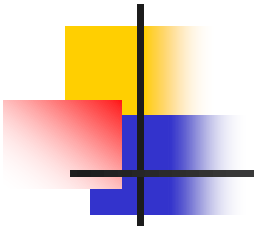


Introduction to Programming

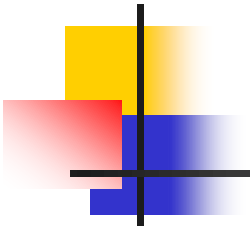
Department of Computer Science and Information Systems

Lecturers: Tingting Han and Steve Maybank

sjmaybank@dcs.bbk.ac.uk

Autumn 2019 and Spring 2020

Week 6: Relational Operators and Boolean Variables

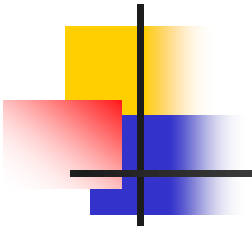


Revision: Strings

- String literals:
  - "Hello", 'World!'
- Length: `len("Hello")`
  - # value 5
- Convert a number to a string:
  - `str(5)`

    # value "5"
  - `str(34.2)`

    # value "34.2"
- String concatenation: `"H"+"W"`
  - # value "HW"



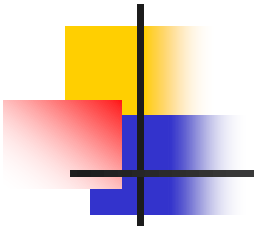
Revision: Strings Indexing

- How are strings indexed?
  - From left to right, starting from 0
- How are individual characters obtained?
  - using [index], e.g. "Cakes"[1]
  - # value "a"
- How to obtain individual characters using negative indices?
  - "Cakes"[-3]
  - # value "k"
- Valid indices for "Cakes"
  - -5? 5? 0? 2.0?
  - -5, -4, -3, -2, -1, 0, 1, 2, 3, 4

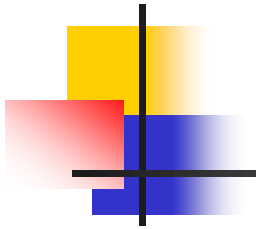
Revision: Escape Sequences

- \"
  - include the character double quote in a string
  - e.g. "a\"b" len("a\"b")=? print("a\"b") ?
  - len("a\"b")= 3, result is a"b
- \n
  - new line
  - e.g. "*\n*", len("*\n*")=? print("*\n*") ?
  - len("*\n*")=3 result is *
*
- \\
  - include the character backslash in a string
  - e.g. "a\\b" len("a\\b")=? print("a\\b") ?
  - len("a\\b")= 3, result is a\b

Revision: Format Specifiers

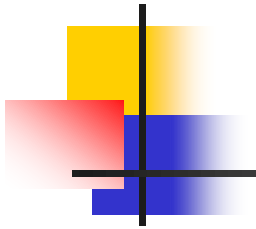


- %5d, e.g. print("%5d" % 56)
  - # three spaces then 56
  - %5d place an integer right justified in a field of 5 characters
- %8.2f, e.g. print("%8.2f" % -586.189)
  - # one space then -586.19
  - %8.2f place a floating point number with two digits after the decimal point right justified in a field of 8 characters. The decimal point and the – sign, if present, each count as characters
- %-9s, e.g. print("%-9s" % "Hello")
  - # Hello then four spaces
  - %-9s place a string left justified in a field of 9 characters



Relational Operators

Python	Math Notation	Description



Relational Operators

Python	Math Notation	Description
>	>	Greater than
>=	≥	Greater than or equal

Relational Operators

Python	Math Notation	Description
>	>	Greater than
>=	≥	Greater than or equal
<	<	Less than
<=	≤	Less than or equal

Relational Operators

Python	Math Notation	Description
>	>	Greater than
>=	≥	Greater than or equal
<	<	Less than
<=	≤	Less than or equal
==	=	Equal
!=	≠	Not equal

