1- Calculations Basic python operators

The order of Operations

```
In [2]: 15 + 30 * 5
```

```
Out[2]: 165
```

the numbers 30 and 20 are multiplied first, and the number 5 is added to their product.

Using Parentheses()

```
In [3]:  (15 + 30) * 5
```

```
Out[3]: 225
```

The parentheses tell Python to do the operation in the parentheses first, and then do the operation outside the parentheses.

Nested Parentheses

```
In [4]: ((15+30) * 5) /10
```

```
Out[4]: 22.5
```

In this case, Python evaluates the innermost parentheses first, then the outer ones, and then the final division operato

2- 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

In [7]:
```
Ali = 100
Ali
```

Out[7]:
```
100
```

Variable names can be made up of letters, numbers and the underscore character (_), but they cant start with a number

In [8]:
```
Uzair_rupees = 30
fahad_rupees = 20
ahmad_rupees = 10
Uzair_rupees + fahad_rupees + ahmad_rupees
```

Out[8]:
```
60
```

3-Strings in Python

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.

In [9]:
```
Uzair = "pharmacist"
print(Uzair)
```

```
pharmacist
```

But you cannot start with single quote and end with double quote. it should be consistent

In [13]:
```
Uzair ='''pharmacist
abbot
pharma'''
Uzair
```

Out[13]:
```
'pharmacist\nabbot\npharma'
```

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

Strings Problems handling

In [14]:
```
string = 'He said, "Aren't can't shouldn't wouldn't."'
print(string)
```

```
  File "C:\Users\EUROTE~1\AppData\Local\Temp/ipykernel_10324/1190097497.py", line 1
    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

In [15]:
```
string1 = '''He said, "Aren't can't shouldn't wouldn't."'''
print(string1)
```

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

Adding a backslash \ or escaping

In [16]:
```
uzi_quote_str = 'He said, "Aren\'t can\'t shouldn\'t wouldn\'t."'
print (uzi_quote_str)
```

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

In [17]:
```
double_quote_str = "He said, \"Aren't can't shouldn't wouldn't.\""
print (double_quote_str)
```

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

Embedding value in strings

(Embedding values, also referred to as string substitution, is programmer-speak for “inserting values.”)

In [19]:
```
runs = 100
message = 'I scored %s points'
print(message % runs)
```

```
I scored 100 points
```

In [21]:
```
nums = 'What did the number %s say to the number %s?'
print(nums % (20, 8))
```

```
What did the number 20 say to the number 8?
```

Multiplying Strings

In [24]:
```
print(10 * 'pharmacy')
```

```
pharmacypharmacypharmacypharmacypharmacypharmacypharmacypharmacypharmacypharmacy
```

Creating a list

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.

In [1]:
```
Mylist =['apple', 'mango', 'aloo', 'chai''']
print(Mylist[2])
```

```
aloo
```

In [5]:
```
# For example we can print the second item in the list.
print(Mylist[1])
```

```
mango
```

In [6]:
```
print(Mylist[0:3])
```

```
['apple', 'mango', '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.

In [8]:
```
# Lists can store all sorts of items, like numbers
my_numbers = [1,23,45,65]
my_numbers
```

Out[8]:
```
[1, 23, 45, 65]
```

In [10]:
```
# They can also hold strings:
uzair_string = ['Which', 'Witch', 'Is', 'Which']
uzair_string
```

Out[10]:
```
['Which', 'Witch', 'Is', 'Which']
```

In [11]:
```
numbers_and_strings = ['Why', 'was', 34, 'afraid', 'of', 54, 'because', 7, 8, 9]
print(numbers_and_strings)
```

```
['Why', 'was', 34, 'afraid', 'of', 54, 'because', 7, 8, 9]
```

In [12]:
```
# List within a list(nested 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']]
```

In [15]:
```
# Adding items to a list
# Before adding an item to a list:
python_list = ['ahmad', 'ali', 'maria', 'husnain', 'abubakar', 'amina']
print(python_list)
```

```
['ahmad', 'ali', 'maria', 'husnain', 'abubakar', 'amina']
```

In [16]:
```
# After adding item to a list via .append
python_list.append('umair')
print(python_list)
```

```
['ahmad', 'ali', 'maria', 'husnain', 'abubakar', 'amina', 'umair']
```

In [18]:
```
del python_list[4]
print(python_list)
# Removing items from the list
# To remove an item from a list we use del short for delete
```

```
['ahmad', 'ali', 'maria', 'husnain', 'amina', 'umair']
```

In [19]:
```
# 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)
```

```
[1, 2, 3, 4, 'I', 'tripped', 'over', 'and', 'hit', 'the', 'floor']
```

In [20]:
```
list1 = [1, 2, 3, 4]
list2 = ['I', 'ate', 'chocolate', 'and', 'I', 'want', 'more']
list3 = list1 + list2
print(list3)
# add list1 and list2 and store it in another variable list3
```

```
[1, 2, 3, 4, 'I', 'ate', 'chocolate', 'and', 'I', 'want', 'more']
```

In [21]:
```
 # multiply a list by a number
list1 = [1, 2]
print(list1 * 5)
```

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

In [22]:
```
# This is actually telling Python to repeat list1 five times, resulting in 1, 2, 1, 2,
1, 2, 1, 2, 1, 2.
```

Out[22]:
```
(1, 2, 1, 2, 1, 2.0)
```

division (/) and subtraction (-) give only errors,

4- Tuples

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

In [23]:
```
ali = (0, 1, 1, 2, 3)
print(ali[3])
```

```
2
```

The main difference between a tuple and a list is that a tuple cannot change once you've created it.

5-Python Maps or dictionary

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.

In [26]:
```
# Creating a map in Python
favorite_sports = ['Ali, Football', 'Omer, Basketball', 'Osman, Baseball',
                   'Abu Bkar, Netball', 'Fatima, Badminton', 'Faizan, Rugby']
print (favorite_sports)
```

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

In [27]:
```
# Modifying a value in a map
favorite_sports = {'Ali' : 'Football',
'Omer' : 'Basketball',
'Osman' : 'Baseball',
'Abu Bakr' : 'Netball',
'Fatima' : 'Badminton',
'Faizan' : 'Rugby'}
print (favorite_sports)
```

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

In [28]:
```
# get Ali favourite sport
print(favorite_sports['Ali'])
```

```
Football
```

In [29]:
```
# Deleting a value in the map
del favorite_sports['Ali']
print(favorite_sports)
```

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

In [30]:
```
# Replacing a value in a map
favorite_sports['Faizan'] = 'Cricket'
print(favorite_sports)
```

```
{'Omer': 'Basketball', 'Osman': 'Baseball', 'Abu Bakr': 'Netball', 'Fatima': 'Badminton', 'Faizan': 'Cricke
t'}
```

In [ ]: