
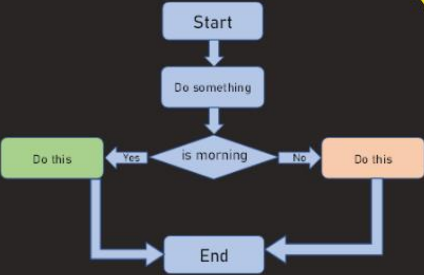


language = "Python"

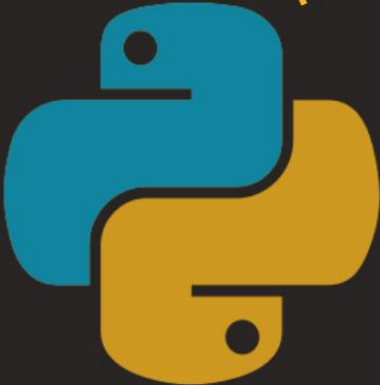
```
print(" I will be a good programmer!")
```



Learn programming with python



while alive:
keepLearnning()



if mygrade < 75 : study_harder()


if not exams :
enjoy()
else :
study()

variables

: * + - / % **

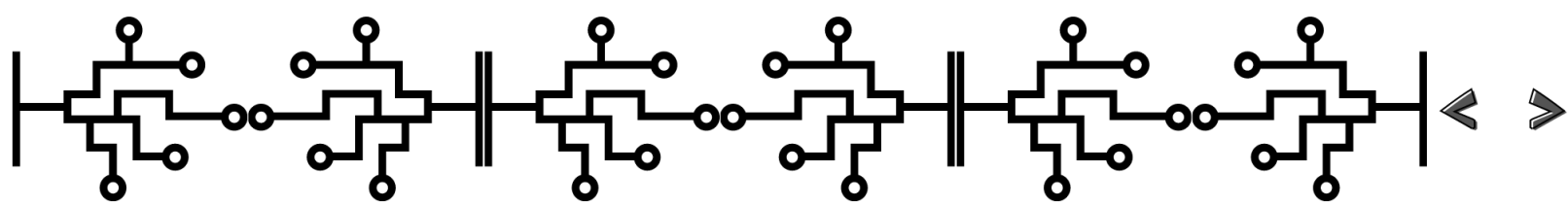
>= <= == != and or not

Algorithms



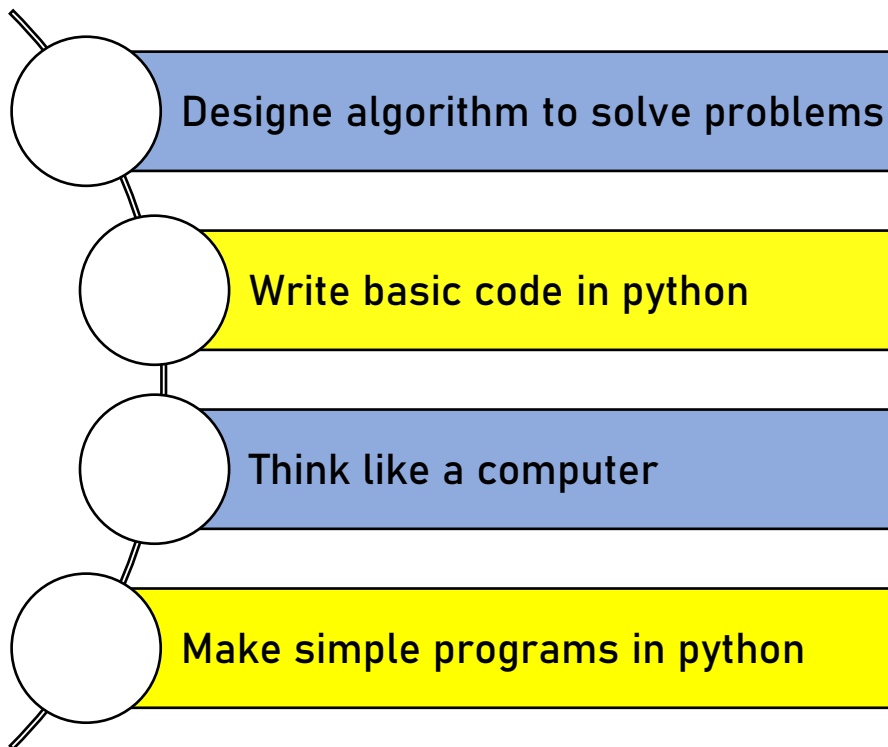
debugging

SALAH ALKMALI



Learn programming with python

At the end of this course, you will be able to:



To be prepared for the future. You must learn the basics of computer programming, no matter what field of work you want to pursue.

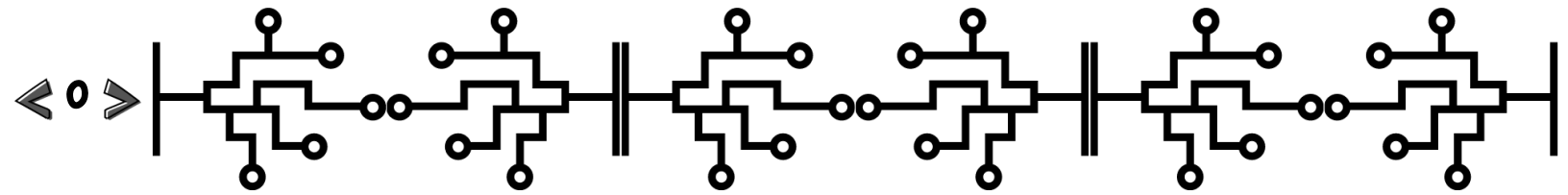


TABLE OF CONTENT

[STARTING WITH CODING](#) ***** 1

❖ [What is coding?](#) ***** 1

❖ [Programming Language](#) ***** 1

[ALGORITHM](#) ***** 3

[MEET PYTHON](#) ***** 5

❖ [Installing python](#) ***** 6

❖ [Using idle](#) ***** 7

[YOUR FIRST PYTHON PROGRAM](#) ***** 9

[COMMENTS](#) ***** 11

[MATH OPERATIONS IN PYTHON](#) ***** 12

[VARIABLES](#)

❖ [Strings](#) ***** 16

❖ [Integers & float](#) ***** 19

❖ [Lists](#) ***** 20

[DECISIONS \(If Statement\)](#) ***** 22

[LOOPS](#)

❖ [For loop](#) ***** 26

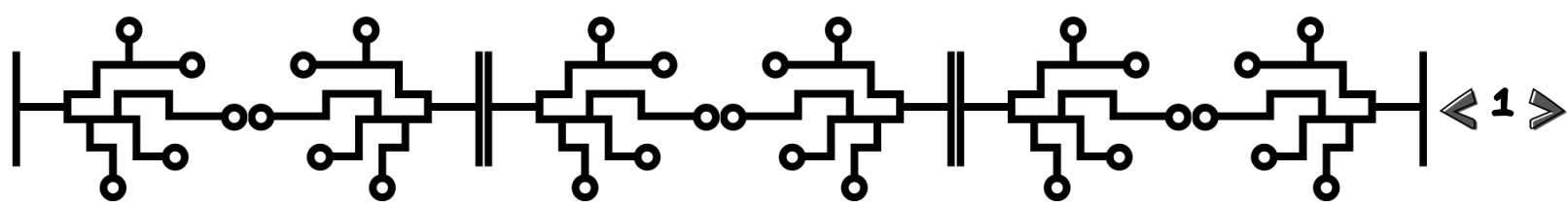
❖ [While loop](#) ***** 28

[Functions](#) ***** 32

[Projects](#) ***** 35

[References](#) ***** 35

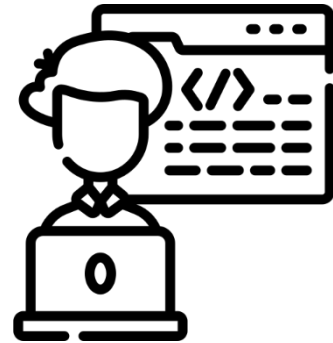
[Glossary](#) ***** 36



STARTING WITH CODING

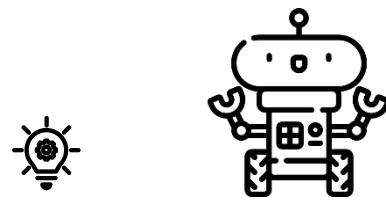
What is coding?

Computer programmers, or “coders”, are people who write step-by-step instructions that can make a computer perform a task. Coders can get computers to do sums, make music, move a robot across a room, or fly a rocket to Mars.



Dumb boxes

A computer can't do anything of its own accord – it just sits there like a dumb box until it's told exactly what to do. Because computers can't think for themselves and can only do as they're told, coders have to do the thinking for them and write their instructions carefully.



By learning how to code, you'll be able to write your own programs and make the computer do what you want. It's a bit like having an electronic pet that you can teach to perform tricks!

Programming languages

Is a type of written language that tells computers what to do. Examples are: Python, Ruby, Java, JavaScript, C, C++, and C#.

Programming languages are used to write all computer programs and computer software.

A computer program is a specific set of commands that will inform the computer that it must do something. The software in a system is a collection of multiple programs.

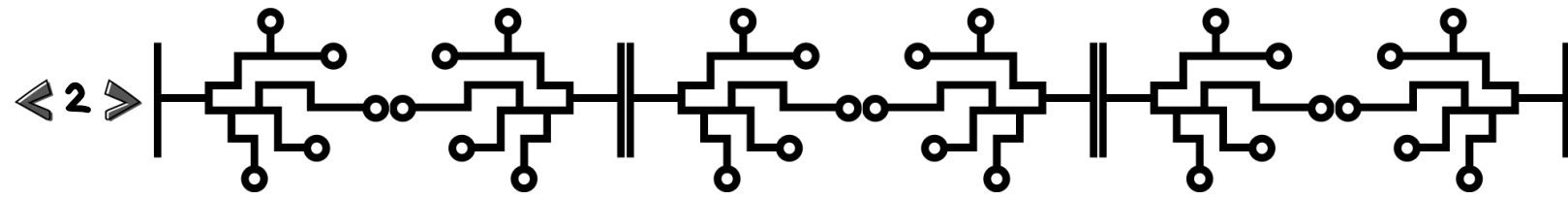
Question: Is the calculator considered to be a software or just a program?

Hello world Program in different languages!

Here an example of a program that prints “HELLO” in 2 different languages , JAVA & PYTHON

```
1 class HelloWorld {
2     public static void main(String[] args) {
3         System.out.println("Hello from java!");
4     }
5 }
```

```
1 print("Hello from Python")
```



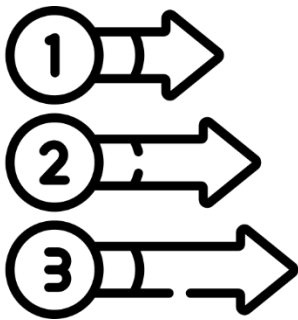
Anyone can code

To be a coder you just need to learn a few basic rules and commands, and then you can start writing programs to suit your skills and interests. If you're into science, for example, you could make an app that draws graphs from the results of your experiments. Or you could use your art skills to design an alien world for your own video game.



▽ Think logically

Coders need to think logically and carefully to write good code. If the instructions aren't quite right or the steps are in the wrong order, a program won't work properly. Think through each step and make sure things happen in a logical order.



▽ Pay attention to detail

If you're good at spot-the-difference puzzles, you'll probably be a great coder. An important skill in coding is spotting mistakes in your code. These mistakes are called bugs, and even tiny bugs can cause big problems. Eagle-eyed coders can pick out spelling mistakes and faults with the logic or order of the instructions. Debugging a program can be tricky, but learning from your mistakes is a great way to improve your coding powers.

Bugs

Bugs are errors in code that make programs behave in unexpected ways. They are so-called because early computers sometimes went wrong when insects got stuck in their circuits!



