Name: ___Sarika Ibnat

Class: Programming&Prototyping

Guidebook

Unit 1

Unit 2

Unit 3

Unit 4

Unit 5

Unit 6

Unit 7

Unit 8

Unit 9

Unit 10

🔙 to cover

Unit 1: Intro to Python

Data/ in /out

🔙to Divider

Python is a programming language that lets you work more quickly and integrate your system more effectively.

You can learn to user Python and see almost immediate gains in productivity and lower maintenance costs.

Variables are spaces in the memory that we say data variables need to be declared

There are 3 main types of data: Boolean which is T/F
Numbers: can use mathematical operations [Inter: whole number Float: Decimal Number]
String: Cannot be evaluated must be enclosed

Naming Conventions in Python
Variables should indicate their purpose
Use underscores to separate words
CamelCase
P   python is case sensitive
Can't start with a number
Keywords like : print, False, True, else, for, while
Variables cannot have key words

In/Out Operations:

String concatenation: To provide more useful output, text is typically combined with information stored in variables through a + sign

Name a variable then use input keyword

Data Type: All values in a programming language have a "type" such as a number, boolean or string that dictates how the computer will interpret it. For example: 7+5 ids interpreted as "7" +"5"
Expression: Any valid unit of code that resolves to a value
Variable: A placeholder for a piece of information that can change

Input & variables

🔙to Divider

1.1) Comp sci is the study of the principles and use of computers
Coding: Using a formal language in order to create software or make programs
The principles include: Data, The internet, Algorithms, Logical Thinking, Ethics & Global impact
Formal Language: designed for a specific reason, follows a strict set of rules & Python is a type of formal language
Natural language: developed over time by people communicating with each other Like english

Formal Language) Very Clear, Consistent-works every time & Literal
Natural language) Ambiguity, Not always literal-metaphors
Syntax rules have 2 types - tokens and structure
Parsing is the structure of the sentence
Semantics is understanding the meaning

The meaning of a computer program is unambiguous and literal and can be understood entirely by analysis of the tokens and structure

1.2
A compiler is a program that takes commands and converts them so that a computer can understand them and execute them. We will do our coding here
The formal name of our compiler is IDE- Integrated Development Environment. It is software or an application that combines multiple tools in one window

1.3
"Print" is how we take information and display it on the screen-also known as output
We use the keyword "print" with a set of parentheses. Anything inside the parentheses will be output to the screen.

1.4
Hardware: The physical machine or Anything you can touch such as a monitor, a keyboard, a mouse

Software: The programs that run on our hardware. These are what computer scientists make: our job is to code and create new software for the hardware to run. Ex: Microsoft Word, Web browsers like Chrome or Safari, Cell phone or tablet apps

Software(Programs): The actual instructions that a computer follows; Written in code
A program is a collection of code that is giving the hardware a task
Computer Hardware: There are many types of computing devices but they all have five things in common.:

- At least one input device
- At least one output device
- A CPU
- Main Memory
- Secondary Memory

CPU=Central Processing Unit, Runs programs by going through the program's instructions, stored in binary
Memory- 2 types [main/RAM and secondary]

- Main memory
  - Also known as **RAM**
  - Short term - is erased when the power turns off
  - Very fast, but expensive
- Secondary memory
  - Long term - stays even after the power turns off
  - Examples: HDD, SSD, flash drives
  - Slower, but relatively cheap

1.5
The computer takes the words in quotes on the first line and stores it in the variable name. The + sign concatenates two strings next to each other "\n" works the same as a return or a enter key and lets you move to the next line when you use them in between words that are in quotations. It is known as an escape sequence. The \ is a escape character that marks the end of a sentence and n means that the computer should move on to a new line. We use escape characters sequences to insert characters in our output which have other meanings and functions in python.

- \n bumps us down a line
- \" prints quotation marks to the screen
- \t inserts a tab in order to line text up on the screen neatly
- \' prints the apostrophe mark
- \\ prints the backslash itself

ASCII stands for the American Standard Code for Information Interchange, which assigns alphanumeric values to keyboard characters.

If we don't want to include the line break, by using , end="" the line break is replaced with whatever is in quotation marks, which in this case is nothing. So it literally tells the computer how to "end" the line.

1.7
Variable is a name for a spot in computer's memory
Different types of data take up different amounts of space in computer's memory

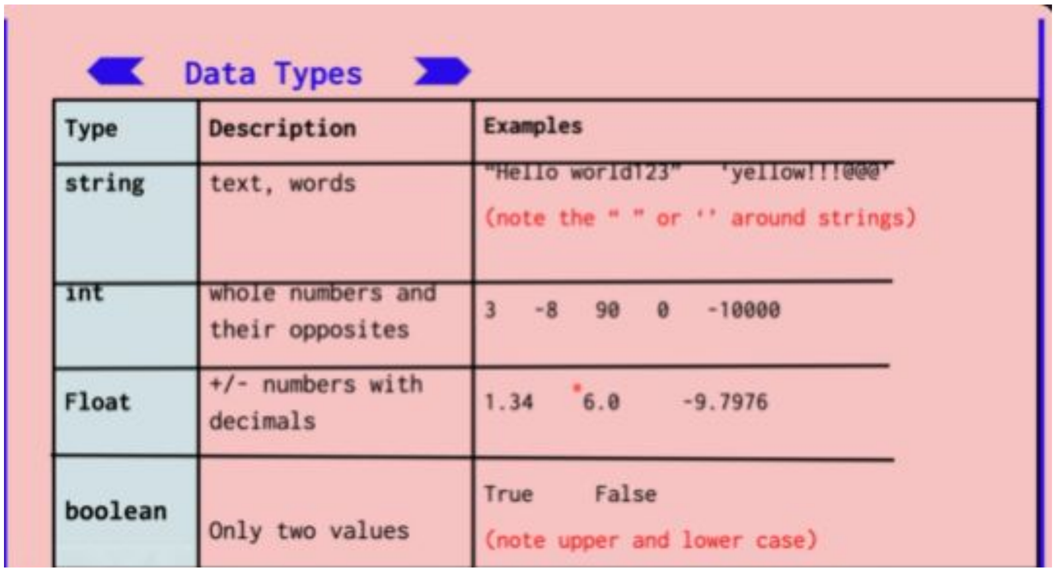
- Integers
  -whole numbers
  -takes up less space in memory
- Floats
  -decimal numbers

-take up more space because they include numbers before and after the decimal point

- strings
  -any combination of letters, numbers and symbols in quotations

Typecasting: The process of converting the value of one data type to another data type. Use the command int()

The str() will take what is inside the parentheses and make that data type a string. We must do this any time we are trying to concatenate words (which are a type of string) with something that is not a string (like the int in our example).
Binary- a number system based on two
Decimal- A number system based on ten, the number system we basically use


Data Types

Type	Description	Examples
string	text, words	"Hello world123" 'yellow!!!@@@' (note the " " or '' around strings)
int	whole numbers and their opposites	3 -8 90 0 -10000
Float	+/- numbers with decimals	1.34 6.0 -9.7976
boolean	Only two values	True False (note upper and lower case)

