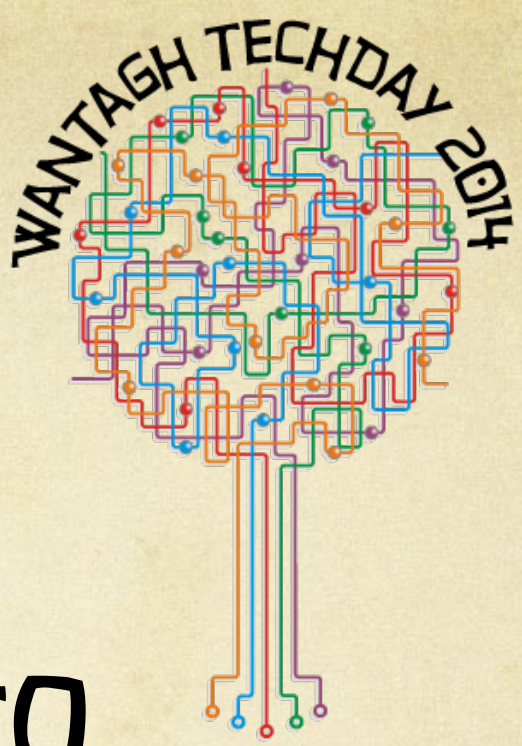


INTRODUCTION TO PROGRAMMING USING PYTHON

CS.002

```
        If a collision occurs with the ball and the paddle, when the ball has a
        negative velocity, the ball's y component of the velocity is negated. If
        there is a collision with the ball and paddle, but the ball is going up,
        nothing happens. If there is a collision between the ball and a brick,
        the brick is removed, and the ball's y component of the velocity is
        negated. Also plays a specific sound depending on what object the ball
        hits."""

        collision = self._getCollidingObject() # Checks for Collisions

        if collision == self._paddle and self._ball.vy > 0:
            pass # nothing happens when the ball is going up and hits the paddle
        elif collision == self._paddle and self._ball.vy < 0:
            if self._soundToggle:
                paddleSound = Sound('bounce.wav')
                paddleSound.play()
            self._ball.vy = - self._ball.vy # change direction
        elif collision in self._bricks:
            if self._soundToggle:
                brickNoise = Sound(BRICKS_AUDIO[self._audioCounter])
                brickNoise.play()
                self._audioCounter = self._audioCounter + 1
                if self._audioCounter == len(BRICKS_AUDIO):
                    self._audioCounter = 0
            self._ball.vy = - self._ball.vy # change direction
            self.view.remove(collision) # remove from view
            self._bricks.remove(collision) # remove from field
            self._score = self._score + 10
            self._scoreKeeper.text = 'Current Score : '+`self._score`

    def _keepInBounds(self):
        """This method keeps the paddle within the boundaries of the window.

        Although short, this helper method is mainly for styling purposes.
```

WHAT IS CODE?

AND HOW IS IT PREVALENT TO TODAY'S WORLD?

CODE

```
01010100100101110010100101010101
01010101001110101011001110001101O
01110010101011001101100000110100
10010101000100110011001001001100
```

```
mov     abx3e321, e32
mov     324aev,231fba1
eax     f2va33,ffa3a34
edx     249gjfa,23f2fa
```

```
number = 30
number = number * 30
aCopy = number
print aCopy
```

BINARY

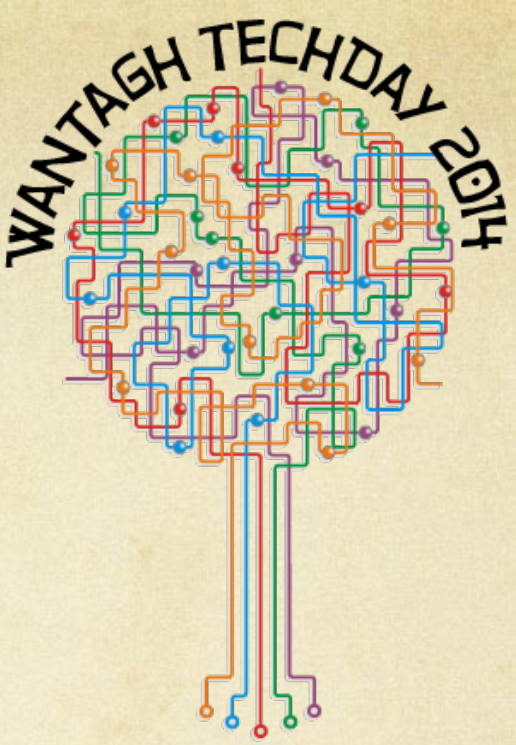
ASSEMBLY

PYTHON

STARTING / USING SUBLIMETEXT 3

- TO START SUBLIMETEXT 3
  - CLICK THE WINDOWS *START* BUTTON, THE START MENU WILL A
  - CLICK ON THE SUBLIMETEXT 3 QUICK-LAUNCH IC 