Homework

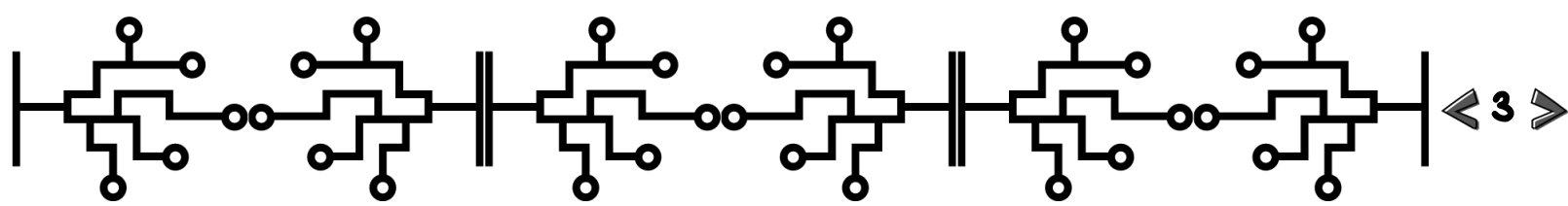
What is a programming language?

List 10 programming languages?

What is an error in a program called?

What is the difference between a program and a software?

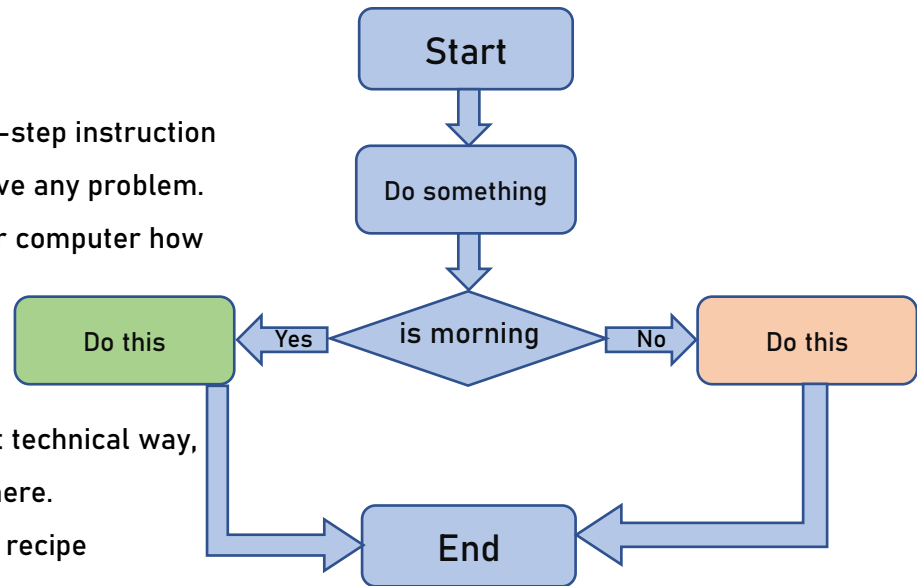
Name a software that can be used to write codes or programs?



ALGORITHM

What Is an Algorithm?

An algorithm is a basic formula or a step-by-step instruction that you can use to complete any task or solve any problem. A programmer will use it to tell a machine or computer how it should perform a specific task.



If you do not look at an algorithm in the most technical way, you will realize that it exists almost everywhere.

Say, your mother or grandmother may use a recipe to prepare a dish. This recipe is an example of an algorithm.

Alternatively, your teacher may have told you to use a specific method to solve a problem in arithmetic.

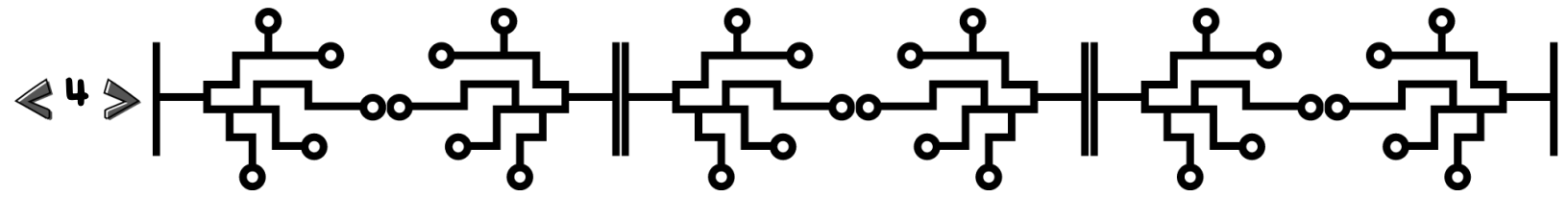
Did you know that your morning routine is also an algorithm?

Activity 1, take a piece of paper and list down what you do every morning.

You Can Write Your Own Algorithm

If you do not want to write down an algorithm about your entire morning routine, you can write about simpler tasks like eating cereal or brushing your teeth. You will soon learn about some important concepts of programming, such as **sequencing** (putting the cereal in a bowl and then pouring milk), **conditional logic** (do not eat if the bowl is empty), and **repetition** (brush the bottom row of teeth four times). If you want to become better at writing algorithms, you should try to add a few more challenges. A computer will never understand what your intentions are if you do not explicitly define something.

Activity 2, What is $345 \div 15$? Show your work.



Let's write some Algorithms.

Problem 1, Everyone can enter this class!

Write algorithm that allows only Grade 11 & Grade 12 to enter this classroom.

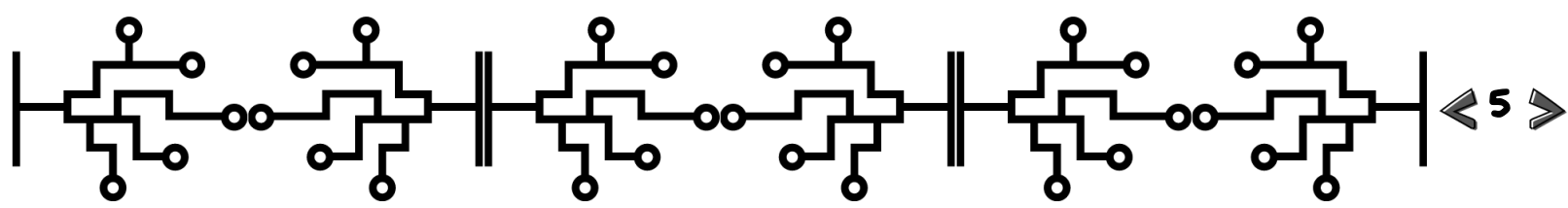
Problem 2, Write algorithm that allows students with specific id numbers to enter the classroom.

Problem 3, Write algorithm that allows student with his password to enter the classroom

Homework

What is an Algorithm?

Write algorithm that test if a given number is ODD or EVEN number.



MEET PYTHON

Python is one of the most popular computer programming languages in the world. It was first released in the 1990s and is now used to build millions of apps, games, and websites.

Why Python?

Python is a great language for getting started with computer programming. Many schools and universities use it to teach coding.

Here are some of the reasons that Python's so useful.

Python

Python programs usually have a name ending with “.py”, which makes them easy to recognize. When you save a program, Python automatically adds “.py” at the end, so you don't need to type it in.



△ Easy to read and write

Python is a text-based computer programming language. You write the instructions using a mixture of English words, punctuation characters, symbols, and numbers. This makes Python code simple to read, write, and understand.

△ Works everywhere

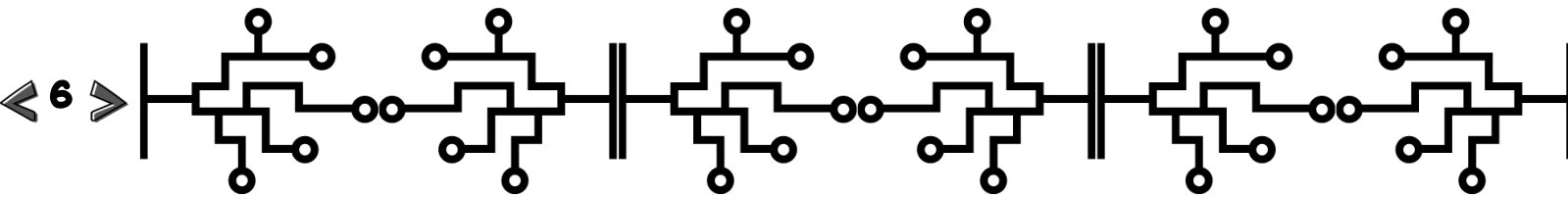
Python is portable. This means you can write and run Python code on lots of different computers. The same Python code will work on PCs, Macs, Linux machines, and Raspberry Pi computers. The programs behave the same way on each machine.

△ Handy tools

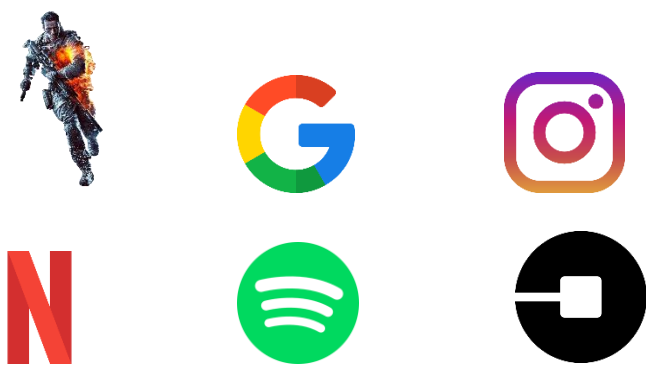
Python is packed with lots of useful tools and preprogrammed code that you can use in your programs. This is called the Standard Library. Using these tools makes it easier and quicker for you to build your own programs.

▷ Great support

Python has well-written documentation. It has a guide to getting started, a reference section for looking up what things mean, and a bunch of example code.



Examples for some companies that uses PYTHON:



The interpreter

Some programming languages use an interpreter. The interpreter is a program that can translate from one programming language into another. Every time you run a Python program, the interpreter translates each line of Python code into a special code that the computer can understand, known as machine code.

INSTALLING PYTHON

1

Go to the python website

Type the address below into your web browser to go to the Python website. Then click on "Downloads" to open the download page.

- <https://www.python.org/>

2

Download Python

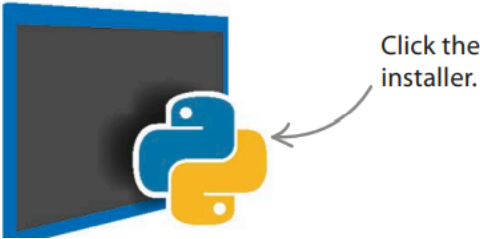
Click on the latest version of Python for Windows, beginning with the number 3. The installer file will download automatically. Of the different installer options, select "executable installer".

- Python 3.6.0a4 - 2016-08-15
 - Windows x86 executable installer
 - Windows x86-64 executable installer

3

Run the installer

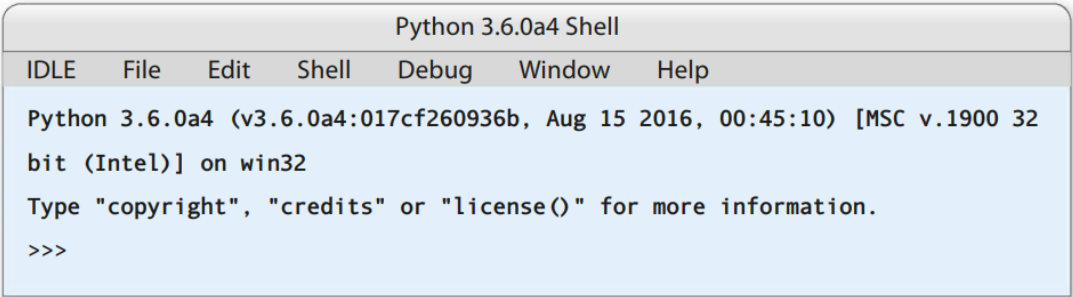
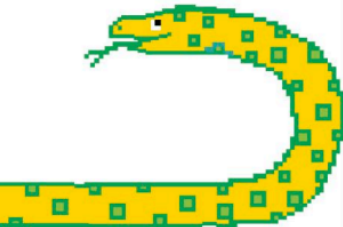
Double-click the installer file to install Python. Choose "install for all users" and click "next" at each prompt, without changing the default settings.