UNit 1 Review

🔙 to divider

TypeError when you try to concatenate integers int() and a string str() objects

NameError: name ‘c’ is not defined

SyntaxError: EOL (end of line) while scanning string literal

Type Coercion:

Ex: print(“hi” + 5)

Fix:

string	str()	str(5) = “5” str(“hey”) = “hey”
integer	int()	int(“4”) =4 int(“hey”) = Error

🔙 to cover

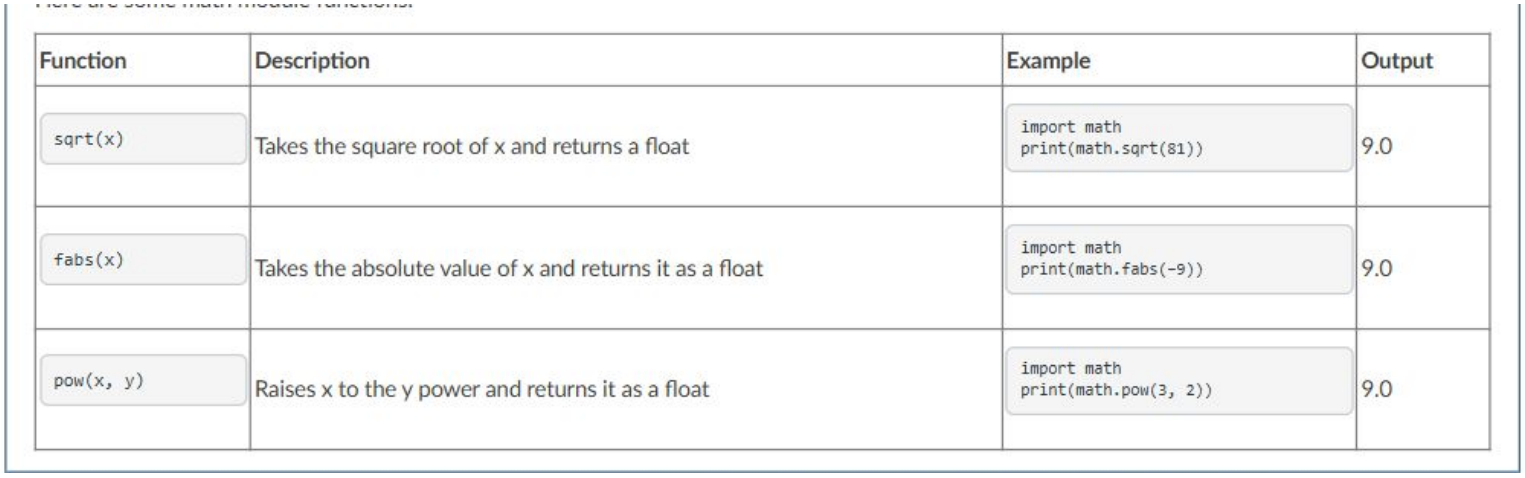
Unit 2: Number Calculations & Data

Parenthesis	(….)	Exponents	**
Multiplication	*	Division	/
Addition	+	Subtraction	-

1 import math

Square Root	math.sqrt(x)	Trig	math.sin(x)
Abs Value	fabs(x)	Degrees	math.degrees()
Log	math.log(x,base)	Pi	math.pi

For more than one line of comment, use ‘’’ ‘’’
While for one line comment use #



Function	Description	Example	Output
`sqrt(x)`	Takes the square root of x and returns a float	`import math` `print(math.sqrt(81))`	9.0
`fabs(x)`	Takes the absolute value of x and returns it as a float	`import math` `print(math.fabs(-9))`	9.0
`pow(x, y)`	Raises x to the y power and returns it as a float	`import math` `print(math.pow(3, 2))`	9.0

Unit 2 Review

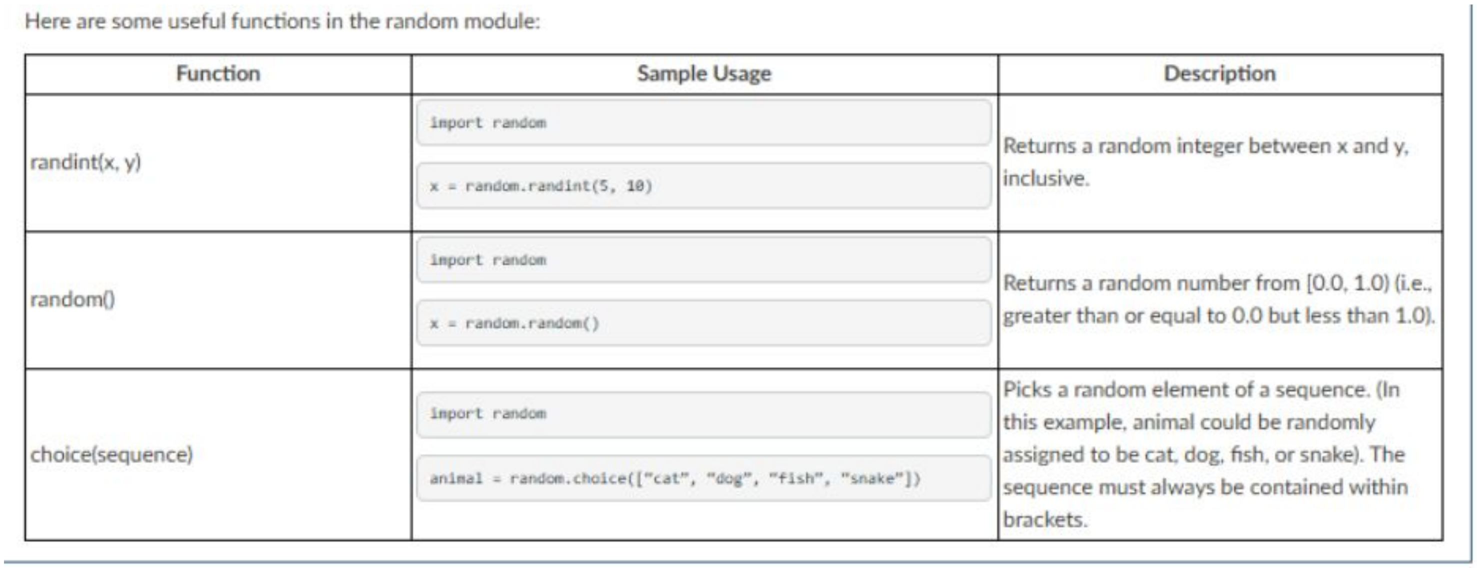
🔙 to divider

2.1)
-Charles Babbage invented the first mechanical computer
-Mauchly and Eckert built the first electronic computer (the ENIAC)
-Alan Turing created the idea of a general Purpose Computing Machine
-Konrad Zuse created the earliest programmed machine.
-In 1948, the first computer ran a program from memory

2.2)
-Python also has the // operator for performing integer division. The // operator rounds down the division to the nearest integer. It also returns an int where as the / operator returns a float
-Symbols that store values are called variables
-divide any number by 0 gets error
Exponentiation has precedence over multiplication, but its precedence goes from right to left! So 2 ** 3 is 8, 2 ** 8 is 256 and 256 * 3 is 768.