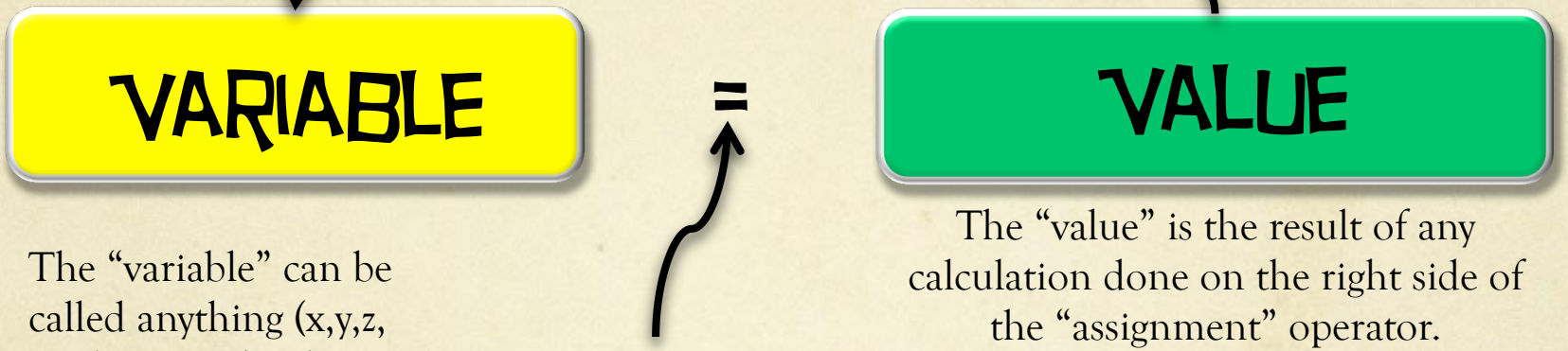
- TO BEGIN USING SUBLIMETEXT 3
  - CLICK *FILE --> NEW FILE (CTRL+N)*
  - SAVE THE NEW FILE
    - CLICK *FILE --> SAVE (CTRL+S)*
    - SAVE THE FILE AS *HELLOWWORD.PY* IN YOUR PROJECT FOLDER
- NOW LET'S WRITE OUR FIRST PROGRAM!



PART ONE

VARIABLES, ASSIGNMENT, AND CONDITIONALS

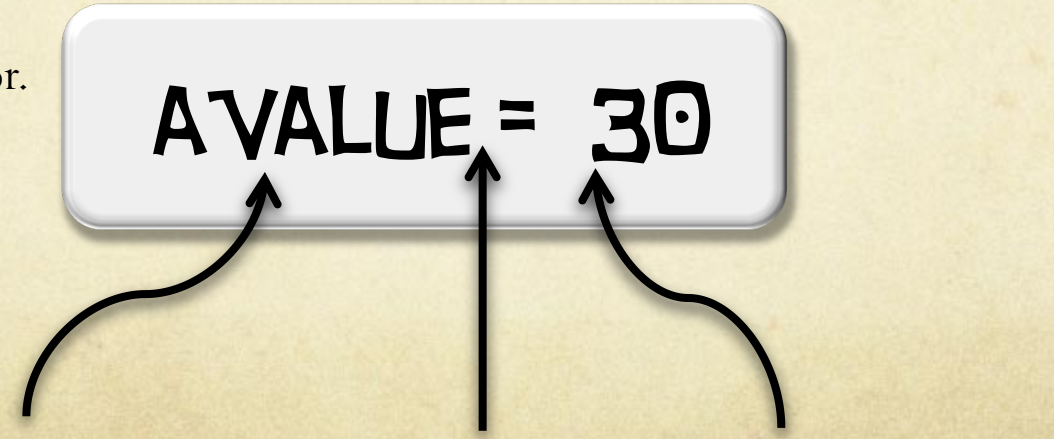
VARIABLES AND ASSIGNMENT



The "variable" can be called anything (x,y,z, apple,orange). The variable "holds" the value calculated on the right side of the assignment operator.

Assignment, NOT **equals**!

The "value" is the result of any calculation done on the right side of the "assignment" operator.
Ex: 30 + 30 (value is 60)
15 * 2 (value is 30)



VARIABLES AND ASSIGNMENT

OPERATION	EVALUATION
NUMBER = 30	NUMBER --> 30
NUMBER = 15 * 2	NUMBER --> 30
NUMBER = 30 – 15	NUMBER --> 15
NUMBER = 3 ** 3	NUMBER --> 27
NUMBER = 2 ** (2 + 3)	NUMBER --> 2 ** (5) --> 32
AVALUE = 5	AVALUE = 5
NUMBER = AVALUE * 3	NUMBER --> (5) * 3 --> 15
NUMBER = AVALUE + 15	NUMBER = (15) + 15 --> 30
NUMBER = 10	NUMBER --> 10
NUMBER = NUMBER ** 2	NUMBER --> (10) ** 2 --> 100

The calculation of the *value* to be assigned to the *variable* follows PEMDAS rules.

A value calculation can include other **variables** (using their stored value), in fact a calculation can include the variables current value, which will be changed to the new value following assignment.

