Operators

GITAA
Transforming careers

In this lecture

- Operators and operands
- Different types of operators
  - Arithmetic
  - Assignment
  - Relational or comparison
  - Logical
  - Bitwise
- Precedence of operators

Operators and operands

- Operators are special symbols that help in carrying out an assignment operation or arithmetic or logical computation
- Value that the operator operates on is called operand

```
In [1]: 2+3
Out[1]: 5
```

Arithmetic operators

- Used to perform mathematical operations between two operands
- Create two variable a and b with values 10 and 5 respectively

```
a=10    b=5
```

Symbol	Operation	Example
+	Addition	In [3]: a+b Out[3]: 15

Arithmetic operators

- Used to perform mathematical operations between two operands
- Create two variable a and b with values 10 and 5 respectively

```
a=10    b=5
```

Symbol	Operation	Example
+	Addition	In [3]: a+b Out[3]: 15
-	Subtraction	In [4]: a-b Out[4]: 5

Arithmetic operators

Symbol	Operation	Example
*	Multiplication	In [5]: a*b Out[5]: 50

Arithmetic operators

Symbol	Operation	Example
*	Multiplication	In [5]: a*b Out[5]: 50
/	Division	In [6]: a/b Out[6]: 2.0

Arithmetic operators

Symbol	Operation	Example
*	Multiplication	In [5]: a*b Out[5]: 50
/	Division	In [6]: a/b Out[6]: 2.0
%	Remainder	In [8]: a%b Out[8]: 0

Arithmetic operators

Symbol	Operation	Example
*	Multiplication	In [5]: a*b Out[5]: 50
/	Division	In [6]: a/b Out[6]: 2.0
%	Remainder	In [8]: a%b Out[8]: 0
**	Exponent	In [7]: a**b Out[7]: 100000

Hierarchy of arithmetic operators

Decreasing order of precedence	Operation
Parentheses	( )
Exponent	**
Division	/
Multiplication	*
Addition and subtraction	+,-

$$A = 7 - 2 \times \frac{27}{3^2} + 4$$

```
In [10]: A=7-2*(27/3**2)+4

In [11]: print(A)
5.0
```

Assignment operators

- Used to assign values to variables

Symbol	Operation	Example
=	Assign values from right side operands to left side operand	a=10 b=5

Assignment operators

- Used to assign values to variables

Symbol	Operation	Example
=	Assign values from right side operands to left side operand	a=10 b=5
+=	Adds right operand to left operand and stores result on left side operand (a=a+b)	In [55]: a+=b ...: print(a) 15

Assignment operators

- Used to assign values to variables

Symbol	Operation	Example
=	Assign values from right side operands to left side operand	a=10 b=5
+=	Adds right operand to left operand and stores result on left side operand (a=a+b)	In [55]: a+=b ...: print(a) 15
-=	Subtracts right operand from left operand and stores result on left side operand (a=a-b)	In [57]: a-=b ...: print(a) 5

Assignment operators

Symbol	Operation	Example
*=	Multiplies right operand from left operand and stores result on left side operand (a=a*b)	In [59]: a*=b ...: print(a) 50

Assignment operators

Symbol	Operation	Example
*=	Multiplies right operand from left operand and stores result on left side operand (a=a*b)	In [59]: a*=b ...: print(a) 50
/=	Divides right operand from left operand and stores result on left side operand (a=a/b)	In [62]: a/=b ...: print(a) 2.0

Relational or comparison operators

- Tests numerical equalities and inequalities between two operands and returns a boolean value
- All operators have same precedence
- Create two variables x and y with values 5 and 7 respectively

```
In [1]: x = 5
In [2]: y = 7
```

Symbol	Operation	Example
<	Strictly less than	In [3]: print(x<y) True

Relational or comparison operators

- Tests numerical equalities and inequalities between two operands and returns a boolean value
- All operators have same precedence
- Create two variables x and y with values 5 and 7 respectively

```
In [1]: x = 5
In [2]: y = 7
```

Symbol	Operation	Example
<	Strictly less than	In [3]: print(x<y) True
<=	Less than equal to	In [4]: print(x<=y) True

Relational or comparison operators

Symbol	Operation	Example
>	Strictly greater than	In [5]: print(x>y) False
>=	Greater than equal to	In [6]: print(x>=y) False

Relational or comparison operators

Symbol	Operation	Example
>	Strictly greater than	In [5]: print(x>y) False
>=	Greater than equal to	In [6]: print(x>=y) False
==	Equal to equal to	In [7]: print(x==y) False