Modulo Operator- '%'
Modular division is different from real division. Modular division tells you the remainder after the division of two numbers.

We must **import** the module first to access the functions, and then specify the module when we use them. In addition, we should provide any information the function may need in the parentheses when we call functions.



Here are some useful functions in the random module:

Function	Sample Usage	Description

randint(x, y)	`import random`
	`x = random.randint(5, 10)`
		Returns a random integer between x and y, inclusive.

random()	`import random`
	`x = random.random()`
		Returns a random number from [0.0, 1.0) (i.e., greater than or equal to 0.0 but less than 1.0).

choice(sequence)	`import random`
	`animal = random.choice(["cat", "dog", "fish", "snake"])`
		Picks a random element of a sequence. (In this example, animal could be randomly assigned to be cat, dog, fish, or snake). The sequence must always be contained within brackets.

2.6)
Big data are sets of large data that are so big or complex that cannot be expressed in megabytes and Gigabytes. A megabyte is 1 million bytes.
Real big data is measured in the terabytes to petabytes. 8 bits in a byte. 1,000 bytes= 1 kilobyte[half a page of text]
1,000 kilobytes= 1 megabyte [a 500 pg- book][It takes a few megabytes to download a song]
1,000 megabytes= 1 gigabyte, or 1,000 thick books [An hd movie might be 2-4 gigabytes in size]
1,000 gigabytes= 1 terabyte, or 1 million thick books or 500 feature length films
1,000 terabytes= 1 petabyte [It would take 13 yrs of movie watching to get thru 1 petabyte of data]

Unit 2 Review

How is big data used:

- Autonomous vehicles/self-driving cars collect big data in order to drive from car censors like cameras, radars. With all these information, the car uses its programming to drive
- Personalized marketing such as is where people receive advertisement specifically chosen for them including shopping trends and patterns through algorithm

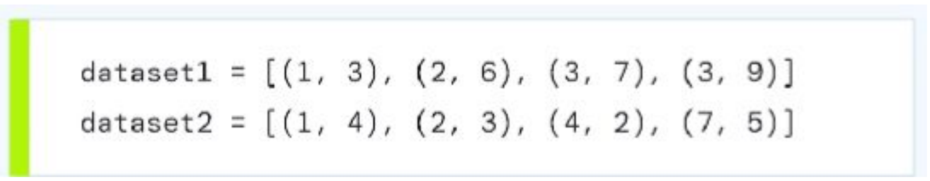
Internet of things is the network of all devices connected to the internet. These are connected for the purpose of connecting and exchanging data.

Big Data	Data set too large and complex for traditional ways of processing
Terabyte	2^40, or one million million (10^12) bytes
Petabyte	1,024 terabytes
Internet of Things	Ever-increasing number of devices that are connected to the internet

min() and max() helps us find the minimum and maximum of a set of integers, floats or even strings based on alphabetical order

To plot points on a graph:

- Start with “import simpleplot”
- Then we create two variables

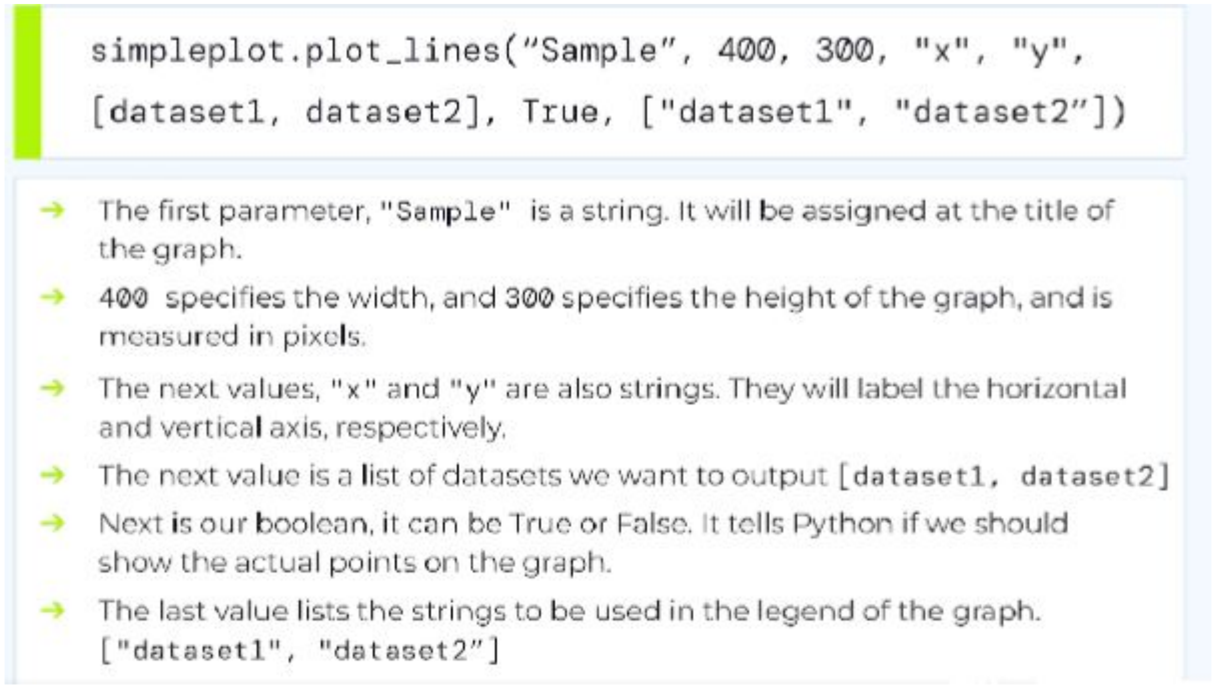


```
dataset1 = [(1, 3), (2, 6), (3, 7), (3, 9)]
dataset2 = [(1, 4), (2, 3), (4, 2), (7, 5)]
```

[These are lists. List Data Type can

Hold more than one value at a time

Unit 2 Review



```
simpleplot.plot_lines("Sample", 400, 300, "x", "y",
[dataset1, dataset2], True, ["dataset1", "dataset2"])
```

- The first parameter, `"Sample"` is a string. It will be assigned at the title of the graph.
- `400` specifies the width, and `300` specifies the height of the graph, and is measured in pixels.
- The next values, `"x"` and `"y"` are also strings. They will label the horizontal and vertical axis, respectively.
- The next value is a list of datasets we want to output `[dataset1, dataset2]`
- Next is our boolean, it can be True or False. It tells Python if we should show the actual points on the graph.
- The last value lists the strings to be used in the legend of the graph. `["dataset1", "dataset2"]`

max()	Finds the largest in a set of values
min()	Finds the smallest in a set of values
simpleplot	Allows us to use functions in order to create graphs with python
plot_lines()	Takes information from a data set and plots it on a graph

🔙 to cover

Unit 3: Making Decisions

Date: 10/08/24

🔙 to divider

Note to Self

1) I learned to create and assign variables in Unit 1 and print strings or numbers that carried to Unit 2
2) In unit2 I learned numbers can be written as integers or floats, concatenate strings and numbers and mathematical operations
3) Questions that are still left unanswered from Unit2 are how to use one print option that will read different expressions, evaluate if they are true or false and vary it's results accordingly
4) I should practice more to use mathematical operations or math.import to solve complex problems like quadratic equations, etc.

