Ch4 Branching (Control)

Friday, September 22, 2023 8:40 AM

- Boolean: True
  False.

- boolean expressions:

  relational

  == equality < >

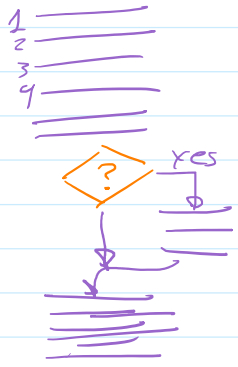
  != not equality <= >=

  or and not.

- Control:

  "if" statement:

  - control execution of code:
  - execute some code "conditionally",
    only when some conditions are true.



- (1) if

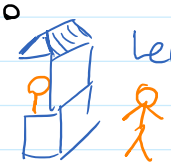
```
if cond :
    block
```

Block of code:
- a sequence of statements.
- begins with indentation increase
- ends when indentation decreases.
- blocks can contain other blocks.

- Demo:

absolute value:

- Demo



Lemonade: - $ 1.25.

Senior discount of %15.

1- ask client how many lemonads to purchase
2- ask client its age.
3- print total to pay.

- (2) if

```
if cond :
    block1
else:
    block 2
```

- if the condition is true block1 is executed
- if the condition is false block2 is executed.

Demo:
check weather a number is positive or negative.

Demo:
check weather a number is even or odd.

- (3) if

```
if cond 1 :
    block 1
elif cond 2 :
    block 2
elif cond 3 :
    block 3
:
:
else:
    block f
```

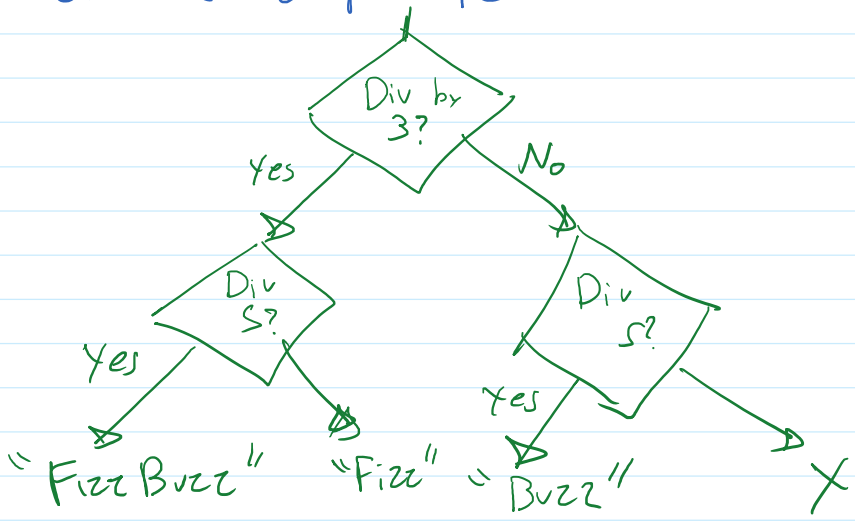
- if cond1 is true, block 1 is executed.
- if cond 1 is false and cond2 is true block2 is executed.
- if cond 1 and cond2 are false and cond3 is true block 3 is executed.
- if cond 1 and cond2 and cond3 are false and cond 4 is true block 4 is executed.

- if no conditions are true the block is executed.

Demo: form (3)

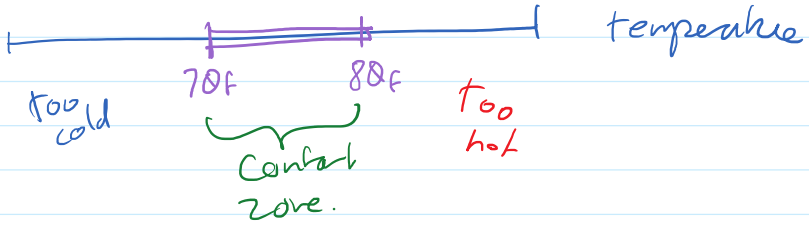
Demo: if statements can be nested.

"FizzBuzz" Given a number: (positive integer)
if the number is divisible by 3 "Fizz"
if the number is divisible by 5 "Buzz"
if the number is divisible by both 3 and 5 "FizzBuzz"
otherwise output the number.



Use Case:
- detection of a range / ranges.

E.g.



- The membership operator
  in not in
  allow you to check whether an element belongs to a sequence type

  elem in seq.

elem in seq.
elem not in seq

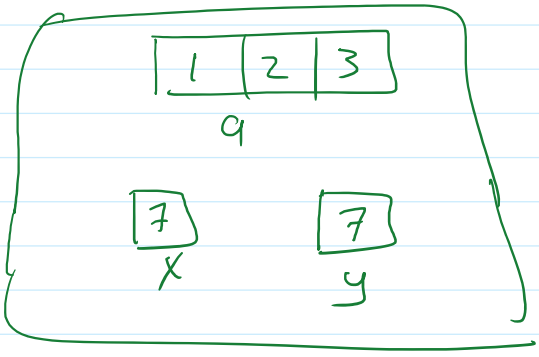
Note: can be expensive if sequence is large.

- the identity operator:

  is not is.

  check whether to variable names refer to the same piece of memory.

```
a = [1, 2, 3]
x = a
x = 7
y = x
```



- the conditional expression

```
if x < y:
    x = x + 1
else:
    x = x - 1
```

changes to same variable.

```
x = x+1 if x<y else x-1
```

syntax:

$expr_1$ if cond else $expr_2$

this expression evaluates $expr_1$ if cond is true otherwise evaluates $expr_2$