Week 1

Sasdijaosi

Day 1 (intro to course)
Notes

- What do theses have in common?
  - 256
  - CCLVI
  - IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIi
- They all represent a number
- Given two collections, can you determine which is bigger WITHOUT COUNTING?
  - We pair the objects off from A and B
  - If B has leftovers
    - B is bigger
  - If there are no leftovers
    - They are of equal size
  - If A has leftovers
    - A is bigger
- “Same Sizeness” abstract notion of number
- Human progress is tied to numbers
- Five operations that are really importan t
  - Addition +
  - Subtraction -
  - Multiplication *
  - Integer Division //
  - Modular division %

Day 2 (Base systems)

- Tally mark activity (how to do operations with tally marks
  - + ~ Concatenate the tally marks
  - - ~ Pair off the smaller subrahend from the bigger. Difference is what left
  - * ~ for each tally mark in the first vactor, make a copy of the second. Then glue em
  - // ~ To get b//a chop a out of b and keep track of the number of chops. c=b//a
  - % This the number of marks remaining after chopping (refer to integer division)
- We can not use tally marks as a base system because it is not scalable with larger numbers
- Denominational number system (roman numerals)
  - Yk this system
  - Arithmetic with this is weird no cap
  - Arithmetic in roman numerals is inconsistent and rly hard
- Denominational number system (money system)

- Denominational number system (binary system)
    - Denominations are powers of 2
    - If you are doing base b: 1, b**2, b**3,.....
- We learned about base 2 and base 4 systems
- Binary example (528)
    - 1000010000
- Packing (528)
    - Binary ~ 10 00 01 00 00
    - Base 4 ~ 2 0 1 0 0
- Base 8 example (528)
    - Binary ~ 001 000 010 000
    - Base 8 ~ 1 0 2 0
- Base 16 (hexadecimal)
    - Alphabet = { 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}
- Base 16 example
    - 0010 0001 0000
    - 2 1 0
- X number
- B base
- x//b is the number with last digit chopped
- X %b is the last digit

Day 3 (Intro to data types)

- Cmw = compiler
- REPL
    - R - Read
    - E - Evaluate
    - P - Print
    - L - Loop
- Python does not have a number limit
- Integer - whole numbers
    - Arithmetic with it is simple
    - Whole numbers you can add subtract multiply and divide
- Floating point number - a decimal number
    - Floats are IEEE double precision floating point numbers
    - All arithmetic can be sued
- Casting - a temporary request to force a number to be a certain data type
    - When an float is casted to an int it is truncated (decimals are cut off)
    - When an integer is turned to a float a decimal is added
- Complex number type
    - To make a complex number use function complex(a,b)
    - a= the real number
    - b= imaginary number
    - All arithmetic can be used

- Variable name rules
    - Must start with alpha character or _
    - Succeeding characters are alphanumeric or _
- Snake notation
    - Separating a variable that has multiple words with underscores (python)
- Camel notation
    - Separating a variable that has multiple words with Capital letters (java)
- Variables do not have types
    - They are just referring things
    - Variables are means to talking to object
    - Variables allow you to refer to objects
    - Variables point at objects
- NC17 explanation
    - Two types of memory
    - Stack and HEAP
    - All the objects live in HEAP
        - Giant storage position
    - The place where all our variables live is Stack
- Variables have no type but are the means to which we can find the object
- Whenever a object loses the variable that refers to it after awhile a “garbage collector” comes and murders it
- Strings
    - Strings store globs of text
    - Syntax for string is that it needs to have parenthesis on both sides
- Boolean
    - True or false basically
- Operational functions
    - >
    - <
    - == (remember to have two)
    - <=
    - >=
    - !=
- Operators
    - And
    - Or
    - Not
- We will be asked what the order of operations is
- Indexes
    - Strings have an index for each letter
    - Starts at 1
    - “Indexes are like rats they live in the walls”
    - Using : (called slicing) you can get the index between em
    - It stops right before the last nume

        - Ex x[1:4]
        - Ex x[:4]
        - Ex x[4:]
        - Ex x[::2]
        - Ex x[1::3]
- Type determines context
    - + adds numbers
    - + concatenates strings
- You can not “add” integers/floats with strings
- upper() - uppercase it
- lower() - lowercase it
- These are called a method/function
- Objects have three attribute
    - State
    - Identity
    - Behavior

Week 2

lol

Day 4 (uh python?)