Overview of Function Syntax

Function name

() with parameters inside

```
def mult_function(x,y):
z= x*y
print("I heart math")
return z
```

colon

Z is a local variable

Indent

return value(optional)

Return can only be put at the end

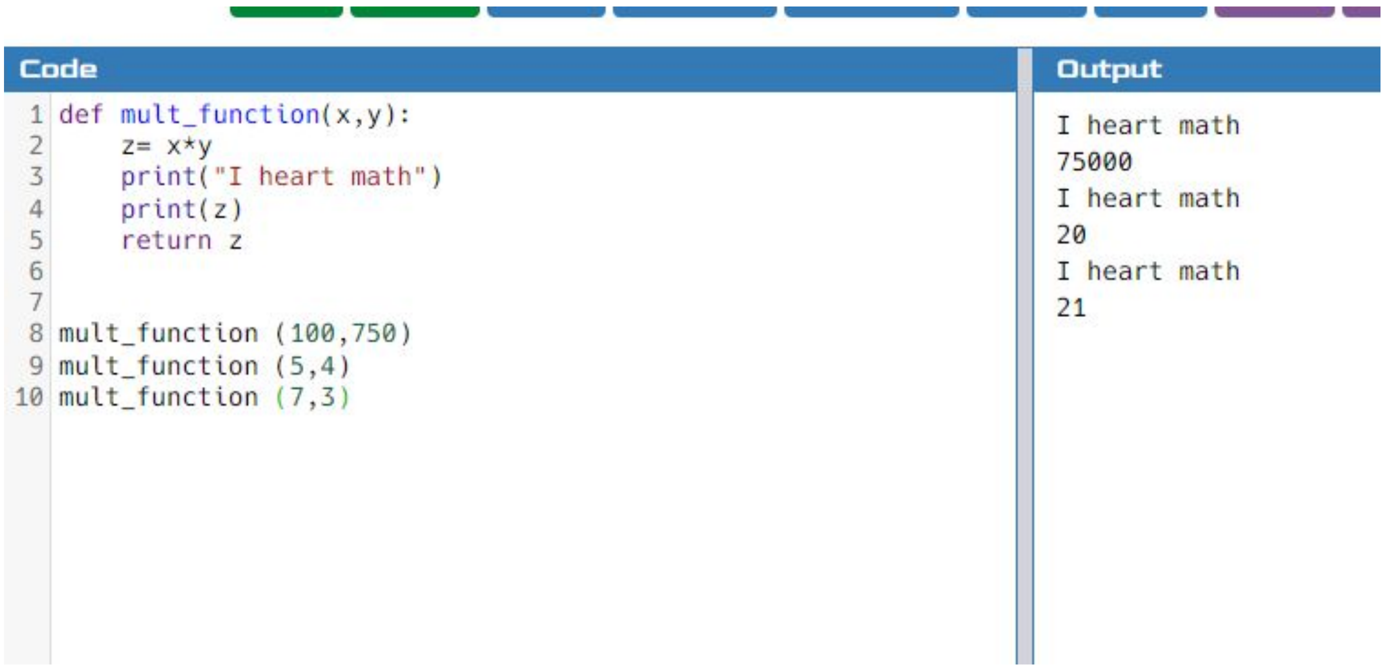
```
mult_function (5,4)
```

() with arguments inside

Function "def"-inition must be FIRST!!!

Unit 3 Review

🔙 to divider



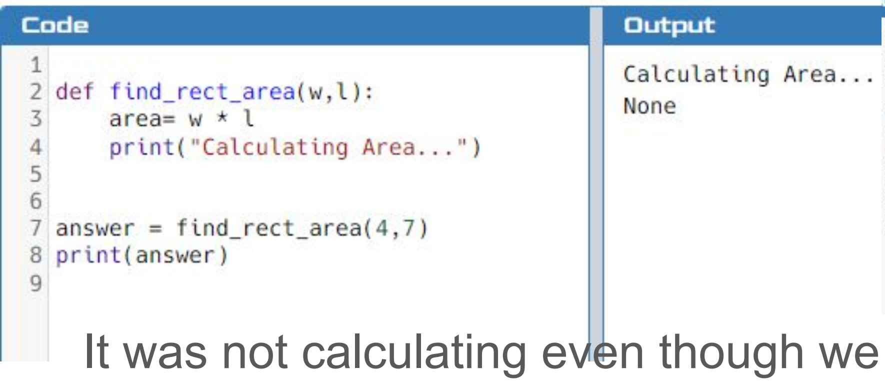
```
def mult_function(x,y):
    z= x*y
    print("I heart math")
    print(z)
    return z


mult_function (100,750)
mult_function (5,4)
mult_function (7,3)
```

```
I heart math
75000
I heart math
20
I heart math
21
```

What is a function?
Examples of built-in functions abs, print:
x= abs(-5) //returns the absolute value of 5
print(x) // prints the value of x

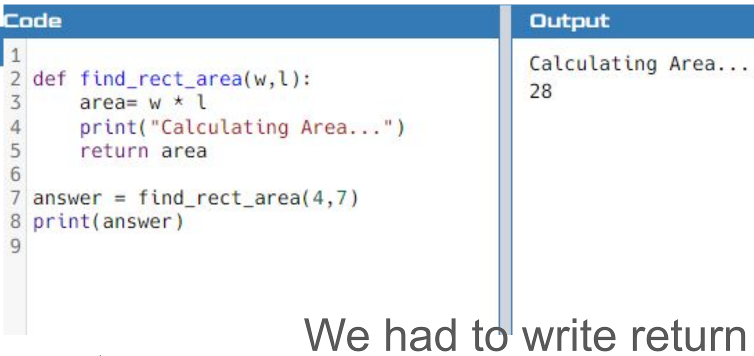
- May or may not have inputs (called "arguments")
- May or may not have outputs ("returns a value")

Functions are useful for efficiency
When you put the function with a variable inside and use parentheses, that's a parameter



```

def find_rect_area(w,l):
    area= w * l
    print("Calculating Area...")


answer = find_rect_area(4,7)
print(answer)

```

```
Calculating Area...
None
```



```

def find_rect_area(w,l):
    area= w * l
    print("Calculating Area...")
    return area

answer = find_rect_area(4,7)
print(answer)

```

```
Calculating Area...
28
```

It was not calculating even though we wrote area variable inside the function

We had to write return area to make the code remember to run it

Unit 3 Review

Function	A named group of code that is designed to do a specific task. They can be built-in to Python, or created by us
Abstraction	The process of reducing complexity by hiding unnecessary information, and focusing on what is needed to make a function work.
Parameters	Extra information sent in between the parentheses of a function

<u>If statement</u>- A structure used to determine if a statement is true or false

If the condition is true, the block of code is executed

If the condition is false, the block of code is skipped

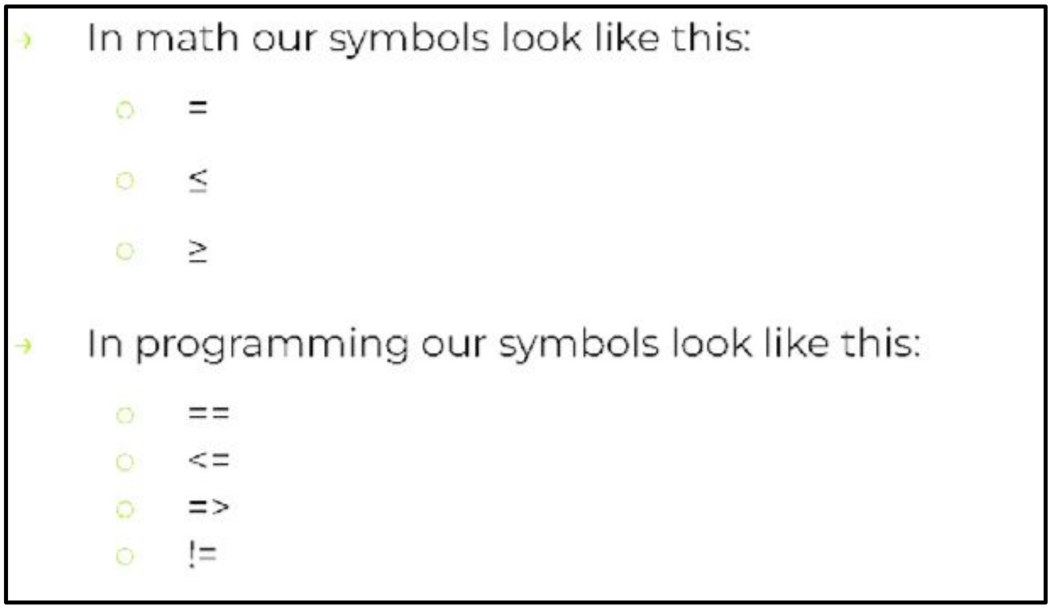
If condition :

Code that it will execute if it's true

<u>Relational Operators:</u> Used to compare how to values relate to each other

- The = is used as an assignment operator, it assigns a value on the right to a value on the left
- The == represents equal to. There is no assignment goin on, only comparison

Unit 3 Review



- In math our symbols look like this:
  - =
  - ≤
  - ≥
- In programming our symbols look like this:
  - ==
  - <=
  - =>
  - !=

If statement	A structure used to determine if a condition is true or false. If true, it executes a block of code that condition evaluates. If false, it skips the section of code.
Rational Operator	Used to compare how two values relate to each other. The result is either true or false.

In order to check for more than one condition at a time, we need to use a <u>logical operator</u>.

A <u>logical operator</u> is a way to connect multiple Boolean expressions

We will look at the key words <u>and</u> , <u>or</u>

<u>And</u> - Checks that BOTH expressions are true, and if they are, the overall set of conditions is true.

Or- Checks if one or the other expression is true. As long as one of them is true, the overall set of conditions is true

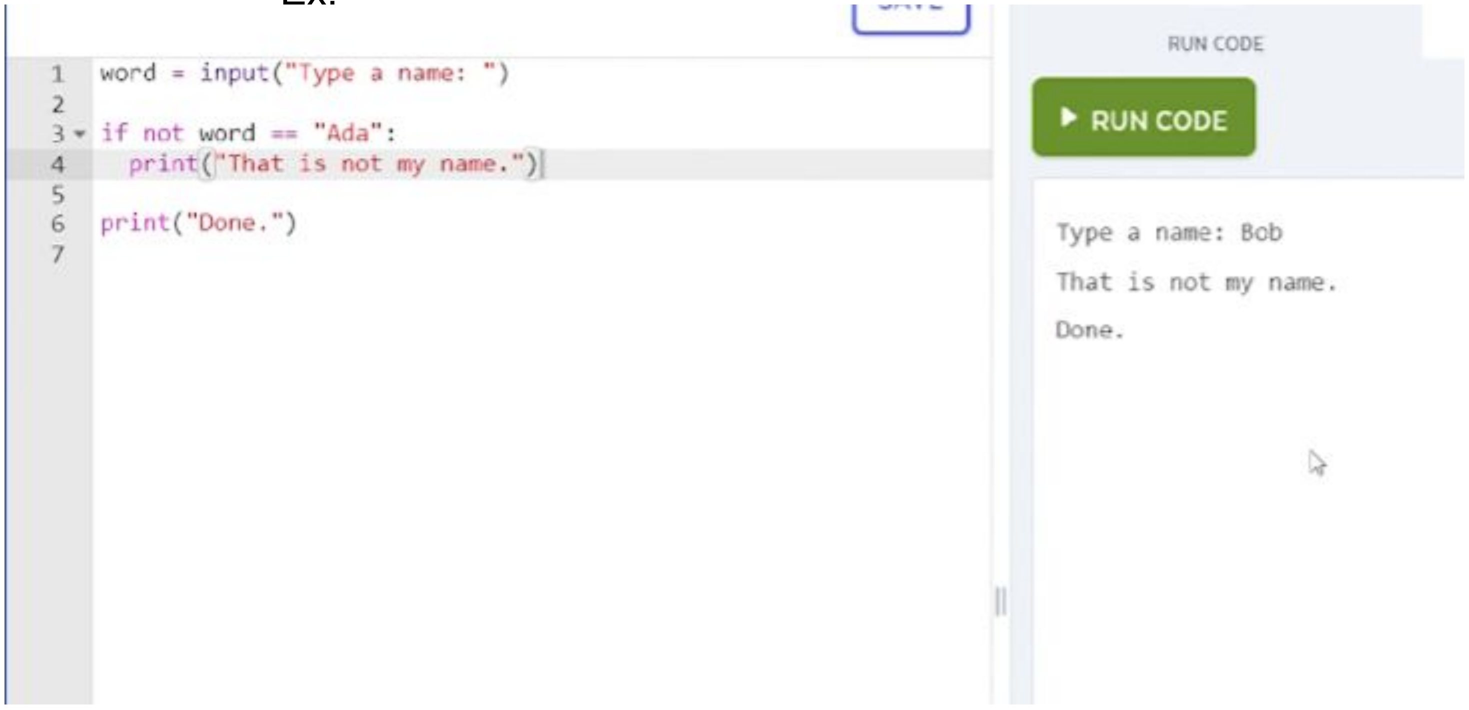
Unit 3 Review

Not operator= Takes the opposite of what the condition evaluated to

If a condition was false, not would make it true.

If it were true, not would make it false.

Ex:

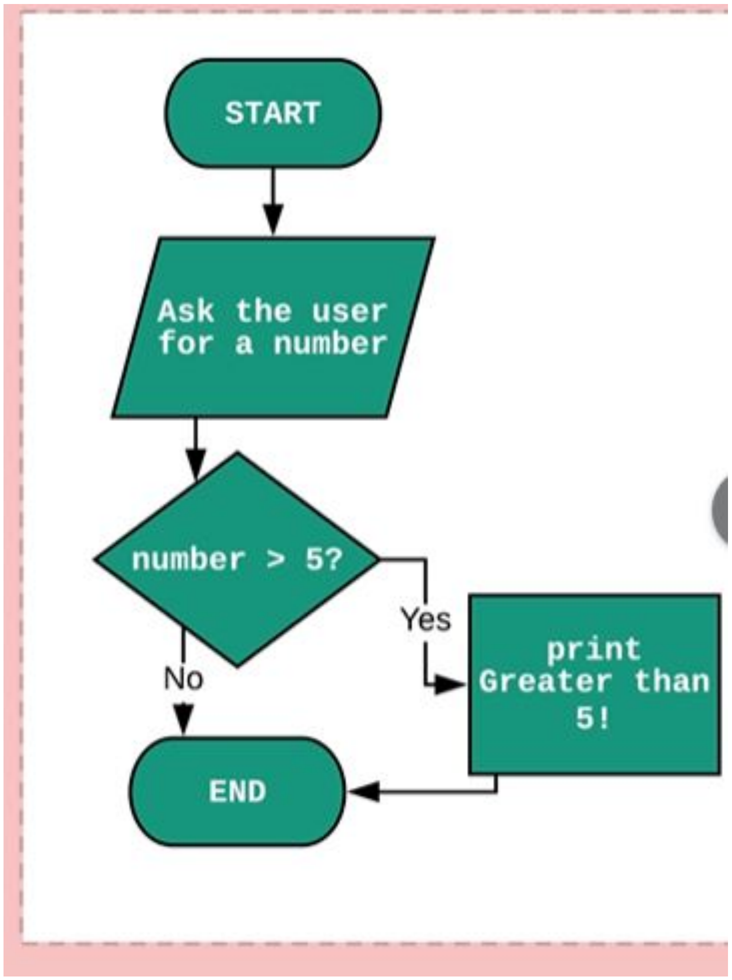


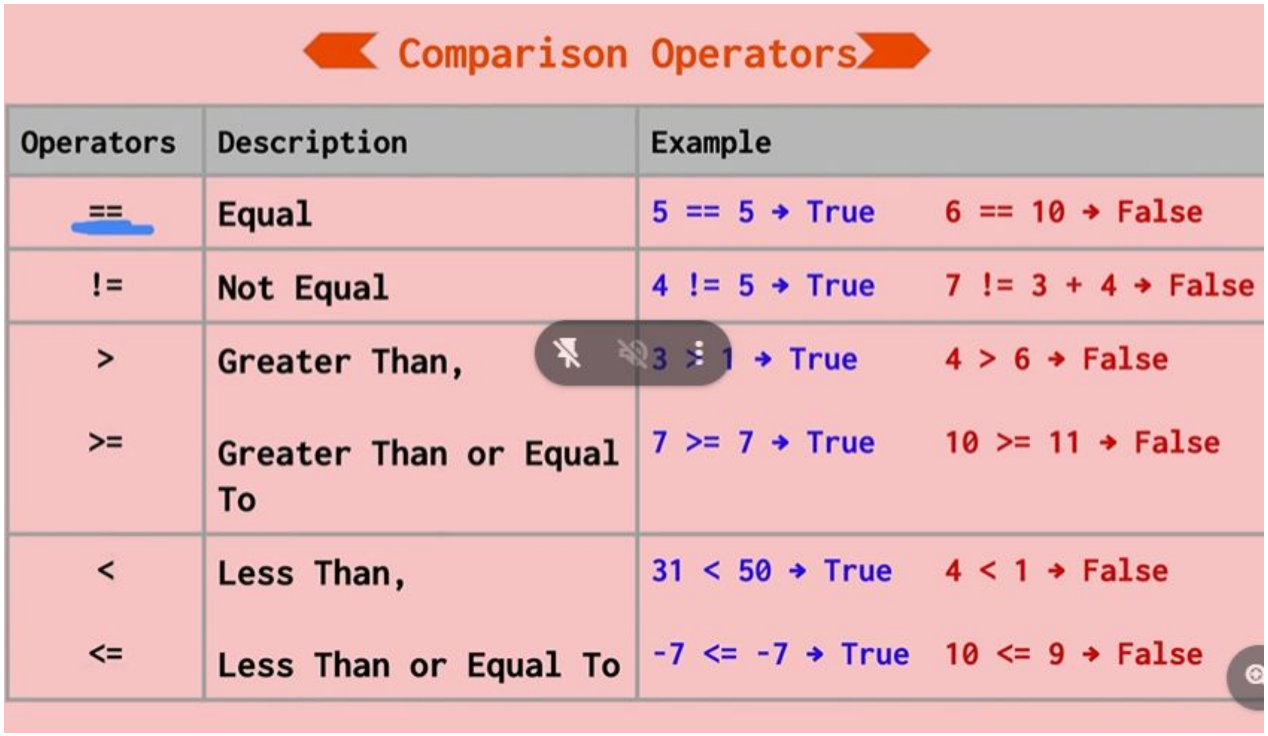
```
word = input("Type a name: ")

if not word == "Ada":
  print("That is not my name.")

print("Done.")
```

```
Type a name: Bob
That is not my name.
Done.
```