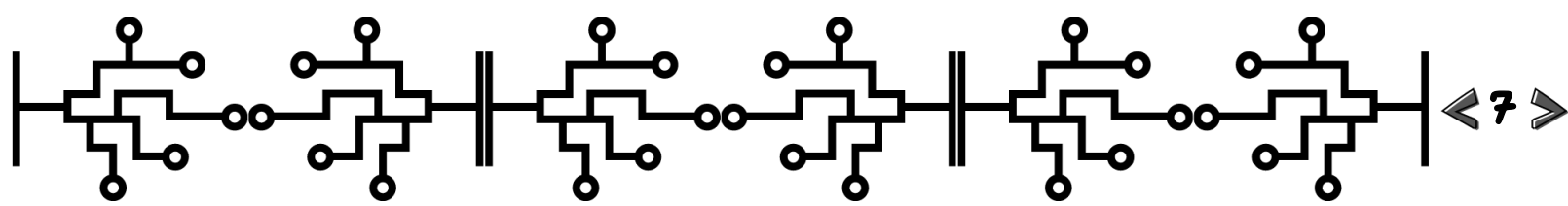


4

Open IDLE

When the installation is finished, check that it was successful by opening the IDLE program. Go to the "Start" menu, choose "All Apps", then select "IDLE". A window like the one below should open up.





USING IDLE

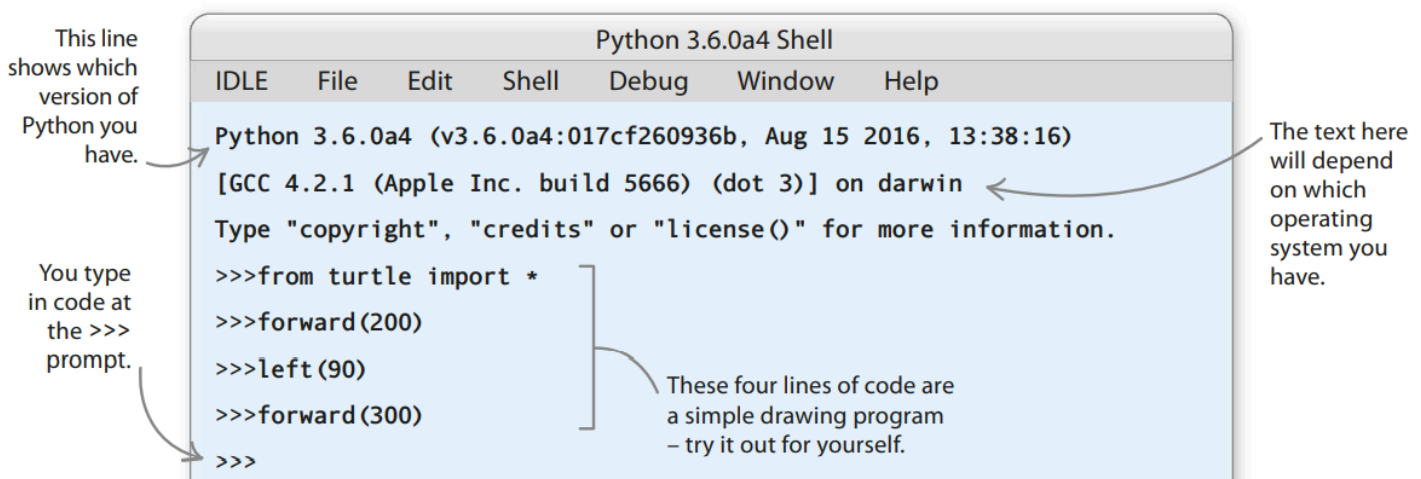
IDLE has two different windows in which you can work. The editor window can be used to write and save programs, while the shell window runs Python instructions immediately.

The shell window

When you open IDLE, the shell window pops up. This is the best place to get started in Python, as you don't have to create a new file first. You just type the code directly into the shell window.

▽ Working in the shell

The code you type can be run straight away, and any messages or "bugs" (errors) are displayed. You can use the shell window like a notepad, to test out snippets of code before you add them into a bigger program.



EXPERT TIPS

Different windows

To help you know which window you should type your code in, we've given each window in IDLE a different colour.

Shell window

Editor window

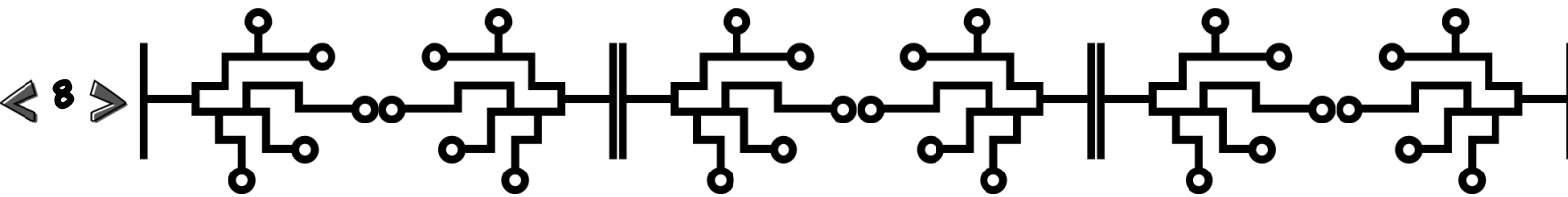
▽ Give the shell a test run

Type each of these code snippets into the shell window and press the enter/return key after each one. The first line displays a message and the second line does a calculation. Can you work out what the third line does?

```
>>> print('I am 10 years old')
```

```
>>> 123 + 456 * 7 / 8
```

```
>>> ''.join(reversed('Time to code'))
```



The editor window

The shell can't save your code, so when you close the shell window the code you typed is lost forever. That's why you should use IDLE's editor window when you work on a project. This window lets you save your code. It also has built-in tools to help you write your programs and to trouble-shoot any errors.

▽ The editor window

To open the editor window in IDLE, click on the File menu at the top and choose New File. An empty editor window will then appear. You'll use the editor window to write and run programs for the projects in this book.

You type the code in here. This program prints a list that tells you which numbers are even and which ones are odd.

Anything you tell Python to print gets displayed in the shell window.

The name of the file is shown here.

You can run your programs from this menu.

The menu bar for the editor window is different to the one for the shell.

EvensandOdds.py

IDLE File Edit Format Run Window Help

```
for counter in range(10):  
    if (counter % 2) == 0:  
        print(counter)  
        print('is even')  
    else:  
        print(counter)  
        print('is odd')
```

I love idling!

Colours in the code

IDLE automatically colours the text to highlight different parts of the code. The colours make it easier to understand the code, and they're useful when you're trying to spot mistakes.

Built-in commands
Python commands, such as "print", are shown in purple.

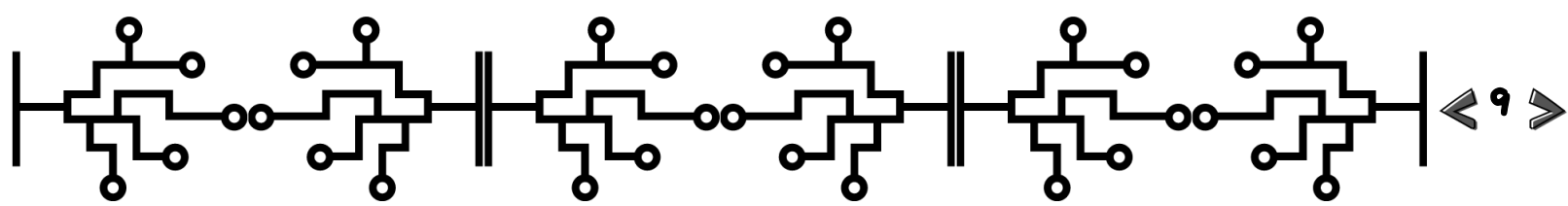
Symbols and names
Most code text is coloured black.

Output
Any text produced when a program runs is blue.

Errors
Python uses red to alert you to any errors in your code.

Keywords
Certain words, such as "if" and "else", are special words that Python uses. They are called keywords and are shown in orange.

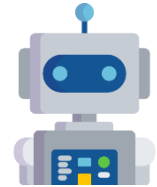
Text in quotes
Text in quote marks is green. A green bracket around text shows you're missing a quote mark.



YOUR FIRST PYTHON PROGRAM

Now that you've installed Python and IDLE, it's time to write your first program in Python. Follow these steps to create a simple program that greets the user with a cheery message.

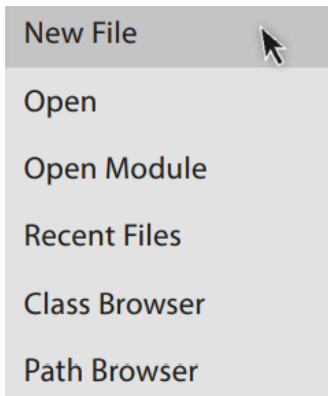
Hello , Salah



Activity 1, write a program that prints welcoming message to a given name.

1. Launch IDLE

Choose New File to create an empty editor window where you can write your program.



2. Type the first line

In the editor window, type this line of text. The word “print” is a Python instruction that tells the computer to display something on the screen, such as the words “Hello, World!”

```
print('Hello, World!')
```

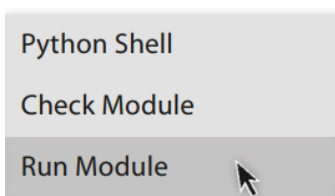
3. Save your file

Before you can run the code, you must save it. Go to the File menu and choose Save.



4. Check it works

Now run the first line of the program to see if it works. Open the Run menu and choose Run Module. You should see the message “Hello, World!” in the shell window.

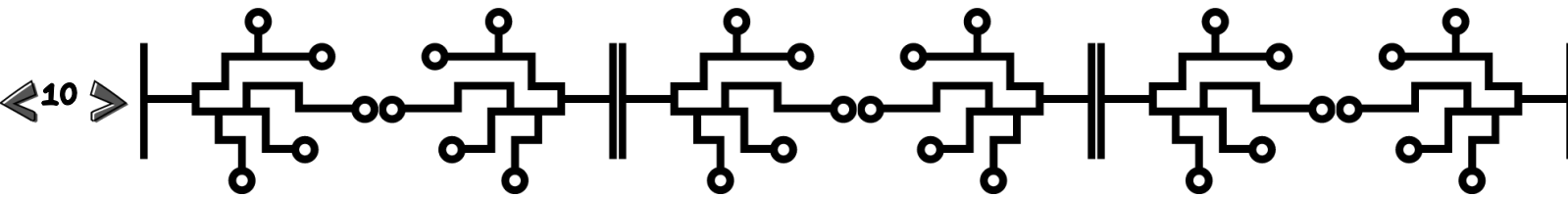


5. Add more lines

Go back to the editor window and add two more lines to your script. Now the middle line asks for your name and then stores it in a variable. The last line uses your name to print a new greeting. You can change it to a different greeting if you prefer – as polite or as rude as you like!

```
print('Hello, World!')
person = input('What's your name?')
print('Hello,', person)
```

This line asks for the user's name and stores it in a variable called “person”.

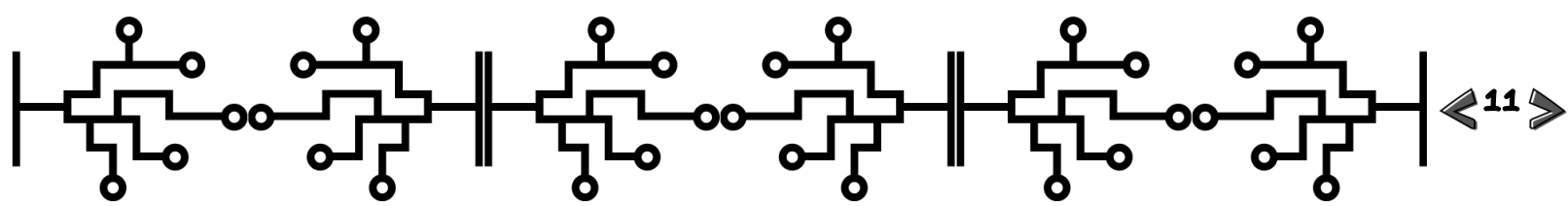


Activity 2, write a program that prints the following output.

```
*
* *
* * *
* * * *
* * * * *
```

Homework

- What is Python?
- What program or software is used to translate a python code into the machine code?
- What are the steps of writing a python code?
- Write a python program that ask a user to enter his name and his age and then print them out.