- we did not have much notes just a lab

Day 5 (more python)

- string syntax
    - Can have single quotes or double quotes
    - make sure to have quotations on both sides
    - three quotes you can do multi line strings

```
string = “lol”
string = ‘hi’
lol = ‘’’ asodijaoisdj
asdoijasoidja
asodijaoisdj ‘’’
```

- by putting r in front you get the raw string

```
 x = r”lol this thing is raw”
```

- ord() function
    - In Python, the ord() function accepts a string of unit length as an argument and returns the Unicode equivalence of the passed argument. In other words, given

string of length 1, the ord() function returns an integer representing the Unicode code point of the character when the argument is a Unicode object, or the value of the byte when the argument is an 8-bit string.

```
ord(“string”)
```

- order a string
    - dictionary ordering
    - compare first if one lesser, report that as less
    - repeat for succeeding characters
- lexicographical
- The scheme of ording the characters is called ASCII
- list
    - a type of variable
    - can hold multiple objects in one variable
    - can hold different objects
    - has an index starting at one

Day 6 (sequential stuff like lists)

- == checks equality in state
- is checks for identity between variables
    - this does not apply for strings

```
x = []
y = []
x == y
true

x is y
false
```

- strings just pool

```
x = “moo”
y = “moo”
x == y
true

x is y
true
```

- This next process is called aliasing
- both of these are pointing to the same object

```
x = []
y = x
y.append(5)
y
[5]
x
[5]
```

- This next data type is called a tuple
- tuples cannot be changed
- they are nonimmunatble objects

```
t = (1,2,3)
t[0]
1
t[0] = 5
ERROR
```

- **<u>python sequence types</u>**
    - lists
        - This is a heterogeneous mutable sequence types
    - tuples
        - This is a heterogeneous immutable sequence type
    - strings
        - This is an immutable sequence of characters
- This is a heterogeneous mutable sequence types
- len() for different sequence types
    - This function will return the number of elements in list or tuple
    - Strings will just be returned with the number of characters
- in function
    - asks if an object is in a tuple or list

```
t = [1,2,3,4]
7 in t
false
4 in t
true
```

- in function for trings
    - asks if a continuous letter combination is in a string

```
x = “cowpie”
“cow” in x
true
“ce” in x
false
```

- You can assign a list to part of a list to slice it

```
t = [1,2,3,4,5]
t[1:3] = []
t
[1,5]
```

- sequences have iterators inside them
- iterators determine how a sequence is walked through
- **important links**
    - Python Built in Types
- dir()
    - tells what things are visible
- method vs. function
    - len(s) ~ function
        - functions the variable is passed just passed through
        - it is not attached to anything
        - they only return a cope
    - c.upper() ~ method
        - methods depends on the state of the object it is attached to
        - it changes the object forever

```
x = [5,1,-3,7,4]

sorted(x)
[-3,1,4,5,7]
x
[5,1,-3,7,4]


x.sort()
x
[-3,1,4,5,7]
```

- sort() is a mutator method because it changes the list

- sorted() is a function because it only returns a copy of the changed thing
- **Range type**
    - range(5)
    - range just gives the numbers between 0 and the number
    - it could also start at another number and stop at the number
    - it can also count by numbers also
    - it also only works on integers

Day 7 (printing)

- Files used for example
    - tree.py
    - triangle.py
- print() function
    - basically prints the stuff between the parenthesis

```
print(“hello”)
hello
```

- something new i learned is that by putting commas between different types of functions you can have it print it just with spaces in the middle of each object.

```
print(1,2,3, True, "Cows", [1,2,3], sep = "|")
```

- # is used for commenting

```
#HI PERSON
```

- dir()
    - The things that are visible
    - The variables that are visible
    - symbol table: dictionary of all variables and their values

Week 3 (started to put date)

python lol

Day 8 (Boss Statements) [8/31/21]

- Files used for examples
    - fun.py
    - uhmcubed.py
    - iffy.py
- Programming has these features
    - Selections

        - You cna decide, based on what your prgram knows (program state), to select code. This includes omitting code we might not want to run
    - Modularization
        - When eating an elephant, heres a tip DONT TAKE TOO BIG OF A BITE
        - Eat a spoonful at a time. When faced with a complex problem =, we break it into simpler problems
        - We keep doing this until the simple problems can be readily solved. We then orchestrate their solutions into a situation as a whole
    - Iteration
        - Sometimes we want to run a segment of code repeatedly
- Languages missing one of these features is not a full - blwn computer language
    - HTML breaks a page into elements, but had no capability of terating
- Every computer language has two types of statements
    - Worker Statements
        - These are grammatically complete imperative sentence with tacit subject”computer/.
    - Boss statements
        - These are grammatically incomplete subordinating clauses
        - Tjese statements control a list of statements called a block
        - Said block will complete the boss statements as a [complex] complete sentence
        - Only boss statements can alter the flow of execution of a program
- def = “To define”

```
#***********FUNCTIONS*************
def square(x):
    return x*x

#**********MAIN***********
num  = 4
newnum = square(num)
print(newnum)

print(f"square {num} = {square(num)}")

OUTPUT
16
square 4 = 16
```

