CS105 Problem Solving

Expressions:
Data Types & Operators

Wholeness

- When working with a model (for business or simulation) the data is often numbers. While for any interaction with humans we need strings (text), and for interacting with the computer we need Booleans (true / false).
- On a fundamental level the computer only has ones and zeros, but can create an infinite variety of content with them. In the physical world we see that the unified field (consciousness) is the source of all creation.

Data Types

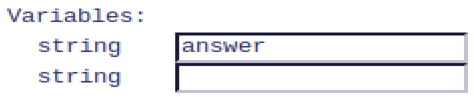
- A variables JavaScript always has a type
  - Types are how the computer interprets the ones and zeros stored at that location
- Flowcharts has the following types:
  - String
  - Number
  - Boolean
  - Array
  - Object

Their Purpose

- Strings
  - Are always for communicating with humans (to some input or output)
  - Examples: Text values like “Hello”
- Numbers
  - Their purpose is always related to a model that is being manipulated
  - Examples: Either natural (int) like 1, 2, and 3 or real (float) like 3.14 and 2.718
- Booleans
  - Their purpose is generally related to program (code) flow (sometimes model)
  - Example: any value can be evaluated to True or False
- Objects and Arrays
  - Both allow us put pieces of data together
  - Example Object: a person has a name and a age
  - Example Array: a shopping list

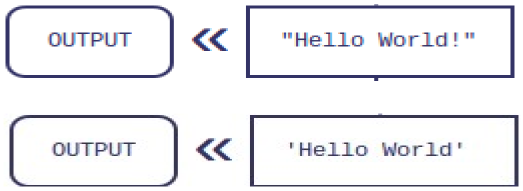
String

- A string is one or more characters
  - It is how the computer works with text data
  - If you store text your variable has to have type string



```
Variables:
  string    answer
  string
```

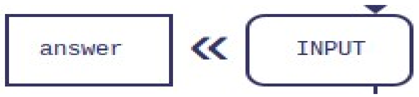
- A String literal is when you ‘hardcode’ some text into your program. These are always in quotes
  - Either double quotes or single quotes



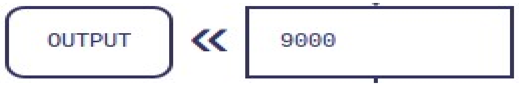
```
OUTPUT « "Hello World!"
OUTPUT « 'Hello World'
```

Input and Output

- Input always gives a string!
  - It always has to store in a variable with type string

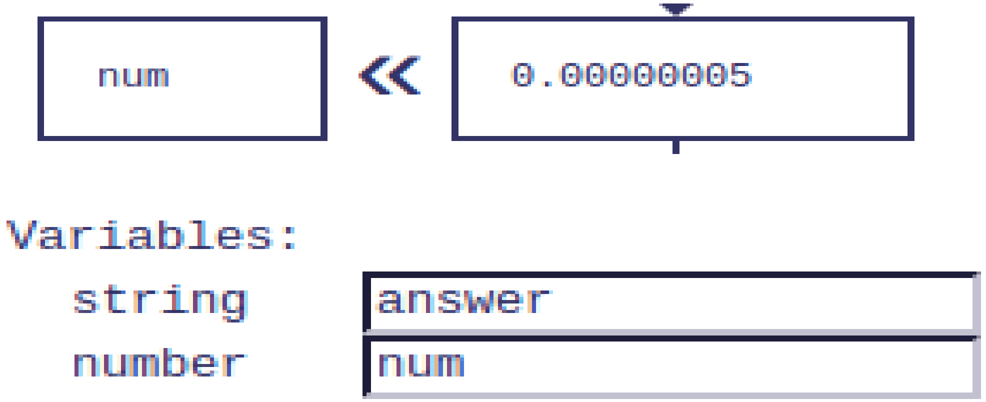


- Output always outputs a string
  - If you give it something else it will convert it for you



Number

- A number can be stored a lot more efficiently in ones and zeros than text can be.
- Therefore the computer uses a different type
- If you store a number your variable has to have type number