Relational or comparison operators

Symbol	Operation	Example
>	Strictly greater than	In [5]: print(x>y) False
>=	Greater than equal to	In [6]: print(x>=y) False
==	Equal to equal to	In [7]: print(x==y) False
!=	Not equal to	In [8]: print(x!=y) True

Logical operators

- Used when operands are conditional statements and returns boolean value
- In python, logical operators are designed to work with scalars or boolean values

Symbol	Operation	Example
or	Logical OR	In [9] print((x>y) or (x<y)) True

Logical operators

- Used when operands are conditional statements and returns boolean value
- In python, logical operators are designed to work with scalars or boolean values

Symbol	Operation	Example
or	Logical OR	In [9] print((x>y) or (x<y)) True
and	Logical AND	In [10]: print((x>y) and (x<y)) False

Logical operators

- Used when operands are conditional statements and returns boolean value
- In python, logical operators are designed to work with scalars or boolean values

Symbol	Operation	Example
or	Logical OR	`In [9] print((x>y) or (x<y))` `True`
and	Logical AND	`In [10]: print((x>y) and (x<y))` `False`
not	Logical NOT	`In [11]: print(not (x==y))` `True`

Bitwise operators

- Used when operands are integers
- Integers are treated as a string of binary digits
- Operates bit by bit
- Can also operate on conditional statements which compare scalar values or arrays
- Bitwise OR (|), AND(&)

Bitwise operators

- Create two variables x and y with values 5 and 7 respectively

```
In [1]: x = 5
In [2]: y = 7
```

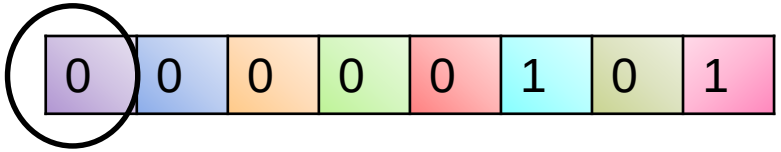
- Binary code for 5 is ***0000 0101*** and for 7 is ***0000 0111***
- ***0*** corresponds to ***False*** and ***1*** corresponds to ***True***
- In bitwise OR (|), operator copies a bit to the result if it exists in either operand
- In bitwise AND (&), operator copies a bit to the result if it exists in both operands

