CISC101 Reminders & Notes

- Test 1 this week
  - Will take place in your tutorial
  - Lab will still follow
- Assignment 1 should be marked in two weeks
  - Look for your grade in Moodle
  - Contact your TA if you have any questions or concerns
- Assignment 2 will be posted next week

Slides courtesy of Dr. Alan McLeod

A Minor in Computing?

- Are you interested in finding out what is required for a Minor in Computing?
  - Please contact our Undergraduate Chair
  - Dr. Bob Tennent (`rdt@cs.queensu.ca`)

Slides courtesy of Dr. Alan McLeod

From Last Time …

- Supplemental notes from Jan. 25th
  - Functions
  - Variable scope
- Notes from Jan. 25th
  - Global variable demo
    - Slides 28
  - Programming style
    - Slides 29-35

Slides courtesy of Dr. Alan McLeod

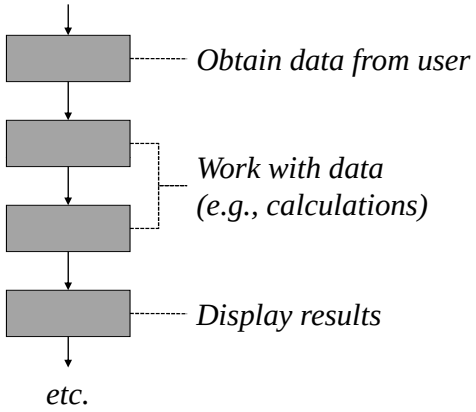
Documentation Strings

- Put a string literal right after the `def` … statement
  - This is a *documentation string*
- It will be regurgitated by the help system
  - Can also be accessed using special variables
  - Demo: WindowWeight.py
    - Try typing `help(main)` at the >>> prompt
- Use triple quotes to place a large document
- Typically used when creating object definitions
  - Classes (which we won't be doing)
- Not a bad habit to get into …

Slides courtesy of Dr. Alan McLeod

So Far …

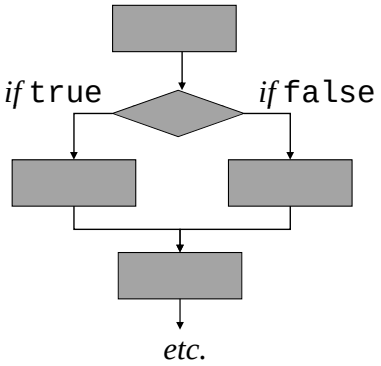
- So far, all our programs have been *linear*



Slides courtesy of Dr. Alan McLeod

Branching Algorithms

- If a program could test a condition, then it would have a choice as to its path of execution



Slides courtesy of Dr. Alan McLeod

Conditionals or "Selection Statements"

- Python has `if`, `if-else` and chained if (`if-elif-else`) statements
- `if` statement syntax

```
if boolean_expression :
   statement1_when_true
   statement2_when_true
   statement3_when_true
   …
```

- Example

```
if num < 0 :
   print("Negative number")
```


Slides courtesy of Dr. Alan McLeod

`if-else` Statement

- Syntax of `if-else` statement

```
if boolean_expression :
   statement(s)_when_true
else :
   statement(s)_when_false
```

- Example

```
if num < 0 :
   print("Negative number – using absolute value")
   num = abs(num)
else :
   print("Non-negative number - OK")
```


Slides courtesy of Dr. Alan McLeod

Boolean Expressions

- We need to know how to construct *expressions* that evaluate to a `bool`
  - a `bool` in Python is either `True` or `False`
- Use *Boolean operators* to create expressions
  - Work like mathematical operators, but they produce Booleans instead of numbers
- Boolean operators are either *unary* or *binary*
  - Unary requires one data value
    - *e.g.*, NOT a
  - Binary requires two data values
    - *e.g.*, a < b, a OR b

Slides courtesy of Dr. Alan McLeod

Unary Boolean Logical Operators

- Python has a Boolean logical operator `not`
  - Must be used on a *single* bool value
  - Sets False to True
  - Sets True to False
    - Just like the Boolean logical operator NOT (!)

a	(! *a*)
0	1
1	0

Slides courtesy of Dr. Alan McLeod

Binary Boolean Logical Operators

- Python has Boolean logical operators `and` and `or`
  - Must be used on *two* `bool` values
  - Work just like the Boolean logical operators AND ($\wedge$) and OR ($\vee$)

a	*b*	($a \wedge b$)
0	0	0
0	1	0
1	0	0
1	1	1

a	*b*	($a \vee b$)
0	0	0
0	1	1
1	0	1
1	1	1

Slides courtesy of Dr. Alan McLeod

Binary Boolean Operators

`>`	greater than
`<`	less than
`>=`	greater than or equal to
`<=`	less than or equal to
`==`	equals
`!=`	not equals
`and`	Boolean logical AND
`or`	Boolean logical OR

All of these result in a `True` or `False` `bool` value

Slides courtesy of Dr. Alan McLeod

Binary Boolean Operators - Cont.

`>`	greater than
`<`	less than
`>=`	greater than or equal to
`<=`	less than or equal to
`==`	equals
`!=`	not equals

These must have a number on both sides or a string on both sides – do not mix types

`and`	Boolean logical AND
`or`	Boolean logical OR

These must have a `bool` value on both sides

Slides courtesy of Dr. Alan McLeod

Binary Boolean Operators - Cont.

- The Python interpreter wants both sides to be the same data type
  - Two numeric values
  - Two string values
- You will get an odd result if you try to compare a string to a number

Slides courtesy of Dr. Alan McLeod

Boolean Operators - Cont.

Precedence rules are now expanded!

1) Math operators in the usual order:
   `**`
   `-` (numeric negation)
   `*` `/` `//` `%`
   `+` `-` (subtraction)
2) Binary Boolean comparison operators:
   `>` `>=` `<` `<=` `==` `!=`
3) Logical Boolean operators:
   `not` `and` `or`
4) Assignment operator: `=`

Slides courtesy of Dr. Alan McLeod

Boolean Expression Examples

```
5 > 2
4 < 3 or 7 > 1
7 != 8
6 + 2 > 9 or 4 == 3 + 1
7 == 7 and 5 > 2 and 6 != 3
5 * 5 >= 5 ** 2
128 % 2 == 0
```


Slides courtesy of Dr. Alan McLeod

Evaluating Another Example

```
not(5 * 4 > 3) or 2 + 3 <= 5 and 6 == 2
not(20  > 3) or 2 + 3 <= 5 and 6 == 2
 not(True)   or 2 + 3 <= 5 and 6 == 2
    False    or 2 + 3 <= 5 and 6 == 2
    False    or   5   <= 5 and 6 == 2
    False    or      True  and 6 == 2
    False    or      True  and  False
             True          and  False
                     False
```


Slides courtesy of Dr. Alan McLeod

Yet Another Example

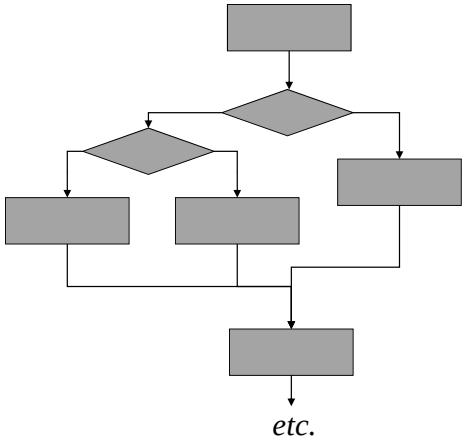
- Of course, these expressions can be used with variables
  - The values of the variables are used by the operators
- What is printed out in this case?

```
a = 12
b = 5
c = 2
print(a > b and b * c < a)
```


Slides courtesy of Dr. Alan McLeod

`if` Statements - Cont.