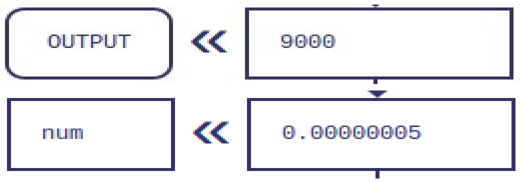


```
num  <<  0.00000005

Variables:
  string    answer
  number    num
```

Number Literal

- A number literal is just the number write out



```
OUTPUT << 9000
num << 0.00000005
```

  - It doesn't need quotes
  - variable names are not allowed to start with a number, therefore the program will not get confused

(strings need quotes to stop confusion with variables)

Integers and Floats

- Most programming languages distinguish between natural numbers and real numbers (they call them integers and floats)
  - Integers are whole numbers like: 5
  - Floats allow fractions like: 534.1236575

- Although we just use Number for both there are a variety of places where the concepts still shows up

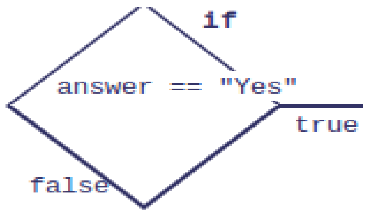
Printing Floats

- Because of the rounding errors that sometimes occur in floats, printing them can be a bit of a hassle.
- You can use the following helper function to print only the specified amount of digits after the period:

```
“123.4675”.toFixed(2) // prints: 123.47
num.toFixed(0)
```

Boolean

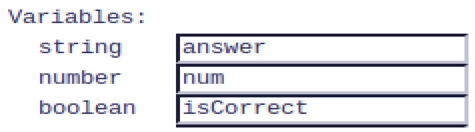
- A boolean value is either “true” or “false”
- The expression in the if statement has to become a boolean



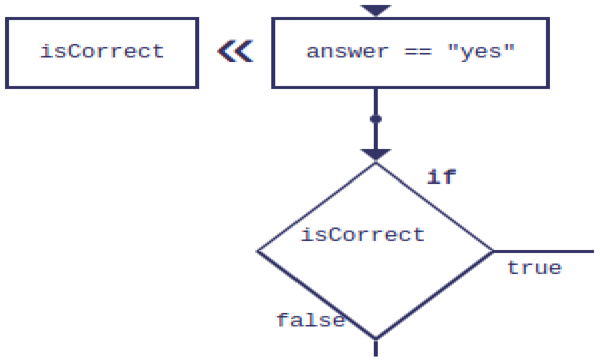
- Booleans can be represented with just a single zero or one, very efficient to store in the computer

Boolean Literals

- There are 2 boolean literals: `true` `false`
  - These are reserved keywords
  - You cannot give a variable the name: true
- Just like other types, you can create a boolean variable, and store boolean data
  - turtle.isPenUp is an example



```
Variables:
  string    answer
  number    num
  boolean   isCorrect
```



Array and Object

- These are more complex types
  - They contain other variables (of any type)
  - The window and turtle used when drawing are examples of objects
  - We will discuss these more next week

Exercise

- Create a flowchart program with
  - A String set to “Hello”
  - A Number set to 5.1234
  - A boolean set to true

- Output all 3 values
  - Output the number with only has 2 places after the period (5.12)

Main Point 1

- A variable has a data type, Allowing the computer to correctly interpret the zeros and ones stored at that location
- It is only by keeping things clear and organized that we can effectively achieve things, TM is a technique to give our mind deep rest, allowing thoughts to become clearer and more organized

Converting between types

- You can always convert between types
  - Often this is even done automatically if the computer recognizes that it’s needed (like output)
- But you can also do it explicitly
  - Because the computer needs it
  - Or to make it clear to readers of your program

Convert to a String

- The easiest way to convert a variable of any other type to a string is to simply concatenate a string to it – the result is always a string

5 + “” → “5”

“” + 7.34 → “7.34”

true + “” → “true”

“” + false → “false”

