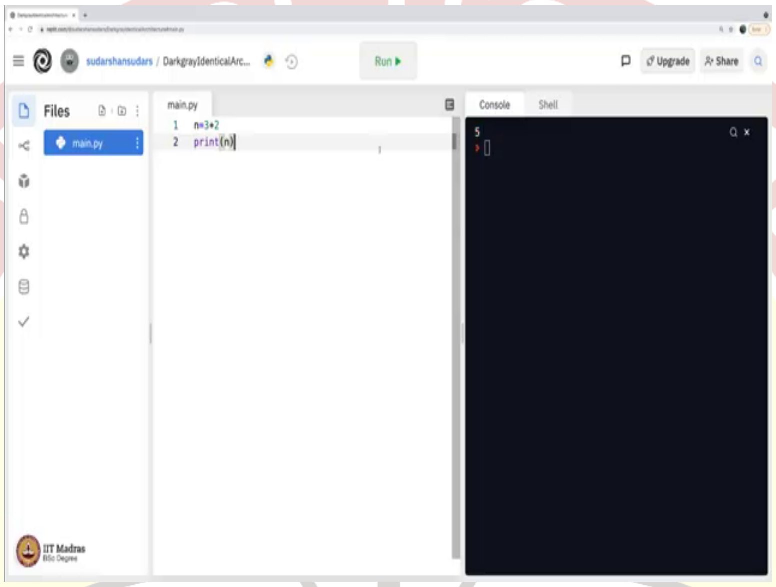
# IIT Madras

ONLINE DEGREE

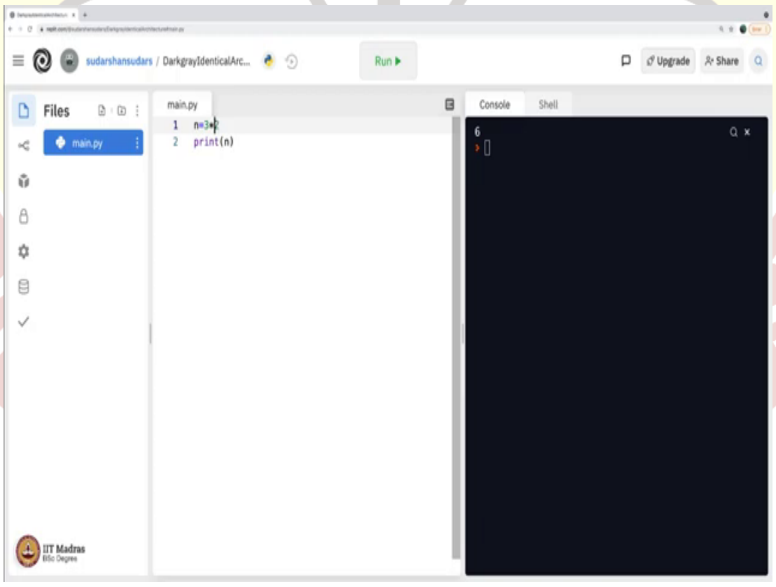
# Programming in Python
**Professor Sudarshan Iyengar**
**Department of Computer Science and Engineering**
**Indian Institute of Technology Ropar**
**Omkar Joshi**
**Course Instructor**
**Indian Institute of Technology Madras Online Degree Program**
**Operators and Expressions 1**

(Refer Slide Time: 0:16)



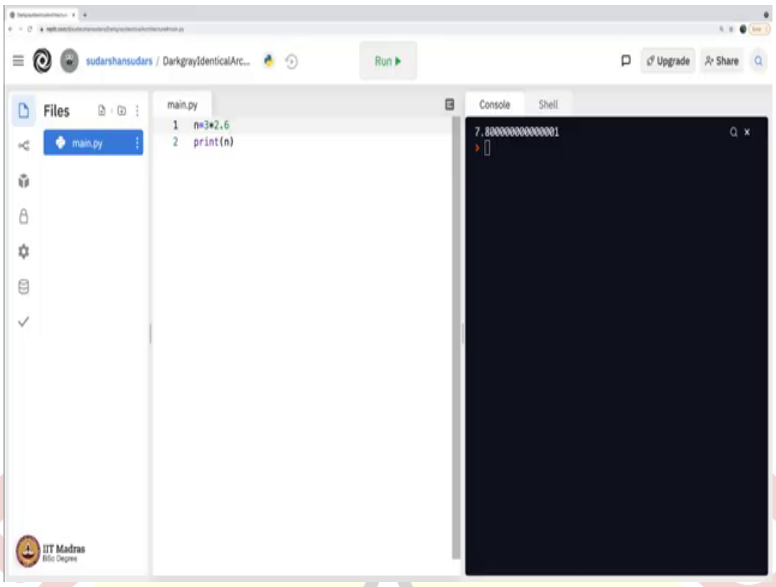
```
n=3+2
print(n)
```

```
5
> 
```



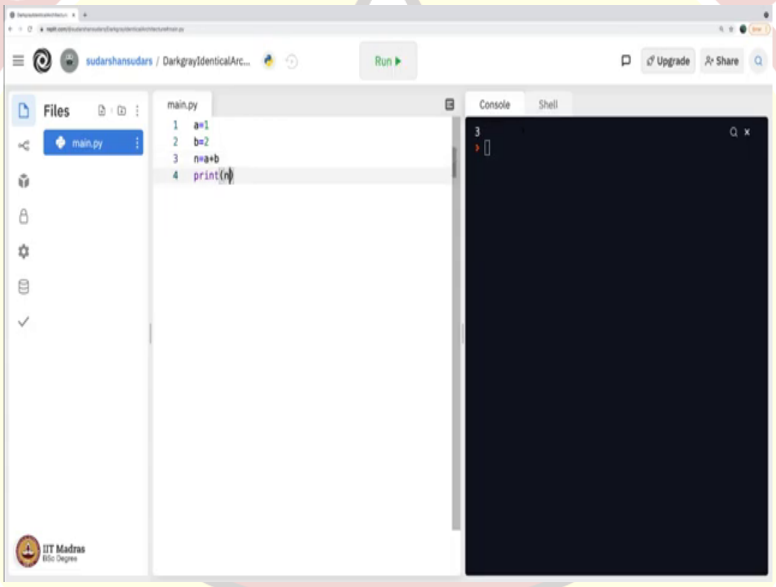
```
n=3*2
print(n)
```

```
6
> 
```



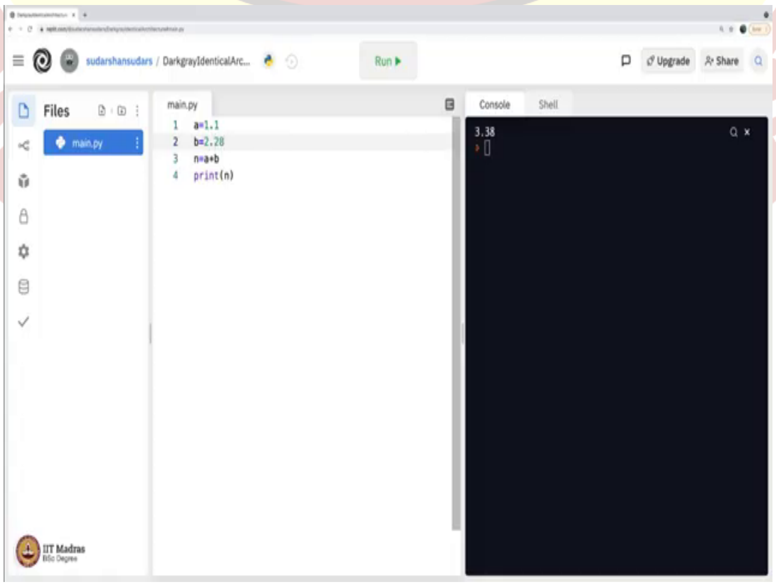
```
n=3*2.6
print(n)
```

```
7.800000000000001
>
```



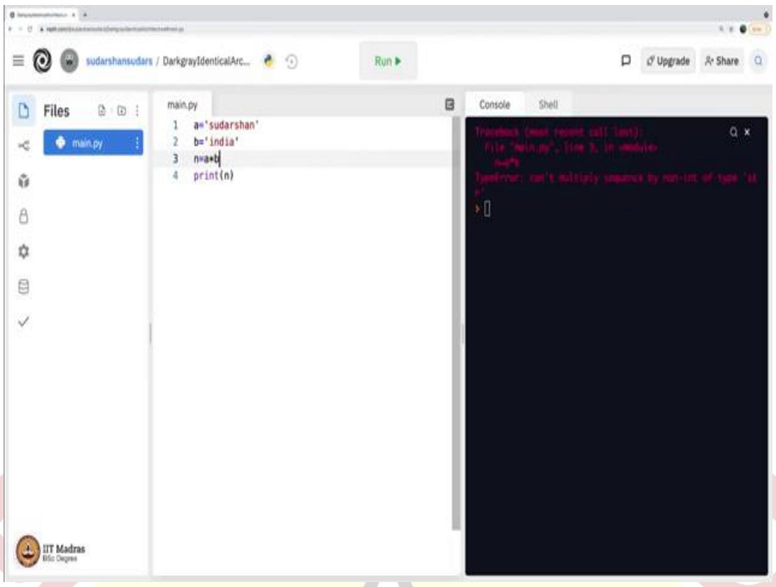
```
a=1
b=2
n=a+b
print(n)
```

```
3
>
```



```
a=1.1
b=2.28
n=a+b
print(n)
```

```
3.38
>
```



```
a='sudarshan'
b='india'
n=a*b
print(n)
```

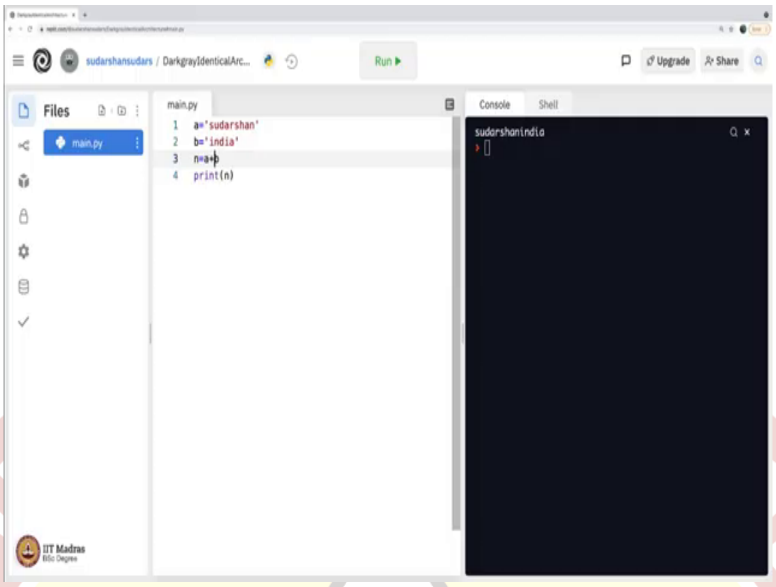
```
Traceback (most recent call last):
  File "main.py", line 3, in <module>
    n=a*b
TypeError: can't multiply sequence by non-int of type 'str'
> 
```

So as anyone would expect it is only obvious that when you say n equals 3 plus 2 and then print n, it prints 5. If you were to say 3 times 2, it prints 6. Then when I say 3 times 2.6, it prints the actual number 7.8. Fine, perfect. So, what about let us say a equals 1, b equals 2 and then I print n equals a plus b. This again behaves naturally as we expect.

If we include floating point numbers again behaves as we expect, but then if you were to simply say a equals sudarshan, b equals India and say n equals a times b, let us see what gets stored in it. It throws an error obviously because it does not know how to multiply these two strings, that is when we may want to recollect our discussion on data types.

This is of the type string, this is another type string, it does not understand, your computer does not understand what is into of two strings, but if it is a number it readily does the multiplication or let us say even division for that matter, correct? But then if I say a plus b, what is your guess? Do you think it will throw error?

(Refer Slide Time: 2:13)



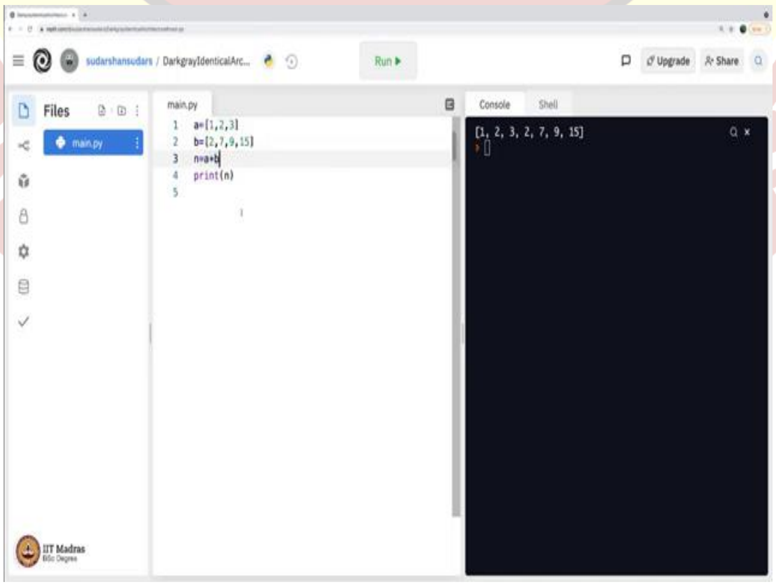
```
a='sudarshan'
b='india'
n=a+b
print(n)
```

```
sudarshanindia
```

Let us check, not at all. What it does? It simply mixes these two strings, mixes as in puts it one next to the other, this is called concatenation in English and more so in computer science this is very often used, they call it concatenate, put two words together, one next to the other. So, whenever you say plus of two strings, the computer seems to understand that.

