AL NAFI,

A company with a focus on education, wellbeing and renewable energy.

"O Allah! I ask You for knowledge that is of benefit..."

اللهم انى أسالك علماً نافعاً و رزقاً طيباً و عملاً متقبلاً

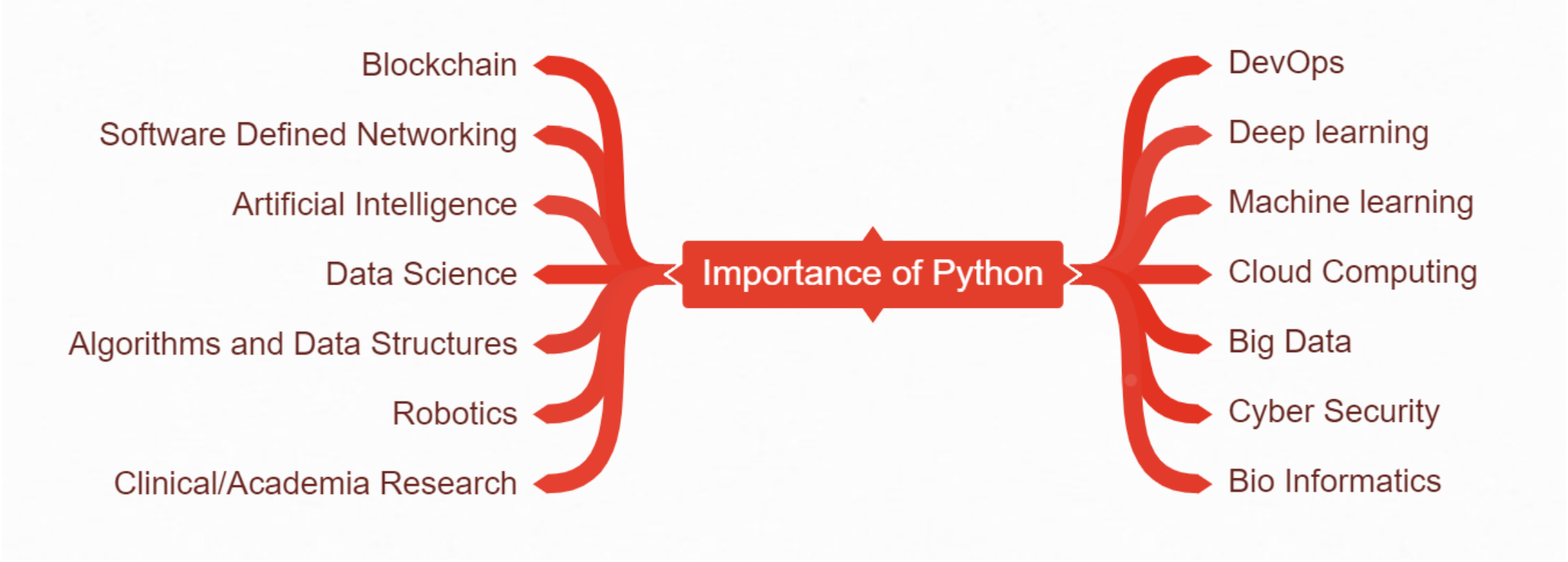
O Allah! I ask You for knowledge that is of benefit, a good provision and deeds that will be accepted.

[Ibn Majah]

Study, Rinse and Repeat

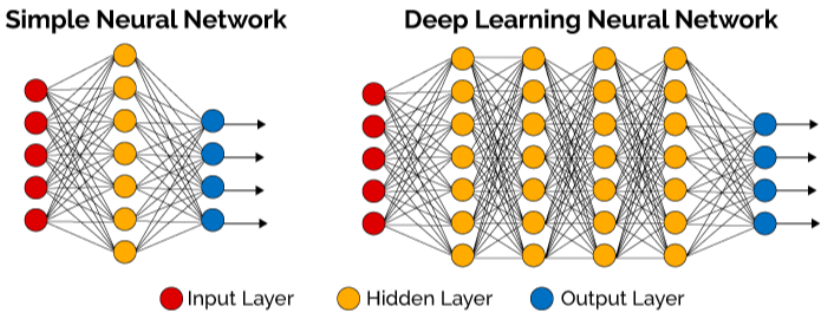
- Please subscribe to our YouTube Channel to be on top of your studies.
- Please keep an eye on zone@alnafi.com emails
- Please review the videos of 30 minutes daily
- Please review the notes daily.
- Please take time for clearing up your mind and reflect on how things are proceeding in your studies.

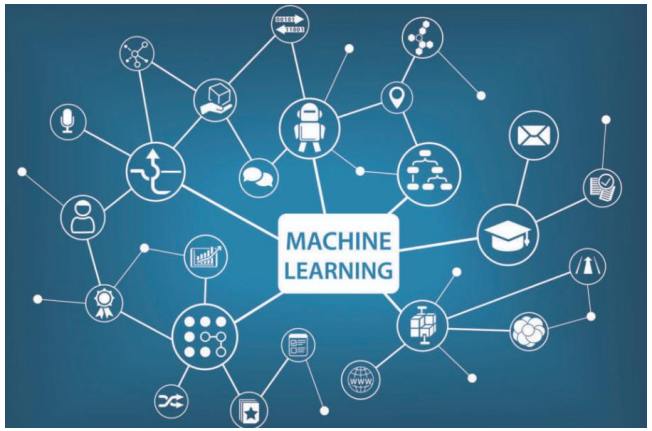
Role Of Python

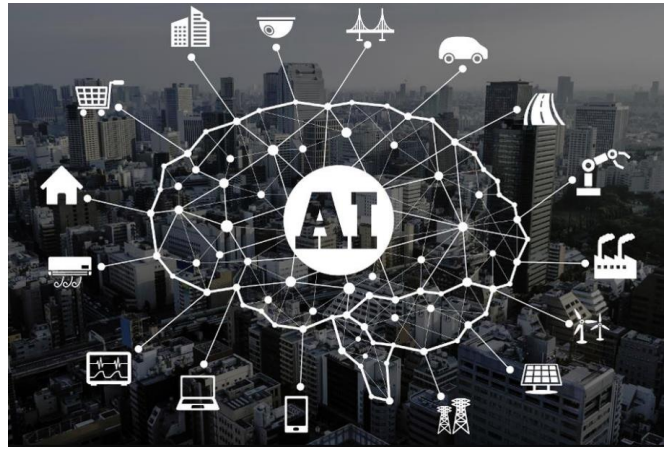


Role of Python

The Actual Role of Python







Calculations

Basic python operators

Symbol	Operation
+	Addition
-	Subtraction
*	Multiplication
/	Division

The order of Operations

```
>>> 5 + 30 * 20
605
```

For example, in the following equation, the numbers 30 and 20 are multiplied first, and the number 5 is added to their product.

Using parentheses ( )

Using Parentheses

- Parentheses can change the order of operations.

The order of Operations

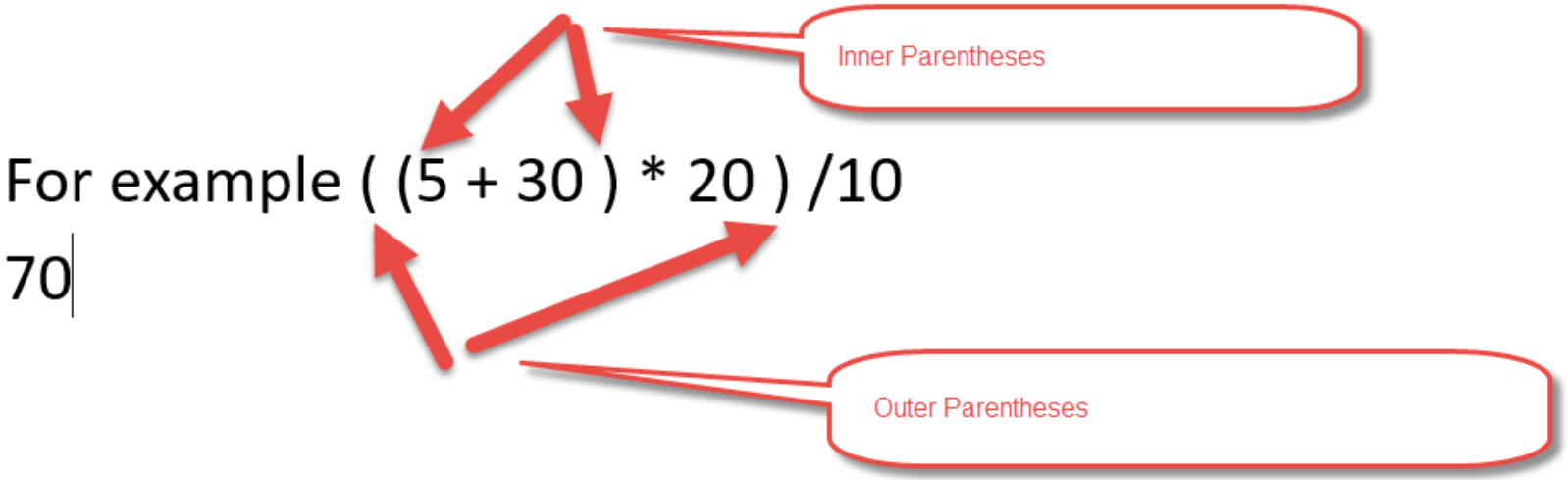
This equation is another way of saying, “multiply 30 by 20, and then add 5 to the result.” The result is 605. We can change the order of operations by adding parentheses around the first two numbers, like so:

```
>>> (5 + 30) * 20
700
```

The result of this equation is 700 (not 605) because the parentheses tell Python to do the operation in the parentheses first, and then do the operation outside the parentheses.

Nested Parentheses

- Inner and outer Parentheses



In this case, Python evaluates the innermost parentheses first, then the outer ones, and then the final division operator. In other words, this equation is saying, “add 5 to 30, then multiply the result by 20, and divide that result by 10.” Here’s what happens:

- Adding 5 to 30 gives 35.
- Multiplying 35 by 20 gives 700.
- Dividing 700 by 10 gives the final answer of 70.

WARNING

Remember that multiplication and division always go before addition and subtraction, unless parentheses are used to control the order of operations.

Variables

Variable in programming means a place to store information such as numbers, text, list of numbers and text etc.

Its like variable is a label for something.

To create a variable name Ali, we use an equal sign **=**

For example Ali = 100

100

Variable names can be made up of letters, numbers and the underscore character (_), but they cant start with a number. You can use anything from single letters such as (a) to long sentences for variable names. But you cannot use space ( )

A short name variable is best

Variable exercise

```
ali_coins=20
fahad_coins=10
abdullah_coins=10

ali_coins+fahad_coins+abdullah_coins

40
```

جزاك اللهُ

Please send us your questions at zone@alnafi.com
We will only answer our Nafi Members. So please quote your membership number within the email.

Python Primer 101

AL NAFI,
A company with a focus on education, wellbeing
and renewable energy.

الْحَمْـدُ للهِ الَّذي بِنِـعْمَتِهِ تَتِـمُّ الصّـالِحات

All Praise is for Allah by whose favor good works are accomplished.

الْحَمْـدُ للهِ على كُـلِّ حال

All Praise is for Allah in all circumstances."

Study, Rinse and Repeat

- Please subscribe to our YouTube Channel to be on top of your studies.
- Please keep an eye on zone@alnafi.com emails
- Please review the videos of 30 minutes daily
- Please review the notes daily.
- Please take time for clearing up your mind and reflect on how things are proceeding in your studies.

Good morning, Class.
Today we will be coding in Python.

Strings in Python

- When programming, we usually call text as ***String.***
- Think of string as a collection of letters, and the terms makes sense.
- All the letters and text in this presentation is a ***string***

Creating strings

We create a string by putting quotes around text because programming languages.

We need to tell a computer whether a value is a number, a string or something else.

```
abubakr = 'first caliph!!'
print(abubakr)
```

Creating strings continued…

One can also use double quotes

abubakr = “first caliph!!”
print(abubakr)

But you cannot start with single quote and end with double quote as you will get an error message.

It has to be consistent.

Syntax and Syntax Error

- Syntax means the arrangement and order of word in a sentence
- Syntax error means that you did something in an order python was not expecting or you missed something.
- EOL means end of line

```
In [28]: abubakr = 'first caliph!!
         print(abubakr)
```

```
  File "<ipython-input-28-c7062b08b869>", line 1
    abubakr = 'first caliph!!
                             ^
SyntaxError: EOL while scanning string literal
```

Adding second line to your command

- You can use 3 single quotes to write a second or third or as many lines as you want to a given command.

```
abubakr = ''first
Righteous
caliph!!''
print(abubakr)
```

Strings Problems handling

silly_string = 'He said, "Aren't can't shouldn't wouldn't."'

print(silly_string)

```
In [58]: silly_string = 'He said, "Aren't can't shouldn't wouldn't."'
         print(silly_string)

           File "<ipython-input-58-0e0f58e888e3>", line 1
             silly_string = 'He said, "Aren't can't shouldn't wouldn't."'
                                      ^
         SyntaxError: invalid syntax
```

When Python sees a quotation mark (either a single or double quote), it expects a string to start following the first mark and the string to end after the next matching quotation mark (either single or double) on that line.

silly_string = '''He said, "Aren't can't shouldn't wouldn't."'''

print(silly_string)

```
silly_string = '''He said, "Aren't can't shouldn't wouldn't."'''

print(silly_string)
```

```
He said, "Aren't can't shouldn't wouldn't."
```

Adding a backslash \ or escaping

\ or excaping means “Yes, I know I have quotes inside my string, and I want you to ignore them until you see the end quote.”

Escaping strings can make them harder to read, so it’s probably better to use multiline strings. Still, you might come across snippets of code that use escaping, so it’s good to know why the backslashes are there. Here are a few examples of how escaping works:

single_quote_str = 'He said, "Aren\'t can\'t shouldn\'t wouldn\'t."'

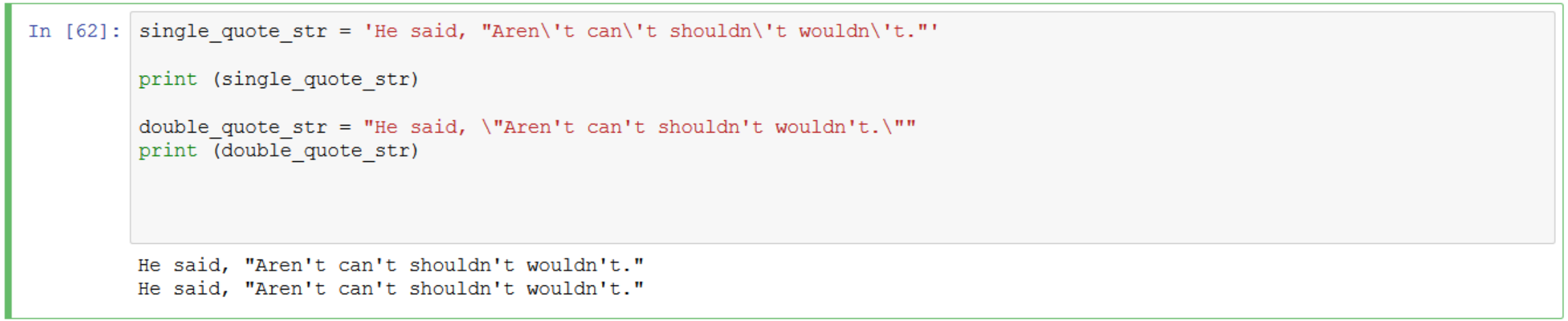
print (single_quote_str)

```
single_quote_str = 'He said, "Aren\'t can\'t shouldn\'t wouldn\'t."'

print (single_quote_str)
```

```
He said, "Aren't can't shouldn't wouldn't."
```

```
single_quote_str = 'He said, "Aren\'t can\'t shouldn\'t wouldn\'t."'
print (single_quote_str)
double_quote_str = "He said, \"Aren't can't shouldn't wouldn't.\""
print (double_quote_str)
```



Embedding value in strings

(Embedding values, also referred to as string substitution, is programmer-speak for “inserting values.”) For example, to have Python calculate or store the number of points you scored in a game, and then add it to a sentence like “I scored points,” use %s in the sentence in place of the value, and then tell Python that value, like this:

myscore = 1000

message = 'I scored %s points'

print(message % myscore)

```
myscore = 1000
message = 'I scored %s points'
print(message % myscore)

```

```
I scored 1000 points
```

Embedding value in strings continued…

nums = 'What did the number %s say to the number %s? Nice belt!!'

print(nums % (0, 8))

```
In [67]: nums = 'What did the number %s say to the number %s? Nice belt!!'
         print(nums % (0, 8))

         What did the number 0 say to the number 8? Nice belt!!
```

When using more than one placeholder, be sure to wrap the replacement values in parentheses, as shown in the example. The order of the values is the order in which they'll be used in the string.

Multiplying Strings

What is 10 multiplied by 5? The answer is 50, of course. But what's 10 multiplied by a? Here's Python's answer:

print(10 * 'a')

aaaaaaaaaa

```
print(10 * 'a')
```

```
aaaaaaaaaa
```

Multiplying Strings continued

```
spaces = ' ' * 25
print('%s 12 DHA Phase 5' % spaces)
print('%s Clifton' % spaces)
print('%s West Snoring' % spaces)
print()
print()
print('Dear Sir')
print()
print('I wish to report that tiles are missing from the')
print('outside toilet roof.')
print('I think it was bad wind the other night that blew them away.')
print()
print('Regards')
print('Sharfoo')
```

Fun Stuff

Try this code and share what happens

```
print(1000 * 'Karachi')
```

Lists vs. string

sharfoo_list='sabzee, fruit, aloo, chai'

print(sharfoo_list)

```
sharfoo_list='sabzee, fruit, aloo, chai'
print(sharfoo_list)
```

```
sabzee, fruit, aloo, chai
```

Creating a list

```
sharfoo_list=['sabzee, fruit, aloo, chai']
print(sharfoo_list[2])
```

Creating a list takes a bit more typing than creating a string, but a list is more useful than a string because it can be manipulated.

For example we can print the third item in the list.

```
sharfoo_list=['sabzee', 'fruit', 'aloo', 'chai']

print(sharfoo_list[2])
```

sharfoo_list=['sabzee', 'fruit', 'aloo', 'chai']

print(sharfoo_list[0:3])

```
sharfoo_list=['sabzee', 'fruit', 'aloo', 'chai']

print(sharfoo_list[0:3])
```

```
['sabzee', 'fruit', 'aloo']
```

Writing [0:3] is the same as saying, “show the items from index position 0 up to (but not including) index position 3”—or in other words, items 0, 1, and 2.

Lists can store all sorts of items, like numbers

some_numbers = [1, 2, 5, 10, 20]

They can also hold strings:

some_strings = ['Which', 'Witch', 'Is', 'Which']

List strings and number example

numbers_and_strings = ['Why', 'was', 6, 'afraid', 'of', 7, 'because', 7, 8, 9]

print(numbers_and_strings)

['Why', 'was', 6, 'afraid', 'of', 7, 'because', 7, 8, 9]

```
numbers_and_strings = ['Why', 'was', 6, 'afraid', 'of', 7,
print(numbers_and_strings)
['Why', 'was', 6, 'afraid', 'of', 7, 'because', 7, 8, 9]
```

```
['Why', 'was', 6, 'afraid', 'of', 7, 'because', 7, 8, 9]
```

```
['Why', 'was', 6, 'afraid', 'of', 7, 'because', 7, 8, 9]
```

List within a list ☺

```
numbers = [1, 2, 3, 4]
strings = ['I', 'kicked', 'my', 'toe', 'and', 'it', 'is', 'sore']
mylist = [numbers, strings]
print(mylist)
```

```
[[1, 2, 3, 4], ['I', 'kicked', 'my', 'toe', 'and', 'it', 'is', 'sore']]
```

Adding items to a LIST

numbers = [1, 2, 3, 4]

strings = ['I', 'kicked', 'my', 'toe', 'and', 'it', 'is', 'sore']

mylist = [numbers, strings]

print(mylist)

```
numbers = [1, 2, 3, 4]
strings = ['I', 'kicked', 'my', 'toe', 'and', 'it', 'is', 'sore']
mylist = [numbers, strings]
print(mylist)
```

```
[[1, 2, 3, 4], ['I', 'kicked', 'my', 'toe', 'and', 'it', 'is', 'sore']]
```

Adding items to a list

Before adding an item to a list:

```
wizard_list = ['spider legs', 'toe of frog', 'eye of newt', 'bat wing', 'slug butter', 'snake dandruff']
print(wizard_list)
```

After adding item to a list via .append

```
wizard_list = ['spider legs', 'toe of frog', 'eye of newt', 'bat wing', 'slug butter', 'snake dandruff']
wizard_list.append('bear burp')
print(wizard_list)
```

Challenge!

Now add the following to the previous list and print.

```
wizard_list.append('mandrake')
wizard_list.append('hemlock')
wizard_list.append('swamp gas')
```

Removing items from the list

To add an item to a list we use **append**

To remove an item from a list we use **del** short for delete

```
wizard_list = ['spider legs', 'toe of frog', 'eye of newt', 'bat wing', 'slug butter', 'snake dandruff']

wizard_list.append('bear burp')

del wizard_list [5]

print(wizard_list)
```

Remember that positions start at zero, so `wizard_list[5]` *actually refers to the sixth item in the list.*

List Arithmetic

We can join two lists by using +

```
list1 = [1, 2, 3, 4]
list2 = ['I', 'tripped', 'over', 'and', 'hit', 'the', 'floor']
print(list1 + list2)
```

We can also add the two lists and set the result equal to another variable.

```
list1 = [1, 2, 3, 4]
list2 = ['I', 'ate', 'chocolate', 'and', 'I', 'want', 'more']
list3 = list1 + list2
print(list3)
```

And we can multiply a list by a number. For example, to multiply list1 by 5, we write list1 * 5:

```
list1 = [1, 2]
print(list1 * 5)
```

```
[1, 2, 1, 2, 1, 2, 1, 2, 1, 2]
```

This is actually telling Python to repeat list1 five times, resulting in 1, 2, 1, 2, 1, 2, 1, 2, 1, 2.

On the other hand, division (/) and subtraction (-) give only errors, as in these examples:

list1 = [1, 2]

list1 / 20

print(list1)

```
list1 = [1, 2]
list1 / 20
print(list1)
```

```
---------------------------------------------------------------------------
--
TypeError                                 Traceback (most recent call las
t)
<ipython-input-115-06d5ab3556d0> in <module>()
      1 list1 = [1, 2]
----> 2 list1 / 20
      3 print(list1)

TypeError: unsupported operand type(s) for /: 'list' and 'int'
```

```
list1 = [1, 2]
list1 - 20
print(list1)
```

```
---------------------------------------------------------------------
--
TypeError                                 Traceback (most recent call las
t)
<ipython-input-116-76a048d0df0e> in <module>()
      1 list1 = [1, 2]
----> 2 list1 - 20
      3 print(list1)

TypeError: unsupported operand type(s) for -: 'list' and 'int'
```

But why? Well, joining lists with + and repeating lists with * are straightforward enough operations.

These concepts also make sense in the real world. For example, if I were to hand you two paper shopping lists and say, “Add these two lists,” you might write out all the items on another sheet of paper in order, end to end.

The same might be true if I said, “Multiply this list by 3.” You could imagine writing a list of all of the list’s items three times on another sheet of paper. But how would you divide a list? For example, consider how you would divide a list of six numbers (1 through 6) in two. Here are just three different ways:

[1, 2, 3] [4, 5, 6]

[1] [2, 3, 4, 5, 6]

[1, 2, 3, 4] [5, 6]

Would we divide the list in the middle, split it after the first item, or just pick some random place and divide it there? There's no simple answer, and when you ask Python to divide a list, it doesn't know what to do, either. That's why it responds with an error.

Tuples

A tuple is like a list that uses parentheses, as in this example:

```
fibs = (0, 1, 1, 2, 3)
print(fibs[3])
```

```
2
```

Here we define the variable fibs as the numbers 0, 1, 1, 2, and 3. Then, as with a list, we print the item in index position 3 in the tuple using print(fibs[3]). The main difference between a tuple and a list is that a tuple cannot change once you've created it.

For example, if we try to replace the first value in the tuple fibs with the number 4 (just as we replaced values in our wizard_list), we get an error message:

```
fibs = (0, 1, 1, 2, 3)
fibs[0]=4
print(fibs[3])
```

```
---------------------------------------------------------------------------
--
TypeError                                 Traceback (most recent call las
t)
<ipython-input-118-deedb4837723> in <module>()
      1 fibs = (0, 1, 1, 2, 3)
----> 2 fibs[0]=4
      3 print(fibs[3])

TypeError: 'tuple' object does not support item assignment
```

So why use tuple?