COMMENTS IN PYTHON

A comment allows you to write notes in English within your programs. And they also are used to disable parts of your program if you need to remove them temporarily.

```
1 # Anything after the # is ignored by python.
2 # Say hello to everyone.
3
4 print("Hello from Python")
5
6 # You can also use a comment to "disable" a piece of code:
7 # print("This won't run.")
8
9 print("This will run.")
```

How do you write comments?

In Python, the hash mark (#) indicates a comment. Anything following a hash mark in your code is ignored by the Python interpreter. (use " shift + 3 " to add #).

Activity 1 (Debugging)

1. Try to run the following code.
2. Find the mistakes and fix them.
3. Add your name and the current date as a comment.

```
1 # This and next line are comments.
2 This program prints my name and the current date.
3
4 PRINT("My name is: Salah")
5 print("The current date and time is: 29/01/2022")
```

Debugging

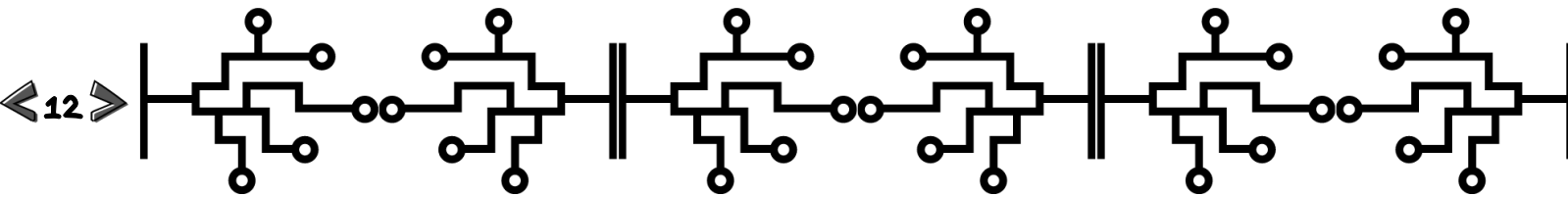
Debugging is the process of finding (and correcting) errors in a computer program. When programmers think that the program may have an error, they debug the program. They look for an error, and when they find it, they try to correct it so the program will work correctly.



Activity 2, Write down the output of the following code!

OUTPUT:

1. # This program prints *
2. print("* ")
3. print(" * ")
4. print(" *")
5. # print(" *")
6. print(" * ")
7. print("* ")
8. # print(" *")



MATH OPERATIONS IN PYTHON

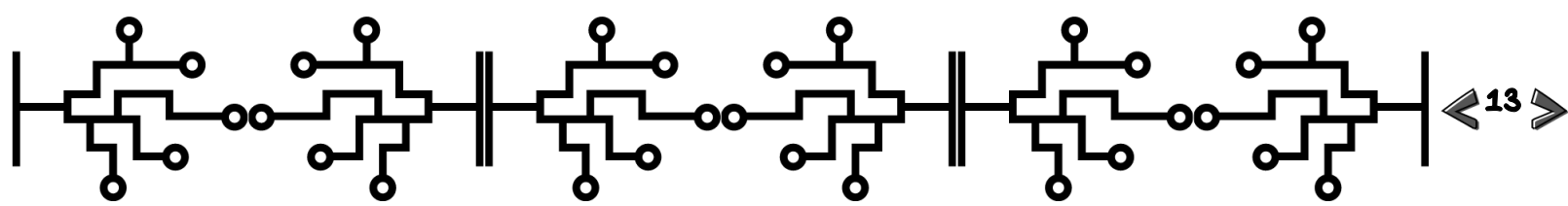
Python allows you to perform numerous mathematical operations, including multiplication, addition, division, and subtraction. There are some basic symbols that you can use in Python called operators, which are used to perform the necessary mathematical operations. The table lists the different mathematical operations that you are allowed to use in Python.

Symbol	Operation
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Remainder
**	Exponentiation
//	Floor division

Activity 1 (Coding)

```
1. # We are trying the different math operations!
2. print("Addition : 9 + 2 = ", 9 + 2)
3. print(" Subtraction: 9 - 2 = ", 9 - 2)
4. print(" Multiplication: 9 x 2 = ", 9 * 2)
5. print(" Division: 9 ÷ 2 = ", 9 / 2)
6. print(" Remainder: 9 % 2 = ", 9 % 2)
7. print(" Exponentiation: 92 = ", 9 ** 2)
8. print(" Floor division: 9 // 2 = ", 9 // 2)
```

1. Copy the given code and run the program.
2. Make a new program that prints the Multiplication table of any number (1 to 12).
3. What will be the output here: `print(30 - 15 * 2 / 5)`
4. Try to run: `print(30 - 15 * 2 / 5)` , is the output the same answer you got for Q3? If Not, Why?



VARIABLES IN PYTHON

If you want to write useful code, you'll need to be able to store and label pieces of information. That's what variables do. Variables are great for all sorts of things from tracking your score in a game to performing calculations and holding lists of items.



- Try out the following codes.

```
1. name = "Salah"
2. print(name)
3. name = "Box"
4. print(name)
```

```
1. name = input("Enter your name")
2. print("Welcome ", name)
```

How to create a variable?

A variable needs a name. Think of a name that will remind you what's inside the variable. Then decide what you want to store in the variable. This is the variable's value. Type the name, followed by an equals sign, followed by the value. We call this "assigning a value" to the variable.

Examples for variables:

age = 15

user_name = "Python"

message = "I am a python programmer"

x = 5

y = 25 - x

answer = x + y

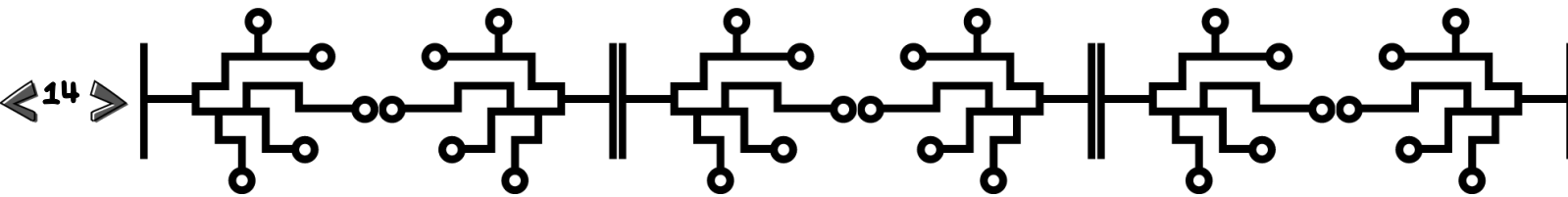


Naming variables

Choosing good names for your variables will make your program easier to understand. For example, a variable tracking a player's lives in a game could be called **lives_remaining**, rather than just **lives** or **lr**. Variable names can contain letters, numbers, and underscores, but they should begin with a letter. Follow the rules shown here and you won't go wrong.

Dos and don'ts

- Start the variable's name with a letter.
- Any letter or number can be used in the name.
- Symbols such as -, /, #, or @ aren't allowed.
- Spaces can't be used.
- An underscore (_) can be used instead of a space.
- Uppercase (capitals) and lowercase letters are different. Python will treat "Score" and "score" as two different variables.
- Avoid words Python uses as commands, such as "print".



Tick (✓) whether the given variable name is accepted in python (has no errors) :

- | | |
|---------------------------------|--------|
| 1. first name = "Salah" | () |
| 2. last_name = "Alkmali" | () |
| 3. num1 = 10 | () |
| 4. 2num = 15 | () |
| 5. print = "Printing" | () |
| 6. gmail@ = "myGmail@gmail.com" | () |
| 7. age = "my age" | () |

Activity 1 (Coding):

Store a message in a variable, and print that message. Then change the value of your variable to a new message, and print the new message.

Challenge 1

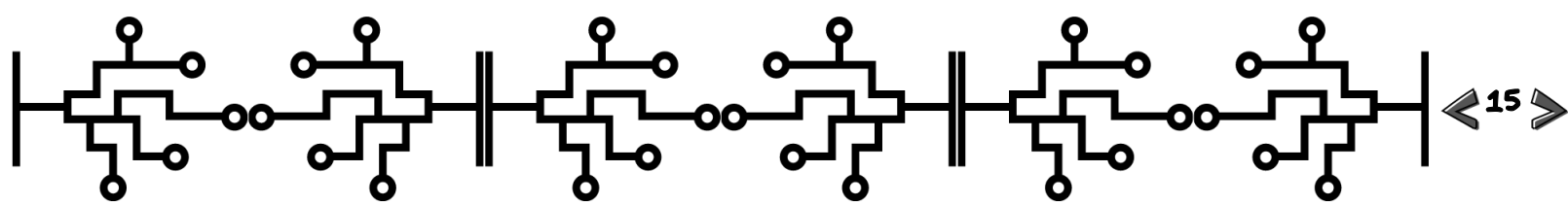
Make a program that swaps the variables values!

Do not use this way:

```
first_number = 6
second_number = 5
```

Expected output:

```
Before swapping :
first_number is 5
second_number is 6
After swapping:
first_number is 6
second_number is 5
```



What I know so far?

What is an Algorithm?

What is Debugging?

What is the output of the following code?

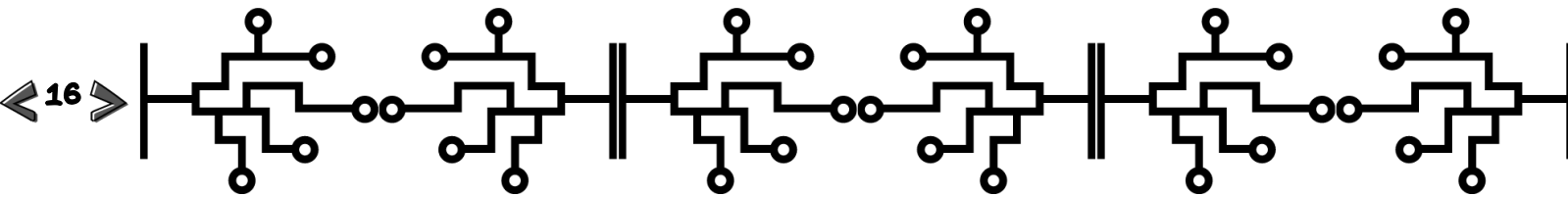
```
1. # print("Welcome to my program")
2. name = input("Please enter your name")
3. print("***** Welcome " , name , " *****")
4. print(10 * ( 40 / 8 ) - 25)
5. print("Goodbye " , name)
```

OUTPUT:

Find the mistakes in the following code and fix them.

```
1. print(" This program has 10 mistakes , Try to find them all"
2. This is a comment.
3. your name = INPUT(enter your name)
4. print("Your name is : " your_name)
5. age = INPUT(enter your age)
6. print("Your age is : " , Age)
7. PRINT("You did it!")
```

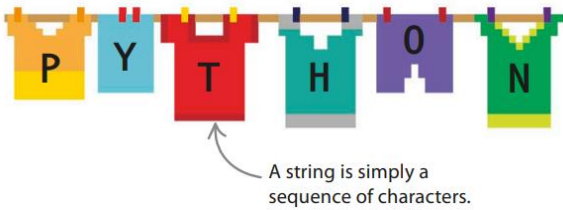
Write a python program that calculates and prints the number of seconds in a day .



TYPES OF VARIABLES IN PYTHON

STRINGS

Coders use the word “string” for any data made up of a sequence of letters or other characters. Words and sentences are stored as strings. Almost all programs use strings at some point. Every character that you can type on your keyboard, and even those you can’t, can be stored in a string.