Date: 10/10/24

Conditionals & Flow Charts



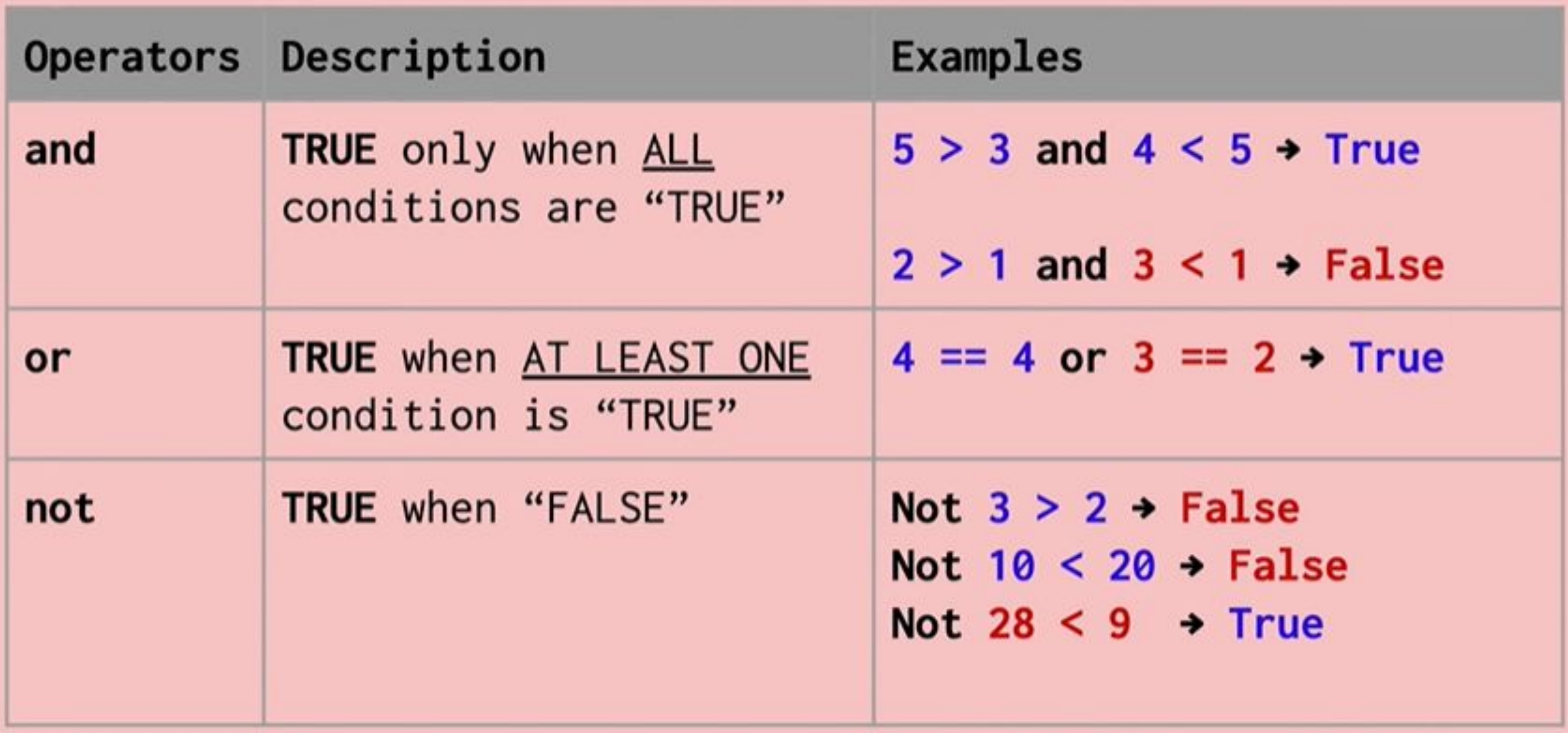


Comparison Operators

Operators	Description	Example	
==	Equal	5 == 5 → True	6 == 10 → False
!=	Not Equal	4 != 5 → True	7 != 3 + 4 → False
>	Greater Than,	3 > 1 → True	4 > 6 → False
>=	Greater Than or Equal To	7 >= 7 → True	10 >= 11 → False
<	Less Than,	31 < 50 → True	4 < 1 → False
<=	Less Than or Equal To	-7 <= -7 → True	10 <= 9 → False

Date: 10/10/24

More than One Condition



Operators	Description	Examples
and	TRUE only when ALL conditions are "TRUE"	5 > 3 and 4 < 5 → True 2 > 1 and 3 < 1 → False
or	TRUE when AT LEAST ONE condition is "TRUE"	4 == 4 or 3 == 2 → True
not	TRUE when "FALSE"	Not 3 > 2 → False Not 10 < 20 → False Not 28 < 9 → True

if

elif

else

Statements let you create conditionals based on the conditions that need to fulfilled for the action/activity to happen

Date: 10/17/24

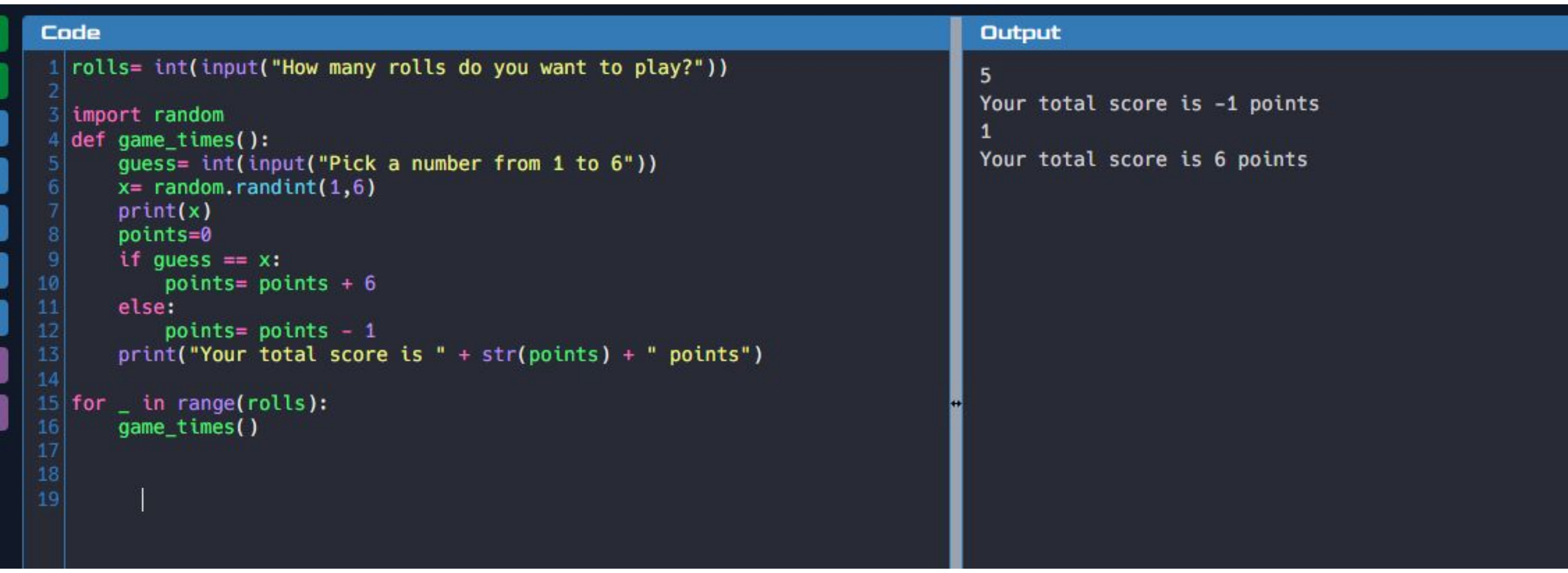
Class Notes:

Start or End ——— ? ——— Action

Start ——— age = input ——— age>=16 ——— out ”yes”

No

Output “No” ——— Elif



```
rolls= int(input("How many rolls do you want to play?"))

import random
def game_times():
    guess= int(input("Pick a number from 1 to 6"))
    x= random.randint(1,6)
    print(x)
    points=0
    if guess == x:
        points= points + 6
    else:
        points= points - 1
    print("Your total score is " + str(points) + " points")

for _ in range(rolls):
    game_times()
```

```
5
Your total score is -1 points
1
Your total score is 6 points
```

This is CFU #9 where we have to ask the user how many times they want to roll the dice, and each time they roll the dice they have to guess a number from 1-6, and if that is the same number that the computer generates, they gain 6 points and for every wrong guess, they lose 1 point.