- Stack
    - Manages function calls
    - The global frame can always be seen
    - Functions that are called only can be seen when they are active

- Heap
    - The warehouse where all objects are kept
- Stdout (standard output)
    - Terminal where output is located
- Local variables only exist in the functions
- What is a function in a math sense?
    - A function has three parts
        - D, a set that is the domain, or the allowable inputs for the functions
        - R, a set that is the codomain (range), or the space of allowable outputs
        - f, a rule tying each element of the domain to some element in the codomain
- len is a mathmatial function
    - D = all sequences
    - R = integer (non-negative integer)
    - f, return the number of items in the sequence according to its iterator.
- Mathematical functions transform a datum with no side effect.
- The print function

Day 9 (Random function) [9/2/2021]

- Files used
    - Randy.py
    - helpfunction.py
    - iffy.py
- Mathematical functions always output the same
- Random
    - Library with a bunch of functions that randomize
- Random.int()
    - random.randint is INCONSISTANT. It is not a pure function

```
import random

#random.randint is INCONSISTANT. It is not a pure function
for k in range(10):
    print(random.randint(1,6), random.randint(1,6))
```

- side effect
    - Something that happens that persists beyond the lifetime of that function
- help()
    - function that tells the documentation of different functions
    - it also tells the documentation of functions from imported libraries
- docs string
    - This is a triple-quoted string right after the function header
- To get help use help() or .__doc__

- How to make it so you own function shows up and tells documentation

```
def square(x):
    """preconditioin: x is a number.
    postcondition: returns the square of x
    """
    return x*x

#***********MAIN***********
print(square.__doc__)
```

- Preconditions
    - specify the number and types of inputs and anything else that should be truen when calling this function
- postconditions
    - this is what is true once the function has done its job.
    - This includes return value and side effect
- if statements
    - if a condition is true the code under it runs
- elif
    - uhm yk what this is idr how to explain
- else
    - executes if all elif an if statements do not run

Week 4

lol

Day 10 (type_checking) [9/7/21]

- files used
    - type_check.py

```
def dublin(x):
    return x*2

def bullet_proof(x:int) -> int:
    return x*2
```

- the -> int just tells that f() returns an integer (but it doesn't force the function to return an integer
- Arguments vs. Parameter
    - Arguments
        - Unlike argument in usual mathematical usage, the argument in computer science is the actual input expression passed/supplied to a function, procedure, or routine in the invocation/call statement
    - Parameter
        - Parameters are the names of the information that we want to use in a function or procedure
        - the parameter is the variable inside the implementation of the subroutine.
- Expressions
    - The term "expression" means a computer program statement that evaluates to some value.
- Complex statement
    - not one instruction, but multiple instructions
- Python is a pass by value language
    - What gets passed to a function, a parameter? A copy of the memory address of the object being returned.
- Python variables only store **MEMORY ADDRESSES** on the heap.
- Python is a dynamically typed language
- Arguments type
    - positional arguments
    - keyword arguments

Day 11 (input and iteration with functions) [9/9/21]

- file used in class
    - hi.py
    - nagging.py
- functions can call themselves. This means there is a method of iteration without using loops. This is called recursion

```
 def print_list(x):
    if(len(x)==0):
        return
    first = x[0]
    rest = x[1:]
    print(first)
    print_list(rest)

#********MAIN*********8
x = ["nyah", "yah", "nyah"]

print_list(x)
```

- Input() function
    - gets input from the user
    - can be anything typed
- c(n,k) - this is how many subsets of size k in a set of size n
- c(n,k) = c(n-1,k-1) + c(n-1,k) [uhm this weird thing he did]
    - c(n,0) = 1
    - c(n,n) = 1
    - c(n,1) = n

Week 5

uhm

Day 12 (for loops) [9/14/21]

- files used
    - urmom.py
- What is an iterable?
    - an iterable is a collection of objects that contain an iterator, which dictates how an object is walked through and what order the items are seen
- Types of iterable
    - lists: one item at a time
    - tuples: one item at a time
    - range: one item at a time
    - strings: one character at a time
- iterable are “food” for for loops
- stargument - it may be treated as a list
- import
    - python has libraries
- https://www.youtube.com/watch?v=crluPcyuchU&list=PL98qAXLA6afuh50qD2MdAj3ofYjZR_Phn&index=27

