Programming Fundamentals and Python

Python – Basic Operations

Number and Types

STRING

- Any data type surrounded by quotation mark (either single or double quotation) is coined as string.

MULTILINE STRING

- Multiline string need multi quotation.

STRING LENGTH

- String length includes the alphabets and spaces as well.

CHECK STRING

- You can check string if it's present or not by using keyword 'in'

```
>>> a='string'
>>> print(a)
string
>>> a='''I
        love
                Pakistan'''
>>> print(a)
I
        love
                Pakistan
>>> a='I Love Pakistan'
>>> print(len(a))
15
>>> print('love' in a)
False
>>> print('Love' in a)
True
>>>
```

String Operator


```
>>> s = 'a\nb\tc'
>>> s
'a\nb\tc'
>>> print(s)
a
b c
>>> S = 'Spam' # Make a 4-character
string, and assign it to a name
>>> len(S) # Length
4
>>> S[0] # The first item in S, indexing
        by zero-based position
                'S'
```

Slicing


```
>>> S[1:3] # Slice of S from offsets 1 through 2 (not 3)
'pa'
>>> S[1:] # Everything past the first (1:len(S))
'pam'
>>> S # S itself hasn't changed
'Spam'
```

Strings are *immutable* in Python i.e. they cannot be changed in place after they are created. For example, a string can‘t be changed by assigning to one of its positions, but new string can always be assigned to the same string. Because Python cleans up old objects

```
>>> S
'Spam'
>>> S[0] = 'z' # Immutable objects cannot be changed
...error text omitted...
```

```
>>> S[0] = 'z' # Immutable objects cannot be changed
...error text omitted...
TypeError: 'str' object does not support item assignment
>>> S = 'z' + S[1:] # But we can run expressions to make new objects
>>> S
'zpam'
>>> 'abc' + 'def' # Concatenation: a new string
'abcdef'
>>> 'Ni!' * 4 # Repetition: like "Ni!" + "Ni!" + ...
'Ni!Ni!Ni!Ni!'
```

- In Python, we can also index backward, from the end—positive indexes count from the left, and negative indexes count back from the right:

```
>>> S[-1] # The last item from the end in S
'm'
>>> S[-2] # The second-to-last item from the end
>>> S # A 4-character string
'Spam'
```

Extended Slicing

The third parameter in square bracket defines

- Difference between the indexes to be printed on output
- Direction of access i.e. negative difference define the access direction from right to left

```
s='Computer'
a=s[::-1]
print(a)
#Output:retupmoC
a=s[1:5:1]
print(a)
# Output:ompu
a=s[1:5:2]
print(a)
# Output:op
a=s[5:1:-1]
print(a)
# Output:tupm
```

Data Type Conversion


```
>>> "42" + 1
TypeError: Can't convert 'int' object to str implicitly
>>> int("42"), str(42) # Convert from/to string
(42, '42')
>>> S = "42"
>>> I = 1
>>> S + I
TypeError: Can't convert 'int' object to str implicitly
>>> int(S) + I # Force addition
43
>>> S + str(I) # Force concatenation
'421'
```

Python Operators

Python divides the operators in the following groups:

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

Arithmetic Operators

- It deals with numeric values and common mathematical operations

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

Assignment Operators

- It assign values and assess in operations

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3

**=	x **= 3	x = x ** 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

Comparison Operators

- Comparison between two values can be made by using comparison operators.

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Logical Operators

- Conditional statements requires logical operators.

Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

Identity Operators

- Identity operators are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location:

Operator	Description	Example
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

Identity Operators

- Identity operators are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location:

Operator	Description	Example
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

Membership Operators

- Membership operators are used to test if a sequence is presented in an object:

Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

Bitwise Operators

- It is used in binary number operations.