The result of the comparing two values using relational operators: True or False

Examples of Relational Operators

- 3<=4
  - True
- 3=<4
  - Error, use <=, not =<
- 3>4
  - False
- 4<4
  - False
- 4<=4
  - True
- 3==5-2
  - True
- 3!=5-1
  - True
- 3=6/2
  - Syntax error, use == to test for equality
- 3==6/2
  - True
- 1.0/3.0 == 0.333333333
  - False, the values are close, but not exactly equal
  - 1.0/3.0 == 0.333333333333333 is True
- "10" > 5
  - Error
  - A string cannot be compared with a number

Relational Operators and Strings

```
name1 = "John"
name2 = "John"
name3 = "Smith"
```

- `name1 == name2`

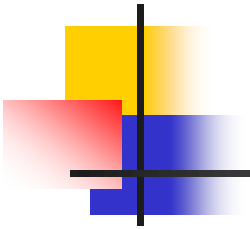
True

- `name1 == name3`

False

- `name1 != name3`

True



Ordering of Single Characters

- All uppercase letters come before lowercase letters
- Numbers come before letters
- The space character comes before all printable characters
- Empty string comes before all non-empty characters

- Example

"" < " " < "0" < "1" < "9" < "A" < "B" < "Z" < "a" < "b" < "z"

Use `ord()` function to check the value of a character.
E.g., `ord("A")` is 65, `ord(" ")` is 32.

ASCII Chart

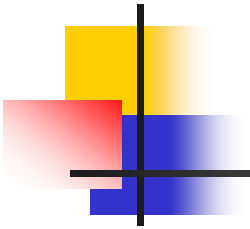
Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(	72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29	)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-_	63	3F	?	95	5F	_	127	7F	DEL

https://www.commfront.com/pages/ascii-chart

Extended ASCII Chart

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	á	192	C0	└	224	E0	α
129	81	ü	161	A1	í	193	C1	┴	225	E1	ß
130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
131	83	â	163	A3	ú	195	C3	├	227	E3	π
132	84	ä	164	A4	ñ	196	C4	─	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
134	86	å	166	A6	ª	198	C6	╞	230	E6	µ
135	87	ç	167	A7	º	199	C7	╟	231	E7	τ
136	88	ê	168	A8	¿	200	C8	╚	232	E8	Φ
137	89	ë	169	A9	⌐	201	C9	╔	233	E9	Θ
138	8A	è	170	AA	¬	202	CA	╩	234	EA	Ω
139	8B	ï	171	AB	½	203	CB	╦	235	EB	δ
140	8C	î	172	AC	¼	204	CC	╠	236	EC	∞
141	8D	ì	173	AD	¡	205	CD	═	237	ED	φ
142	8E	Ä	174	AE	«	206	CE	╬	238	EE	ε
143	8F	Å	175	AF	»	207	CF	╧	239	EF	∩
144	90	É	176	B0	░	208	D0	╨	240	F0	≡
145	91	æ	177	B1	▒	209	D1	╤	241	F1	±
146	92	Æ	178	B2	▓	210	D2	╥	242	F2	≥
147	93	ô	179	B3	│	211	D3	╙	243	F3	≤
148	94	ö	180	B4	┤	212	D4	╘	244	F4	⌠
149	95	ò	181	B5	╡	213	D5	╒	245	F5	⌡
150	96	û	182	B6	╢	214	D6	╓	246	F6	÷
151	97	ù	183	B7	╖	215	D7	╫	247	F7	≈
152	98	ÿ	184	B8	╕	216	D8	╪	248	F8	°
153	99	Ö	185	B9	╣	217	D9	┘	249	F9	∙
154	9A	Ü	186	BA	║	218	DA	┌	250	FA	·
155	9B	¢	187	BB	╗	219	DB	█	251	FB	√
156	9C	£	188	BC	╝	220	DC	▄	252	FC	ⁿ
157	9D	¥	189	BD	╜	221	DD	▌	253	FD	²
158	9E	Pts	190	BE	╛	222	DE	▐	254	FE	■
159	9F	ƒ	191	BF	┐	223	DF	▀	255	FF	

https://www.commfront.com/pages/ascii-chart



Lexicographic Ordering of Strings

- Python's relational operators compare strings in lexicographic order.
  - Lexicographic order: similar to the way in a dictionary
  - string1 < string2, if string1 comes before string2 in a dict.
    - "Hammer"<"Hello"
  - string1 == string2, if string1 are string2 are identical
- How does Python compare strings?
  - E.g. "catch" and "cart"?
  - "coal" and "coat"?
  - "tone" and "ton"?