ARITHMETIC OPERATIONS

- Assume variable *a* holds 10 and variable *b* holds 20, then:

Operator	Description	Example
+	Addition - Adds values on either side of the operator	a + b will give 30
-	Subtraction - Subtracts right hand operand from left hand operand	a - b will give -10
*	Multiplication - Multiplies values on either side of the operator	a * b will give 200
/	Division - Divides left hand operand by right hand operand	b / a will give 2
%	Modulus - Divides left hand operand by right hand operand and returns remainder	b % a will give 0
**	Exponent - Performs exponential (power) calculation on operators	a**b will give 10 to the power 20
//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed.	9//2 is equal to 4 and 9.0//2.0 is equal to 4.0

ADVANCED ASSIGNMENT OPERATORS

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand	c = a + b will assigne value of a + b into c
+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand	c += a is equivalent to c = c + a
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand	c -= a is equivalent to c = c - a
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand	c *= a is equivalent to c = c * a
/=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand	c /= a is equivalent to c = c / a
%=	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand	c %= a is equivalent to c = c % a
**=	Exponent AND assignment operator, Performs exponential (power) calculation on operators and assign value to the left operand	c **= a is equivalent to c = c ** a
//=	Floor Dividion and assigns a value, Performs floor division on operators and assign value to the left operand	c //= a is equivalent to c = c // a

NOW TRY THIS YOURSELF

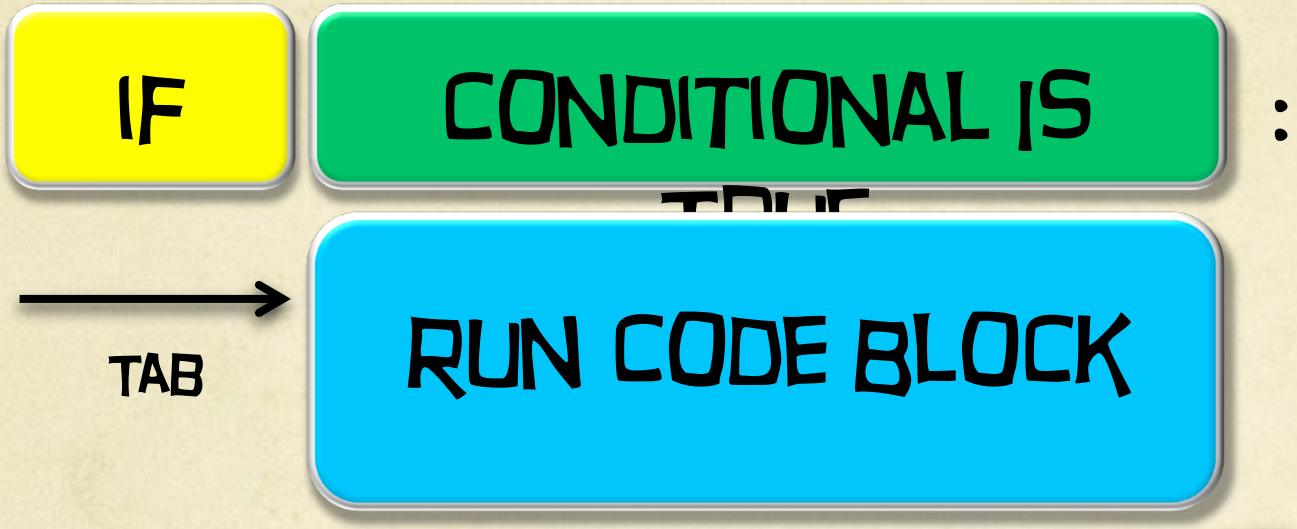
Remember to include a *print* statement:

If you want python to output the variable *"aValue"* (can be called anything...)

Type:

```
print aValue
```

CONTROLLING ASSIGNMENT
CONDITIONALS



The *Assignment*

The *Conditional*

```
NUMBER = 0
IF (NUMBER == 0):
    ACOPY = 10
    NUMBER = ACOPY * 2

PRINT NUMBER
```

The *print statement*.
Ex:
print variable

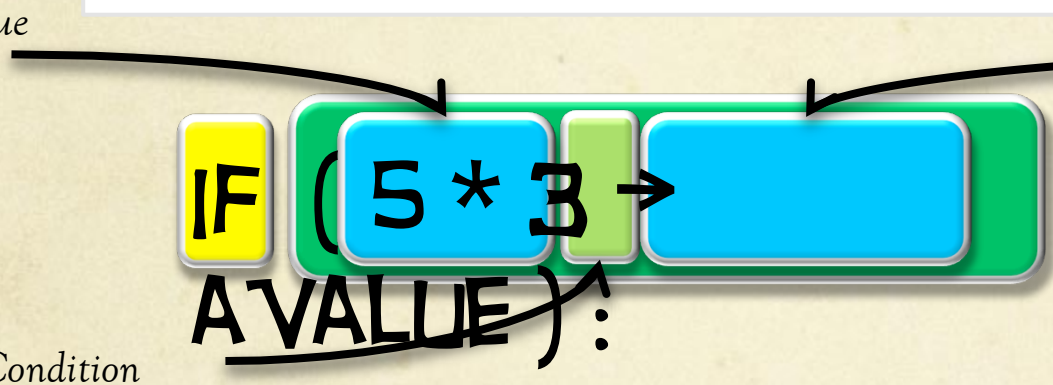
The equation in here is known as a *logic operation*. All conditionals will either be *True* or *False*.
If true, the program runs the code block, otherwise it skips it.