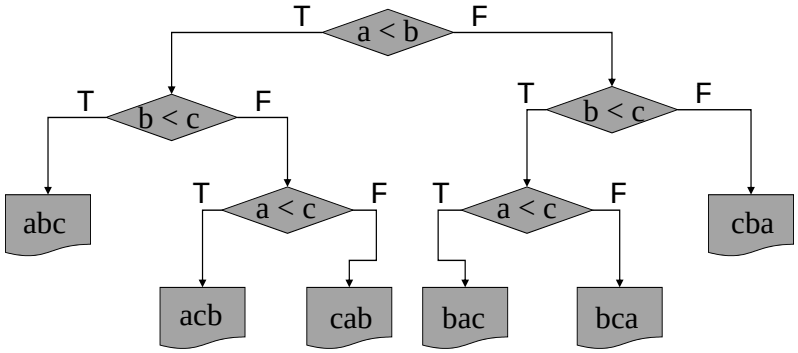
- You will often have to nest `if` statements



Slides courtesy of Dr. Alan McLeod

Nested `if` Statements

- For example, what if you have three numbers a, b & c and you want to print them in order?
  - Demo: SortThree.py



Slides courtesy of Dr. Alan McLeod

`if-elif` Statements

- In the code in SortThree.py, you might have noticed this construct:

```
else :
    if a < c :
```

- This kind of thing occurs so often that it can be shortened to:

```
elif a < c :
```


Slides courtesy of Dr. Alan McLeod

`if-elif` Statements - Cont.

- This leads to a common construct often called a *chained* `if` construct

```
if condition1 :
   statement(s)
elif condition2 :
   statement(s)
elif condition3 :
   statement(s)
else :
   statement(s)
```

- You can have as many `elif`s as you want
- The `else` is optional

```
if condition1 :
   statement(s)
elif condition2 :
   statement(s)
elif condition3 :
   statement(s)
```


Slides courtesy of Dr. Alan McLeod

`if-elif` Statements - Cont.

- There is nothing in this construct that you could not make with normal `if-else` statements
- For some kinds of conditionals, the `if-elif` might be easier to put together
- For example, consider letter grades
  - Demo: GradeCodes.py
    - 0 to 50 is F
    - 50 to 70 is C
    - 70 to 85 is B
    - 85 to 100 is A
    - Anything else is an illegal grade

Slides courtesy of Dr. Alan McLeod

`if-elif` Statements - Cont.

- Good lab exercises!
  - Rewrite GradeCodes.py using a nested `if` construct
  - Rewrite SortThree.py with a chained `if` construct
- In both cases, compare the versions
  - Is one more efficient than the other?
    - Count the maximum number of comparisons that are made
  - Which version is easier to write and debug?

Slides courtesy of Dr. Alan McLeod

Comparing Strings

- Boolean comparison operators work on strings

  `"abc" == "abC"` gives `False`
  `"abc" < "abcd"` gives `True`
  `"A" < "a"` gives `True`
  `"a" < "b"` gives `True`

- These comparisons are based on the ASCII code values of the characters compared
  - See Appendix C in the textbook

Slides courtesy of Dr. Alan McLeod

ASCII

- American Standard Code for Information Interchange
  - Each character is represented by a byte (8 bits)
- Maps integers from 0-127 to characters
  - A-Z → 65-90
  - a-z → 97-122
- ASCII was a standard
  - Now the *Unicode* scheme is commonly used
    - Uses more bits
    - Includes more alphabets
  - Unicode is compatible with ASCII

Slides courtesy of Dr. Alan McLeod

ASCII Table (Lower Half)

	0	1	2	3	4	5	6	7	8	9
0	nul	soh	stx	etx	eot	enq	ack	bel	bs	ht
1	nl	vt	ff	cr	so	si	dle	dc1	dc2	dc3
2	dc4	nak	syn	etb	can	em	sub	esc	fs	gs
3	rs	us	sp	!	"	#	$	%	&	‘
4	(	)	*	+	,	-	.	/	0	1
5	2	3	4	5	6	7	8	9	:	;
6	<	=	>	?	@	A	B	C	D	E
7	F	G	H	I	J	K	L	M	N	O
8	P	Q	R	S	T	U	V	W	X	Y
9	Z	[	\	]	^	_	’	a	b	c
10	d	e	f	g	h	i	j	k	l	m
11	n	o	p	q	r	s	t	u	v	w
12	x	y	z	{	l	}	~	del		

Slides courtesy of Dr. Alan McLeod

ASCII and Python

- BIFs in Python can convert characters
  - A character is a string of length one
- `ord(aChar)`
  - Returns the ASCII value of the given character
- `chr(anInt)`
  - Returns the character for the given ASCII value

Slides courtesy of Dr. Alan McLeod