Summary of Lexicographic Ordering

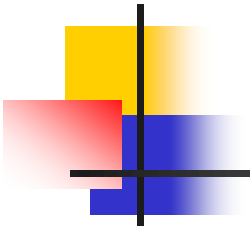
- Given strings s1, s2, find the longest string s such that

  s1 = s + u1 s2 = s + u2

  s is the longest common string between s1 and s2

- If u1 == u2 == "", then s1 == s2 (see Example 5)
- If u1 == "" and u2 ! = "", then s1 < s2 (see Example 4)
- If u1 != "" and u2 == "", then s1 > s2 (see Example 3)
- If u1 != "" and u2 != "", then
- if u1[0] < u2[0] then s1 < s2 (see Example 2)
- if u1[0] > u2[0] then s1 > s2 (see Example 1)

	s1	s2	u1	u2	compare	order
Example 1	"catch"	"cart"	"tch"	"rt"	"t" > "r"	s1 > s2
Example 2	"coal"	"coats"	"l"	"ts"	"l" < "t"	s1 < s2
Example 3	"tone"	"ton"	"e"	""	"e" > ""	s1 > s2
Example 4	"pit"	"pith"	""	"h"	"" < "h"	s1 < s2
Example 5	"pitch"	"pitch"	""	""	"" == ""	s1 == s2



Boolean Variables

- Variables of type bool have the value True or the value False, e.g.

  failed = False

  passed = True

- True and False are special values, not numbers or strings.
- True and False are reserved words
  - What about true and false?

Boolean Operators

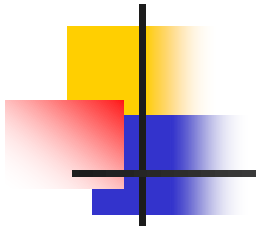
- A Boolean operator takes one or more Boolean values as input and produces a Boolean value as output.
- Example: `and`

  input: two Boolean values `True`, `True`

  output: `True`

  `and` yields `True` if **both** inputs are `True`

- Wave the flag when the road is clear and the cars are ready
  - `flag = True and True`
  - The Boolean variable `flag` has the value `True`



Truth Tables

A	B	A `and` B
True	True	True
True	False	False
False	True	False
False	False	False

Boolean Operators

- A Boolean operator takes one or more Boolean values as input and produces a Boolean value as output.
- Example: `or`

  input: two Boolean values `True`, `False`

  output: `True`

  `or` yields `True` if at least one input is `True`

- Fail the game if bumping into a poisonous mushroom or time is up
  - `fail = True or False`
  - The Boolean variable `fail` has the value `True`

Truth Tables

A	B	A `and` B
True	True	True
True	False	False
False	True	False
False	False	False

A	B	A `or` B
True	True	True
True	False	True
False	True	True
False	False	False

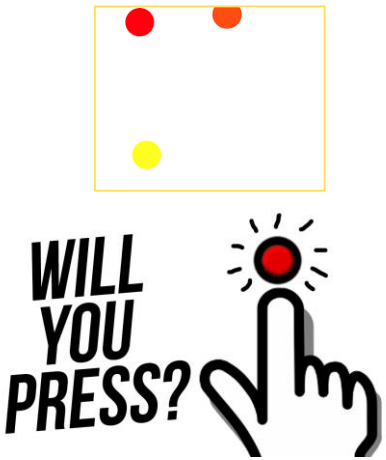
Boolean Operators

- A Boolean operator takes one or more Boolean values as input and produces a Boolean value as output.
- Example: `not`

  input: ONE Boolean value `False`

  output: `True`

  `or` yields `True` if the input is `False`



- Press the button if colour is not purple
  - `press = not isPurple`
  - The Boolean variable `press` has the value `False` if `isPurple` is `True`