Why would you use a tuple instead of a list? Basically because sometimes it is useful to use something that you know can never change. If you create a tuple with two elements inside, it will always have those two elements inside.

Python Maps

In Python, a map (also referred to as a dict, short for dictionary) is a collection of things, like lists and tuples. The difference between maps and lists or tuples is that each item in a map has a key and a corresponding value. For example, say we have a list of people and their favorite sports. We could put this information into a Python list, with the person's name followed by their sport, like so:

Creating a map in Python

favorite_sports = ['Ali, Football', 'Omer, Basketball', 'Osman, Baseball', 'Abu Bkar, Netball', 'Fatima, Badminton', 'Faizan, Rugby']

print (favorite_sports)

```
favorite_sports = ['Ali, Football', 'Omer, Basketball', 'Osman, Baseball', 'Abu Bkar, Netball', 'Fatima, Badminton', 'Fa

print (favorite_sports)
```

```
['Ali, Football', 'Omer, Basketball', 'Osman, Baseball', 'Abu Bkar, Netball', 'Fatima, Badminton', 'Faizan, Rugby']
```

Modifying a value in a map.

Now, if we store this same information in a map, with the person's name as the key and their favorite sport as the value, the Python code would look like this:

```
favorite_sports = {'Ali' : 'Football',
                   'Omer' : 'Basketball',
                   'Osman' : 'Baseball',
                   'Abu Bakr' : 'Netball',
                   'Fatima' : 'Badminton',
                   'Faizan' : 'Rugby'}

print (favorite_sports)
```

We use colons to separate each key from its value, and each key and value is surrounded by single quotes. Notice, too, that the items in a map are enclosed in braces ({}), not parentheses or square brackets.

Now to get Ali favourite sport, we access ou rmap favourite_sports using his name as

```
favorite_sports = {'Ali' : 'Football',
                   'Omer' : 'Basketball',
                   'Osman' : 'Baseball',
                   'Abu Bakr' : 'Netball',
                   'Fatima' : 'Badminton',
                   'Faizan' : 'Rugby'}

print(favorite_sports['Ali'])
```

Deleting a value in the map

```
favorite_sports = {'Ali' : 'Football',
                   'Omer' : 'Basketball',
                   'Osman' : 'Baseball',
                   'Abu Bakr' : 'Netball',
                   'Fatima' : 'Badminton',
                   'Faizan' : 'Rugby'}

del favorite_sports['Ali']

print(favorite_sports)
```

Replacing a value in a map

```
favorite_sports = {'Ali' : 'Football',
                   'Omer' : 'Basketball',
                   'Osman' : 'Baseball',
                   'Abu Bakr' : 'Netball',
                   'Fatima' : 'Badminton',
                   'Faizan' : 'Rugby'}

favorite_sports['Faizan'] = 'Cricket'

print(favorite_sports)
```

We replace the favorite sport of Rugby with Cricket by using the key Faizan. As you can see, working with maps is kind of like working with lists and tuples, except that you can’t join maps with the plus operator (+). If you try to do that, you’ll get an error message:

جزاك اللهُ

Please send us your questions at zone@alnafi.com
We will only answer our Nafi Members. So please quote your membership number within the email.

AL NAFI,
A company with a focus on education,
wellbeing and renewable energy.

Python Primer 102a

Asking questions with IF and ELSE

Dua of the day

The righteous dua that Abu Bakr (r) made at the end of his lifetime.

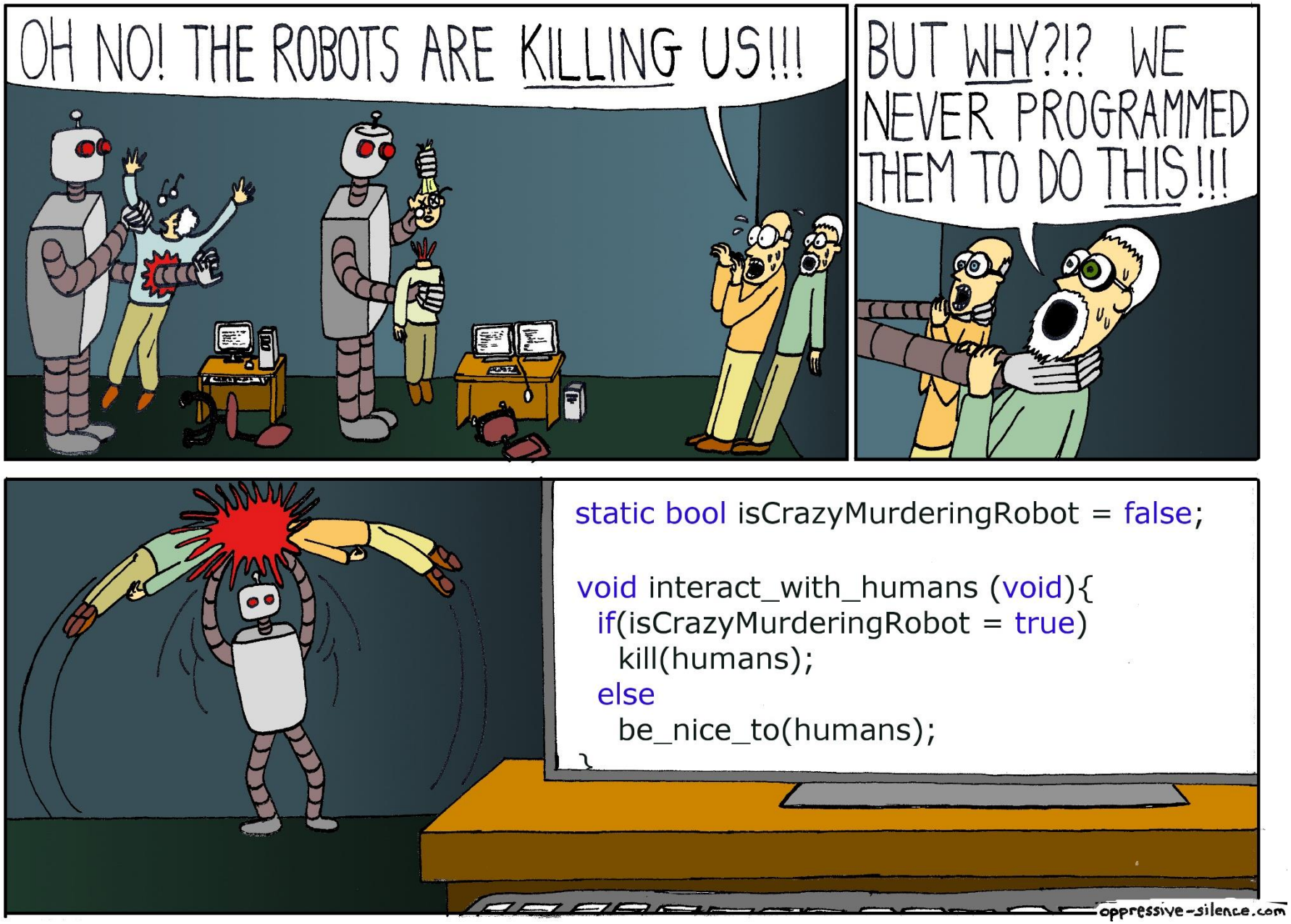
اللَّهُمَّ اجْعَلْ خَيْرَ زَمَانِيْ آخِرَهُ، وَخَيْرَ عَمَلِيْ خَوَاتِمَهُ، وَخَيْرَ أَيَّامِيْ يِوْمَ أَلْقَاكَ

"O Allah, let the best of my lifetime be its ending, and my best deed be that which I seal [my life with], and the best of my days the day I meet You."

Reference: The Dua is taken from Sheikh Omer Sulaiman Dua Compilation.

Study, Rinse and Repeat

- Please subscribe to our YouTube Channel to be on top of your studies.
- Please logon to our website https://alnafi.com/login/
- Use your username and password to logon
- Please keep an eye on zone@alnafi.com emails
- Please review the videos of 30 minutes daily
- Please review the notes daily.
- Please take time for clearing up your mind and reflect on how things are proceeding in your studies.



Conditions **If and Else**

In programming, we often ask yes or no questions, and decide to do something based on the answer.

For example, we might ask, “Are you older than 20?” and if the answer is yes, respond with “You are too old!”

These sorts of questions are called conditions, and we combine these conditions and the responses into **if statements**.

Conditions can be more complicated than a single question, and if statements can also be combined with multiple questions and different responses based on the answer to each question.

IF Statements

```
age = 13
if age > 20:
    Print ('you are too old')
```

```
In [3]: age = 13
        if age > 20:
            Print ('you are too old')
```

An if statement is made up of the if keyword, followed by a condition and a colon (:), as in if age > 20:

The lines following the colon must be in a block, and if the answer to the question is yes (or true, as we say in Python programming), the commands in the block will be run.

Now, let’s explore how to write blocks and conditions.

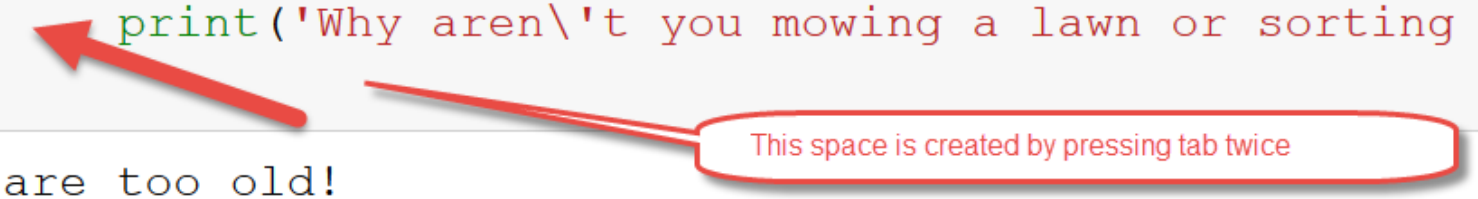
Python Block

A BLOCK IS A GROUP OF PROGRAMMING STATEMENTS

A block of code is a grouped set of programming statements.

For example, when if age > 20: is true, you might want to do more than just print “You are too old!” Perhaps you want to print out a few other choice sentences, like this:

```
age = 25
if age > 20:
        print('You are too old!')
        print('Why are you here?')
        print('Why aren\'t you mowing a lawn or sorting papers?')
```




```
You are too old!
Why are you here?
Why aren't you mowing a lawn or sorting papers?
```

This block of code is made up of three print statements that are run only if the condition age > 20 is found to be true.

Each line in the block has four spaces at the beginning, when you compare it with the if statement above it. Let’s look at that code again, with visible spaces:

The importance of Space in Python

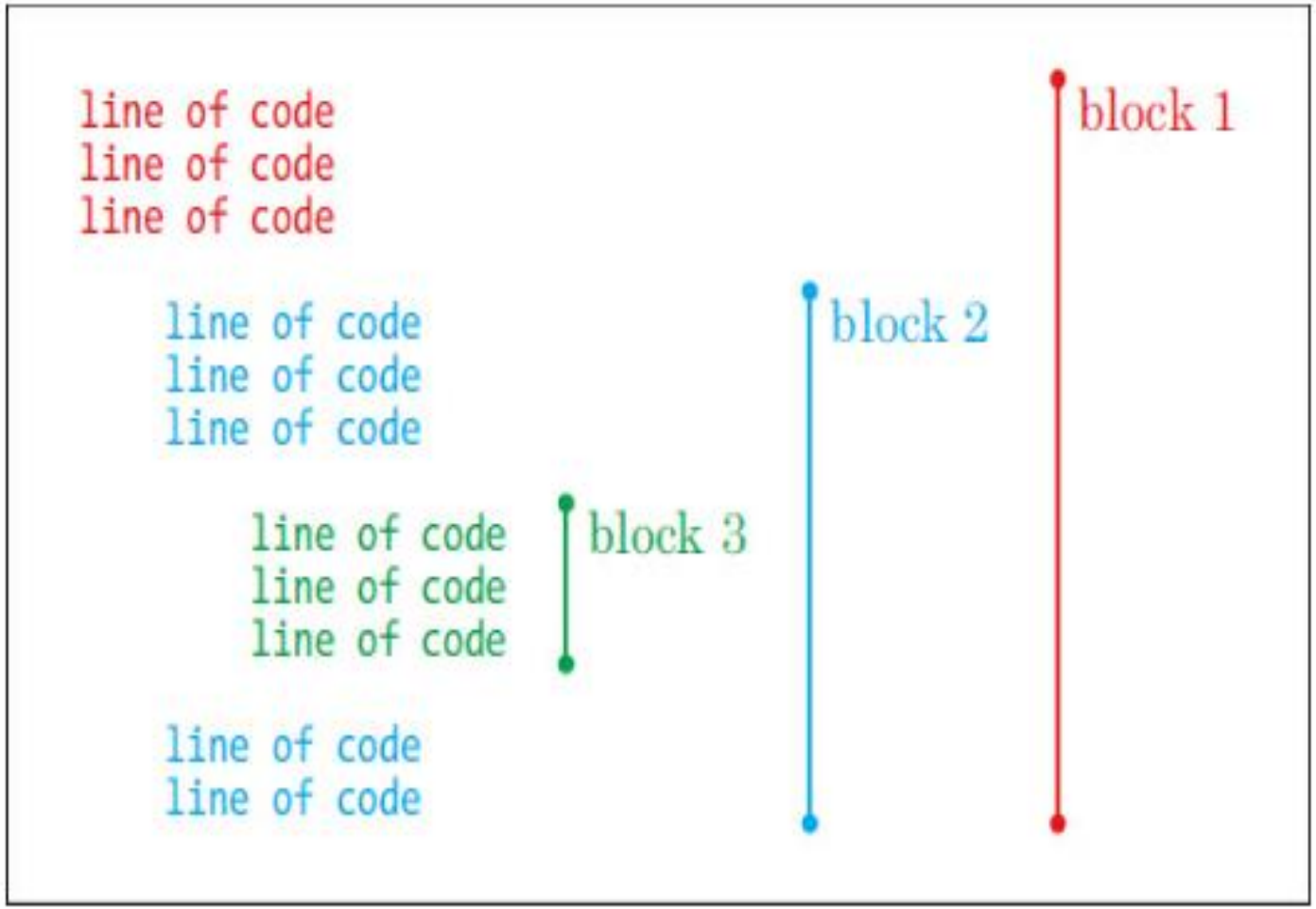
Space equals block

In Python, whitespace, such as a tab (inserted when you press the TAB key) or a space (inserted when you press the spacebar), is meaningful. Code that is at the same position (indented the same number of spaces from the left margin) is grouped into a block, and whenever you start a new line with more spaces than the previous one, you are starting a new block that is part of the previous one, like this:

Block means programming statements!!

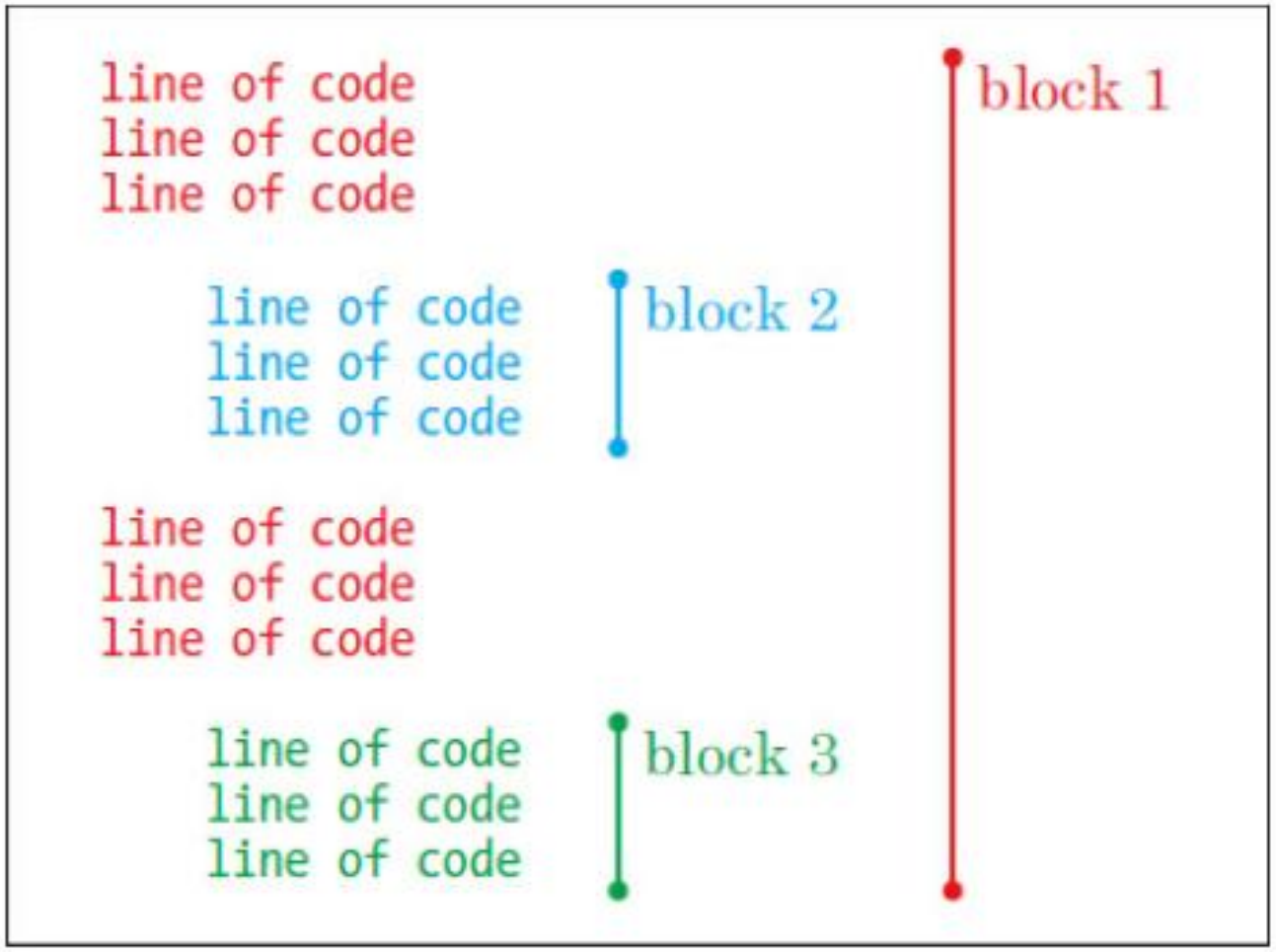
We group statements together into blocks because they are related. The statements need to be run together. When you change the indentation, you're generally creating new blocks.

The following example shows three separate blocks that are created just by changing the indentation.



We group statements together into blocks because they are related. The statements need to be run together. When you change the indentation, you're generally creating new blocks.

The following example shows three separate blocks that are created just by changing the indentation.



Here, even though blocks 2 and 3 have the same indentation, they are considered different blocks because there is a block with less indentation (fewer spaces) between them.

For that matter, a block with four spaces on one line and six spaces on the next will produce an indentation error when you run it, because Python expects you to use the same number of spaces for all the lines in a block.

If the spaces are out of whack

```
age=21
if age > 20:
        print('You are too old!')
            print('Why are you here?')
                print('Why aren\'t you mowing a lawn or sorting papers?')
```

```
  File "<ipython-input-43-255d82701839>", line 4
    print('Why are you here?')
    ^
IndentationError: unexpected indent
```

Points to Remember for spacing

Use consistent spacing to make your code easier to read. If you start writing a program and put four spaces at the beginning of a block, keep using four spaces at the beginning of the other blocks in your program.

Also, be sure to indent each line in the same block with the same number of spaces.

جزاك اللهُ

Please send us your questions at zone@alnafi.com
We will only answer our Nafi Members. So please quote your membership number within the email.

AL NAFI,
A company with a focus on education,
wellbeing and renewable energy.

Python Primer 102b

Asking questions with IF and ELSE

Dua of the day

The righteous dua that Abu Bakr (r) made at the end of his lifetime.

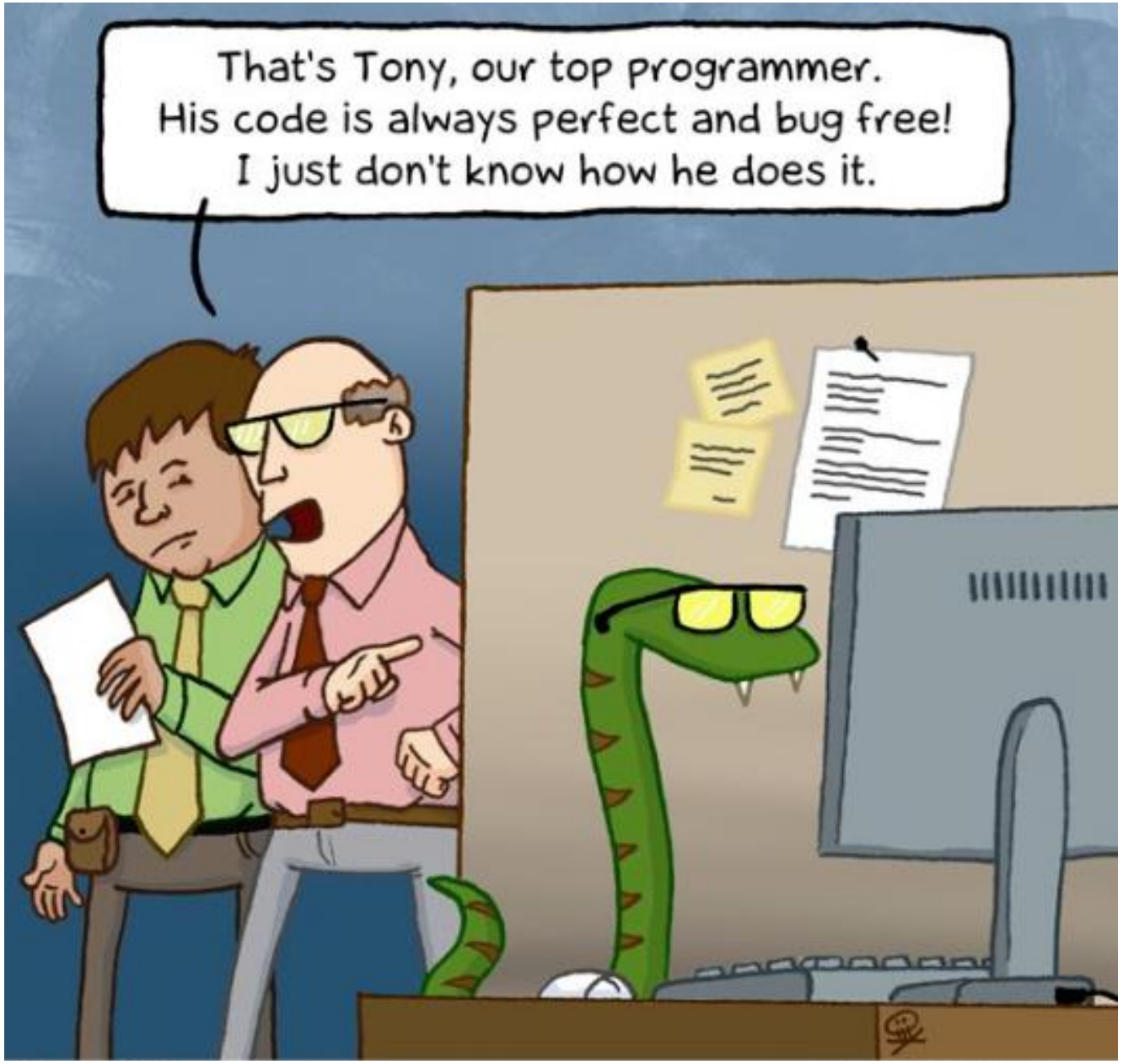
اللَّهُمَّ اجْعَلْ خَيْرَ زَمَانِيْ آخِرَهُ، وَخَيْرَ عَمَلِيْ خَوَاتِمَهُ، وَخَيْرَ أَيَّامِيْ يِوْمَ أَلْقَاكَ

"O Allah, let the best of my lifetime be its ending, and my best deed be that which I seal [my life with], and the best of my days the day I meet You."

Reference: The Dua is taken from Sheikh Omer Sulaiman Dua Compilation.