CONTROLLING ASSIGNMENT

CONDITIONALS

TYPES OF CONDITIONS

==	EQUALS
!=	NOT EQUALS
→	GREATER THAN
←	LESS THAN



IF

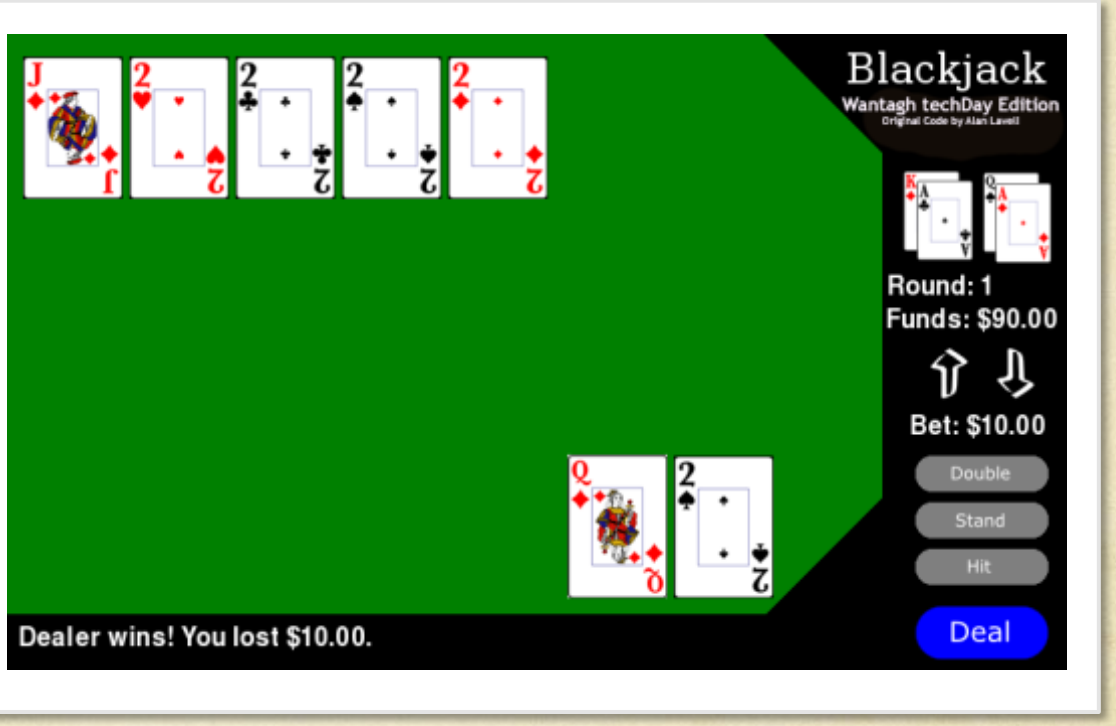
15 == 15	TRUE
15 == 16	FALSE
3 + 4 == 7	TRUE
8 != 7	TRUE
98 → (3*15)	TRUE

Each side of the *conditional statement* is calculated exactly like the right side of an *assignment* of operation. The resulting values on each side are compared to one another according to the type of *condition* imposed (==, != , > , <) and returns either *True* or *False*.

NOW TRY THIS YOURSELF

A PRACTICAL APPLICATION
BLACKJACK

- Blackjack is a simple card game. See the provided rule sheet.
- Each of you has your own copy of the games python code in your personal folder.
- There are two *"bugs"* in the games code. Let's use what we've learned thus far to fix the first one.
- Scroll to line 534



A PRACTICAL APPLICATION

BLACKJACK

A Broken "Bet Button"

- Run the game (CTRL+B)
- Note that the bet buttons don't work (when you click them the amount of money being bet doesn't go up or down)
- Think about the steps needed to raise a bet:
  - 1. Check to see if your bet is less than the total money available
  - 2. If (1) is true, increase the bet by a certain amount.
- Think about the steps needed to lower a bet:
  - 1. A bet cannot be negative, how would you check this using conditionals?
  - Assuming (1) is true, decrease the bet by a certain amount.

A Practical Application

Blackjack

A Broken "Bet Button"

- Raising the bet.

```
536     #---------------------------------------------------------------------------
537     #---------------------- Only Type New Code In The Area Below ---------------
538     #---------------------------------------------------------------------------
539
540
541
542
543     #---------------------------------------------------------------------------
544     #---------------------------------------------------------------------------
545     #---------------------------------------------------------------------------
546
```

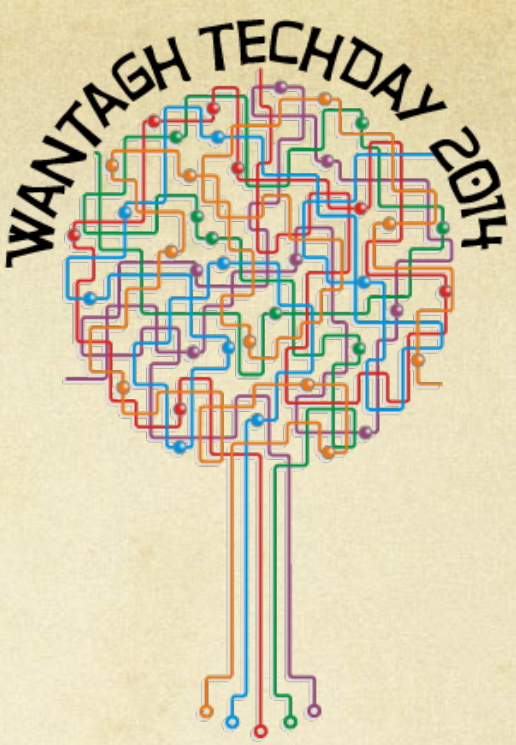
A Practical Application

Blackjack

A Broken "Bet Button"

- Lowering the bet.

```
578         #-----------------------------------------------------------------------
579         #---------------------- Only Type New Code In The Area Below ----------
580         #-----------------------------------------------------------------------
581
582
583
584
585
586         #-----------------------------------------------------------------------
587         #-----------------------------------------------------------------------
588         #-----------------------------------------------------------------------
589
```