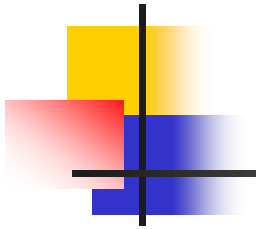
Truth Tables

A	B	A `and` B
True	True	True
True	False	False
False	True	False
False	False	False

A	B	A `or` B
True	True	True
True	False	True
False	True	True
False	False	False

A	`not` A
True	False
False	True

Boolean Operator Examples



- 0 < 200 `and` 200 < 100
  - `False`
- 0 < 200 `or` 200 < 100
  - `True`
- 0 < 200 `or` 100 < 200
  - `True`
- x=-7

  0>x `or` x<100 `and` x>50
  - (0>x `or` x<100) `and` x>50 is False
  - 0>x `or` (x<100 `and` x>50) is True
  - The `and` operator has a higher precedence than the `or` operator
  - So 0>x `or` x<100 `and` x>50 is True

Boolean Operator Examples

- 0 < 200 `and` 200 < 100
  - `False`
- 0 < 200 `or` 200 < 100
  - `True`
- 0 < 200 `or` 100 < 200
  - `True`
- x=-7

  0>x `or` x<100 `and` x>50
  - (0>x `or` x<100) `and` x>50 is False
  - 0>x `or` (x<100 `and` x>50) is True
  - The `and` operator has a higher precedence than the `or` operator
  - So 0>x `or` x<100 `and` x>50 is True
- `not` (0 < 200)
  - `False`
- `not` (0 < 200) `or` (100 < 200)
  - not ((0 < 200) `or` (100 < 200)) is False
  - (not (0 < 200)) `or` (100 < 200) is True
  - The `not` operator has a higher precedence than the `and/or` operator
- `frozen` == True
  - `frozen`
  - There is no need to compare a Boolean variable with `True`
- `frozen` == `False`
  - `not frozen`
  - It is clearer to use `not` than to compare with `False`

The Operators and, or

- Avoid confusing the operators `and`, `or`

- E.g. x in the range 0 to 100 inclusive

  0 <= x `and` x <= 100

- E.g. x outside the range 0 to 100 inclusive

  x < 0 `or` x > 100

Chaining Relational Operators

- The expression

```
0 <= value <= 100
```

is equivalent in Python to

```
value >= 0 and value <= 100
```

- The expression

```
a < x > b
```

is equivalent to

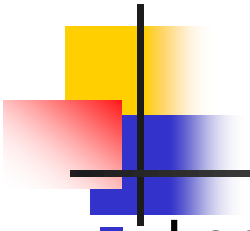
```
a < x and x > b
```

- Example:

```
x = 9
```

- `3 < x > 8`
  - `True`
- `8 < x > 3`
  - `True`
- `3 < x > 9`
  - `False`

Short Circuit Evaluation



- Logical expressions `x and y` and `x or y` are evaluated left to right.
- Evaluation stops when the value of the expression is known.
- Examples:

  True `or` Anything

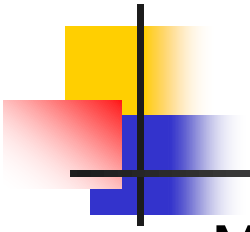
  ```
  # True
  ```

  False `and` Anything

  ```
  # False
  ```

Short Circuit Evaluation

- Another example:
  - fun() is a user-defined function
    - It will print "Yes" in the shell and return True
  - What will the following statements do in the shell?
    - True `and` fun()
      - Print "Yes", True
    - False `and` fun()
      - No print, False
    - True `or` fun()
      - No print, True
    - False `or` fun()
      - Print "Yes", True
- Yet another example:
  - quantity > 0 `and` price/quantity < 10
    - # False if quantity == 0



De Morgan's Law

- Motivation example

Charge a higher shipping rate if the destination is not within the continental United States

- Not US
- part of the US, but not continental: Alaska and Hawaii

```
if not (country == "USA" and state != "AK" and state != "HI") :
        shippingCharge = 20.00
```

When `not` is applied on the outermost level of the condition, it becomes harder to understand what it means.

