

COMP1021  
Introduction to Computer Science

# More on Operators

David Rossiter and Gibson Lam

# Outcomes

- After completing this presentation, you are expected to be able to:
  1. Explain the use of the various kinds of Python operators
  2. Write code to represent `True` or `False` using numbers, lists, tuples or strings
  3. Apply operator precedence in expressions

# Python Operators

- We already know we can do common maths things in Python, i.e. + - / \*

```
print(100 - 25 * 4 + 120 / 5)
```



```
24.0
```

- These things are called *operators*
- This presentation gives you summaries of different types of operators
- You have already used most of them
- We will also look at some related things

# Arithmetic Operators

- Basic operators:

+ - / \* %

- ‘Advanced’ operators:

\*\* means ‘to the power of’

// means ‘do division,  
return the integer result’

-x means the same as ‘-1 \* x’

2\*\*3

8

3\*\*2

9

3//3

1

4//3

1

5//3

1

6//3

2

7//3

2

8//3

2

x=10

-x

-10

# Comparison Operators

*Reminder*

- For comparing two values:

`a < b` returns `True` if `a` is less than `b`

`a <= b` returns `True` if `a` is less than or equal to `b`

`a > b` returns `True` if `a` is greater than `b`

`a >= b` returns `True`  
if `a` is greater than or equal to `b`

`a == b` returns `True` if `a` is equal to `b`

`a != b` returns `True` if `a` is not equal to `b`

- All of them return `False` otherwise

# Logical Operators

*Reminder*

- Logical operators work with Boolean values, i.e.

True or False

$a$  and  $b$       if both condition  $a$  and condition  $b$  are True,  
the result is True; otherwise, it's False

$a$  or  $b$         if either condition  $a$  or condition  $b$  is True,  
the result is True; otherwise, it's False

not  $a$           if  $a$  is True, then the result is False;  
if  $a$  is False, then the result is True

# Summary

*Reminder*

- Here is a summary of the input and output:

<b>a</b>	<b>b</b>	<b>a and b</b>	<b>a or b</b>	<b>not a</b>
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True

# Using Other Things as True/False

- Any number other than 0 means `True`
  - 0 means `False`
- An empty list `[]`, tuple `()` or string `""` means `False`
  - Non-empty list/ tuple/ string means `True`



# Using Other Things as True/False

*Python sees  
this as True*

```
if "^o^":  
    print("yes")
```



yes

*Python sees this empty  
string as False*

```
if "":  
    print("yes")
```



*nothing is printed*

# Using the Equals Sign

- You use the equals sign to put things into a variable, i.e. `age = 25`
- Sometimes you may want to do something like this (adding one to the variable `count`):


```
count = count + 1
```


- When you are doing something to the **same** variable Python has a shortcut, like this:

```
count += 1
```

# Using Shortcuts with the Equals Sign

- You can use the equals sign with most arithmetic operators, for example:


`calories = calories + 800`  `calories += 800`

`pigs = pigs * 5`  `pigs *= 5`

`cakes = cakes / students`  `cakes /= students`

`marks = marks - 20`  `marks -= 20`

`hello = hello + "!"`  `hello += "!"`

 *As you can see, this works for strings too,  
not just numerical values*

# Operators for Lists, Tuples and Strings

- These operators are used by lists, tuples and strings:

`x + y` concatenates (=put together) two lists, tuples or strings

`x * n` concatenates `n` copies of `x`

`a in x` returns `True` if `a` is in collection `x` and `False` otherwise

`a not in x` returns `False` if `a` is in collection `x` and `True` otherwise

# Using 'in' with Strings

- Using the `in` operator you can test for a string inside another string, like this:

```
if "shark" in "baby shark dance":  
    print("yes")
```



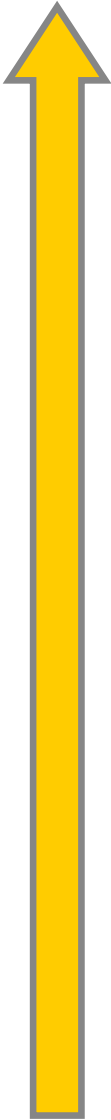
yes

# Operator Precedence

- If we ask Python to calculate  $2 + 3 * 4$  what will the result be?
  - You might think the answer is  $5 * 4$  which is 20
  - You are wrong!
  - This is because  $*$  has *precedence* over  $+$
  - So  $3 * 4$  will be calculated first, then the result (12) will be added to 2, so the answer is 14
- If you always use brackets, e.g.  $2 + (3 * 4)$ , then you don't need to worry about precedence, but you need to understand what happens when there aren't any brackets

# The Precedence Table

Increasing precedence



- *Highest precedence* -

( )

\* \*

-x, +x

\*, /, %, //

+, -

<, >, <=, >=, !=, ==

in, not in

logical not

logical and


logical or

- *Lowest precedence* -



*So if you use  
brackets ()  
they override  
everything*

# Precedence Example 1

$$x = 17 / 2 * 3 + 2$$


- / and \* have higher precedence than +, so they are handled first
- / and \* have equal precedence, so the one on the left (/) is evaluated first

- So the answer is:

$$\begin{aligned} &= ( (17 / 2) * 3 ) + 2 \\ &= 27.5 \end{aligned}$$



# Precedence Example 2

$$x = 19 \% 4 + 15 / 2 * 3$$


- $\%$ ,  $/$  and  $*$  have higher precedence than  $+$ , so they are handled first
- $\%$ ,  $/$  and  $*$  have equal precedence, so the one on the left is evaluated first, which is  $\%$ , then  $/$ , then  $*$

- So the answer is:

$$\begin{aligned} &= (19 \% 4) + ((15 / 2) * 3) \\ &= 25.5 \end{aligned}$$

# Precedence Example 3

$$x = 17 / 2 \% 2 * 3 * 3$$


- $**$  has a higher precedence than the others, so it is handled first
- $/$ ,  $\%$ , and  $*$  have equal precedence, so the one on the left ( $/$ ) is evaluated first, then  $\%$ , then  $*$

- So the answer is:

$$= ( (17 / 2) \% 2 ) * (3 * 3)$$

$$= ( (17 / 2) \% 2 ) * 27$$

$$= 13.5$$

# Precedence Example 4

```
english_is_spoken = True
need_visa = False
married_to_singapore_person = False
want_to_visit_singapore = True
visit_singapore = english_is_spoken \
    and not need_visa or married_to_singapore_person \
    and want_to_visit_singapore
print(visit_singapore)
```

- What is printed?

- *Highest precedence* -

...



logical not

logical and

logical or

- *Lowest precedence* -

# Precedence Example 4

```
english_is_spoken = True
need_visa = False
married_to_singapore_person = False
want_to_visit_singapore = True
visit_singapore = (english_is_spoken \
    and (not need_visa)) or (married_to_singapore_person \
    and want_to_visit_singapore)
```

• Here brackets have been added to indicate the order

- *Highest precedence* -

...



logical not

logical and

logical or

- *Lowest precedence* -

**(True and (not False)) or (False and True)**

# Precedence Example 4

```
(True and (not False)) or (False and True)
= (True and True) or (False and True)
= True or False
= True
```

```
visit_singapore = (english_is_spoken \
    and (not need_visa)) or (married_to_singapore_person \
    and want_to_visit_singapore)
```

• Here brackets have been added to indicate the order

```
print(visit_singapore)
```