Day 13 (more loops and files) [9/16/21]

- Files used in class
    - and i loop.py
    - filestuffig.py
- Reversed loop

```
for i in reversed(10):
    print(“lol”)
```

- enumerate()

    - gives each object in an iterator like a genuine index.
    - It outputs a list with each object in a tuple containing the object and the index of that object
    - Pretty cool function idk when to use it itho
    - We can use this in functions too
- You can apply enumerate with files also.
- Importing stuff ~ there are three ways of importing something
    - intimate import
        - ex: from math import *
    - regular import
        - import things into programs this way
        - ex: import math
    - selective import
        - importing the functions that you want
        - This is good for getting a couple of things from a module
        - ex: from math import sin, cos
- dir()
    - tells what functions are currently in the program
- math module
    - a frick ton of math functions that you can import
- sys module
    - The sys module in Python provides various functions and variables that are used to manipulate different parts of the Python runtime environment
    - There is a function in here that can increase the max amount of recursions than occur
- The largest amount of recursions that can occur is 1000
- a for loop and an if statement can be used to make a filtering algorithm
- Indefinite Loops ~ While loop
    - the while loop keeps looping until the statement is no longer true

Day 14 (algorithms) [9/17/2021]

- Files used
    - powerstack.py
- Algorithm 1 ~ Divide and Conquer Power Algorthm

```
def power(x,n):
      if n == 0:
            return 1
      if n < 0:
            x = 1/x
            n = -n
      return n*power(x,n-1)

#basic algorithm but uses a fuck ton of frames
```

```
#heres a better one
```

- Dr. Morrison's algorithm in regular nothation
    - 3^10 = 9^5
    - = 9*9^4
    - = 9*81^2
    - = 9 * 6561
    - = 59049

Week 6

wap

Day 15 (Hashing) [9/20/21]

- Files used
    - kay.py
- A hash function is a function whose domain is some type of object and whose codomain is the integers. Within the lifetime of a given program, a hash function is a mathematical function. Generally, hash functions are defined in terms of object state. As a result, only immutible objects are hashable.
- Hashability Hashing is a technique that allows for the rapid retrieval of data from sets and dictionaries. Hashing breaks a set into "buckets"; deterining the bucket to look in involves calling a hash function on an object. This is a quick operation. These buckets are maintained by an object called a hash table.
- When the hash table becomes too crowded, it is enlarged and all objects are rehashed and placed into the newer larger table.
- Here is a good way to think of a hash table. It is a giant array of memory, say of size M. When an object is to be inserted in this table, we mod by M to get result loc and place that object in a list that lives at entry loc. When the table crowds, the lists get too long and efficiency is lost. At that time, rehashing occurs.
- Mutable objects are not hashable because hash functions depend on object state and the must be mathematical functions, and therefore be consistent.
- Hash method in Python is a module that is used to return the hash value of an object. In programming, the hash method is used to return integer values that are used to compare dictionary keys using a dictionary look up feature
- A set is an unordered data structure containing no duplicates. A dictionary is an associative array that maintains paired objects. Both are hashed data structures, in that they only admit the insertion of hashable objects.
- Hashing
    - Depend on object state

    - Only work on mathematical functions
- Hashing uses
    - Passwords
    - Security
- set()
    - another type of array making shit
    - not ordered in any way
    - you can add tuples to a set but you can not add lists
    - also you can not have multiples of an object
- Set notation things (im sorry Al)
    - A
- Dictionaries
    - There is a “key” and a value is tied to it
    - keys MUST be hashable

Day 16 (making a concordance of war and peace) [9/21/21]

- so dr. morrison decided to make a concordance of war and peace
- yeahhh i dont really understand it
- tuples are ordered lexicographically

Day 17 (binary search algorithm) [9/24/21]

- You kinda know this algorithm
- Basically you get the middle element and see if it is greater or larger
- if greator or larger cut it in half
- keep doing until you either get the element or it doesnt exists

Week 7

lol

Day 18 (more algorithms) [9/27]

- files used
    - eloel.py
- If a function f is continuous on an interval]a,b] and if f changes sign on that interval, then it must have a root in [a,b]
- Let’s approximate the square root of 2. Put f(x) = x^2. f(x) = 2. f changes again on [1,2]

Day 19 Bozo sort [9/30]

- files used
    - bozosort.py
    - bestiesort.py
- Sort Technique Number 1
    - get the smallest element of the list