De Morgan's Law

- It tells us how to negate `and` and `or` conditions
- Version 1

  `not(A and B)` is the same as `(not A) or (not B)`
- Version 2

  `not(A or B)` is the same as `(not A) and (not B)`
- To prove Version 2 from Version 1, write

```
  not (A or B)
= not (not not A or not not B)   #not not = identity
= not not (not A and not B)      # apply Version 1
= not A and not B                #not not = identity
```

Example of De Morgan's Law

- Charge a higher shipping rate if the destination is not within the continental United States
  - part of the US, but not continental: Alaska and Hawaii
- if **not** (country == "USA" and state != "AK" and state != "HI") :

  shippingCharge = 20.00
- if(country != "USA" or state == "AK" or state == "HI") :

  shippingCharge = 20.00

Usually it is a good idea to push negations to the innermost level

PFE Review Question R3.20

- Of the following pairs of strings, which comes first in lexicographic order?

  > "Tom", "Jerry"

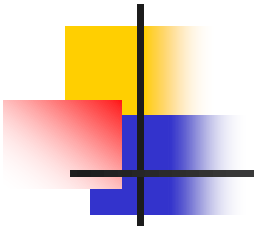
  < "Tom", "Tomato"

  > "church", "Churchill"

  < "car manufacturer", "carburettor"

  < "36", "A1"

  < "36", "a1"

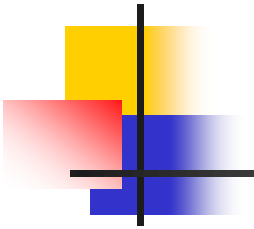


Examples

- Let x, y, z be variables with integer values. Construct a Boolean expression that is True if and only if exactly one of x, y, z is equal to zero.

- Construct a Boolean expression with the following truth table, using one or more of the operators and, or, not.

A	B	Expression
True	True	False
True	False	True
False	True	True
False	False	False



Examples

- Let x, y, z be variables with integer values. Construct a Boolean expression that is True if and only if exactly one of x, y, z is equal to zero.

- (x==0 and y!=0 and z!=0)

or (x!=0 and y==0 and z!=0)

or (x!=0 and y!=0 and z==0)

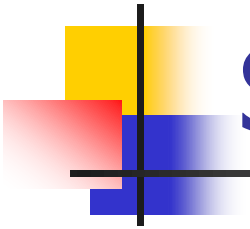
Examples

- Construct a Boolean expression with the following truth table, using one or more of the operators and, or, not.

A	B	Expression
True	True	False
True	False	True
False	True	True
False	False	False

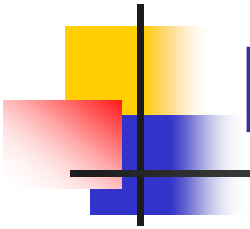
exclusive or

- (A and not B) or (not A and B)
- (A or B) and not (A and B)
- (A or B) and (not A or not B)



Saturday Sessions

- One-to-one session
  - Help with your Python questions
- Every Saturday
  - Until 28th March
- 20-min time slots
  - 15:00, 15:20, 15:40, 16:00, 16:20, 16:40, 17:00, 17:20, 17:40
- Venue: MAL 151
- Tutor: Donal
- Free, but you need to book first



How to Book

- By email
  - Put "Booking Out Of Hours Session" in the title
  - State which week you want
- First come first served
  - You may state which slot you prefer, but not always available
- Book at least one working day in advance
  - The latest a booking can be made will be on the Thursday of that week
- Which email account?
  - fd@dcs.bbk.ac.uk