Now you can ask me this question why is it that the computer understands plus and not minus or subtraction, I mean, subtraction or multiplication? It is programmed that way, it is for convenience that they use plus for concatenation of two strings.

(Refer Slide Time: 2:58)



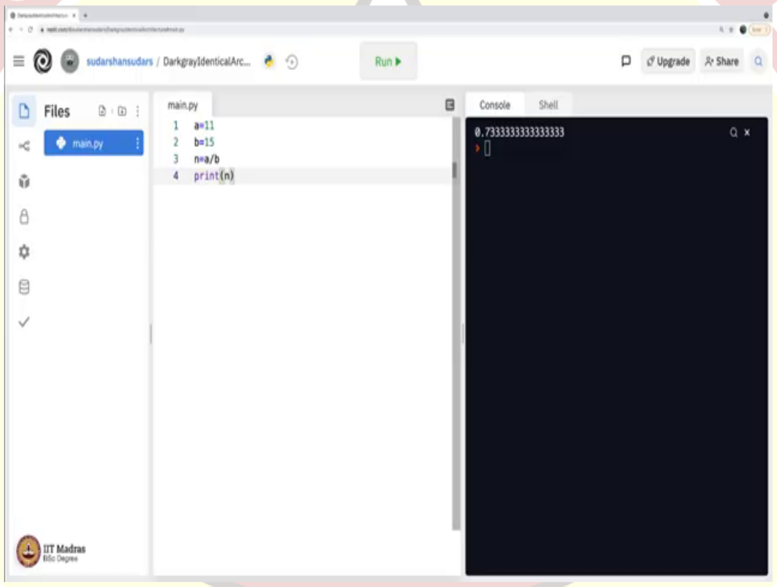
```
a=[1,2,3]
b=[2,7,9,15]
n=a+b
print(n)
```

```
[1, 2, 3, 2, 7, 9, 15]
```

So, let us go ahead and do something more here, let us say a equals 1, 2, 3, let us explore, what if b becomes 7, 9 and 15, what will a plus b be? a plus b simply gets, this is called union in mathematics. What is union? Union is simply, you take this list and this list and put them together. In fact, it is not even union, it is simply put them together, make them, make it together. For example, if you had another 2 here, it will include that 2 too here as you can see.

Union means there should not be a repetition, that is the idea of sets that you would have studied in your high school days, anyway let us not get there, it is not required here. All that this does is simply puts them one next to the other and creates a new list. So, now you understand a data type matters when we say an operator like plus, when we put what is the data type of a and b matters for the computer to give the right output?

(Refer Slide Time: 4:15)

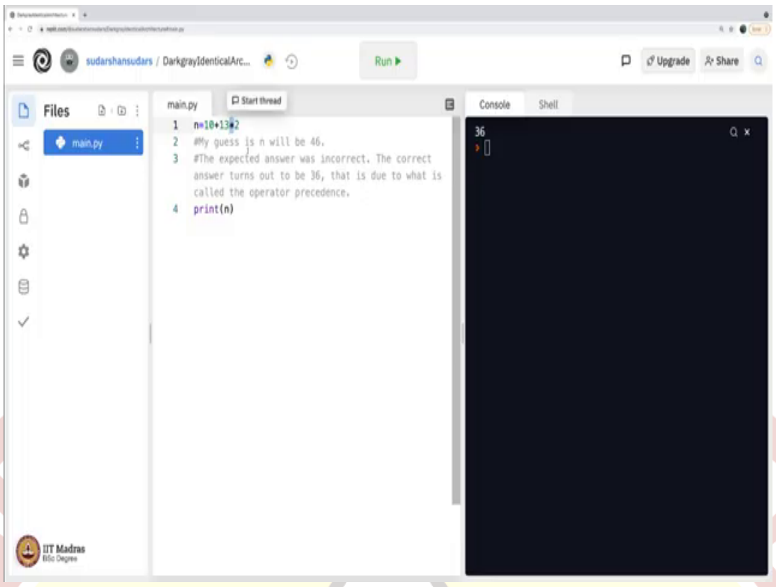


```
a=11
b=15
n=a/b
print(n)
```

```
0.7333333333333333
›
```

So, with this let us go ahead to the next idea, which is let us say a equals 11, b equals 15, I say n equals a by b, I say print n. Again as expected it prints the floating point, the float number. Float is something that is not integer as I keep saying, so it is 0.7333. Fine, so far so good.