Study, Rinse and Repeat

- Please subscribe to our YouTube Channel to be on top of your studies.
- Please logon to our website https://alnafi.com/login/
- Use your username and password to logon
- Please keep an eye on zone@alnafi.com emails
- Please review the videos of 30 minutes daily
- Please review the notes daily.
- Please take time for clearing up your mind and reflect on how things are proceeding in your studies.



Conditions in Python

A condition is a programming statement that compares things and tells us whether the criteria set by the comparison are either True (yes) or False (no).

For example, age > 10 is a condition, and is another way of saying, “Is the value of the age variable greater than 10?” This is also a condition: hair_color == ‘black', which is another way of saying, “Is the value of the hair_color variable black?”

We use symbols in Python (called operators) to create our conditions, such as equal to, greater than, and less than.

Conditions in Python

Symbol	Definition
==	Equal to
!=	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

Be sure to use a double equal sign (==) when defining an equal-to condition.

Greater than

```
age=10
if age > 10:
        print ('you are too old for my jokes!')
```

```
age=20
if age > 10:
        print ('you are too old for my jokes!')
```

```
you are too old for my jokes!
```

Equals to

```
age=10
if age == 10:
        print ('you are too old for my jokes!')
```

```
you are too old for my jokes!
```

Try now all the conditions

Symbol	Definition
==	Equal to
!=	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

Else statement example

```
print("Want to hear a dirty joke?")


age = 12
if age == 12:
        print("A pig fell in the mud!")
else:
        print("Shh. It's a secret.")
```

Else statement another example

```
print("Want to hear a dirty joke?")


age = 10
if age == 12:
        print("A pig fell in the mud!")
else:
        print("Shh. It's a secret.")
```

جزاك اللهُ

Please send us your questions at zone@alnafi.com
We will only answer our Nafi Members. So please quote your membership number within the email.

AL NAFI,
A company with a focus on education,
wellbeing and renewable energy.

Python Primer 102c

Asking questions with IF and ELSE

Dua of the day

The righteous dua that Abu Bakr (r) made at the end of his lifetime.

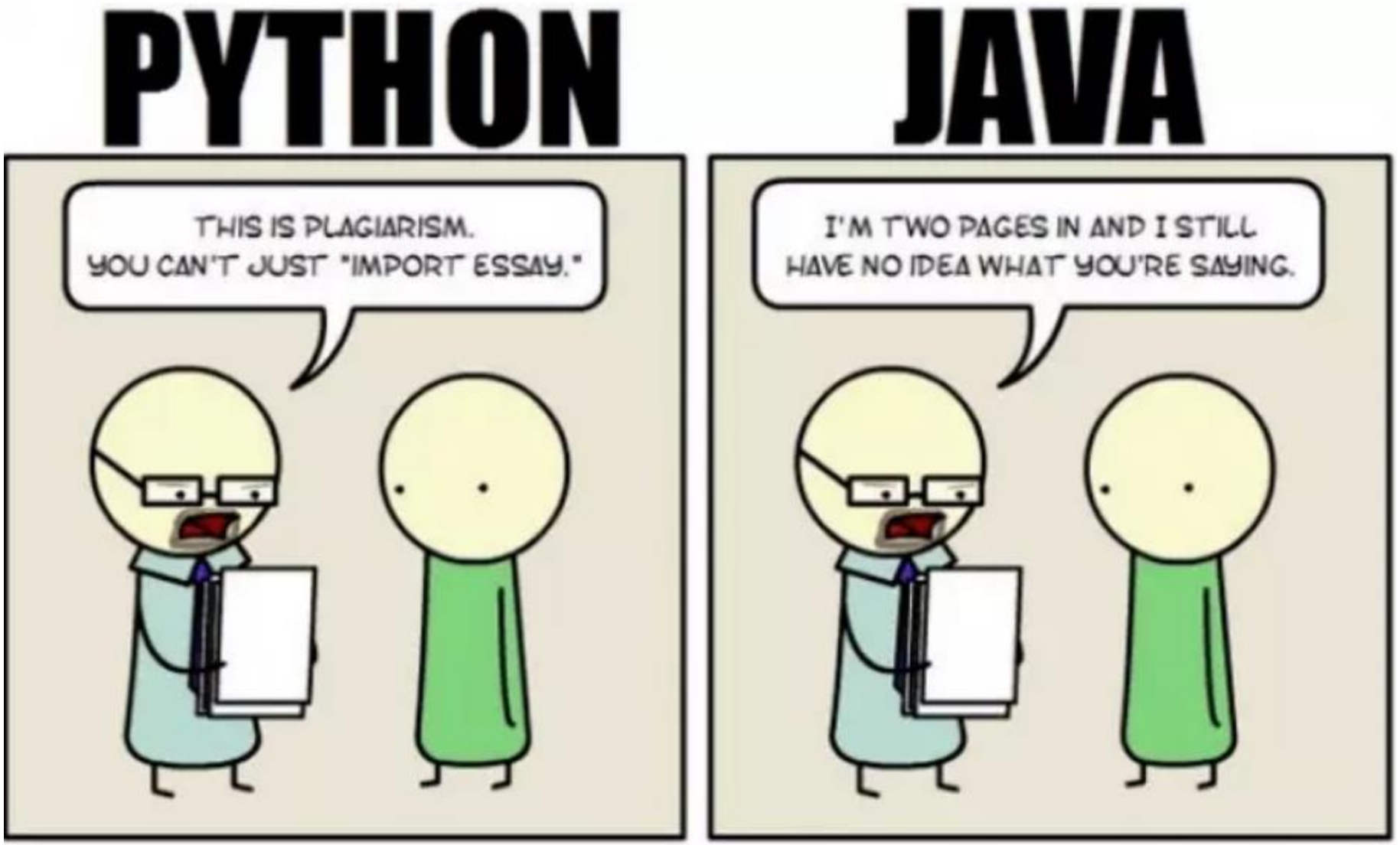
اللَّهُمَّ اجْعَلْ خَيْرَ زَمَانِيْ آخِرَهُ، وَخَيْرَ عَمَلِيْ خَوَاتِمَهُ، وَخَيْرَ أَيَّامِيْ يِوْمَ أَلْقَاكَ

"O Allah, let the best of my lifetime be its ending, and my best deed be that which I seal [my life with], and the best of my days the day I meet You."

Reference: The Dua is taken from Sheikh Omer Sulaiman Dua Compilation.

Study, Rinse and Repeat

- Please subscribe to our YouTube Channel to be on top of your studies.
- Please logon to our website https://alnafi.com/login/
- Use your username and password to logon
- Please keep an eye on zone@alnafi.com emails
- Please review the videos of 30 minutes daily
- Please review the notes daily.
- Please take time for clearing up your mind and reflect on how things are proceeding in your studies.



IF AND ELIF (else-if) STATEMENTS

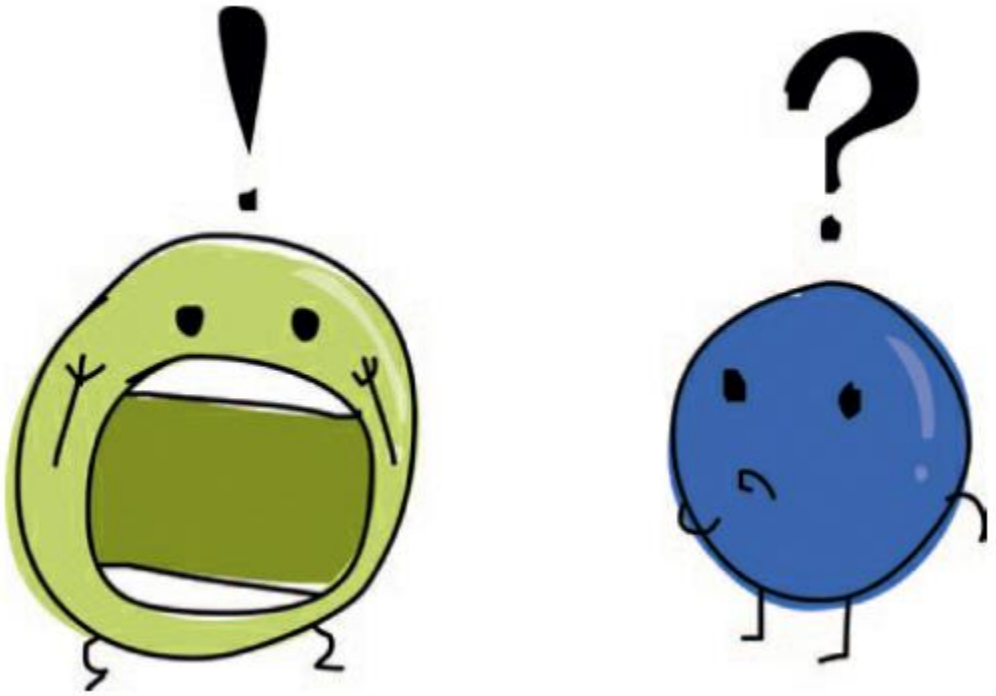
We can extend an if statement even further with elif (which is short for else-if).

For example, we can check if a person's age is 10, 11, or 12 (and so on) and have our program do something different based on the answer.

These statements are different from if-then-else statements in that there can be more than one elif in the same statement:

```
age = 12
if age == 10:
    print("What do you call an unhappy cranberry?")
    print("A blueberry!")
elif age == 11:
    print("What did the green grape say to the blue grape?")
    print("Breathe! Breathe!")
elif age == 12:
    print("What did 0 say to 8?")
    print("Hi guys!")
elif age == 13:
    print("Why wasn't 10 afraid of 7?")
    print("Because rather than eating 9, 7 8 pi.")
else:
    print("Huh?")
```

Combining Conditions



You can combine conditions by using the keywords and and or, which produces shorter and simpler code. Here's an example of using or:

```
age=10

if age == 10 or age == 11 or age == 12 or age == 13:

    print('What is 13 + 49 + 84 + 155 + 97? A headache!')
else:
    print('Huh?')
```

```
age=15

if age == 10 or age == 11 or age == 12 or age == 13:

    print('What is 13 + 49 + 84 + 155 + 97? A headache!')
else:
    print('Huh?')
```

Using greater than or equal-to operator (>=) and less-than-equal-to operator(<=)

```
age=10

if age >= 10 and age <= 13:

    print('What is 13 + 49 + 84 + 155 + 97? A headache!')

else:
    print('Huh?')
```

```
age=14

if age >= 10 and age <= 13:

    print('What is 13 + 49 + 84 + 155 + 97? A headache!')

else:
    print('Huh?')
```

Variables with no Value-None

Just as we can assign numbers, strings, and lists to a variable, we can also assign nothing, or an empty value, to a variable. In Python, an empty value is referred to as **None**, and it is the absence of value.

And it’s important to note that the value None is different from the value 0 because it is the absence of a value, rather than a number with a value of 0. The only value that a variable has when we give it the empty value None is nothing. Here’s an example:

None variable example

```
myval = None
if myval == None:
    print("The variable myval doesn't have a value")
```

This is useful when you only want to calculate a value for a variable if it hasn’t already been calculated.

جزاك اللهُ

Please send us your questions at zone@alnafi.com
We will only answer our Nafi Members. So please quote your membership number within the email.

AL NAFI,
A company with a focus on education,
wellbeing and renewable energy.

Python Primer 102d

Asking questions with IF and ELSE

Dua of the day

The righteous dua that Abu Bakr (r) made at the end of his lifetime.

اللَّهُمَّ اجْعَلْ خَيْرَ زَمَانِيْ آخِرَهُ، وَخَيْرَ عَمَلِيْ خَوَاتِمَهُ، وَخَيْرَ أَيَّامِيْ يِوْمَ أَلْقَاكَ

"O Allah, let the best of my lifetime be its ending, and my best deed be that which I seal [my life with], and the best of my days the day I meet You."

Reference: The Dua is taken from Sheikh Omer Sulaiman Dua Compilation.

Study, Rinse and Repeat

- Please subscribe to our YouTube Channel to be on top of your studies.
- Please logon to our website https://alnafi.com/login/
- Use your username and password to logon
- Please keep an eye on zone@alnafi.com emails
- Please review the videos of 30 minutes daily
- Please review the notes daily.
- Please take time for clearing up your mind and reflect on how things are proceeding in your studies.

Difference between Strings and numbers

User input is what a person enters on the keyboard—whether that's a character, a pressed arrow or ENTER key, or anything else.

User input comes into Python as a string, which means that when you type the number 10 on your keyboard, Python saves the number 10 into a variable as a string, not a number.