    - swap to the start of the list
    - go throught the whole list
- Sort technique number 2 (adjacent pairs)
    - Pointer at end
    - goes from beginner swap adjacent pairs if they are out of order
    - max values ar pushed to the front
- Sort technique number 3 (trickle down)
    - everything on the left side is ordered
    - pointer is on element 1
    - every time the things is bigger the number is trickled down

Day 20 (uhm) [10/1/21]

- file used
    - lol.py

Week 8

lol

Day 21 (Primitive Java Types) [10/5/21]

- Files used today
    - hello.java
- Program LIfe cycles
    - Python
        - You write source
        - you save to disk
        - you run python on it
        - you code begins execution and dies if an error occurs
        - If no error occurs, your program has run successfully
    - Java
        - You write source
        - you save it to disk
        - You compile. If the program does not make syntactic sense, no code generation occurs and there is nothing to execute. Errors occurring here are compile time errors
        - If your code compiles, you get a file containing executable code. YOu are not out of the woods. If you do something dumb, such as accessing a nonexistant index in a string, you get an exception, or run-time error
        - Barring any run-time error, your code successfully executes.

```java
public class Hello
{
    public static void main(String[] args)
    {
        System.out.println("Hello, Java!");
    }
}
```

- public class hello
    - class name must equal file name
- public static void main(String[] args)
    - main thing ig
- System.out.println(“Hello, Java!”)
    - print function, “ln” makes a new line
- jhel
    - the interactive repil for java (the equivalent of typing python in the cmd window)
- In python you hafta declare the type of the variable
    - so like you gotta tell whether the object is a string, integer, or whatever
- comments
    - // ~ on line comment
    - /* blah blah */
- The java integer type is a 32 bit integer. If you go to the top and add one you wind up with the very bottom
    - going above the size will result in a type overflow
    - to reiterate: int is a 32 bit integer stored in two’s complement notation
- booleans
    - basically the same but they are not capitalized
    - true and false are the boolean constants
- Arithmetic
    - Addition, subtraction, and multiplication are the same
    - division of integers is integer division
    - For power of things you gotta call a math power function
- Float types (there are two)
    - float is a 32 bit single precision IEEE 754 number
    - double is a 64 bit double precision IEEE 754 number
        - more like python’s float and we will use it more
- Math.function()
    - you don't hafta import it
    - it is also capilized
    - gots all the functions we like
- byte

    - data type that stores a byte
- short
    - short is a two byte data type
- long
    - long is a 64 bit integer
- char
    - characters (like a bit of a string)
    - represented by unicode
    - java characters are UNICODE
- Java has 8 primitive data types
- Summary of data types
    - data types used to represent integers
        - byte
        - short
        - int
        - long
    - boolean
    - float
    - double
    - char
- Primitive data types store their objects directly and are on the outside.
- Every other data types store the address to the object like python
- There is also no global objects in java\
- If a data type is not a primitive data type then it is an object
- Strings object
    - just declare it as a string
    - strings are a different types
    - considered just an object
- ArrayList object
    - a list
    - same concept different rules than python tho
- Math object
    - math lol idk
- If the data type is uppercase then it is an object, if it is lowercase then it is a primitive data type
- for loops
    - for(initializer; test; between)
    - initializer executes once
    - test is carried out, if true, loop's block executes
    - between -> test -> block until test is false
- Java is block scope while python is function scope

Day 22 (java class) [10/7/2021]

- files used
    - class.py
    - Point.java
- A class is a blueprint for the creation of objects. When you make a class, you will specify state and behavior. The JVM creates objects from classes and gives them identity.
- making something “final” means that it cannot be changed.
- encyclopedia of java

Day 23 (java bestie) [10/8/2021]

- files used
    - Commando.java
    - Wabbit.java
- both java and python
    - class This is a blueprint for the creation of objects. In it, you specify state and behavior.
    - instance: Objects created using a class are called instances of the class.
    - method: this is function attached to an object. All functions in Java are methods. Methods can depend on object state.
- java stuff
    - constructor: this it the OBGYN of Java objects created by class. An imporant postcondtion is for all state to be explicitly initialized. It runs every time a new object is creatd by a call to new. The constructor must have the same name as the class, and it is the only method with no return type.
    - new Use this keyword when creating a new object. It causes the constructor of the class to be called.
    - tacit call to new: This occurs when making Strings. It is idomatic in Java not to use new when creating a stirng. 99.44% of the time you will do this.

```
String s = "fooment";
```