Operator	Name	Description
&	AND	Sets each bit to 1 if both bits are 1
|	OR	Sets each bit to 1 if one of two bits is 1
^	XOR	Sets each bit to 1 if only one of two bits is 1
~	NOT	Inverts all the bits
<<	Zero fill left shift	Shift left by pushing zeros in from the right and let the leftmost bits fall off
>>	Signed right shift	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off

Python Operators

- Operations are mainly performed on values and variables.
- Routine mathematical operations like subtraction, multiplication and division can be performed in the similar way as addition operation performed below:

```
>>> 123+456 #Addition
579
>>> 123**2 #Power
15129
>>> 2.0 >= 1 # Greater than or equal: mixed-type 1 converted to 1.0
True
>>> 2.0 == 2.0 # Equal value
True
>>> 2.0 != 2.0 # Not equal value
False
```

Basic arithmetic operator examples

Add

Subtract

Multiply

Divide

Floor Division

Exponent

```
x = 15
y = 4

# Output: x + y = 19
print('x + y =',x+y)

# Output: x - y = 11
print('x - y =',x-y)

# Output: x * y = 60
print('x * y =',x*y)

# Output: x / y = 3.75
print('x / y =',x/y)

# Output: x // y = 3
print('x // y =',x//y)

# Output: x ** y = 50625
print('x ** y =',x**y)
```

Output

```
x + y = 19
x - y = 11
x * y = 60
x / y = 3.75
x // y = 3
x ** y = 50625
```

Comparison operator examples

Greater than

Less than

Equal to

Not Equal to

Greater than or equal t

Less than or equal to

```
x = 10
y = 12

# Output: x > y is False
print('x > y is',x>y)

# Output: x < y is True
print('x < y is',x<y)

# Output: x == y is False
print('x == y is',x==y)

# Output: x != y is True
print('x != y is',x!=y)

# Output: x >= y is False
print('x >= y is',x>=y)

# Output: x <= y is True
print('x <= y is',x<=y)
```

Output

```
x > y is False
x < y is True
x == y is False
x != y is True
x >= y is False
x <= y is True
```

Logical operator examples

And

Or

Not

```
x = True
y = False

print('x and y is',x and y)

print('x or y is',x or y)

print('not x is',not x)
```

Output

```
x and y is False
x or y is True
not x is False
```

Identity operator examples

Is

Is not

```
x1 = 5
y1 = 5
x2 = 'Hello'
y2 = 'Hello'
x3 = [1,2,3]
y3 = [1,2,3]

# Output: False
print(x1 is not y1)

# Output: True
print(x2 is y2)

# Output: False
print(x3 is y3)
```

Output

```
False
True
False
```

Membership operator examples

In

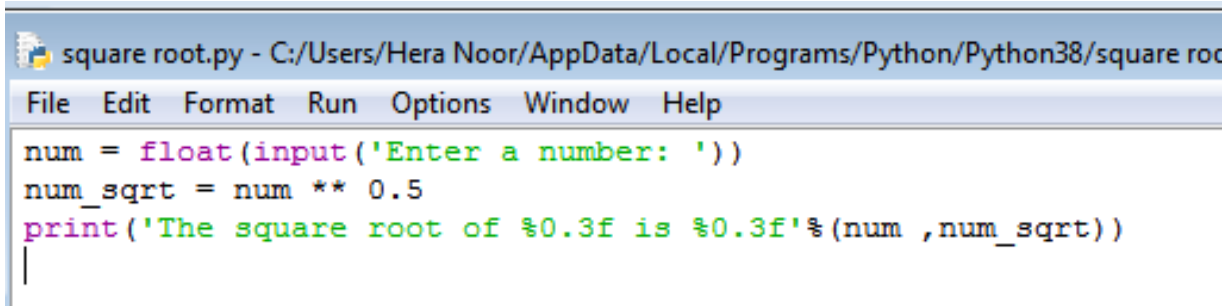
In not

```
x = 'Hello world'
y = {1:'a',2:'b'}

# Output: True
print('H' in x)

# Output: True
print('hello' not in x)

# Output: True
print(1 in y)

# Output: False
print('a' in y)
```

Output

```
True
True
True
False
```

Square Root



OUTPUT