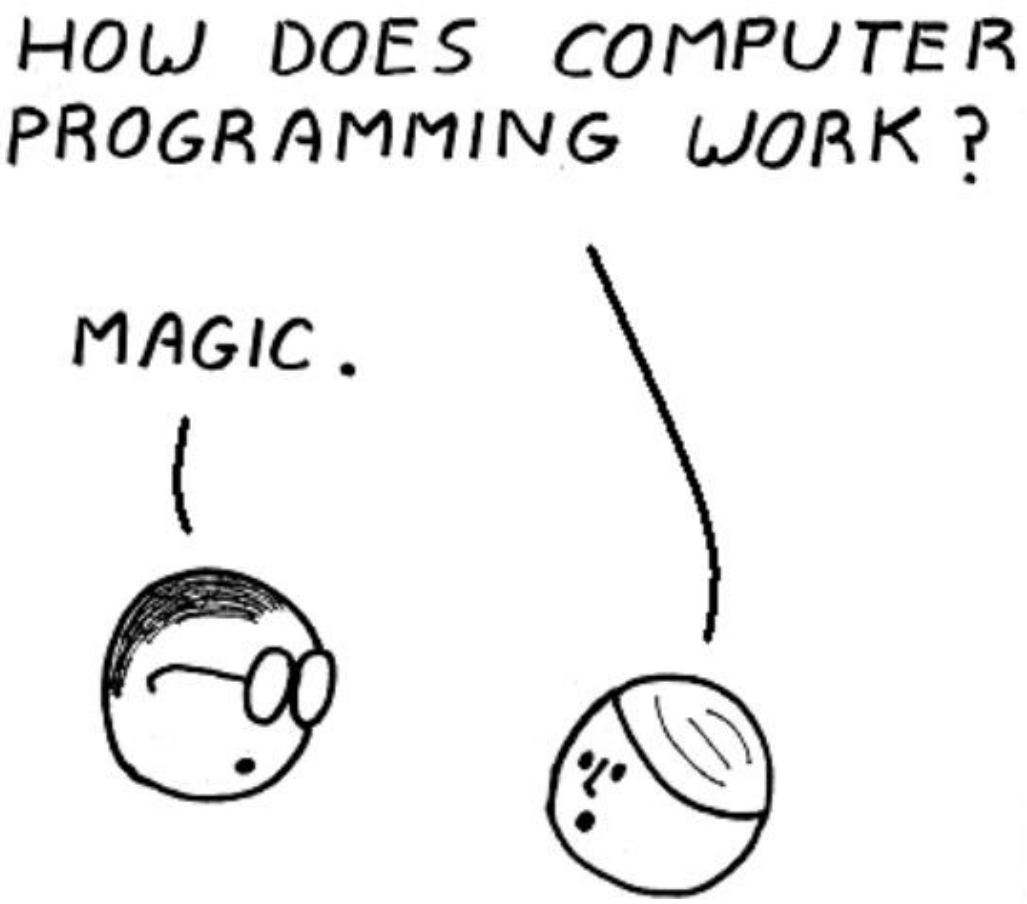
What’s the difference between the number 10 and the string '10'? Both look the same to us, with the only difference being that one is surrounded by quotes.

But to a computer, the two are very different. For example, suppose that we compare the value of the variable age to a number in an if statement, like this:

```
age=10
if age == 10:
    print("What's the best way to speak to a monster?")
    print("From as far away as possible!")
```

```
age=’10’
if age == 10:
    print("What's the best way to speak to a monster?")
    print("From as far away as possible!")
```

Python Magic



String into a number using int

Converting variables into numbers and numbers into variables

For example, you can convert the string '10' into a number with int:

```
age = '10'
converted_age = int(age)
```

Number into a string using str

The variable converted_age would now hold the number 10.

To convert a number into a string, use str:

```
age = 10
converted_age = str(age)
```

Example

```
age = '10'
converted_age = int(age)
if converted_age == 10:
    print("What's the best way to speak to a monster?")
    print("From as far away as possible!")
```

What if we use a number with a decimal ?

```
age = '10.5'
converted_age = int(age)
if converted_age == 10:
    print("What's the best way to speak to a monster?")
    print("From as far away as possible!")
```

For decimal numbers use **float**

A ValueError is what Python uses to tell you that the value you have tried to use isn’t appropriate.

To fix this, use the function float instead of int. The float function can handle numbers that aren’t integers.

```
age = '10.5'
converted_age = float(age)
if converted_age == 10:
    print("What's the best way to speak to a monster?")
    print("From as far away as possible!")
```

جزاك اللهُ

Please send us your questions at zone@alnafi.com
We will only answer our Nafi Members. So please quote your membership number within the email.

AL NAFI,
A company with a focus on education,
wellbeing and renewable energy.

Python Primer 103

Loops

Dua of the day

The righteous dua that Abu Bakr (r) made at the end of his lifetime.

اللَّهُمَّ اجْعَلْ خَيْرَ زَمَانِيْ آخِرَهُ، وَخَيْرَ عَمَلِيْ خَوَاتِمَهُ، وَخَيْرَ أَيَّامِيْ يِوْمَ أَلْقَاكَ

"O Allah, let the best of my lifetime be its ending, and my best deed be that which I seal [my life with], and the best of my days the day I meet You."

Reference: The Dua is taken from Sheikh Omer Sulaiman Dua Compilation.

Study, Rinse and Repeat

- Please subscribe to our YouTube Channel to be on top of your studies.
- Please logon to our website https://alnafi.com/login/
- Use your username and password to logon
- Please keep an eye on zone@alnafi.com emails
- Please review the videos of 30 minutes daily
- Please review the notes daily.
- Please take time for clearing up your mind and reflect on how things are proceeding in your studies.

Using Loops

To print Salam five times in Python, you could do the following:

```
print(“Salam")
print(“Salam")
print(“Salam")
print(“Salam")
print(“Salam")
```

Python Loops

Use **for** and **while** for creating loops

The **Range** function can be used to create a list of numbers ranging from a starting number up to the number just before the ending number.

That may sound a little confusing. Let's combine the range function with the list function to see exactly how this works.

But this is rather tedious. Instead, you can use a for loop to reduce the amount of typing and repetition, like this:

```
for x in range(0, 5):
    print('salam')
```

Using range and list together

```
print(list(range(10, 20)))
```

In the case of the for loop, the code at is actually telling Python to do the following:

Start counting from 0 and stop before reaching 5.

For each number we count, store the value in the variable x.

```
for x in range(0, 5):
    print('salam %s' %x)
```

If we remove the for then what happens!

```
x = 0
print('salam %s' % x)
x = 1
print('salam %s' % x)
x = 2
print('salam %s' % x)
x = 3
print('salam %s' % x)
x = 4
print('salam %s' % x)
```

Not using range and list

```
wizard_list = ['spider legs','toe of frog', 'snail tongue',
               'bat wing', 'slug butter', 'bear burp']
for i in wizard_list:
    print(i)
```

Without the loop we have a long code

```
wizard_list = ['spider legs', 'toe of frog', 'snail tongue',
               'bat wing', 'slug butter', 'bear burp']
print(wizard_list[0])
print(wizard_list[1])
print(wizard_list[2])
print(wizard_list[3])
print(wizard_list[4])
print(wizard_list[5])
```

Printing it twice

```
hugehairypants = ['huge', 'hairy', 'pants']
for i in hugehairypants:
        print(i)
        print(i)
```

Mind the gap

```
hugehairypants = ['huge', 'hairy', 'pants']
for i in hugehairypants:
        print(i)
         print(i)
```

Gaps or spaces needs to be consistent

```
hugehairypants = ['huge', 'hairy', 'pants']
for i in hugehairypants:
        print(i)
        for j in hugehairypants:
                print(j)
```

Another kind of loop while

A for loop isn't the only kind of loop you can make in Python. There's also the while loop.

A for loop is a loop of a specific length, whereas a while loop is a loop that is used when you don't know ahead of time when it needs to stop looping.

```
x = 45
y = 80
while x < 50 and y < 100:
        x = x + 1
        y = y + 1
        print(x, y)
```

جزاك اللهُ

Please send us your questions at zone@alnafi.com
We will only answer our Nafi Members. So please quote your membership number within the email.

AL NAFI,
A company with a focus on education,
wellbeing and renewable energy.

Python Primer 104

Code and Functions Modules

Dua of the day

The righteous dua that Abu Bakr (r) made at the end of his lifetime.

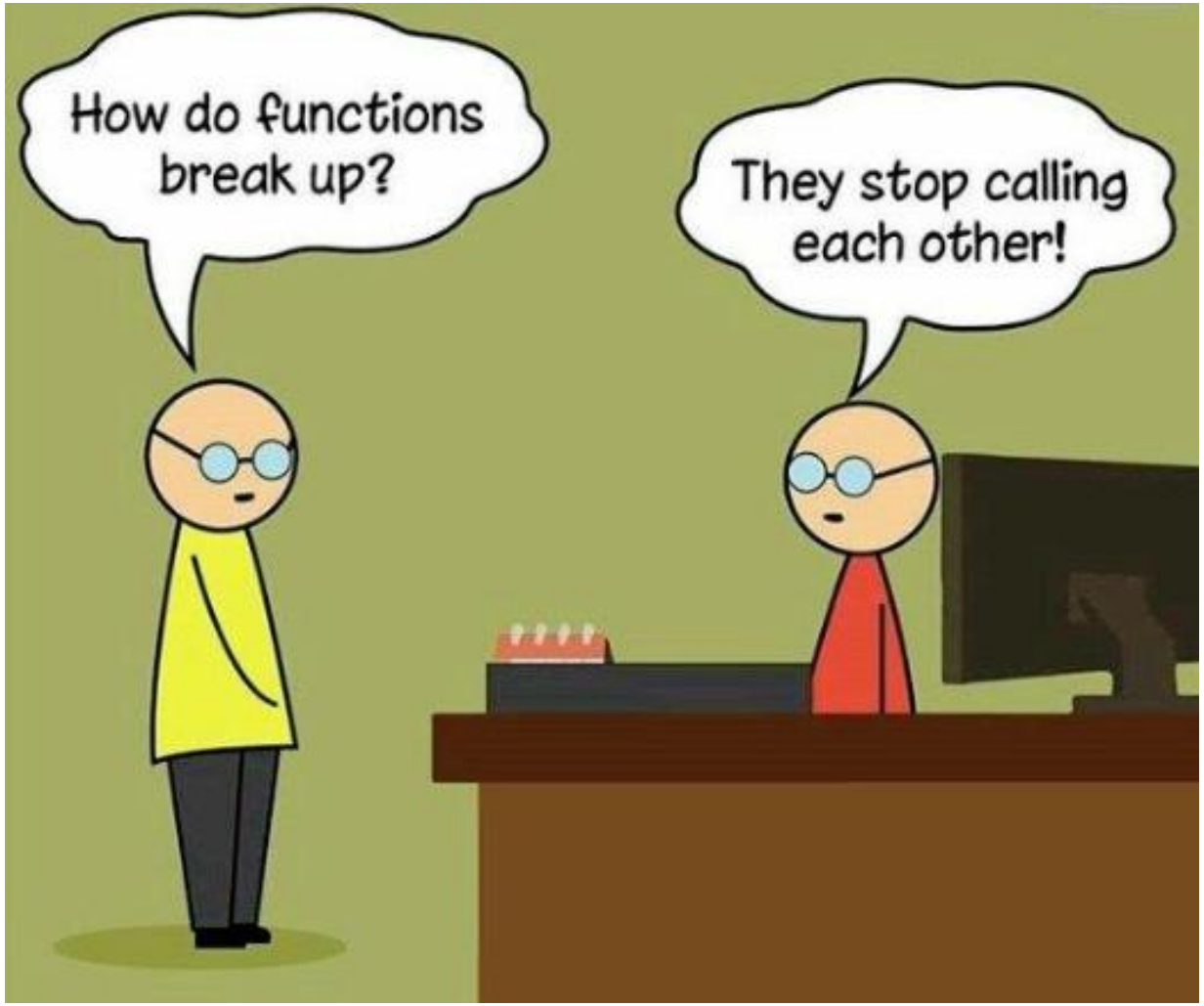
اللَّهُمَّ اجْعَلْ خَيْرَ زَمَانِيْ آخِرَهُ، وَخَيْرَ عَمَلِيْ خَوَاتِمَهُ، وَخَيْرَ أَيَّامِيْ يِوْمَ أَلْقَاكَ

"O Allah, let the best of my lifetime be its ending, and my best deed be that which I seal [my life with], and the best of my days the day I meet You."

