AI Programming

[Day-5]

Program: Raghav has been given a challenge to write a python program to add 5 and 7. How can he do so?

Solution:

```
a,b = 5,7
print(a+b)
```

Output: 12

Program 2: Vigyan was asked to print the following using the print function. Help him in writing a python program for the same. (The Golden Ratio has the same letters as The God Relation.)

Solution:

```
a = "The Golden Ratio has the same letters as The God Relation."
print(a)
```

Output:

```
The Golden Ratio has the same letters
as The God Relation.
```

Program 3: Indentation refers to the spaces at the beginning of a code line. Where in other programming languages the indentation in code is for readability only, the indentation in Python is very important. When the following code is run, it gives an Indentation Error which is a compile-time error. Fix the code below and produce the intended output

i = 4
j = 3
if (i>j):
print(i)

Solution:

```
i = 4
j = 3
if (i>j):
  print(i)
```

Output: 4

Program 3: Mohit is a very naughty child, when his brother was away from his laptop, he changed his original program to the following:

x = input()

y = input()

if x>y:

output("x is greater than y")

else:

print("y is greater than x")

This program does not throw an error when it is run, rather it throws an error during runtime. These kinds of errors are known as runtime errors. If we give x=1, y=2, the program runs fine, but when we give x=2 and y=1, the program will throw an error. Correct this code so that it is error free.

Solution:

```
x = float(input())
y = float(input())
if x>y:
  print("x is greater than y")
else:
  print("y is greater than x")
```

Output:

```
1
2
y is greater than x
```

Program : The id() can be used to get the memory address of a variable. Consider the following code and tell if the id() functions will return the same value or not (as the value to be printed via print()) ? Why?

num = 13
print(id(num))
num = num +3
print (id(num))
num = num -3

```
print(id(num))
num = "Hello"
print(id(num))
```

Solution:

```
num = 13
print(id(num))
num = num+3
print(id(num))

num = num-3
print(id(num))
num ="Hello"
print(id(num))

'''
it is because when we assign 13 to the num
there is a memory alocated to 13
and num was pointing to that address and
when we assign 10 the interpreater created
10 and
when we change it 13 again interpreater
start to point again to the same location
as before
because it was not deleted in the
compilation
```

Output:

```
137944973296240
137944973296336
137944973296240
137943934857904
```