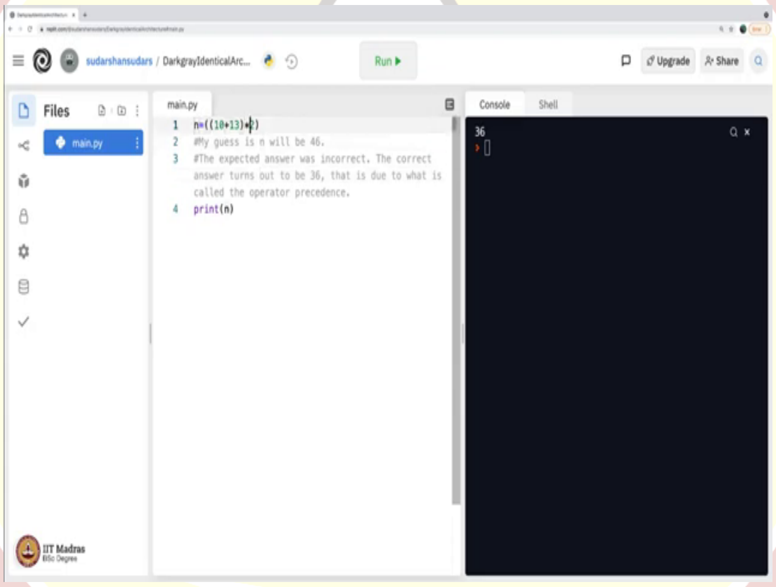
(Refer Slide Time: 4:42)



main.py

```python
n=10+13*2
#My guess is n will be 46.
#The expected answer was incorrect. The correct answer turns out to be 36, that is due to what is called the operator precedence.
print(n)
```

```
36
> 
```



main.py

```python
n=((10+13)*2)
#My guess is n will be 46.
#The expected answer was incorrect. The correct answer turns out to be 36, that is due to what is called the operator precedence.
print(n)
```

```
36
> 
```

Now, what if we were to simply say n equals, let us say 10 plus 13, again does the obvious here, but then what if I said 10 plus 13 times 2. What is it that you would expect here? I would expect this to be 10 plus 13, let us say it goes from left to right, first it does 10 plus 13, this becomes 23, and then multiplies 23 into 2, which is 46, this is my guess.

You comment using hash here, my guess is n will be how much, 10 plus 13 is 23 times 2 is 46. Let us see if this is the answer. No, this is not the answer, the answer seems 36. Why? That is because your computer does not do it the way you expect it to do, it does the way it was programmed to do. Internally it was told to calculate such expressions, arithmetic expressions in a particular way.

And that particular way is that whenever you have a plus and a multiplication, give priority to multiplication, which means come immediately to the place where there is into, the star and multiply these two numbers, you get 26, and then add 10 to it, you get 36, and that is precisely what is being displayed here, not the typical first guess that I did, 10 plus 13 is 23 times 2 is 46.

No, it first comes to into and then does plus. This concept is called operator precedence, let me just comment that too, the expected answer was incorrect, the correct answer turns out to be 36 that is due to what is called the operator precedence. It is a very complicated term for something as simple as some operators, by operator we mean into, plus they all are called operators. Some operators are precedence over the other.

They are executed first over the other. This is an important concept, in all programming classes they teach this, although in my personal opinion this is not very important. For a simple reason that we can always do what it takes for us to simply put braces, brackets and then tell what you want. Here the moment you put your brackets, 10 plus 13 will get executed first, 23, into 2 is 46, it has to be that.

Now the computer cannot do into first and then do plus next. It did that when no brackets were there like this, but the moment you put brackets the computer understands what you expect it to do, it will indeed show 46 as you can see here. So, now you understand what are operators, your plus, your star, your minus, your division, slash, all of them are called operators and when you put them in tandem, when you put them together the way we did before, how was that.

Let me put that, when you put them continuously like this and confuse the computer, the computer indeed does not get confused, it does what is called operator precedence rule with which it first does the multiplication and then does the addition. As I repeat do not break your head so much about this concept, it is just that you need to know what happens when you give an arithmetic expression like this.

What is more important is trying to code and get the output that is expected, rest are all just details. So, let us see more about these operators and operator precedence and arithmetic expressions in our next video.