Reference: The Dua is taken from Sheikh Omer Sulaiman Dua Compilation.

Study, Rinse and Repeat

- Please subscribe to our YouTube Channel to be on top of your studies.
- Please logon to our website https://alnafi.com/login/
- Use your username and password to logon
- Please keep an eye on zone@alnafi.com emails
- Please review the videos of 30 minutes daily
- Please review the notes daily.
- Please take time for clearing up your mind and reflect on how things are proceeding in your studies.



Recycling code functions and modules

Talk about recycling

In the programming world, reuse is just as important. Obviously, your program won't disappear under a pile of garbage, but if you don't reuse some of what you're doing, you'll eventually wear your fingers down to painful stubs through overtyping. Reuse also makes your code shorter and easier to read.

Functions Come handy ☺

Using list and range

list(range(0, 5))

Or try another example

list(range(0, 1000))

Why functions?

When you're writing simple programs, functions are handy. Once you start writing long, more complicated programs, like machine learning, deep learning, AI, and games functions are essential (assuming you want to finish writing your program this century).

Parts of Functions

```
testfunc('mary')
def testfunc(myname):
        print('hello %s' % myname)
```

Multiple ways for writing functions

```
testfunc('Mohammad', 'Ali')

def testfunc(fname, lname):
testfunc('Mohammad', 'Ali')

        print('Hello %s %s' % (fname, lname))
```

Creating variables

```
firstname = 'Mohammad'
lastname = 'Omar'
testfunc(firstname, lastname)
print (firstname, lastname)
```

Using return

```
def variable_test ():
        first_variable = 10
        second_variable = 20
        return first_variable * second_variable
print(variable_test())
```

Variable Outside the functions

```
another_variable = 100
def variable_test2():
    first_variable = 10
    second_variable = 20
    return first_variable * second_variable * another_variable

print(variable_test2())
```

Python Modules

Python - **Modules**. ... A **module** is a **Python** object with arbitrarily named attributes that you can bind and reference. Simply, a **module** is a file consisting of **Python**code. A **module** can define functions, classes and variables. A **module** can also include runnable code.

OR

Modules are used to group functions, variables, and other things together into larger, more powerful programs. Some modules are built into Python, and you can download other modules separately.

Modules examples

```
import time
print(time.asctime())
```

Here, the import command is used to tell Python that we want to use the module time.

The function asctime is a part of the time module that returns the current date and time, as a string.

Another module example

```
age=11
if age >= 10 and age <= 13:
    print('What is 13 + 49 + 84 + 155 + 97? A headache!')
else:
    print('Huh?')
```

جزاك اللهُ

Please send us your questions at zone@alnafi.com
We will only answer our Nafi Members. So please quote your membership number within the email.

AL NAFI,
A company with a focus on education,
wellbeing and renewable energy.

Python Primer 105

Python Classes and Objects

Dua of the day

The righteous dua that Abu Bakr (r) made at the end of his lifetime.

اللَّهُمَّ اجْعَلْ خَيْرَ زَمَانِيْ آخِرَهُ، وَخَيْرَ عَمَلِيْ خَوَاتِمَهُ، وَخَيْرَ أَيَّامِيْ يِوْمَ أَلْقَاكَ

"O Allah, let the best of my lifetime be its ending, and my best deed be that which I seal [my life with], and the best of my days the day I meet You."

Reference: The Dua is taken from Sheikh Omer Sulaiman Dua Compilation.

Study, Rinse and Repeat

- Please subscribe to our YouTube Channel to be on top of your studies.
- Please logon to our website https://alnafi.com/login/
- Use your username and password to logon
- Please keep an eye on zone@alnafi.com emails
- Please review the videos of 30 minutes daily
- Please review the notes daily.
- Please take time for clearing up your mind and reflect on how things are proceeding in your studies.

Objects in Python

Objects are a way of organizing code in a program and breaking things down to make it easier to think about complex ideas.

Objects explanation

A giraffe is a type of mammal, which is a type of animal. A giraffe is also an animate object—it's alive. Now consider a sidewalk.

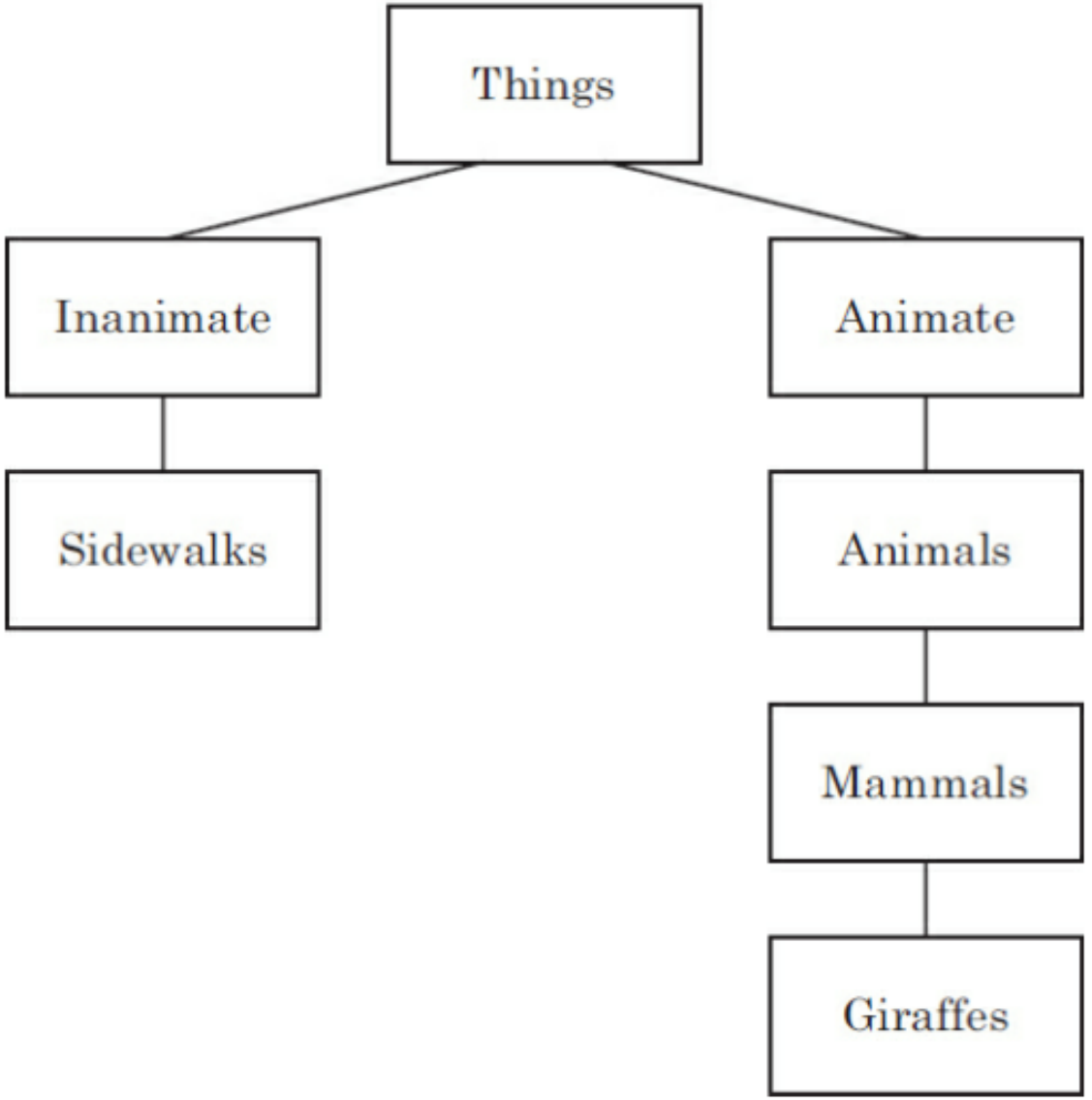
There's not much to say about a sidewalk other than it's not a living thing. Let's call it an inanimate object (in other words, it's not alive).

The terms mammal, animal, animate, and inanimate are all ways of classifying things.

Objects into classes

In Python, objects are defined by classes, which we can think of as a way to classify objects into groups.

Here is a tree diagram of the classes that giraffes and sidewalks would fit into based on our preceding definitions:



Creating a class

```
class Things:
    pass
class Inanimate(Things):
    pass
class Animate(Things):
    pass
class Sidewalks(Inanimate):
    pass
class Animals(Animate):
    pass
class Mammals(Animals):
    pass
class Giraffes(Mammals):
    pass
```

Adding objects to classes

Qamoo = Giraffes()

Defining functions of Classes

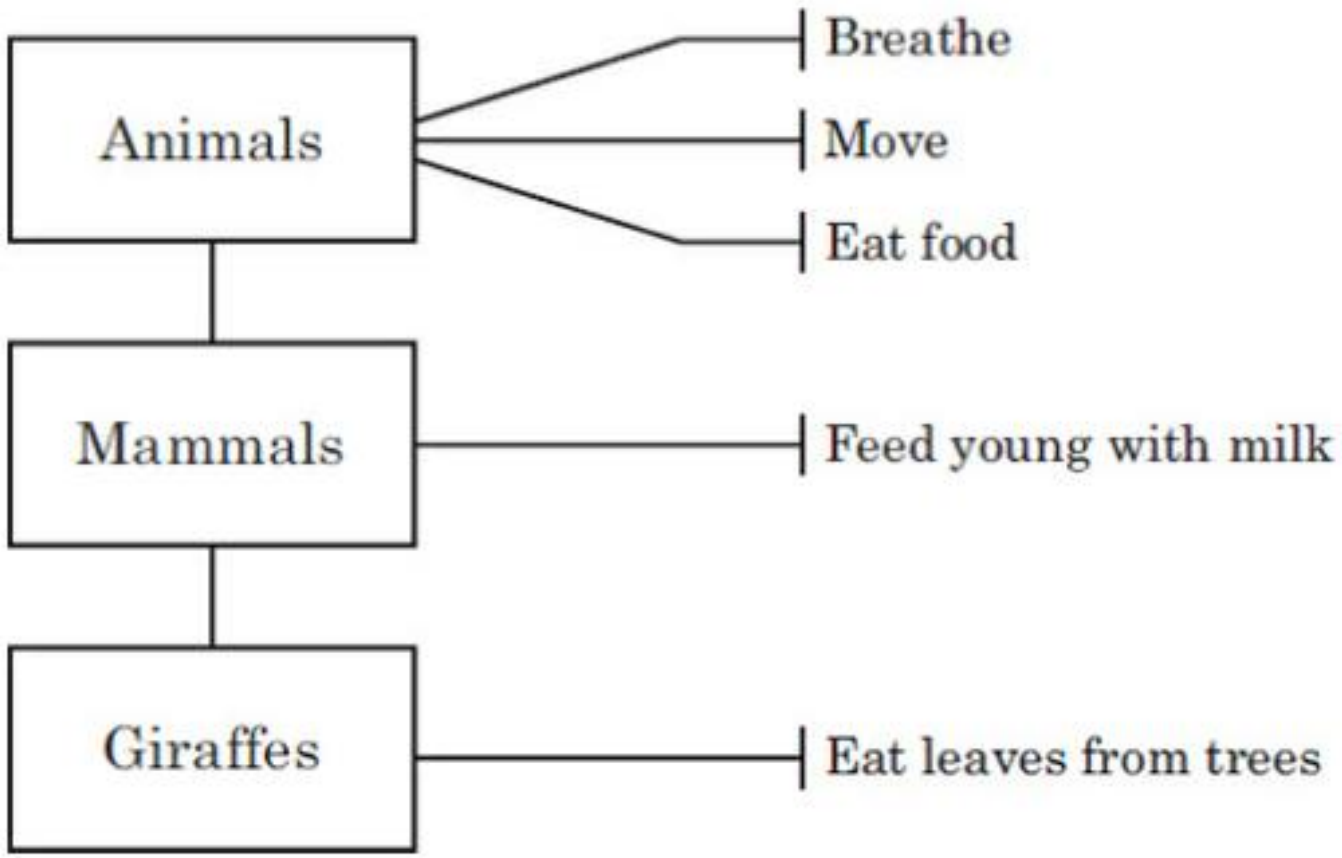
Functions are a way to reuse code ☺

code. When we define a function that is associated with a class, we do so in the same way that we define any other function, except that we indent it beneath the class definition. For example, here's a normal function that isn't associated with a class:

```
def this_is_a_normal_function():
    print('I am a normal function')

class ThisIsMySillyClass:
    def this_is_a_class_function():
        print('I am a class function')
    def this_is_also_a_class_function():
        print('I am also a class function. See?')
```