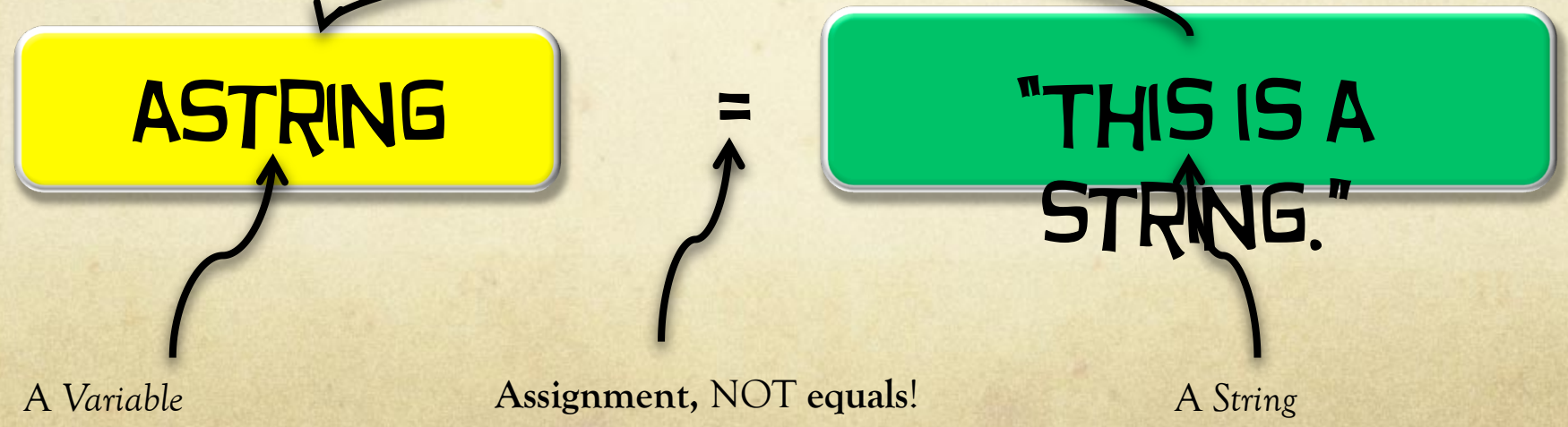


PART TWO

STRINGS, LISTS, AND LOOPS

STRINGS

- A *VARIABLE* CAN STORE THINGS BESIDES NUMBERS AS ITS VALUE.
  - TEXT, LISTS, ETC. . . *(WE WILL GET TO THIS LAST ONE IN A FEW MINUTES)*
- A *STRING* IS A VALUE THAT HOLDS TEXT, WHICH CAN BE A SINGLE LETTER OR AN ENTIRE PARAGRAPH.
  - A STRING IS MADE SIMPLY BY PUTTING " " AROUND A WORD OR PHRASE.
    - EX: *"THIS IS A STRING"* CAN BE ASSIGNED TO A VARIABLE LIKE A NUMBER.