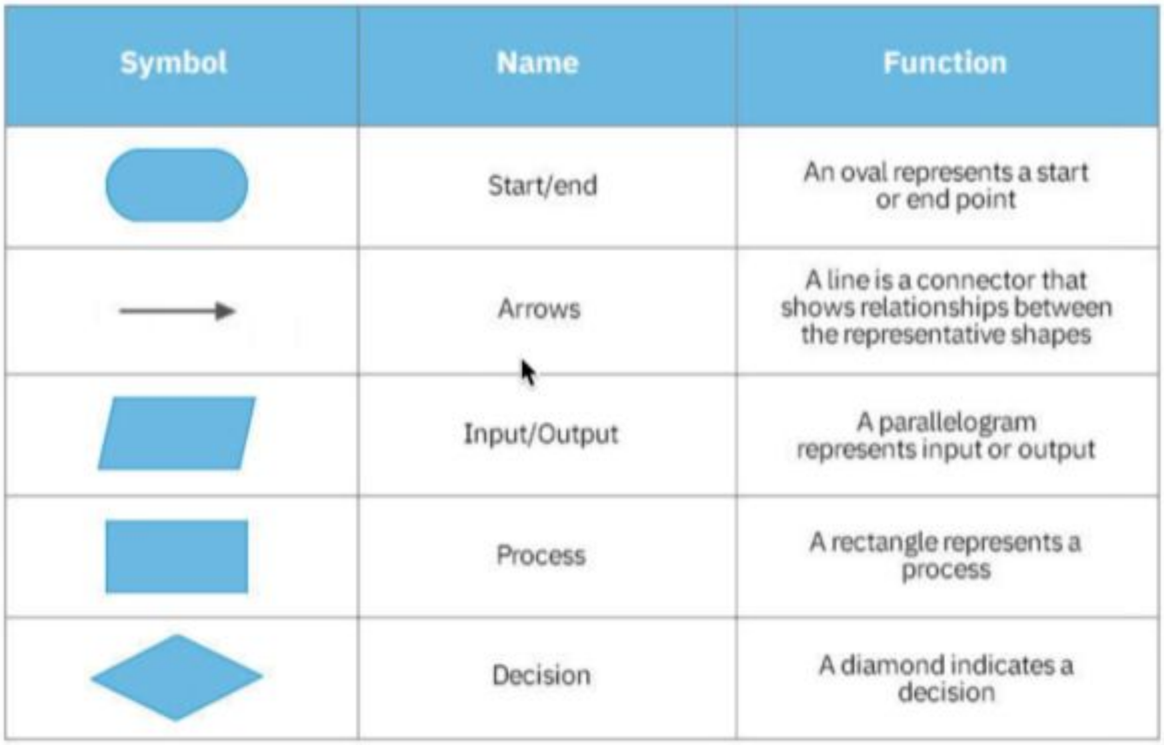
I’m struggling to write the code so that the program prints the user’s total score after all the rolls are played. I tried making points a global variable but that means whenever I use points to keep track of the score inside the function game_times(), it doesn’t calculate that score at the end when I try to print scores outside of the function. Therefore, the best I could do is make points a local variable inside the function

Date: 10/17/24

Which means after every rolling the dice, it prints out the total score, which is not entirely correct. Also, I don't know how to call a function the same amount of times as another variable. So I googled that and wrote the section of the code

"for _ in range(rolls)" which calls the function, "rolls" number of times

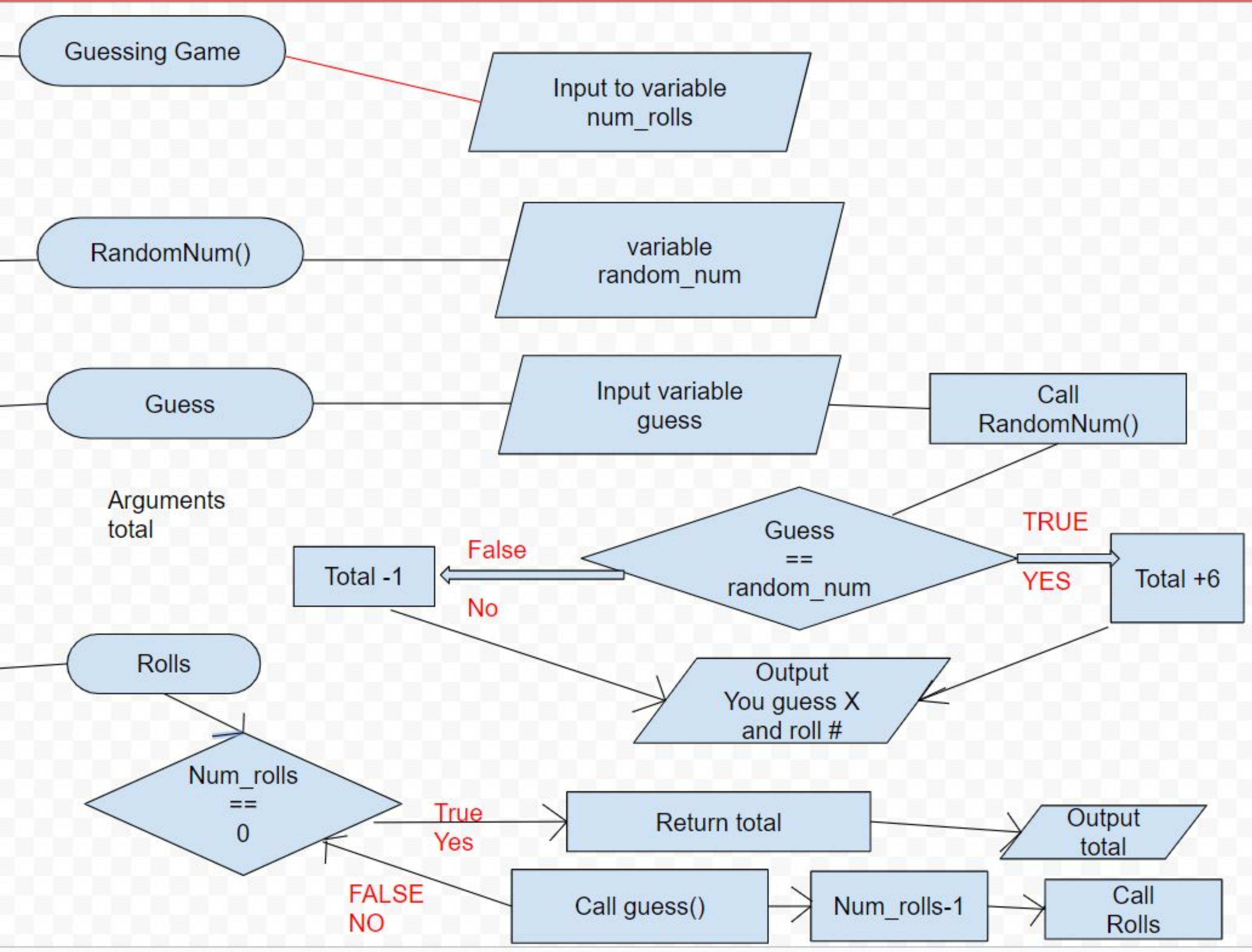
Date: 10/18/24



Symbol	Name	Function
⬭	Start/end	An oval represents a start or end point
→	Arrows	A line is a connector that shows relationships between the representative shapes
▱	Input/Output	A parallelogram represents input or output
▭	Process	A rectangle represents a process
◆	Decision	A diamond indicates a decision

Date: 10/18/24

Flowchart



Date: 10/22/24

Algorithm is a process or set of instructions for a computer to follow to complete a desired task.

Algorithms must have: -have an order -have clear instructions -stop in a finite amount of time -produce a result/solution -operations that can be done by our computer more efficiently than by a human

We analyze algorithms to predict performance and help choose which algorithm is best to use.