python/python_operators.asp
https://docs.python.org/3.4/library/operator.html
https://www.tutorialspoint.com/python/python_basic_operators.htm

TOPIC 5 NOTES

1. **Arithmetic operators:** Arithmetic operators are used to perform mathematical operations like addition, subtraction, multiplication and division.

OPERATOR	DESCRIPTION	SYNTAX
+	Addition: adds two operands	x + y
-	Subtraction: subtracts two operands	x - y
*	Multiplication: multiplies two operands	x * y
/	Division (float): divides the first operand by the second	x / y
//	Division (floor): divides the first operand by the second	x // y
%	Modulus: returns the remainder when first operand is divided by the second	x % y

```
# Examples of Arithmetic Operator
a = 9
b = 4

# Addition of numbers
add = a + b
# Subtraction of numbers
sub = a - b
# Multiplication of number
mul = a * b
# Division(float) of number
div1 = a / b
# Division(floor) of number
div2 = a // b
# Modulo of both number
mod = a % b

# print results
print(add)
print(sub)
print(mul)
print(div1)
print(div2)
print(mod)
```

Output:

```
13

5

36

2.25

2

1
```

2. **Relational Operators:** Relational operators compares the values. It either returns **True** or **False** according to the condition.

OPERATOR	DESCRIPTION	SYNTAX
>	Greater than: True if left operand is greater than the right	x > y
<	Less than: True if left operand is less than the right	x < y
==	Equal to: True if both operands are equal	x == y
!=	Not equal to - True if operands are not equal	x != y
>=	Greater than or equal to: True if left operand is greater than or equal to the right	x >= y
<=	Less than or equal to: True if left operand is less than or equal to the right	x <= y

```
# Examples of Relational Operators
a = 13
b = 33

# a > b is False
print(a > b)

# a < b is True
print(a < b)

# a == b is False
print(a == b)

# a != b is True
print(a != b)

# a >= b is False
print(a >= b)

# a <= b is True
print(a <= b)
```

Output:

```
False

True

False

True

False

True
```

3. **Logical operators:** Logical operators perform **Logical AND**, **Logical OR** and **Logical NOT** operations.

OPERATOR	DESCRIPTION	SYNTAX
and	Logical AND: True if both the operands are true	x and y
or	Logical OR: True if either of the operands is true	x or y
not	Logical NOT: True if operand is false	not x

```
# Examples of Logical Operator
a = True
b = False

# Print a and b is False
print(a and b)

# Print a or b is True
print(a or b)

# Print not a is False
print(not a)
```

Output:

```
False

True

False
```

Bitwise operators: Bitwise operators acts on bits and performs bit by bit operation.

OPERATOR	DESCRIPTION	SYNTAX
&	Bitwise AND	x & y
|	Bitwise OR	x | y
~	Bitwise NOT	~x
^	Bitwise XOR	x ^ y
>>	Bitwise right shift	x>>
<<	Bitwise left shift	x<<

```
# Examples of Bitwise operators
a = 10
b = 4

# Print bitwise AND operation
print(a & b)

# Print bitwise OR operation
print(a | b)

# Print bitwise NOT operation
print(~a)

# print bitwise XOR operation
print(a ^ b)

# print bitwise right shift operation
print(a >> 2)

# print bitwise left shift operation
print(a << 2)
```

Output:

```
0

14

-11

14

2

40
```

4. **Assignment operators:** Assignment operators are used to assign values to the variables.

OPERATOR	DESCRIPTION	SYNTAX
=	Assign value of right side of expression to left side operand	x = y + z
+=	Add AND: Add right side operand with left side operand and then assign to left operand	a+=b a=a+b
-=	Subtract AND: Subtract right operand from left operand and then assign to left operand	a-=b a=a-b
*=	Multiply AND: Multiply right operand with left operand and then assign to left operand	a*=b a=a*b
/=	Divide AND: Divide left operand with right operand and then assign to left operand	a/=b a=a/b
%=	Modulus AND: Takes modulus using left and right operands and assign result to left operand	a%=b a=a%b

//=	Divide(floor) AND: Divide left operand with right operand and then assign the value(floor) to left operand	a//=b	a=a//b
**=	Exponent AND: Calculate exponent(raise power) value using operands and assign value to left operand	a**=b	a=a**b
&=	Performs Bitwise AND on operands and assign value to left operand	a&=b	a=a&b
|=	Performs Bitwise OR on operands and assign value to left operand	a|=b	a=a|b
^=	Performs Bitwise xOR on operands and assign value to left operand	a^=b	a=a^b
>>=	Performs Bitwise right shift on operands and assign value to left operand	a>>=b	a=a>>b
<<=	Performs Bitwise left shift on operands and assign value to left operand	a <<= b	a= a << b

5. **Special operators:** There are some special type of operators like-
   - **Identity operators-**
     is and **is not** are the identity operators both are used to check if two values are located on the same part of the memory. Two variables that are equal does not imply that they are identical.

is	True if the operands are identical
is not	True if the operands are not identical

```
# Examples of Identity operators
a1 = 3
b1 = 3
a2 = 'Programming in Python'
b2 = 'Programming in Python'
a3 = [1,2,3]
b3 = [1,2,3]


print(a1 is not b1)


print(a2 is b2)

# Output is False, since lists are mutable.
print(a3 is b3)
```

Output:

```
False

True

False
```

- **Membership operators**-
  in and **not in** are the membership operators; used to test whether a value or variable is in a sequence.

in	True if value is found in the sequence
not in	True if value is not found in the sequence

```
# Examples of Membership operator
x = 'Prorgamming in Python'
y = {3:'a',4:'b'}


print('P' in x)

print('programming' not in x)

print('Python' not in x)

print(3 in y)

print('b' in y)
```

Output:

```
True
```

True

False

True

False

Revision questions

1. What is the output of the following code : print**(9//2)**
2. What is the answer of this expression, 22 % 3 is?
3. (T/F) x = 8, print(x > 3 or x < 20)
4. Which function overloads the >> operator?
   (A) more()
   (B) gt()
   (C) ge()
   (D) None of the above

5. Which operator is overloaded by the or() function?
   (A) ||
   (B) |
   (C) //
   (D) /
6. What is the output of the following program :

```
i = 0
while i < 3:
      print i
      i++
      print i+1
```

   (A) 0 2 1 3 2 4
   (B) 0 1 2 3 4 5
   (C) Error
   (D) 1 0 2 4 3 5