    - signature: The signature of a method is the list of types in its parameter list. The parameter names are not a part of the sig.
    - method name overloading: Several methods can have the same name if they have different sigs. Use this only for closely-related methods.
    - void Use this keyword as a return type when a method has no return value. It is an inheritance from the C language.
    - public When used on a class, this says the class is visible outside of its file. When used on a method, it says the method is visible outside of the class.
    - private When used on a method or a state variable, , this says the item is not visible outside of its file.
    - void Use this keyword as a return type when a method has no return value. It is an inheritance from the C language.
    - public String toString() This method allows you to create a string representation

of your object.
    - public boolean equals(Object o) This method allows you to decide when your object is "equal" to another object. Note that objects of different types should never be declared equal. This is the species test.
    - package: This is a collection of related classes.
    - module: This is a collection of related packages.
    - Object This is the root class in the Java class hierarchy. All classes, including user-defined ones, come equipped with all of the Object methods.
    - Override
        - a thing that “overrides” a method in a class and the compiler checks that you did that method right and will return a compiler error if it is wrong.
    -
- Python stuff
    - dunder method: This is a method in a Python class that has a special puropose.
    - __init__ This method is called when a new instance of a class is created. It acts in a manner similar to a constructor in Java.
    - __eq__ This method is called when == is used to compare two objects.
    - __str__ This method gives a string representation of an object. It is automatically called on an object by the print function.
- Making arrays
    - arrays are fixed length

```
int[] nums = new int[10]; //makes an integer array of 10 zeros

// im pretty sure u gotta put values for each place

//printing arrays

System.out.println(Arrays.toString(nums));


String[] arr = new String[10]; //makes a array of null strings you gotta
put some shit in there
```

Week 9

lol

Day 23 (arrays and wrappers in java) [10/12/21]

- **The class ArrayList<T>** The T is the *type parameter*. Classes using a type parameter are called *generic classes*.
- The type parameter represents the type of item that is present in the array list. We will write a very simple generic class Pair<S, T>.
- Autoboxing ~ the automatic conversion that the Java compiler makes between the primitive types and their corresponding object wrapper classes.
- unboxing ~ Converting an object of a wrapper type (Integer) to its corresponding primitive (int) value
- use primitive types in loops

Day 24 (more more java lava) [10/14/21]

- material from s6.pdf and s7.pdf

For loops in java and then python

```
for(Type item: collection) /*Java*/
{
    Block
}

for item in collection: #Python
    Block
```

a for loop that i do not know how it works

```
for(intit: test: between) /*Java*/
{
    BLOCK
}
```

if else statements java and python

```
if(predicate) /*Java*/
{
     TRUEBLOCK
}
else if(predicate2)
{
     TRUE2BLOCK
```

```
}
else
{
     FALSEBLOCK
}

if predicate: #Python
   TRUEBLOCK
elif predicate:
   TRUEBLOCK2
else:
    FALSE
```

Switch case statement

```
public class StandOld
{
    public String fruit(char c)
    {
        String out = "";
        switch(c)  //chars, integer types, Strings, enums
        {
            case 'a': case 'A':
                out = "apple";
                break;   //each case MUST end with a break.
            case 'b': case 'B':
                out = "blueberry";
                break;
            case 'c': case 'C':
                out = "cherry";
                break;
            case 'd': case 'e':
                out = "kumquat";
                break;
            default:
                out = "No fruit with this letter";
        }
        return out;
    }
}
```

Ternary Operator
In java then python

```
public class Foo /*Java*/
{
    public double f(double x)
    {
        return x < 0? -x:x;
    }
}

#python
def f(x):
    return -x if x < 0 else x
```

- Big Integers
    - This class gives Java a Python-like arbitrary-precision integer type. You will notice some important differences with Python, most of which are fairly cosmetic.
    - Documentation on java site
    - You hafta import this class to use it

Recursion and starguments in Java

```
import java.math.BigInteger;
public class Fact
{
    public static BigInteger factorial(int n) /*recursion*/
    {
        if(n < 0)
        {
            throw new IllegalArgumentException();
        }
        if(n == 0 || n == 1)
        {
            return BigInteger.ONE;
        }
        return BigInteger.valueOf(n).multiply(factorial(n - 1));
    }
    public static int atom(int... nums)
    {
        int out = 0;
        for(int k: nums)
        {
            out += k;
```

```
        }
        return out;
    }

    public static void main(String[] args)
    {
        System.out.println(factorial(100));
        System.out.println(factorial(3));
        //System.out.println(factorial(-3));
    }

}
```

While loops java

```
/*regular while loops*/
while(predicate);
{
    BLOCK
}


/*while loop but the block executes at least once*/
do
{
    BLOCK
} while (predicate);
```