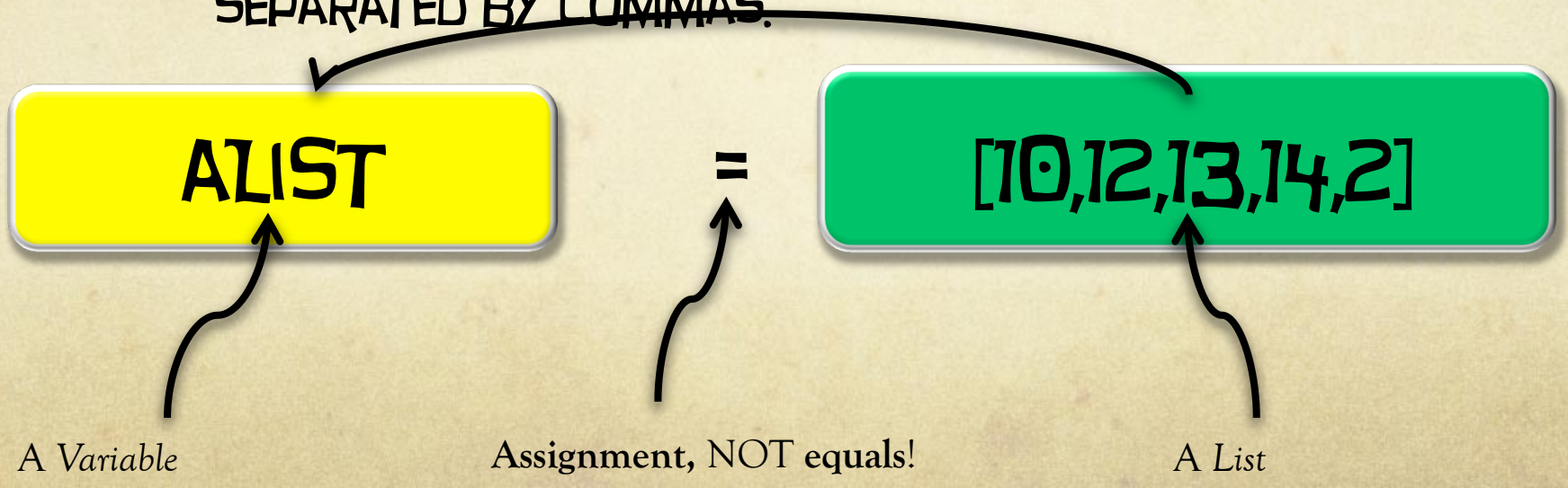
STRINGS

- WHAT CAN STRING BE USED FOR?
  - COMBINE WITH *CONDITIONALS* AND *PRINT* COMMANDS TO ALLOW THE PROGRAM TO GIVE FEEDBACK TO THE USER.
- HOW CAN WE MANIPULATE STRINGS?
  - STRINGS *CAN* BE ADDED TOGETHER.
    - EX:
  - STRINGS *CANNOT* BE SUBTRACTED FROM ONE ANOTHER.
- STRING EXAMPLES:
  - "A" A SINGLE CHARACTER STRING.
  - "123213" NUMBERS CAN BE STRINGS TOO.
  - "DANIEL" A NAME.
  - "SENTENCE'S CAN BE STRINGS TOO." A SENTENCE.

NOW TRY THIS YOURSELF

Lists

- A *variable* can store things besides numbers as its value.
  - Lists, text , etc... *(we already covered the second one)*
- A *list* is a collection of a bunch of similar items (numbers, strings, etc...).
  - A list is made by putting a series of values (numbers or strings) together inside brackets [ ], separated by commas.



LISTS

EXAMPLES OF LISTS

MYLIST = [3,4,1,63,12] LIST OF NUMBERS

MYLIST = ["ASTRING","ANOTHERSTRING"] LIST OF STRINGS

MYLIST = [ANOTHERLIST,ANOTHERLIST2] LIST OF LISTS

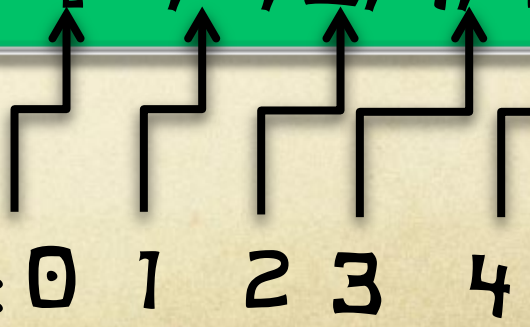
- ADDING AN ITEM TO A LIST:
  - MYLIST.APPEND(ITEM)
- REMOVING AN ITEM FROM A LIST:
  - MYLIST.REMOVE(ITEM)
- COMBINING LISTS:
  - GIVEN ALIST = [1,2,3] AND ALIST2 = [4,5,6]
    - SUMLIST = ALIST + ALIST2 = [1,2,3,4,5,6]
- HOW DO WE ACCESS ITEMS THAT WE PUT IN OUR LIST?

LISTS

- INDEXING
  - A LIST HOLDS A COLLECTION OF ITEMS, EACH OF WHICH IS IN A SPECIFIC POSITION WITHIN A LIST.
  - YOU CAN RETRIEVE ITEMS FROM A LIST THROUGH A PROCESS CALLED *INDEXING*.
    - ALIST[POSITION] WILL RETURN THE VALUE STORED AT ALIST'S POSITION
    - POSITIONS START AT 0

ALIST = [10,12,13,14,2]

ALIST[2] --> 13

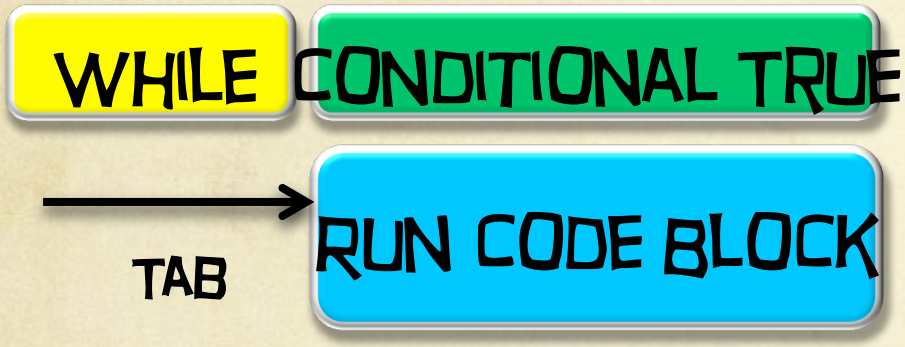


NOW TRY THIS YOURSELF

LOOPS

- A *LOOP* IS A POWERFUL WAY TO REPEAT THE SAME BLOCK OF CODE MULTIPLE TIMES.
- TWO IMPORTANT TYPES OF LOOP:

WHILE LOOP



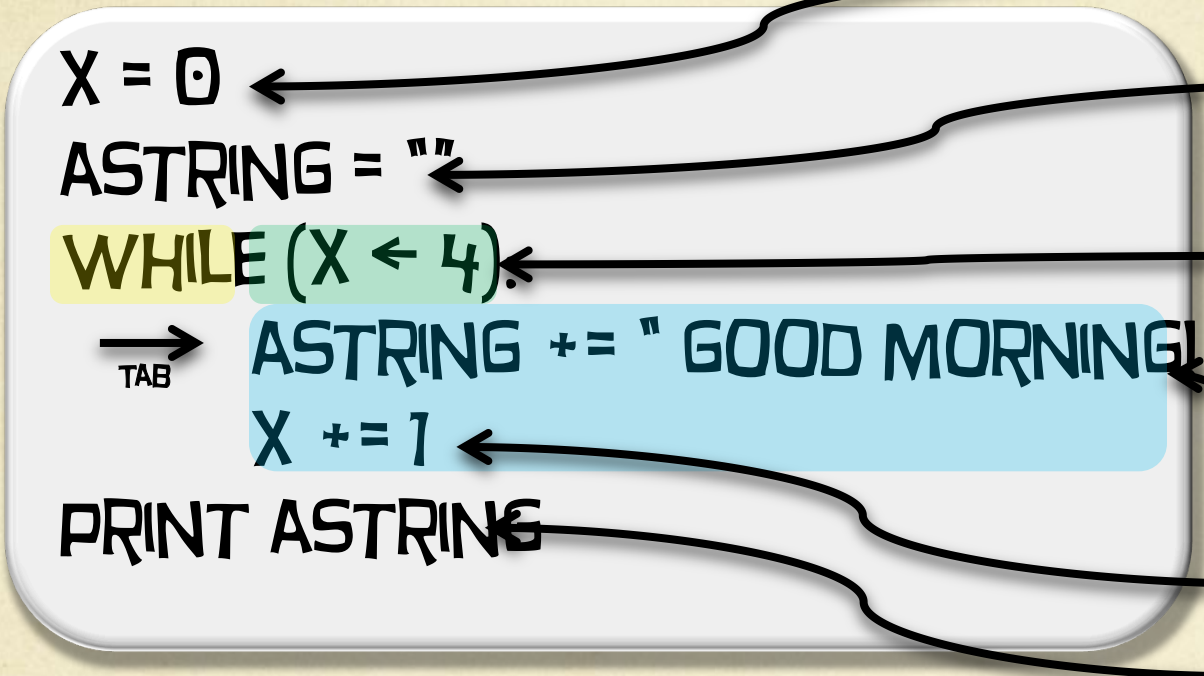
- A *WHILE LOOP* IS SIMILAR TO AN *IF STATEMENT* IN THAT THE CODE BLOCK WILL KEEP REPEATING WHILE THE GIVEN CONDITIONAL IS STILL

FOR LOOP

- A *FOR LOOP* WILL RUN THE CODE BLOCK *ONCE FOR EVERY VALUE IN A LIST*. FOR ANY GIVEN CYCLE OF THE CODE BLOCK, *X* HOLDS THE

LOOPS

- EXAMPLE OF A *WHILE LOOP*:



- RESULT: " GOOD MORNING! GOOD MORNING! GOOD MORNING! GOOD MORNING!"

LOOPS

- WHAT IS THE *RANGE(N)* FUNCTION?
  - IN PYTHON, TYPING RANGE(*N*) WHERE *N* IS A NUMBER WILL GIVE YOU A *LIST OF ELEMENTS* STARTING FROM 0 AND GOING UP BY ONE (1,2,3,ETC...) UNTIL *ONE LESS THAN N*.
    - *EX: RANGE(5)* == [0,1,2,3,4]
    - THIS IS AN EASY AND COMMONLY USED WAY TO MAKE LISTS FOR *FOR LOOPS*.

- EXAMPLE OF A *FOR LOOP*:

The variable *x* will cycle through the range 0 to 4

The *range* here is: 0,1,2,3,4

```
FOR X IN RANGE(0,5):
    PRINT (2 * X)
```

TAB

- RESULT: 0 2 4 6 8

NOW TRY THIS YOURSELF

A PRACTICAL APPLICATION

BLACKJACK

An improper deck of cards

- Run the game (CTRL+B)
- Note that of the cards being "*dealt*", there are no numbered cards? Also, where are the diamond cards?
- Scroll to line 114
  - Note that the diamond cards are all missing?
  - Note that of the cards being "*dealt*", there are no numbered cards. There are only *royal spade, club and heart cards*.
- Lets fix this!

A PRACTICAL APPLICATION

BLACKJACK

An improper deck of cards

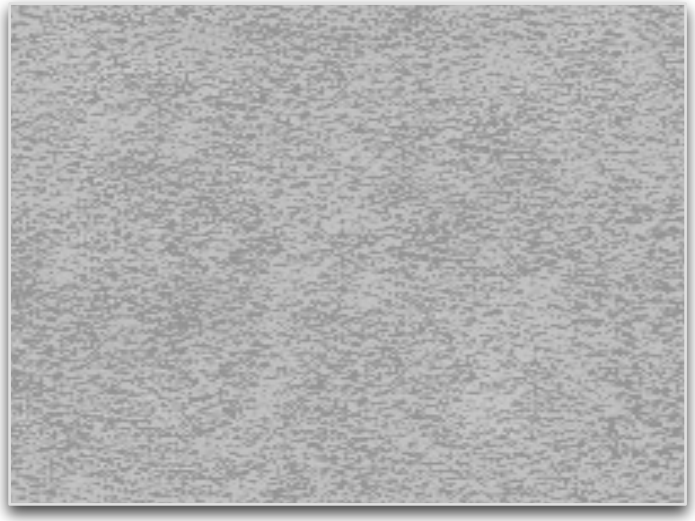
```
#------------------------------------------------
#---------------- Lists -------------------------
#------------------------------------------------

#s = Spade, h = Heart, c = Club, d = Diamond
#j = Jack, q = queen, k = king,
royaltySpades = ["sj","sq","sk","sa"]
royaltyHearts = ["hj","hq","hk","ha"]
royaltyClubs = ["cj","cq","ck","ca"]

#Where are the Diamonds royals? Make the a list similar to the ones above with diamond royals!

[redacted]

# We want our deck to look like this one below when we are done. But
# instead we have the four small decks we have created. How can we use
# python arrays to create the one below?
#   deck = ['sj', 'sq', 'sk', 'sa', 'hj', 'hq', 'hk', 'ha', 'cj', 'cq', 'ck', 'ca', 'dj', 'dq', 'dk', 'da']

[redacted]
```