Bitwise OR on integers

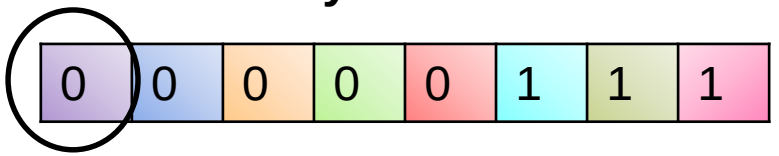
Code and output in console

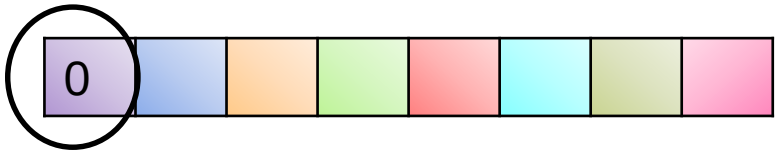
```
In [47]: x | y
Out[47]: 7
```

Binary code for 5



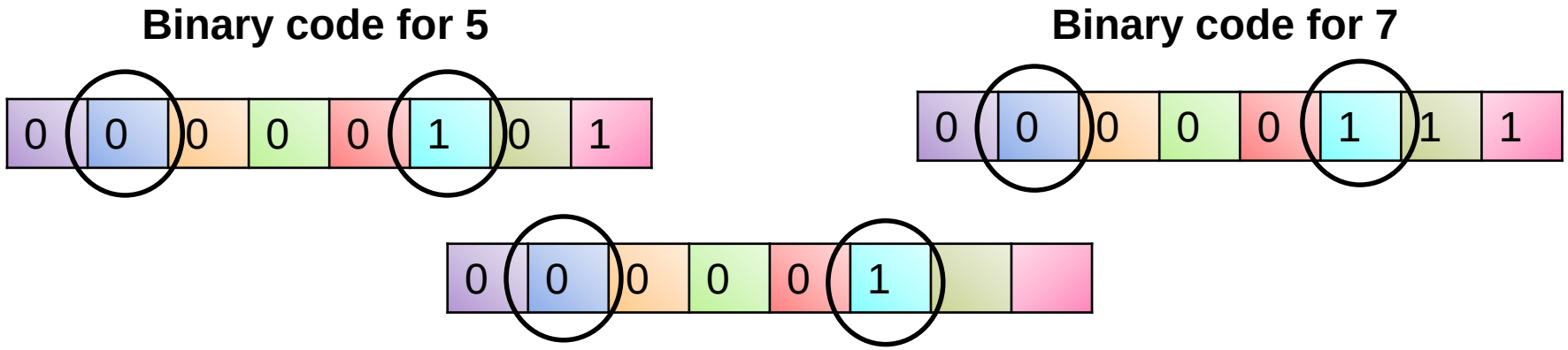
Binary code for 7





- 0 present in corresponding positions, therefore resultant cell is also 0

Bitwise OR on integers

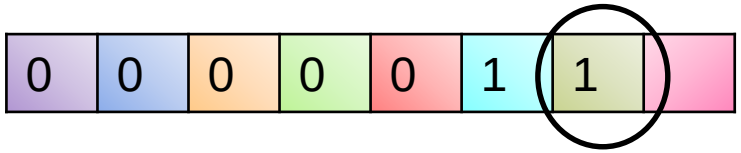


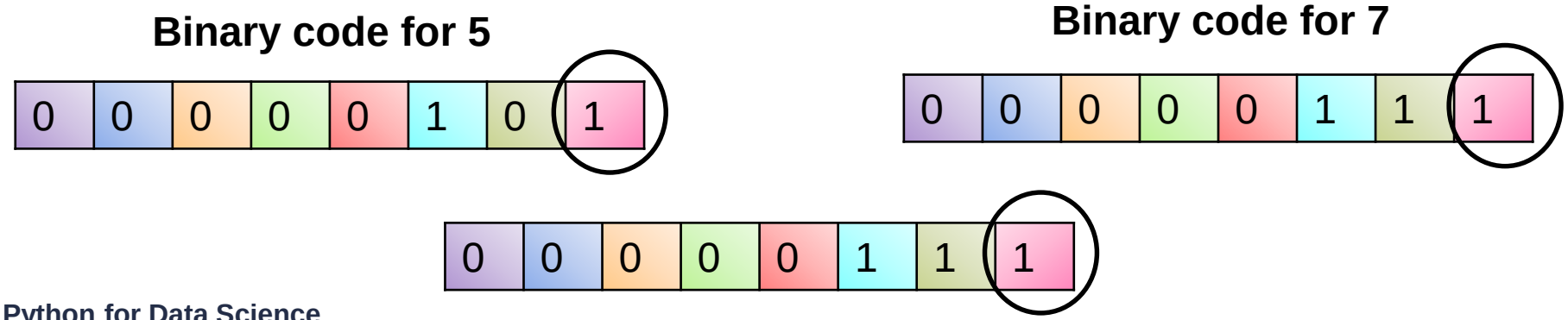
- 0 present in positions 2-5, therefore resultant cell will also contain 0
- In the 6th position, 1 is present in both operands and hence resultant will also contain 1

Bitwise OR on integers

- The 7th position has 0 in the first operand and 1 in the second

- Since this is an OR operator, only the True condition is considered





Bitwise operators

- Bitwise operators can also operate on conditional statements

Symbol	Operation	Example
|	Bitwise OR	In [9]: print((x<y)|(x==y)) True

Bitwise operators

- Bitwise operators can also operate on conditional statements

Symbol	Operation	Example
|	Bitwise OR	In [9]: print((x<y)|(x==y)) True
&	Bitwise AND	In [10]: print((x<y)&(x==y)) False

Precedence of operators

Decreasing order of precedence	Operation
Parentheses	( )
Exponent	**
Division	/
Multiplication	*
Addition and subtraction	+,-
Bitwise AND	&

Precedence of operators

Decreasing order of precedence	Operation
Bitwise OR	|
Relational/ comparison operators	==, !=, >, >=, <, <=
Logical NOT	not
Logical AND	and
Logical OR	or

Summary

- Important operators
  - Arithmetic
  - Assignment
  - Relational
  - Logical
  - Bitwise


operation == "MIRROR_X":
irror_mod.use_x = True
irror_mod.use_y = False
irror_mod.use_z = False
operation == "MIRROR_Y":
irror_mod.use_x = False
irror_mod.use_y = True
irror_mod.use_z = False
operation == "MIRROR_Z":
irror_mod.use_x = False
irror_mod.use_y = False
irror_mod.use_z = True
selection at the end -add
_ob.select= 1
er_ob.select=1
ntext.scene.objects.acti
("Selected" + str(modifier
irror_ob.select = 0
bpy.context.selected_ob
ata.objects[one.name].se
int("please select exact

THANK YOU