Week 10

lol

Day 25 (Terminal + BIG FRACTION) [10/19/2021]

- Terminal notes Video
    - cd
        - changes to a directory (how i get into a folder)
    - ..
        - goes to file above

    - javac
        - complies program
    - java
        - runs program
    - tab
        - completes file it for you automatically
    - dir
        - shows files

Week 11

lol

Day 26 (Interfaces) [10/25/21]

- files used
    - Circle.java
    - Shape.java
    - Rectangle.java
- What is an interface? An interface is a contract. An interface consists of a collection of method headers
- An interface is also a data type and you can declare varialbes of interface type and you can use interface types as type parameters in generic classes such as ArrayList<T>.
- This magical property is called polymorphism
- It means "having multiple shapes." A variable of interface type can point at
- Who signs the contract? A class can sign an interface contract using the keyword implements.
- Visibility: Type of variable determines what methods are visible
    - Meaning the interface can only see the methods inside the interface
- Delegation : Execution is delegated to the object
    - Interfaces are types

Day 27 (Collections) [10/26/21]

- Today we are learning more about collections
- Types: Both classes and interfaces represent types. Here are the rules for being a sub/supertype
    - If T is a subtype of S, then S is a supertype of T.
    - Every type is a sub and supertype of itself.
    - If T is a class type and S is an interface type and T implements S, then T is a subtype of S.
- Rules for pointing
    - A variable of a given type can point at any object that is a subtype of the variable's type. For example, you can do these things.

- List<String> ell = new ArrayList<>();
- List<String> ell = new LinkedList<>();
- List<String> ell = new Vector<>();

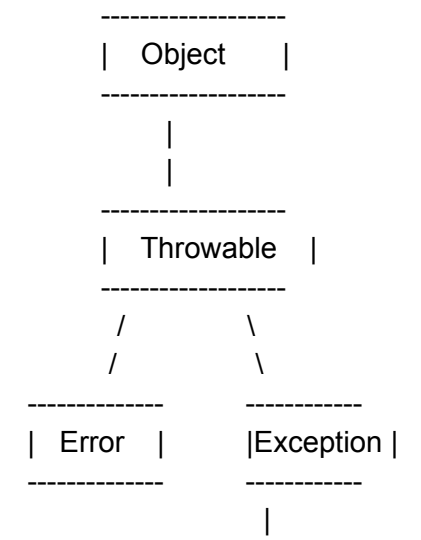
Day 28 (More stuff) [10/28/21]

- Composition
  - THe most common relationships between classes. It is a “has a” relationship
  - A BigFraction has two BigIntegers as a state. This is composition
- Inheritance: The Basics
  - One-Parent rule. You can have only one parent class
- Abstract classes and the abstract keyword
  - Any class can be declared abstract and and thereby by class-strated in the sense that you can’t make instances
  - A bodiless method MUST be declared abstract
  - Any class contain and abstract method must be declared abstract
  - A non-abstract class class is called called concrete
  - Any concrete class must implement all methods form:
    - Any interfaces its implements
    - Any abstract classes it inherits from
    - THese contracts can be fulfilled via inheritance

Day 29 (El o el) [10/29/2021]

-

The Throwable Subtree



Inevitable ----------
Death! | RuntimeException |

Largely programmer
goofs:
IllegalArgumentException
ArithmeticException
IndexOutOfBoundsException

It is optional to guard against runtime exceptions. All other exceptions must be handled by the try-catch-throw-finally mechanism.

Week 12

el o el

Day 30 (Java FIles) [11/2/2021]

- Java.nio.path
    - Path This is an interface with a static method of. This method can be called two ways.

```
jshell> Path p = Path.of("foo", "baz", "bar");
p ==> foo/baz/bar

jshell> Path q = Path.of("foo/baz/bar");
q ==> foo/baz/bar

jshell> p.equals(q)
$3 ==> true
```

    - You can use it to create absolute or relative paths.
    - You will notice an (abortive) static service class Paths; don't use it. Prefer Path.of.
    - Paths refer to possible locations in your file system.
    - Useful methods
        - getParent()
        - isAbsolute()
        - startsWith(String prefix)
        - endsWith(String suffix)
- java.nio.Files
    - There are a variety of ways of getting file properties.
    - exists(Path p)
    - isReadable(Path p)
    - isWriteable(Path p)

    - isExecutable(Path p)
    - isHidden(Path p)
    - isRegularFile(Path p)
    - isDirectory(Path p)
    - size(Path p)
- Reading a File
    - Files.newBufferedReader(Path p)
    - startsWith(String prefix)
    - endsWith(String suffix)