A PRACTICAL APPLICATION

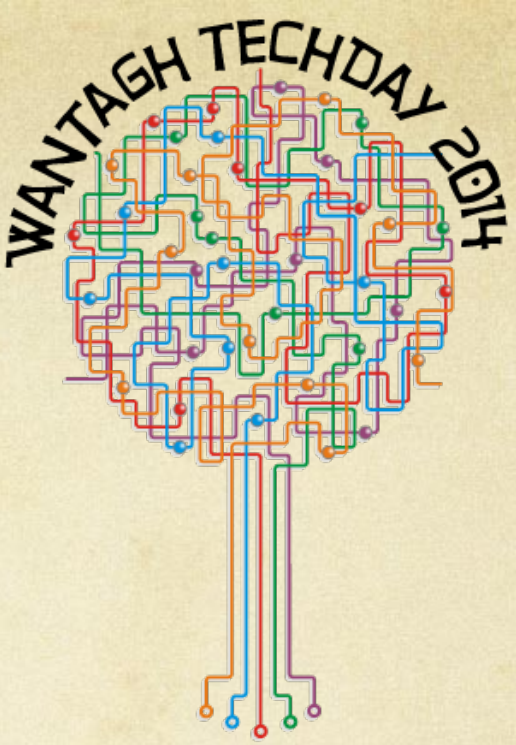
BLACKJACK

An improper deck of cards

```
#-----------------------------------------------
#--------------- Loops -------------------------
#-----------------------------------------------
```



```
#-----------------------------------------------
#-----------------------------------------------
#-----------------------------------------------
```



PART THREE

REVIEW AND TURTLES

REVIEW

ANY QUESTIONS?

TURTLE

- WHAT IS A TURTLE?
  - A LITTLE *"ARROW"* WHOSE MOVEMENT CAN BE CONTROLLED.
  - TURTLE CONTROLS:
    - MOVEMENT *(←DISTANCE→ IS A NUMBER IN PIXELS)*
      - TURTLE.FORWARD(←DISTANCE→)
      - TURTLE.BACKWARD(←DISTANCE→)
      - TURTLE.LEFT(←DEGREES→)
      - TURTLE.RIGHT(←DEGREES→)
    - SPECIAL
      - TURTLE.CLEAR() CLEARS THE *TURTLE SCREEN*
      - TURTLE.UP() *"PICKS UP"* PEN SO TURTLE WON'T DRAW LINE
      - TURTLE.DOWN() *"PUTS DOWN"* PEN SO TURTLE WILL DRAW LINE
      - TURTLE.HOME() BRINGS THE TURTLE BACK TO ITS ORIGINAL

TURTLE

- LET'S TRY PLAYING WITH SOME OF THESE COMMANDS!
  - OPEN TURTLEBLANK.PY WHICH IS FOUND IN YOUR TECHDAY STUDENT FOLDER!
  - GO TO FILE --> SAVE AS:
    - RENAME AND RE-SAVE THE FILE AS TURTLETEST1.PY IN YOUR TECHDAY STUDENT FOLDER.
    - EACH TIME YOU WANT TO MAKE A NEW TURTLE DRAWING, REMEMBER TO RESAVE THE FILE UNDER A NEW NAME (EX: TURTLETEST2.PY).
  - REMEMBER TO ONLY ADD TURTLE ACTION CODE IN THE APPROPRIATE SPACES, THE OTHER LINES OF CODE ARE THERE TO MAKE SURE TURTLE WORKS PROPERLY!

TURTLE

```
1  #Includes special turtle code built-in to python
2  import Tkinter
3  import turtle
4
5  #-----------------------------------------------------------
6  #---------- Insert Initial Commands Here -------------------
7  #-----------------------------------------------------------
8
9  #Here can go things such as changing the color of the
10 #background, the color of the turtle, the size of
11 #line drawn by the turtle.
12
13 #-----------------------------------------------------------
14 #-----------------------------------------------------------
15 #-----------------------------------------------------------
16 turtle.home() # Start Program / Put Turtle in Center ----
17 #-----------------------------------------------------------
18 #---------- Insert Movement Commands Here -----------------
19 #-----------------------------------------------------------
20
21
22
23
24
25 #-----------------------------------------------------------
26 #-----------------------------------------------------------
27 #-----------------------------------------------------------
28 turtle.exitonclick() # Exits Program On Click -----------
29 #-----------------------------------------------------------
30 #-----------------------------------------------------------
31 #-----------------------------------------------------------
```

TURTLE
CHALLENGES

- GO TO YOUR TECHDAY STUDENT FOLDER. YOU WILL FIND A DOCUMENT CALLED TURTLECHALLENGES.PDF WHICH CONTAINS THE INSTRUCTIONS AND TUTORIAL GUIDES FOR EACH CHALLENGE. FOLLOW ALONG AS BEST AS YOU CAN, IF YOU GET STUCK REFER TO THE TUTORIAL DOCUMENTATION, OR ASK ONE OF US FOR HELP!
- GOOD LUCK!