A string is simply a sequence of characters. Anything inside quotes is considered a string in Python, and you can use single(‘This is a single quote’) or double quotes(“This is a double quote”) around your strings like this:

```
1. message = “ This is a string “
2. message = ‘ This is also a string’
```

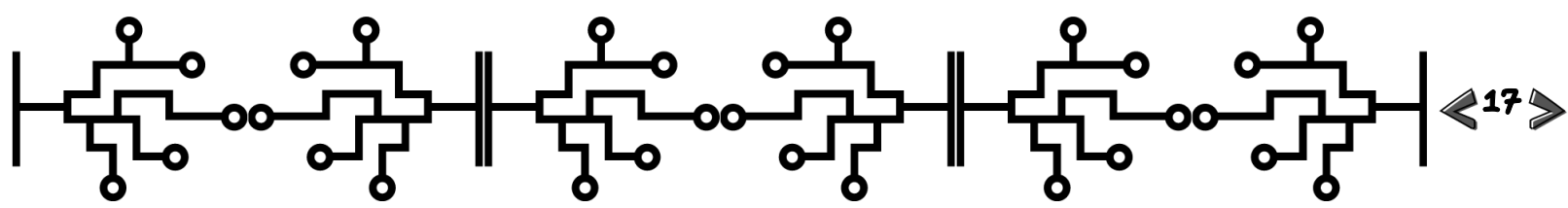
```
3. message = “ “ “
    This is multiple lines string
    “ “ “
```

Combining strings

Variables become useful when you combine them to make new variables. If you add two strings together, you can store the combination in a new variable. Try this out.

```
1. name = ' Salah '
2. greeting = 'Welcome to Earth, '
3. message = greeting + name
4. print(message)

OUTPUT: Welcome to Earth, Salah
```



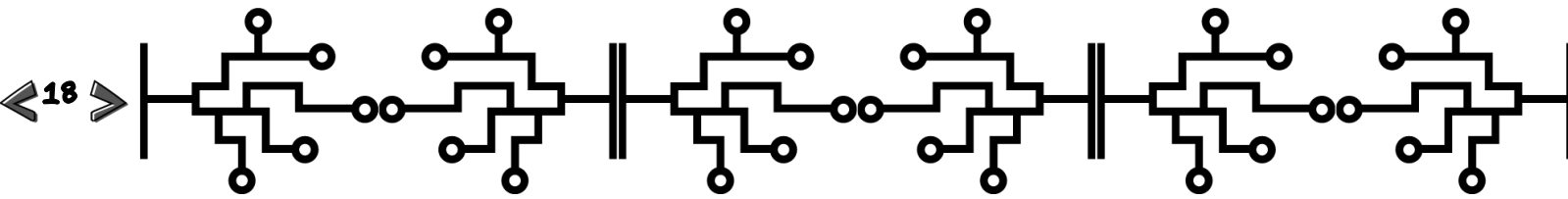
Embedding Values in Strings:

You can use a **placeholder** to display the messages in a string. This will allow you to embed any values within the string. The placeholder is the percentage symbol, which will act like the marker for where the value should be added. Embedding values is the same as inserting values.

1. `name = "Ali "`
2. `message = "Welcome to earth %s , you are a good programmer! "`
3. `print(message % name)`

Exercise 1

1. `my_name = input("Enter your name")`
2. `my_age = input("Enter your age")`
3. `my_height = input("Enter your height")`
4. `my_weight= input("Enter your weight")`
5. `my_eyes= input("Enter your eyes color")`
6. `my_hair= input("Enter your hair color")`
7. `print("Let's talk about %s." % my_name)`
8. `print("He's %d inches tall." % my_height)`
9. `print("He's %d pounds heavy." % my_weight)`
10. `print("Actually that's not too heavy.")`
11. `print("He's got %s eyes and %s hair." % (my_eyes, my_hair))`
12. `print("If I add %d, %d, and %d I get %d." % (my_age, my_height, my_weight, my_age + my_height + my_weight))`



Some useful functions with strings

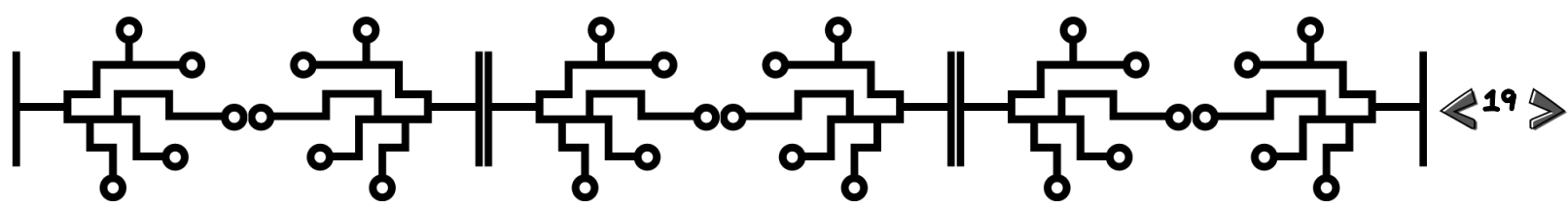
1. `len()` → Length of a string : to find out how many characters there are in a string.
2. `title()` → displays each word in title case, where each word begins with a capital letter.
3. `upper()` → convert strings to UPPERCASE
4. `lower()` → convert strings to lowercase

Exercise 2

1. `quote = "A person who never made a mistake never tried anything new."`
2. `print("here you will see all the letters in uppercase: %s." % quote.upper())`
3. `print("here you will see all the letters in lowercase: %s." % quote.lower())`
4. `print("here you will see the first letter of each word capital: %s." % quote.title())`
5. `print("The Length of the quote is : %d." % len(quote))`

Challenge 2 : What will be the output of the following code? Explain why you see that output?

1. `first_number = input(" Please enter a number")`
2. `second_number = input(" Please enter a number")`
3. `output = first_number + second_number`
4. `print(output)`



INTEGERS AND FLOATS

In coding, whole numbers are called integers, while numbers with a decimal point in them are known as floats. Programs usually count things using integers. Floats are more often used for measurements.

Using numbers

Variables can be used to store numbers. You can use them with symbols to do calculations, just like you do in math. (Go to page 12 to see the different math operations in python)

Exercise 1

Simple calculator that ask the user to enter 2 numbers and perform adding , subtracting , multiplication and division

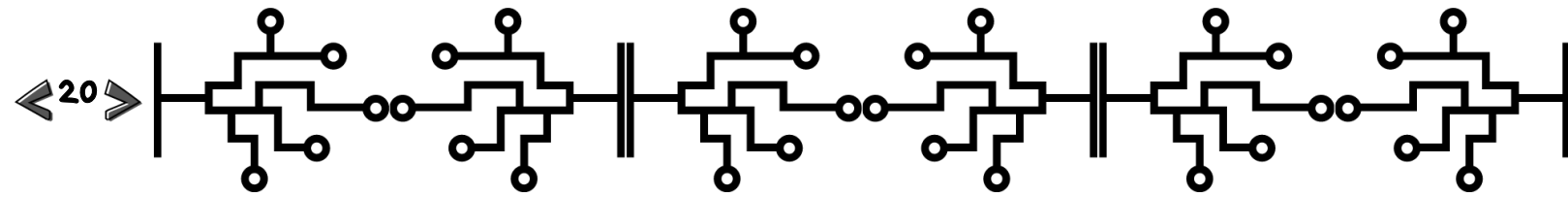
```
1. first_number = int(input(" Please enter the first number"))
2. second_number = int(input(" Please enter the second number"))
3. print("%s + %s = %s " % (first_number , second_number , first_number + second_number) )
4. print("%s - %s = %s " % (first_number , second_number , first_number - second_number) )
5. print("%s * %s = %s " % (first_number , second_number , first_number * second_number) )
6. print("%s / %s = %s " % (first_number , second_number , first_number / second_number) )
```

NOTE: input function asks the user to enter a data and store this data as string!

to tell python that we want the user to enter a number we must put input function inside int() ,
int() makes sure that the user input is an INTEGER (number).

Question: How can we make sure that the user input will be a DECIMAL number ?

Activity 1 : Change the type of the input in Exercise 1 to float instead of int , What is the output of the same code? Is it the same output? Why , Why not?



LISTS IN PYTHON

A *list* is a collection of items in a particular order. You can make a list that includes the letters of the alphabet, the digits from 0–9, or the names of all the people in your family.



When you want to store a lot of data, or perhaps the order of the data is important, you may need to use a list. A list can hold many items together and keep them in order. Python gives each item a number that shows its position in the list. You can change the items in the list at any time.

```
1. # To declare a list in python, make all your items in square brackets [ ]
2. grades = [ 6 , 7 , 8 , 11 , 12 ]
3. grade6_students = ["Omer" , "Amin" , "Leena" , "Mysa"]
4. # the next line will print all the grades
5. print(grades)
6. # next line will print 6 (the first item in grades list)
7. print(grades[0])
8. # next line will print Leena
9. print(grade6_students[2])
10. # next line will print an error message , What is the error here!
11. print(grade6_students[5])
```

List index

Lists are ordered collections, so you can access any element in a list by telling Python the position, or index, of the item desired.

IMPORTANT NOTE

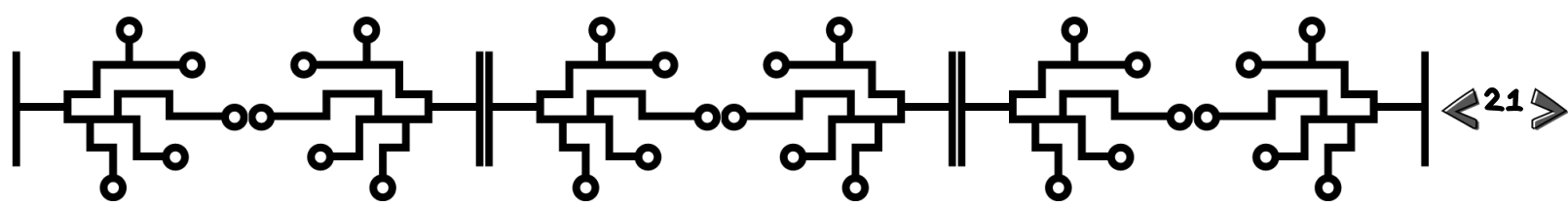
In programming we start counting from 0 not from 1, and that's why we got " 6 " in the 7th line .

Error: (list index out of range)

You will see this error message when you try to access an item that is not exist in the list.

Challenge 1

print sixth grade students' names in a capital LETTERS !



Adding Items to Lists

If you want to add new items to a list, you should use the append function.

```
grade6_students.append("Ahhad")
```

Removing Items From Lists

You should use the del command if you want to remove any items from a list.

```
del grade6_students[1]
```

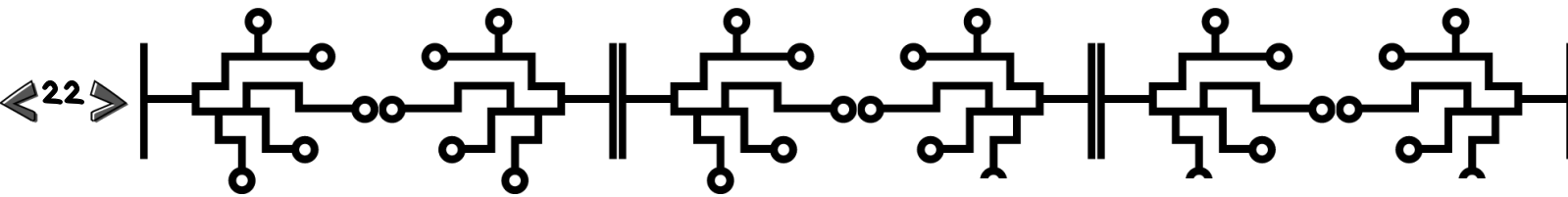
Join Lists

You can join two or more lists by adding them. The operation is the same as adding two numbers.

```
1. grade6_students = ["Omer", "Amin", "Leena", "Mysa"]
2. grade7_students = ["Hams", "Ahmed", "Maria", "Shihab"]
3. all_students = grade6_students + grade7_students
4. print(all_students)
```

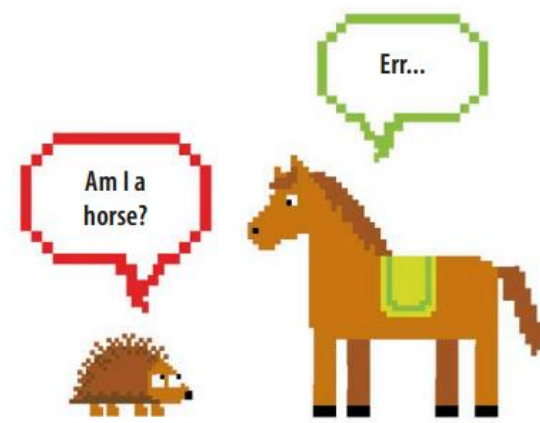
Challenge 2

Store the names of a few of your friends in a list called names. print a message to them. The text of each message should be the same, but each message should be personalized with the person's name.



MAKING DECISIONS IN PYTHON

Every day you make decisions about what to do next, based on the answers to questions you ask yourself. For example, “Is it raining?”, “Have I done my homework?”, “Am I a horse?” Computers also make decisions by asking questions.



Asking the Right Questions

The questions that computers ask themselves usually involve comparing one thing with another. For example, you can ask the question, “Are you older than 20?” If the answer to that question is ‘yes,’ you can respond by saying, “You are too old!”, otherwise respond by saying, “You are too young!”.

```
>>> answer_one = True
>>> answer_two = False
```

Variable

Boolean value

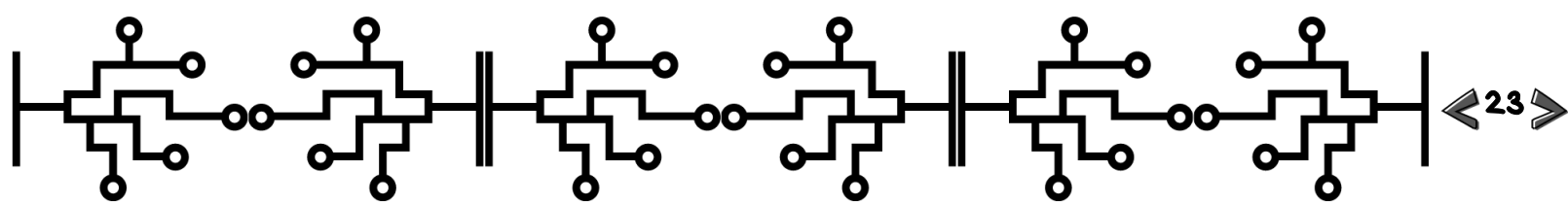
▷ Boolean values

The answers to the questions computers ask, have only two possible values: **True** or **False**. Python calls these two values Boolean values, and they must always start with a capital letter. You can store a Boolean value in a variable.

▽ Logical operators

These symbols tell computers to make comparisons. Programmers call them logical operators. You may have used some of them in math. The words “and” and “or” can also be used as logical operators in computer code.

Symbol	Meaning
==	equal to
!=	not equal to
<	less than
>	greater than



If Statements

An if statement has the 'if' keyword, a condition, and a colon. The lines of code that follow the colon should always be within a block.

This is how we declare if statement in python.

if (keyword)

age < 10 (condition)

: (colon tells python to run the if statement block only if the condition is true)

There will always be four spaces before the beginning of every line of code after the if statement. Look at the code and view the spaces again.

This space called **indentation**, and you have to be careful with spaces in python, you may have errors because of the indentation.

```
1 age = int(input("Enter your age: "))
2
3 if age < 10:
4     print("You are a kid") } code block
5
6 if age ≥ 10 and age < 20:
7     print("You are a teenager")
8
9 if age ≥ 20:
10    print("You are an adult")
```

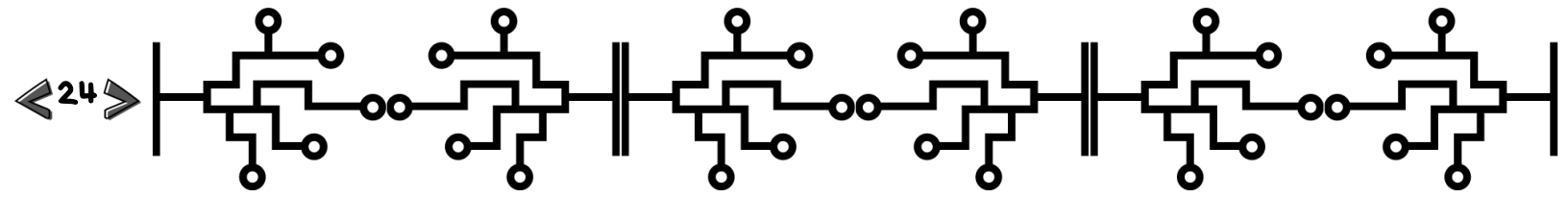
Exercise 1

Try to run this code and then find the errors and fix them

1. print("Welcome to my program! ")
2. if 10 < 12 :
3. print("That is true!")
4. if 10 != 10
5. print("That is a wrong statement! ")
6. print("Goodbye")

Challenge 1

1. Write a python program that tells you if a given number is an odd number or an even number.
2. Search on the internet about how can you use else if statement!



LEARN WITH CODING

Example 1



```
1 # A program that test if a given number is a multiple of 7
2
3 number_to_check = int(input("Enter any multiple of 7: "))
4
5 if number_to_check % 7 == 0 :
6     print("Bengo! , You are coreect")
7
8 else:
9     print("Opps, you got that wrong")
```

Remember

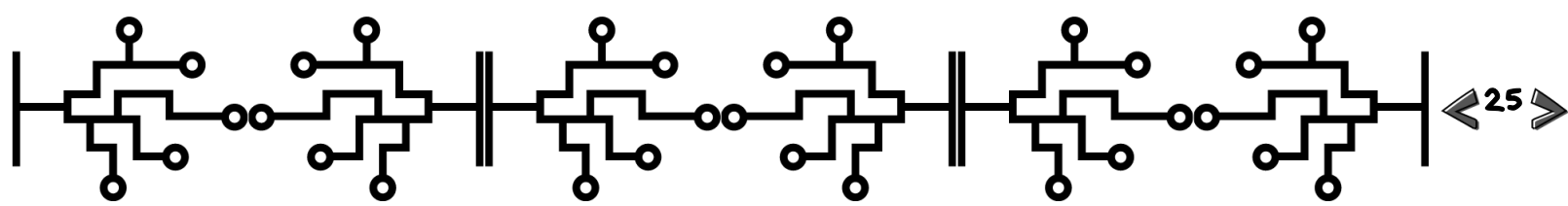
The % is the reminder operation.

If the given number can be evenly divided by 7 , that means this number is a multiple of 7.

Example 2



```
1 # A program that test multiple scenarios !
2
3 weather = input ("What is the forecast for today? (rain/snow/sun) ")
4
5 if weather == "rain":
6     print("Remember your umbrella!")
7
8 elif weather == "snow":
9     print('Remember your gloves!')
10
11 elif weather == 'sun':
12     print('Dont forget your sunglasses!')
13
14 else:
15     print('I dont know what to do with this weather')
```



Example 3

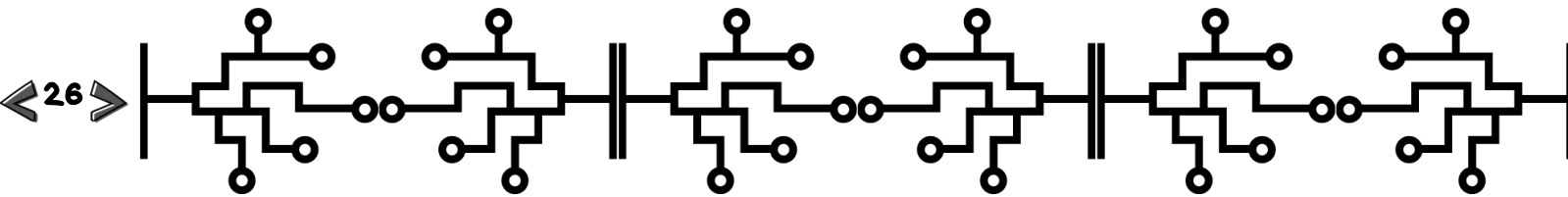


```
1 # A program that check if a given name is in the list of students.
2
3 students = ["Omer", "Leena", "Rama", "Jane", "Amin"]
4
5 name = input("Enter your name: ")
6
7 if name in students:
8     print("Welcome %, you are in the list" % name)
9
10 else:
11     print("You are not in the list")
12
13     add_to_list= input("Do you want to be added to the list? (y/n) ")
14
15     if add_to_list == "y":
16         students.append(name)
17         print("Your name is added to the list")
18
19     else:
20         print("Your name is not added to the list")
21
```

Example 4



```
1 name = input("Enter your name: ")
2 password = input("Enter your password: ")
3
4 if name == "python" and password == "2022":
5     print("Welcome to the python class")
```

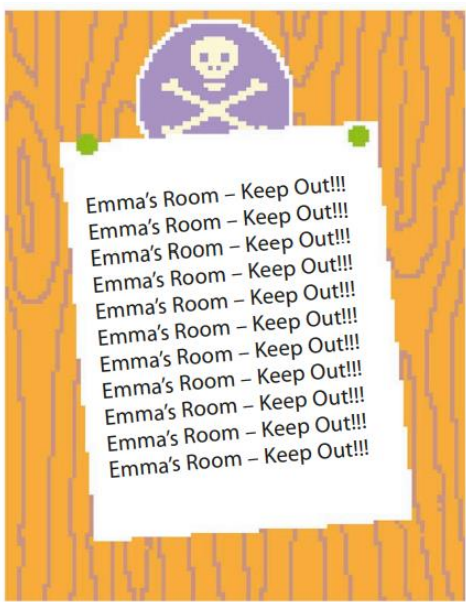
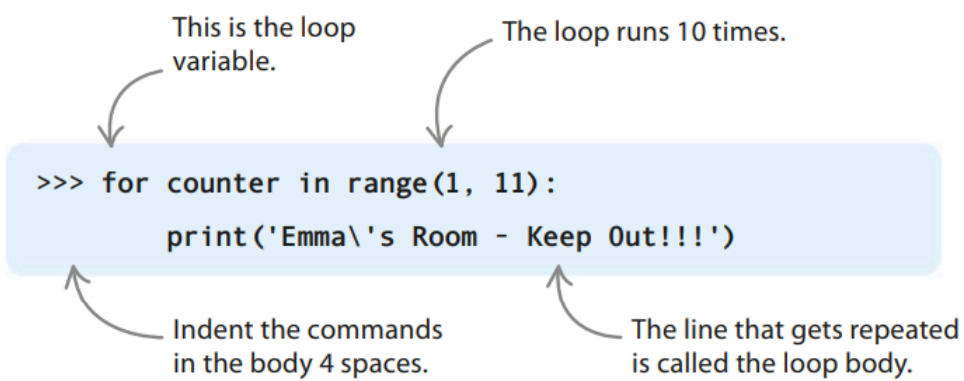


LOOPS IN PYTHON

Computers are great at doing boring tasks without complaining. Programmers aren't, but they are good at getting computers to do repetitive work for them – by using loops. A loop runs the same block of code over and over again. There are several different types of loop.

FOR LOOP:

When you know how many times you want to run a block of code, you can use a for loop. In this example, Emma has written a program to make a sign for her door. It prints “Emma’s Room – Keep Out!!!” ten times.



▽ Loop variable

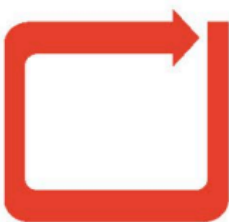
The loop variable keeps track of how many times we’ve gone round the loop so far. The first time round it’s equal to the first number in the list specified by **range(1, 11)**. The second time round it’s equal to the second number in the list, and so on. When we’ve used all the numbers in the list, we stop looping.