Day 31 (Java files continued) [11/4/2021]

- void delete(Path p) This will delete an empty directory or a regular file sitting at the specified path. A NoSuchFileException is thrown if the path points at a nonexistent item.
- void copy(Path source, Path dest, CopyOption... opts) This will copy a file sitting at source to path dest.
- byte[] readAllBytes(Path p) This will read all of the bytes into a byte array.
- String readString(Path p) This will read a text file into a String.
- List<String> readAllLines(Path p) This will read all of the lines of a file into a list of strings.
- Stream<Path> list(Path p) This provides a stream of paths in p if p is a directory and throws an exception otherwise.
- What can you get Stream from?
    - Collection<T>: Any class implementing this interface. Subinterfaces include List<T> and Set<T> and Queue<T>.
    - ArrayList<T>
    - TreeSet<T>
    - HashSet<T>
    - Arrays.stream will return a stream of the array's elements.
- Also, a BufferedReader has a lines() method that can offer you a Stream<String>. Files.list will give you a stream of a directory's contents.
- Filtering Streams
    - There is an interface called predicate

Week 13

el o el

Day 32 (Character smth) [11/9/2021]

- Characterclassese
    - This is the super-little language of character wildcards.
- Here are list character classes. These can contain asciicographical ranges. Inside of these, - is a magic character and must be defanged with a \. This is also true for [and].
    - [a-j] matches any character in the asciicographical range a-j.

    - [aqm5$] matches any of the characters (list character class)
    - [a-zA-Z] matches any alpha character
    - [0-9] matches any digit
- NOTty Character Classes
    - [^a-z] matches anything BUT a-z.
    - ^ must go at start or it's just a ^.
    - [^a-zA-Z0-9] matches any non alphanumeric character.
- Special Character Classes
    - . matches any character except \n
- Regexese
    - This is a little language that specifies patterns in text.
- Juxaposition "and then immediately."
    - a[bcd][0-9]
    -
    - starts with a
    - then a character b, c or d
    - then a 0-9.
    -
    - ad7
    - cowabungad7
    -
    - ..eati..
    -
    - any two chs except newline
    - e
    - a
    - t
    - i
    - any two chs except newline
    -
    - defeating
    - cheating
    - treating
    - creating
    - creation
    -
    -
    - ^..eati..$
    -
    - cheating
    - treating
    - ideation
    -
    -

- Match a valid telephone number of the form XXX-XXX-XXXX
-
- ^[2-9][0-9][0-9]-[2-9][0-9][0-9]-[0-9][0-9][0-9][0-9]$
- ^[2-9][0-9]{2}-[2-9][0-9]{2}-[0-9]{4}$
-
- Multiplicity operator is a postfix unary operator.
- Juxtapositoin has lower precedence.
-
- {m, n} at least m but not more than n
- {m,} at least m
- {m} exactly m
- * zero or more of
- + one or more of
-
- \s+

Day 33 (Regular Expressions even more) [11/12/2021]

- Attention VSCode Users! You can search programs using regexes! We vim users have had this all along.
- [a-zQ5&#] list/range character class
- [^a-zQ5&#] snotty character class: anything but chars shown
- m this non-magic character is just.... m.
- . any character except \n
- \s whitespace
- \S anything not whitespace
- \d digit
- \D non-digit
- ^ start of the line
- $ end of line
- \w any alpha character or _
- \W anything but an alpha character or _
-
- Magic characters in list character classes:
-
- [start list character class
-] end list character class
- - indicates a range
- ^ when first, makes a list character class snotty
-
- Assertions :
-
- ^ start of the line
- $ end of line
- \b word boundary

- \B not a word boundary
-
- Multiplicity Operators
-
- * zero or more
- + one or more
- {n} exactly n of
- {m, n} at least m but not more than n of
- {m,} zero or more
- ? once or not at all
-
- Operators
-
- juxtaposition "and then immediately"
- | or
- (...) override the order of operations
-
-
- Magic Toggler:
-
- \
-
- Example \? is a question mark, not a multiplicity operator
- d is the nonmagic letter d, \d is a digit.
-
- Java
- java.util.regex package. Note that in Java a \\ is needed for a \.
-
- Python:
- re the regular expression package
-
- match an integer: ^[+\-]?\d+$
-
- Amy Sheck is bonkers
- John Morrison is loony!
- John Morrison is crazy
-
- ^(Amy Sheck|John Morrison) is (bonkers|loony|crazy) [!.]?$
-