Adding class characteristics as functions

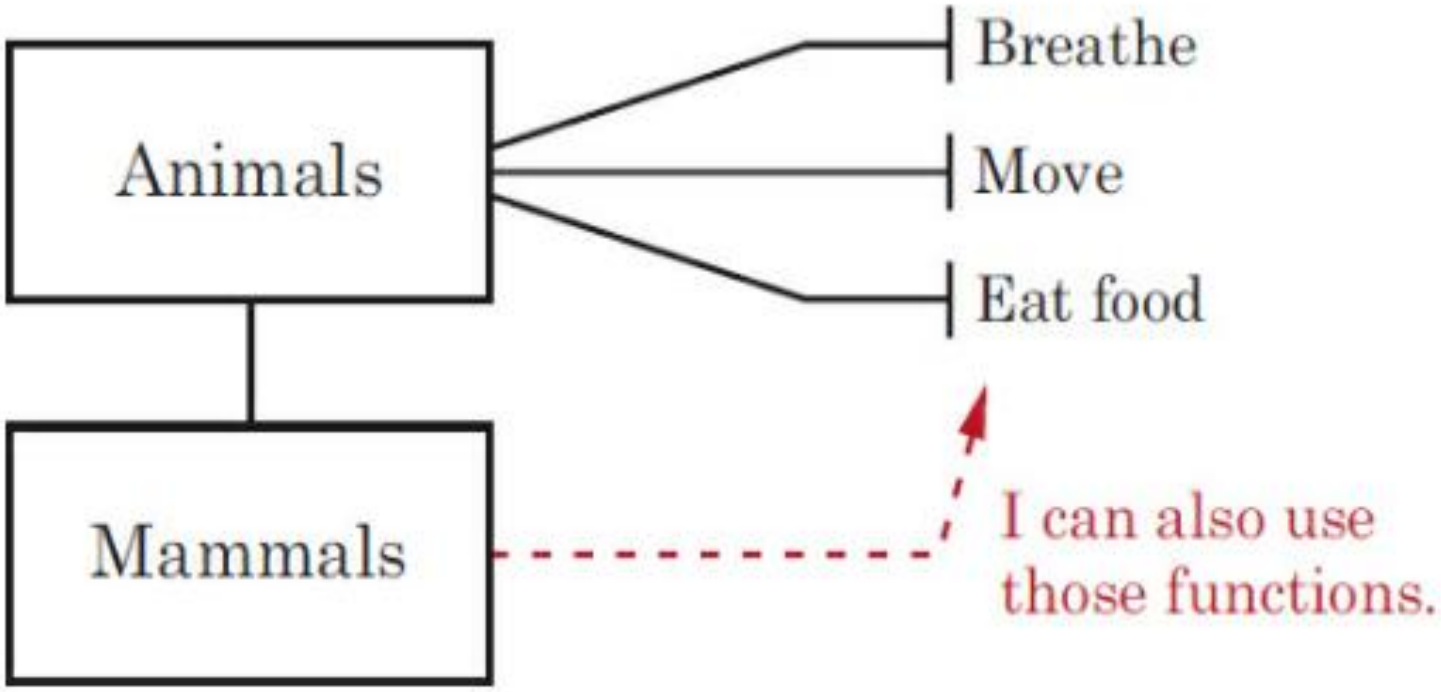


```
class Animals(Animate):
    def breathe(self):
        pass
    def move(self):
        pass
    def eat_food(self):
        pass
```

Why use classes and objects?

We’ve now added functions to our classes, but why use classes and objects at all, when you could just write normal functions called breathe, move, eat_food, and so on?

Inherited Functions



Functions calling other functions

We have now created a function that combines two other functions, which is quite common in programming.

Often, you will write a function that does something useful, which you can then use inside another function.

Initializing an object

Sometimes when creating an object, we want to set some values (also called properties) for later use. When we initialize an object, we are getting it ready to be used.

جزاك اللهُ

Please send us your questions at zone@alnafi.com We will only answer our Nafi Members. So please quote your membership number within the email.

Soon we will completely move to the portal to answer any further questions and will remove the email answering fuction.

AL NAFI,
A company with a focus on education,
wellbeing and renewable energy.

Python Primer 106

Python build-in functions

Dua of the day

The righteous dua that Abu Bakr (r) made at the end of his lifetime.

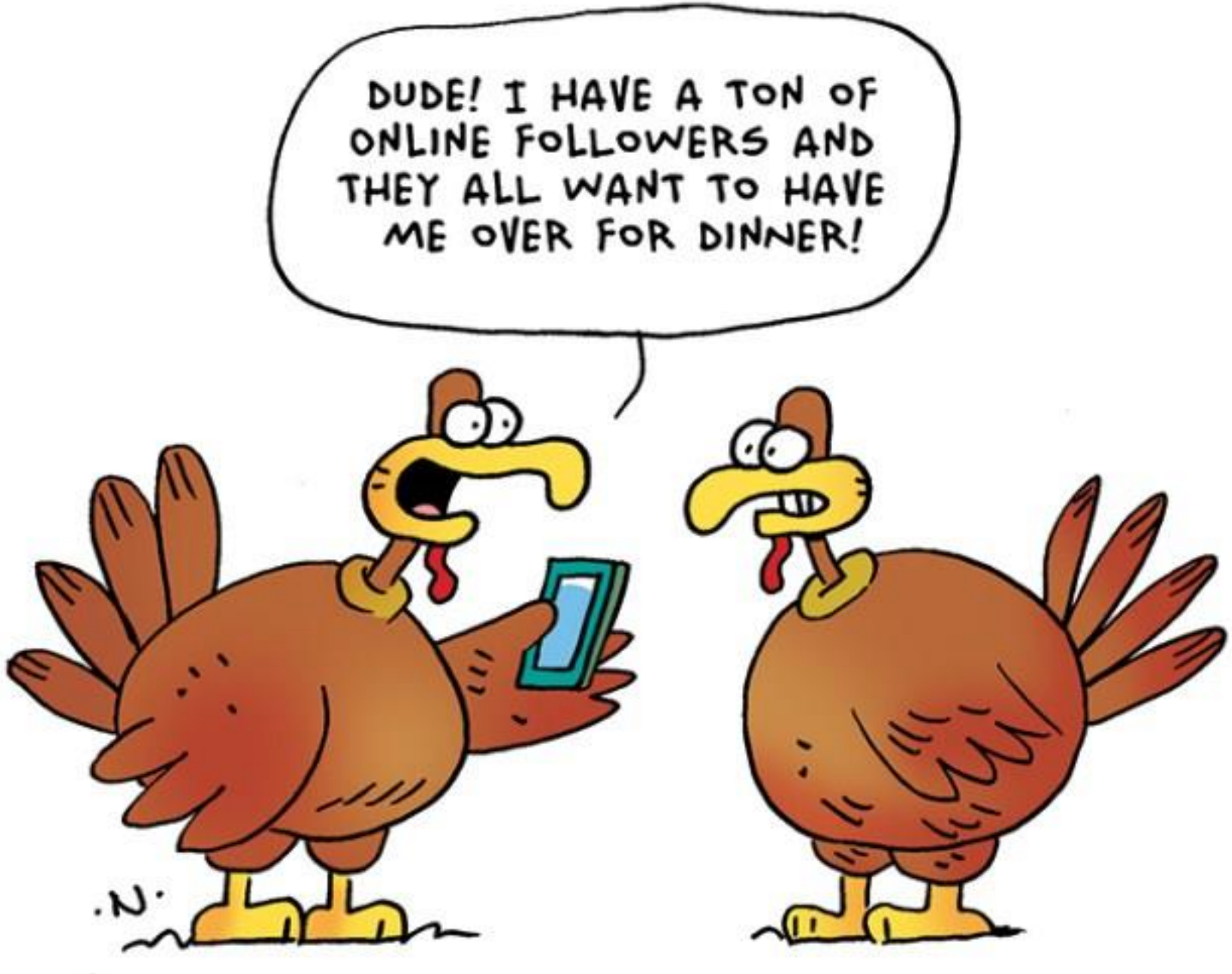
اللَّهُمَّ اجْعَلْ خَيْرَ زَمَانِيْ آخِرَهُ، وَخَيْرَ عَمَلِيْ خَوَاتِمَهُ، وَخَيْرَ أَيَّامِيْ يِوْمَ أَلْقَاكَ

"O Allah, let the best of my lifetime be its ending, and my best deed be that which I seal [my life with], and the best of my days the day I meet You."

Reference: The Dua is taken from Sheikh Omer Sulaiman Dua Compilation.

Study, Rinse and Repeat

- Please subscribe to our YouTube Channel to be on top of your studies.
- Please logon to our website https://alnafi.com/login/
- Use your username and password to logon
- Please keep an eye on zone@alnafi.com emails
- Please review the videos of 30 minutes daily
- Please review the notes daily.
- Please take time for clearing up your mind and reflect on how things are proceeding in your studies.



Python Build-in Functions

Python has number of built in functions.

And they are always available without, importing any external python code via modules or packages.

There are around 68 build-in functions.

Python Build-in Function's

		Built-in Functions		
abs()	dict()	help()	min()	setattr()
all()	dir()	hex()	next()	slice()
any()	divmod()	id()	object()	sorted()
ascii()	enumerate()	input()	oct()	staticmethod()
bin()	eval()	int()	open()	str()
bool()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	
delattr()	hash()	memoryview()	set()	

Python abs

The syntax of abs() method is:

abs(num)

Python abs ()

abs() Parameters

The abs() method takes a single argument:

num - number whose absolute value is to be returned. The number can be:

- integer
- floating number
- complex number

Return value from abs()

- The abs() method returns the absolute value of the given number.
- For integers - integer absolute value is returned
- For floating numbers - floating absolute value is returned
- For complex numbers - magnitude of the number is returned

abs examples

```
#random Integer
integer = -20
print('Absolute value of -20 is:', abs(integer))

#random floating number
floating = -30.33
print('Absolute value of -30.33 is:', abs(floating))
```

Python bool()

The bool() method converts a value to Boolean (True or False) using the standard truth testing procedure.

bool([value])

bool() parameters

It's not mandatory to pass a value to bool(). If you do not pass a value, bool() returns False.

In general use, bool() takes a single parameter value.

Return value for bool()

The bool() returns:

False if the value is omitted or false

True if the value is true

what in bool is considered **false**

The following values are considered false in Python:

None

False

Zero of any numeric type. For example, 0, 0.0, 0j

Empty sequence. For example, (), [], ''.

Empty mapping. For example, {}

objects of Classes which has __bool__() or __len()__ method which returns 0 or False

All other values except these values are considered true.

bool examples

```
test = []
print(test,'is',bool(test))

test = [0]
print(test,'is',bool(test))

test = 0.0
print(test,'is',bool(test))
```

```
test = None
print(test,'is',bool(test))

test = True
print(test,'is',bool(test))

test = 'Easy string'
print(test,'is',bool(test))
```

Python dir function

The dir() method tries to return a list of valid attributes of the object.

The syntax of dir() is:

dir([object])

dir() Parameters

The dir() takes maximum of one object.

object (optional) - dir() attempts to return all attributes of this object.

Return Value from dir()

The dir() tries to return a list of valid attributes of the object.

If the object has __dir__() method, the method will be called and must return the list of attributes.

If the object doesn't have __dir__() method, this method tries to find information from the __dict__ attribute (if defined), and from type object. In this case, the list returned from dir() may not be complete.

If object is not passed to the dir() method, it returns the list of names in the current local scope.

Example 1: How dir() works?

```
number = [1, 2, 3]
print(dir(number))

print('\nReturn Value from empty dir()')
print(dir())
```

Covering all the build-in functions in Python

We are creating 10 different videos to cover all the build-in functions in a detailed manner which will become an add-on to the python primer.

The videos will be shared over the weekend inshAllah.

جزاك اللهُ

Please send us your questions at zone@alnafi.com We will only answer our Nafi Members. So please quote your membership number within the email.

Soon we will completely move to the portal to answer any further questions and will remove the email answering fuction.