First loop



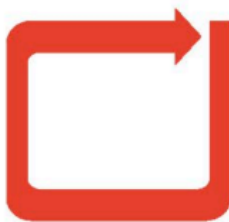
Loop variable = 1

Second loop



Loop variable = 2

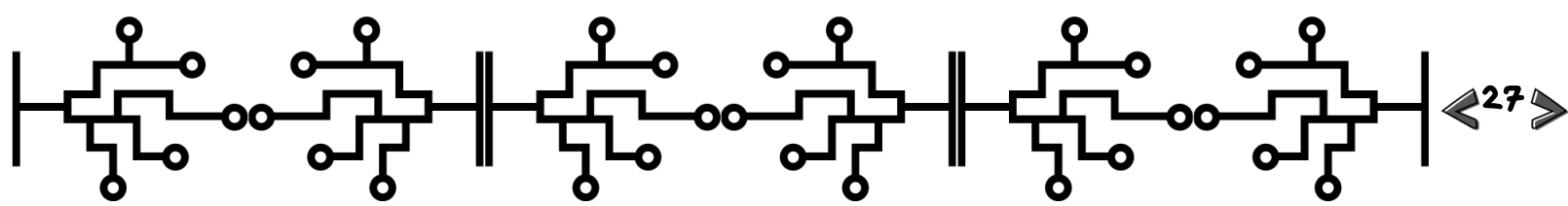
Third loop



Loop variable = 3

Range

In Python code, the word “range” followed by two numbers within brackets stands for “all the numbers from the first number to one less than the second number”. So **range(1, 4)** means the numbers 1, 2, and 3 – but not 4. In Emma’s “Keep Out” program, **range(1, 11)** is the numbers 1, 2, 3, 4, 5, 6, 7, 8, 9, and 10.



For loop Examples:

Example 1

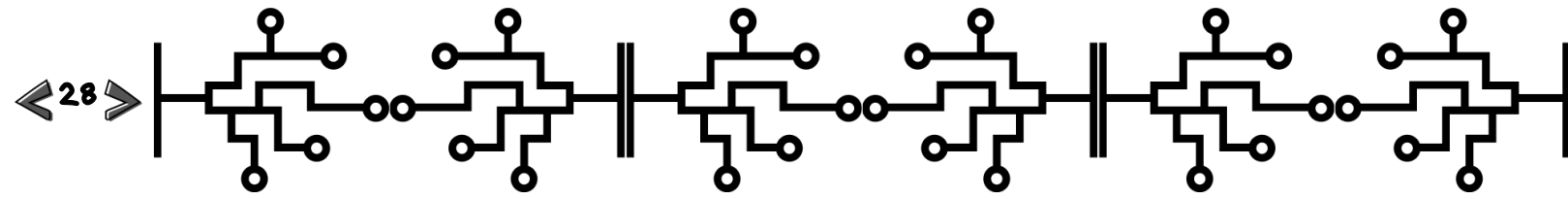
```
1 # print all the numbers from 1 to 100
2 for i in range(1,101):
3     print(i)
4
5 # print all the odd numbers from 1 to 100
6 for i in range(1,101,2):
7     print(i)
8
9 # print all the even numbers from 1 to 100
10 for i in range(2,101,2):
11     print(i)
```

Example 2

```
1 # A Program that prints all the list items
2
3 students = ["Omer", "Leena", "Rama", "Jane", "Amin"]
4
5 for student in students:
6     print(student)
```

Challenge 1

Make the code in example 2 prints the names in capital letters.



WHILE LOOPS

What happens if you don't know how many times you want to repeat the code? Do you need a crystal ball or some other way of seeing into the future? No, it's okay! You can use a while loop.

In this example, Ahmed has written a program to keep track of how many blocks there are to make a tower. Read through the code and see if you can figure out how it works.

```
1 blocks = 0
2 answer = 'y'
3
4 while answer == 'y':
5     blocks = blocks + 1
6     print("You have %d blocks" % blocks)
7     answer = input("Do you want to add another block? (y/n) ")
```

▷ Loop condition

A while loop doesn't have a loop variable that's set to a range of values. Instead it has a loop condition. This is a Boolean expression that can be either True or False. It's a bit like a bouncer at a disco asking you if you've got a ticket. If you have one (True), head straight for the dance floor; if you don't (False), the bouncer won't let you in. In programming, if the loop condition isn't True, you won't get into the loop

Infinite loops

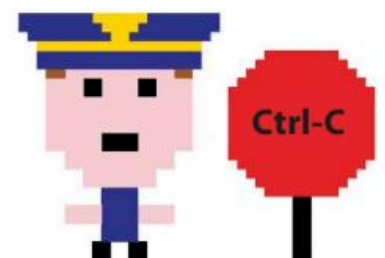
Sometimes you may want a while loop to keep going for as long as the program is running. This kind of loop is called an infinite loop. Lots of video-game programs use an infinite loop known as a main loop.

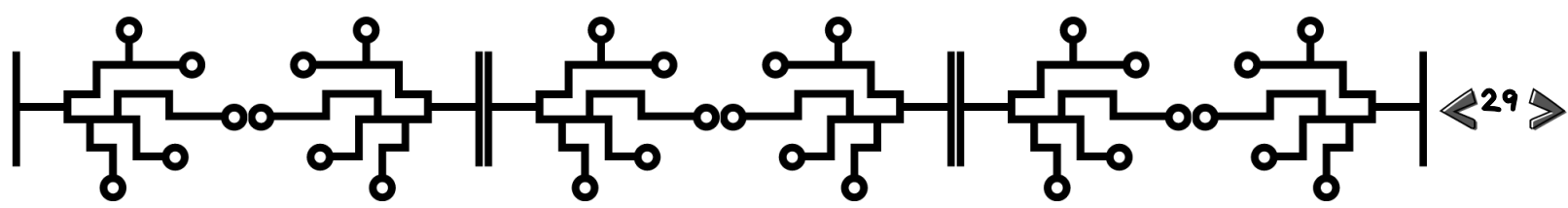
```
1 while True:
2     print("Hi, I am doing an infinite loop ! ")
3
```

Escaping infinity

To exit a while loop immediately without running any remaining code in the loop, use the **break** statement.

Or you can make the loop condition False, it's important to make sure that the body of a while loop does something that could make the loop condition False. But don't worry – if you accidentally code an infinite loop, you can escape from it by pressing the C key while holding down the Ctrl (control) key. You may have to press Ctrl-C several times before you quit the loop.





Some Python Projects



```
1 # A program that ask the user to enter a number and
2 # print out the sum of all the numbers from 1 to that number.
3
4 number = int(input("Enter a number: "))
5 sum = 0
6 for i in range(1,number+1):
7     sum = sum + i
8 print("The sum of all the numbers from 1 to %d is %d" % (number,sum))
9
```



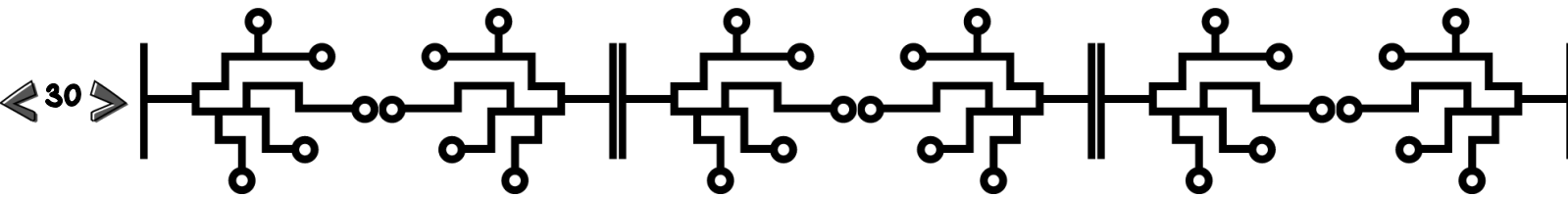
```
1 # A program that ask the user to enter a number and
2 # print out the factorial of that number.
3
4 number = int(input("Enter a number: "))
5 factorial = 1
6 for i in range(1,number+1):
7     factorial = factorial * i
8 print("The factorial of %d is %d" % (number,factorial))
```

Challenge 1

Make a list that contains students' grades. [87, 75 , 60 , 98 , 94 , 69]

Declare a variable sum that store the total amount of grades for all the students. (use a for loop)

Find the average of the students' grades.



What I have learned

Write down the output of the following codes:

1. -----

```
1 if 15 > 20:
2     print("This is false")
3 else:
4     print("This is true")
```

2. -----

```
1 print("5 to the power of 2 is %d" % (5**2))
```

Find the errors in the following code and fix them:

```
1 $ A program that prints a greating message to all the students in the list.
2 # There are 4 errors in this program.
3
4 student list = ["Omer", "Leena", "Rama", "Jane", "Amin"]
5
6 for student in student_list
7     print("Hello %s, welcome to the class" % studnt)
```

Write a python program that inverse a list.

list = [1 , 2 , 3 , 4]

Expected output:

4

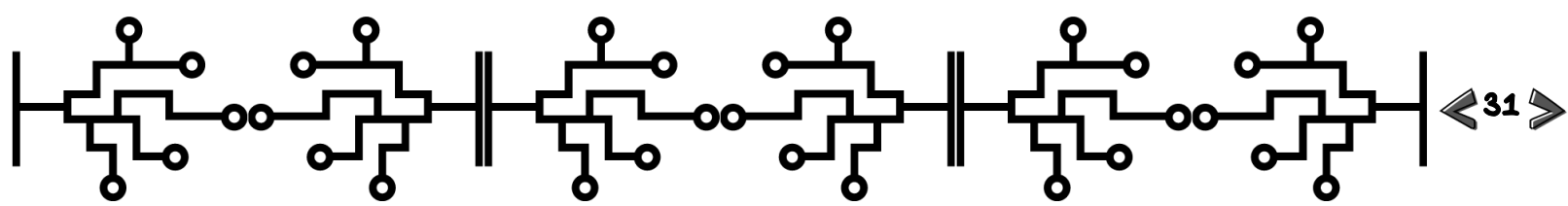
3

2

1

Write a python program that prints the absolute value of a given number.

Write a python program that prints the perfect cubes of the numbers from 1 to 10.



PYTHON QUIZ PROJECT

In this project, you'll build a quiz. Even though the questions are about computers, this project can be easily modified to be about any other topic.

Instructions

Make a friendly user interface (as you will see in the output picture).

Ask the user to enter his name.

Declare the questions list and the answers list:

```
questions = [" Which of the following is not a system software? \n 1. Windows \n 2. Linux \n 3. Compiler \n 4. MS Word \n",
             " Is a portable computer that includes a physical keyboard? \n 1. Laptop \n 2. Desktop \n 3. Tablet \n 4. Smartphone \n",
             " Which of following is not an input device? \n 1. Keyboard \n 2. Mouse \n 3. Monitor \n 4. Scanner \n",
             " Which of the following is not a type of computer? \n 1. Desktop \n 2. Laptop \n 3. Mobile phone \n 4. None \n",
             " Motherboard is an external hardware device. \n 1. True \n 2. False \n"]
```

```
answers = ["4", "1", "3", "4", "2"]
```

declare a variable to keep track on the user score.

Use a for loop to ask the user all the 5 questions and check his answers if he got the right answer print " correct " and increase the score by 10. If he got a wrong answer, print " wrong "

When he finishes all the questions print out his score.

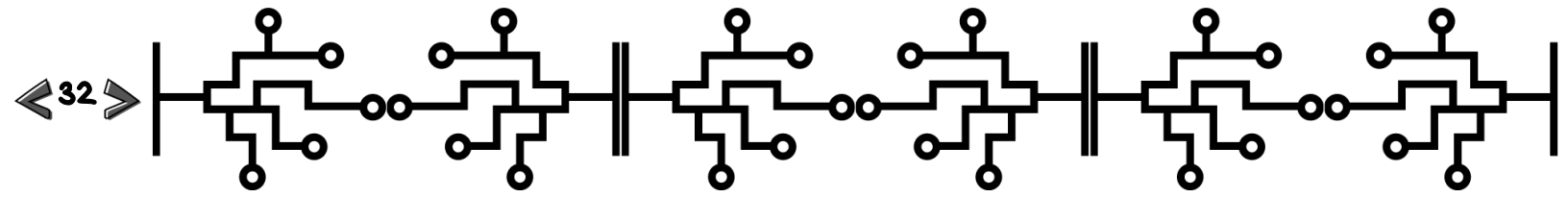
```
***** Welcome to the quiz *****
***** You have 5 questions *****
***** Good luck *****

Enter your name: Salah

Hi Salah, let's start the quiz

Which of the following is not a system software?
1. Windows
2. Linux
3. Compiler
4. MS Word
4
Correct

Is a portable computer that includes a physical keyboard?
1. Laptop
2. Desktop
3. Tablet
4. Smartphone
1
Correct
```



Functions in python

Programmers love shortcuts that make writing code easier.