String to a Number

- To turn a string into a number either:

```
parseInt(“5”)
ParseInt(“5.9”)  // still gives 5
```

- Or:

```
parseFloat(“5”) // 5
ParseFloat(“5.9”) // 5.9
```

- Or:

```
“5” – 0   // 5
“5.9” – 0 // 5.9
```

To Boolean

- Anything used in a boolean context (inside a diamond) will automatically be converted:
  - The empty string “” becomes false
  - All other strings become true
  - 0 becomes false
  - All other numbers become true

- Also adding ! Before something turns it into a boolean (and flips it). Putting !! before something is also possible
  - We will discuss the not operator more coming up

Comparing with Boolean

- When a boolean is used with another type
  - Either comparing or with addition / subtraction
  - It becomes 1 for true, 0 for false

- In general when comparing two types the type with lower precision (boolean in this case) is turned into the type with higher precision
  - After which the two values (of same type) are compared

Exercise

- Check to see if the following are correct by making a small program that outputs correct or incorrect for:
    - “” == false
    - true == 1
    - “12” == 12

Main Point 2

- You can convert data from one type into another. These changes are on the surface, as deep down the computer always just uses ones and zeros.
- We see that harmony exists in diversity

Expressions

- Expressions contain values and operators. They are evaluated to a single value:

```
5 + 8 / 3
“Hello ” + name + “! How are you?”
“text”
4
x
```

- Expressions are composed of one or more variables or literals.
  - When there are more (variables or literals) it has operators to indicate how they are going to be combined to form a single value
  - The result of an expression is always a single value

Operators

- Each data type has operators to combine variables or literals of that type
- As we saw yesterday, there are also comparison operators that create boolean types

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

- Just like in math, operators have precedence

  5 + 4 * 8 != (5 + 4) * 8

String Operator

- The string data type only has one operator
  - The concatenation operator: +

- Unfortunately this uses the same symbol as used for the addition operator with numbers
- The algorithm the computer uses to decide between concatenation and addition is:
  - Only if both sides are numbers is it addition
  - Lets try this out in web-raptor!

Number Operators

- The following number operators exist:
  - Addition: +
  - Subtraction: -
  - Multiplication: *
  - Division: /
  - Remainder (aka modulo): %

- Where just like in math, multiplication, division (and modulo) have precedence over add / sub

Check if it is a Multiple

- Today some of the homework exercises ask you to check if a number is a multiple of another number
- You can check this with the remainder operator
  - If the remainder is 0 it is an exact multiple
  - Example 21 % 7 == 0
  - But 13 % 3 == 1 therefore not a multiple of 3!

Boolean Operators

- There are 3 boolean operators
  - Not: !
  - And: &&
  - Or: ||
- Not flips (true to false, and false to true)

A	B	A && B
T	T	T
F	T	F
T	F	F
F	F	F

A	B	A || B
T	T	T
F	T	T
T	F	T
F	F	F

Operator Precedence

Name	Symbol
Parenthesis	(…)
Boolean not	!
Multiplication, division, remainder	*, /, %
Concatenation or addition, subtraction	+, -
Less than, greater than	<, >, <=, >=
Equality, inequality	==, !=
Boolean and	&&
Boolean or	||

When in doubt, add more parenthesis.
“Bugs are expensive, parenthesis are free”

Exercise

- Write expressions that show
    - What the remainder is of 17 / 5
    - The result of: 5 + 3 * 2
    - The result of: true > 2 || !(false + 1)

- Are the outputs what you expected them to be?

Main Point 3

- Expressions can contain a variety of operators, just be careful that that 1) you’re using the right operators for the data type and 2) you’re not caught by surprise by operator precedence
- Just like the principle of diving, we want to take the right angle and let go, let the computer do the work. But in order to take this right angle our mind has to be clear, which is where TM comes in.

Summary

- We saw the 3 main data types: Strings, Numbers, Booleans
- It’s relatively easy to convert from one type to another
- Expressions use operators to combine 2 pieces of data into a new value