TOPIC : Artificial Intelligence with Python - INMOVIDU - Jahnavi N

GETTING STARTED WITH PYTHON PROGRAMMING

1) printing , Comments

In [1]:
```
print("hai aspirants")
```

```
hai aspirants
```

In [2]:
```
print("hello")
print("world")
```

```
hello
world
```

In [3]:
```
print("hai",end=" ")
print("aspirants ")
```

```
hai aspirants
```

In [4]:
```
print("hai\naspirants")
```

```
hai
aspirants
```

In [ ]:
```
# now a is printed
#three comment
#print("hai")
""" comment """
```

we could even go the options available such as marksown, raw notebook convert or heading when required

2) Format -functionality for complex variable substitutions and value formatting.

In [6]:
```
x=100
```

In [7]:
```
print("we are ",x)
```

```
we are  100
```

In [8]:
```
print("{0} am {1}!!".format("you","aspirant","here ") )
```

```
you am aspirant!!
```

In [9]:
```
x=10
y=20
print("value x is {0} and y is {1}".format(x,y))
```

```
value x is 10 and y is 20
```

3) VARIABLES

A Python variable is a reserved memory location to store values. Python has no command for declaring a variable. A variable is created the moment you first assign a value to it "SO PYTHON IS DYNAMICALLY TYPED"

x=100 x will be the variable as it is storing the data

4) DATA TYPES

In [10]:
```
###############int
x=20
y=100
print(x,y)
```

```
20 100
```

In [11]:
```
##############float
x=3.2
print(type(x))
print(x)

print("type of x is ",type(x))
print("type of {} is {}".format(x,type(x)))
```

```
<class 'float'>
3.2
type of x is  <class 'float'>
type of 3.2 is <class 'float'>
```

In [12]:
```
################String
s="CORONA "
k=str("2021")
print(s,k)
```

```
CORONA  2021
```

In [13]:
```
s='a'
print(type(s))
```

```
<class 'str'>
```

In python there is no character data type, a character is a string of length one. It is represented by str class

In [16]:
```
###############BOOLEAN
z=True
y=False

print(type(z))
```

```
<class 'bool'>
```

we cannot store our values in reserved words for example True = 20 is in correct

5) RESERVED WORDS

In [17]:
```
from IPython import display
display.Image("reserved.jpg")
```

Out[17]:

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

all the above are reserved words

6) OPERATORS

In [7]:
```
from IPython import display
display.Image("operators.jpg")
```

Out[7]:

ARITHMETIC OPERATORS

1) + Addition x+y

2) - Subtraction x-y

3) *Multiplication xy*

4) / Division x/y

5) % Modulus x%y

6) **Exponentiation xy**

7) // Floor Division x//y

In [19]:

```
a=10
b=2
print(a**b)
```

```
9
```

In [1]:
```
a=10
b=2
print(a%b)
```

```
0
```

In [2]:
```
a=10
b=2
print(a*b)
```

```
20
```

RELATIONAL OPERATORS OR COMPARISION OPERATORS

Relational operators are used to compare two variables. It will return true , if the comparison or relation is true. otherwise it will return false.

In [20]:
```
from IPython import display
display.Image("http://www2.hawaii.edu/~takebaya/cent110/selection/relational_operato
```

Out[20]:

Operator	Description
>	greater than
<	less than
==	equal to
<=	less or equal to
>=	greater or equal to
!=	lnot equal to

In [21]:
```
a = 500
b = 500
print(a!=b)
```

```
False
```

In [22]:
```
print(a != b)
print(a < b)
print(a <= b)
print(a > b)
print(a >= b)
```

```
False
False
True
False
True
```

ASSIGNMENT OPERATORS

Assignment operators are used in Python to assign values to variables.

In [8]:
```
display.Image("https://www.engineeringbigdata.com/wp-content/uploads/assignment-oper
```

Out[8]:

Python Assignment Operators		
Operator	**Example**	**Equal to**
=	a = 20	a = 20
+=	a += b	a = a + b
-=	a -= b	a = a - b
*=	a *= b	a = a * b
/=	a /= b	a = a / b
%=	a %= b	a = a % b
//=	a //= b	a = a // b
**=	a **= b	a = a ** b
&=	a &= b	a = a & b
|=	a |= b	a = a | b
^=	a ^= b	a = a ^ b
>>=	a >>= b	a = a >> b
<<=	a <<= b	a = a << b

In [24]:
```
a=2
b=4


a**=b
print(a)
```

```
16
```

In [3]:
```
a=2
b=4
a=a**b
print(a)
```

```
16
```

#BOTH OF THE ABOVE CELLS GIVES SAME ANSWER IN ALL ASSIGNMENT OPERATIONS WE DO USE AN " = " SYMBOL BECAUSE ONLY IT COULD ASSIGN THE VALUE

BITWISE OPERATORS

& Bitwise AND x & y

| Bitwise OR x | y

~ Bitwise NOT ~x

^ Bitwise XOR x ^ y

>> Bitwise right shift x>>

<< Bitwise left shift x<<

AND , OR

& Returns 1 if both the bits are 1 else 0.

| Returns 1 if either of the bit is 1 else 0.

In [9]:
```
display.Image("https://introcs.cs.princeton.edu/java/71boolean/images/truth-table.pn
```

Out[9]:

NOT

x	*x'*
0	*1*
1	*0*

AND

x	*y*	*xy*
0	*0*	*0*
0	*1*	*0*
1	*0*	*0*
1	*1*	*1*

OR

x	*y*	*x+y*
0	*0*	*0*
0	*1*	*1*
1	*0*	*1*
1	*1*	*1*

XOR

x	*y*	*x⊕y*
0	*0*	*0*
0	*1*	*1*
1	*0*	*1*
1	*1*	*0*

----or exmaple----------

a = 2 = 0010 (Binary)

b = 3 = 0011 (Binary)

a|b = 0011=3

---------and example-------

a = 2 = 0010 (Binary)

b = 3 = 0011 (Binary)

a|b = 0010=2

In [29]:
```
a=2
b=3
```

In [30]:
```
print(a|b)
```

```
3
```

In [31]:
```
a=2
b=3
print(a&b)
```

```
2
```

left and right shift

Shifts the bits of the number to the left and fills 0 on voids left as a result.

In [4]:
```
a=4
a=a>>2
print(a)
```

```
1
```

In [33]:
```
a=2

a=a<<1
print(a)
```

```
4
```

NOT ~

Returns one's compliement of the number. Example: a = 10 = 1010 (Binary) ~a = ~1010 = -(1010 + 1) = -(1011) = -11

In [34]:
```
print(~10)
```

```
-11
```

XOR

Bitwise xor operator: Returns 1 if one of the bit is 1 and other is 0 else returns false

In [36]:
```
print(2^3)
```

```
1
```

LOGICAL OPERATORS

Logical operators in Python are used for conditional statements are true or false.

For AND operator – It returns TRUE if both the operands (right side and left side) are true

For OR operator- It returns TRUE if either of the operand (right side or left side) is true

For NOT operator- returns TRUE if operand is false

In [37]:
```
a = 0
b = 1
print('a and b is',a and b)
print(('a or b is',a or b))
print(('not a is',not a))
```

```
a and b is 0
('a or b is', 1)
('not a is', True)
```

In [38]:
```
a=10
b=2

print(a>3 or b<1)
```

```
True
```

In [10]:
```
a=10
b=2

print(a>3 and b<1)
```

```
False
```

In [14]:
```
b=1 # TRUE
print(not b)
```

```
False
```

In [15]:
```
b=0
print(not b)
```

```
True
```

-------- ##########--------

In [ ]:
```

```