One of the most common shortcuts is to give a name to a block of code that does an especially useful job. Then, instead of having to type out the whole block each time you need it, you simply type its name.

These named blocks of code are called functions.

Here's a simple function named `greet_user()` that prints a greeting:

```
1 def greet_user():
2     print("Hello!")
3
4 greet_user()
```

Try to call `greet_user()` more than once, What will be the output?

Here's a simple function with parameter named `greet_user(name)` that prints a greeting with the user name:

```
1 def greet_user(name):
2     print("Hello %s!" % name)
3
4 greet_user("Python")
```

Function terms

There are a number of special words that coders use when talking about functions.

Call To use a function.

def to declare a function.

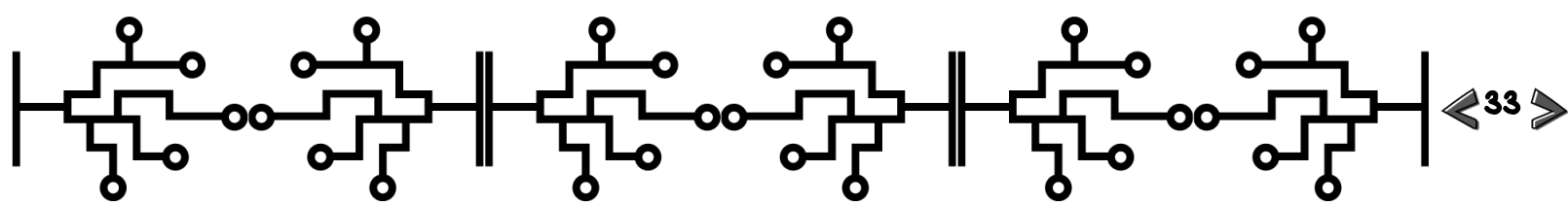
Parameter A piece of data (information) that you give to a function to use.

Return value Data that you pass from a function back to the main code. You get it using the keyword `return`.

The function now expects you to provide a value for username each time you call it.

Challenge 1

Write a function called `favorite_book()` that accepts one parameter, `title`. The function should print a message, such as One of my favorite books is Alice in Wonderland. Call the function, making sure to include a book title as an argument in the function call.



Here's a simple function that returns a value:

```
1 def add_numbers(a,b):  
2     return a + b  
3  
4 sum = add_numbers(5,6)  
5 print(sum)
```

Important note

It's important to define your functions before you use them in your main code. When you're learning to code with Python, it's helpful to put your functions at the top of your file, after any import statements. By doing this, you won't make the mistake of trying to call a function before you've defined it.

Built-in functions

Python has a number of built-in functions that you can use in your code. These are helpful tools that let you do lots of tasks, from inputting information and showing messages on the screen to converting one type of data into another. You've already used some of Python's built-in functions, such as `print()` and `input()`.

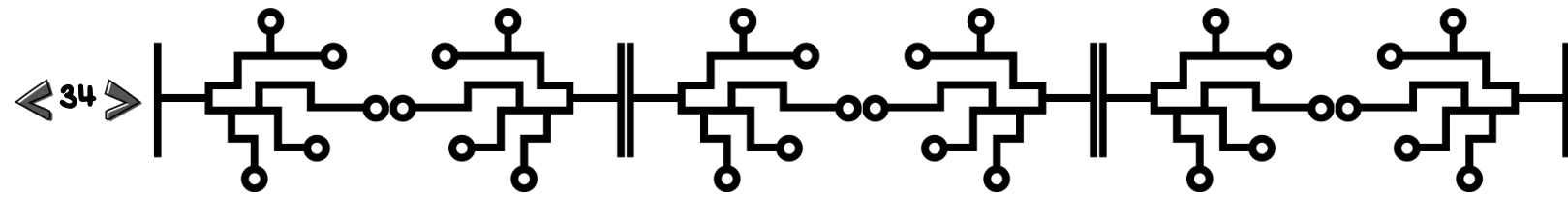
```
>>> name = input('What is your name?')  
What is your name? Sara  
>>> greeting = 'Hello' + name  
>>> print(greeting)  
Hello Sara
```

This asks the user to type in their name.

This shows the content of the variable `greeting` on the screen.

How to use a function

Using a function is also known as “calling” it. To call a function, you just type the function's name, followed by a set of brackets that contain any parameters you want the function to work with. Parameters are a bit like variables that belong to the function, and they allow you to pass data between different parts of your program. When a function doesn't need any parameters, the brackets are left empty.



Calculator using python

```
# A calculator program

def add(a,b):
    return a + b

def subtract(a,b):
    return a - b

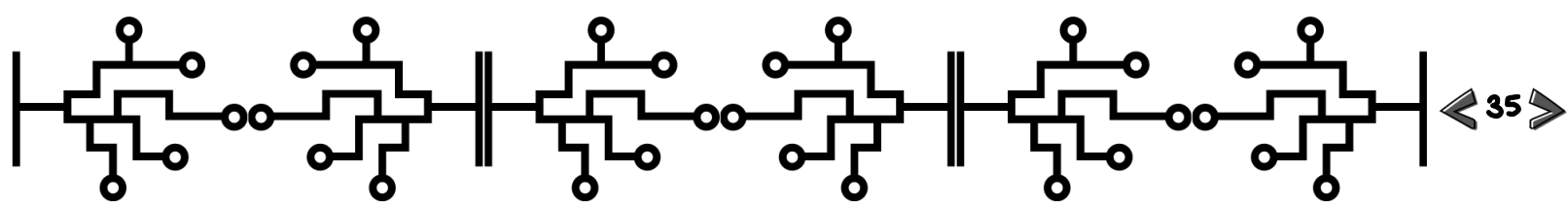
def multiply(a,b):
    return a * b

def divide(a,b):
    return a / b

use_calculator = True

while use_calculator:
    print("What do you want to do?\n1.Add\n2.Subtract\n3.Multiply\n4.Divide\n5.Exit \n")

    choice = int(input("Enter your choice: "))
    print()
    if choice == 1:
        first_number = int(input("Enter the first number: "))
        second_number = int(input("Enter the second number: "))
        print("The sum is %d" % add(first_number, second_number))
    elif choice == 2:
        first_number = int(input("Enter the first number: "))
        second_number = int(input("Enter the second number: "))
        print("The difference is %d" % subtract(first_number, second_number))
    elif choice == 3:
        first_number = int(input("Enter the first number: "))
        second_number = int(input("Enter the second number: "))
        print("The product is %d" % multiply(first_number, second_number))
    elif choice == 4:
        first_number = int(input("Enter the first number: "))
        second_number = int(input("Enter the second number: "))
        print("The quotient is %d" % divide(first_number, second_number))
    elif choice == 5:
        print("Thank you for using our calculator")
        use_calculator = False
    else:
        print("Invalid choice")
```



SOME PROJECTS AND CHALLENGES:

1. A program that checks if a given number, is prime number.
2. A program that prints all the prime numbers between 1 and 100.
3. A program that prints the smallest number in a list.
4. A program that prints the greatest value in a list.
5. A program that finds the GCF of 2 numbers.
6. A program that prints the number of seconds in a given number of days.
7. A program that orders numbers in a list from least to greatest.
8. A program that prints the area of a square.
9. A program that prints the numbers from 100 to 1.
10. A program that prints the Fibonacci sequence.

REFERENCES:

Boyini, K. (2018). Page Rank algorithm and implementation using Python [Online Forum Comment].

<https://www.tutorialspoint.com/page-rank-algorithm-and-implementationusing-python>

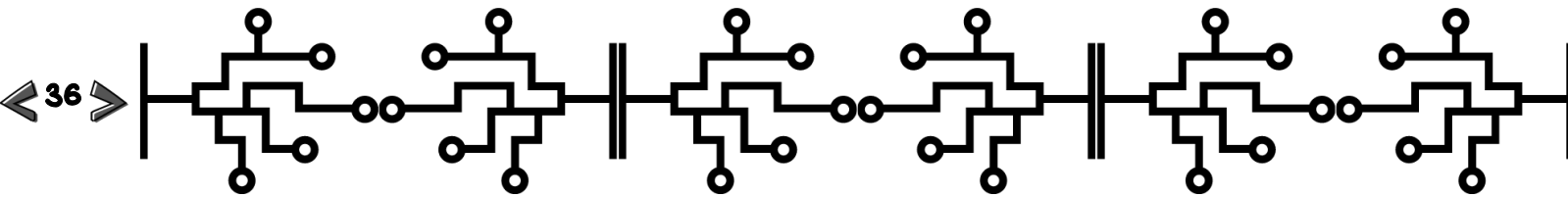
Computer coding for kid foreword by carol vorderman

Python Programming for Kids: Beginners' Guide With Easy-to-Learn Activities to Unlock the Adventurous World of Python Programming by Simon Weber

Learn python the hard way third edition by zed a. Shaw

Black Hat Python: Python Programming for Hackers and pentesters by justin seitz

Python Crash Course A Hands-On , Project-Based Introduction to Programming by Eric Matthes.



GLOSSARY

float

A number with a decimal point in it.

flowchart

A diagram that shows a program as a sequence of steps and decisions.

Boolean expression

A statement that is either True or False, leading to two possible outcomes.

Data

information, such as text, symbols, and numerical values.

Function

Code that carries out a specific task, working like a program within a program. Also called a procedure, subprogram, or subroutine.

Indent

When a block of code is placed further to the right than the previous block. An indent is usually four spaces. Every line in a particular block of code must be indented by the same amount.

Bug

An error in a program's code that makes it behave in an unexpected way.

Debug

To look for and correct errors in a program.

Call

To use a function in a program.

index number

A number given to an item in a list. In Python, the index number of the first item will be 0, the second item 1, and so on.

Condition

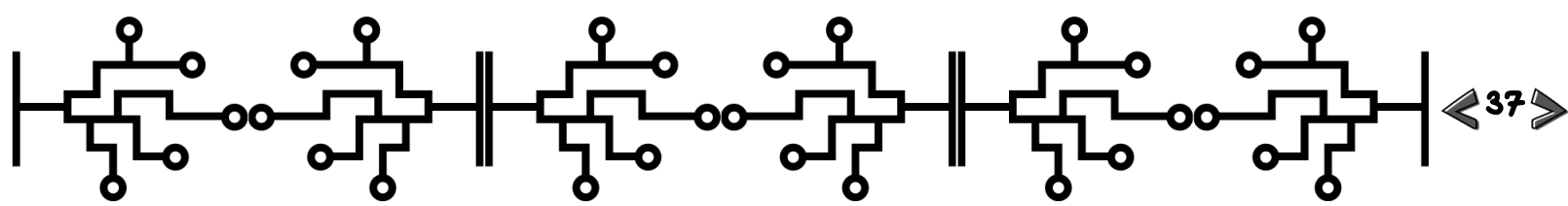
A "True or False" statement used to make a decision in a program. See also Boolean expression.

Integer

A whole number. An integer does not contain a decimal point and is not written as a fraction.

Python

A popular programming language created by Guido van Rossum. It is a great language for beginners to learn.



Operator

A symbol that performs a specific function: for example, “+” (addition) or “-” (subtraction).

Output

Data that is produced by a computer program and viewed by the user.

return value

The variable or data that is passed back after a function has been called (run).

List

A collection of items stored in numbered order.

Parameter

A value given to a function. The value of a parameter is assigned by the line of code that calls the function.

Run

The command to make a program start.

Loop

A part of a program that repeats itself, removing the need to type out the same piece of code multiple times.

Program

A set of instructions that a computer follows in order to complete a task.

Statement

The smallest complete instruction a programming language can be broken down into.

Variable

A place to store data that can change in a program, such as the player's score. A variable has a name and a value.

Programming language

A language that is used to give instructions to a computer.

String

A series of characters. Strings can contain numbers, letters, or symbols, such as a colon.

