

1.

Syntax:-

`x=5` # Defining variable 'x'

`type(x)` # Checking the data type of the variable 'x'

Output : int =>i.e., Data type of variable 'x' is integer.

`y="John"` # Defining variable 'y'

`type(y)` # Checking the data type of the variable 'y'

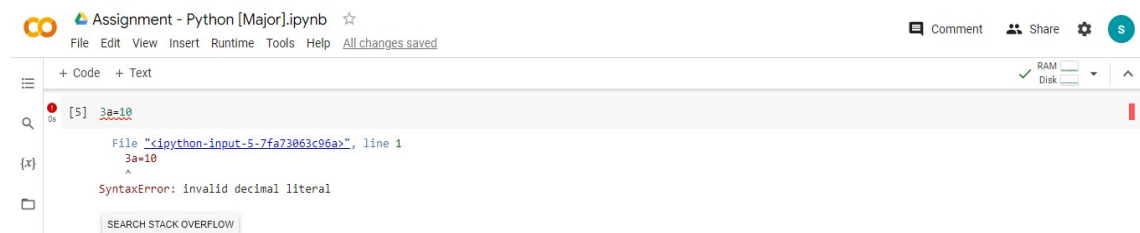
Output : str => i.e., Data type of variable 'y' is String.



A screenshot of a Jupyter Notebook titled "Assignment - Python [Major].ipynb". The notebook shows two code cells. The first cell contains `[1] x=5` and `[3] type(x)`, with the output `int` displayed. The second cell contains `[2] y="John"` and `[4] type(y)`, with the output `str` displayed. The interface includes a top bar with "File Edit View Insert Runtime Tools Help" and a right sidebar with "Comment Share" and a status icon.

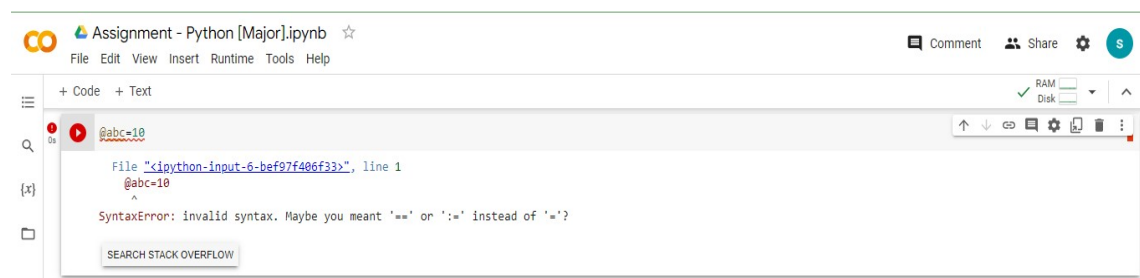
2.

i. `3a=10` #invalid syntax – Identifier can't start with a number.



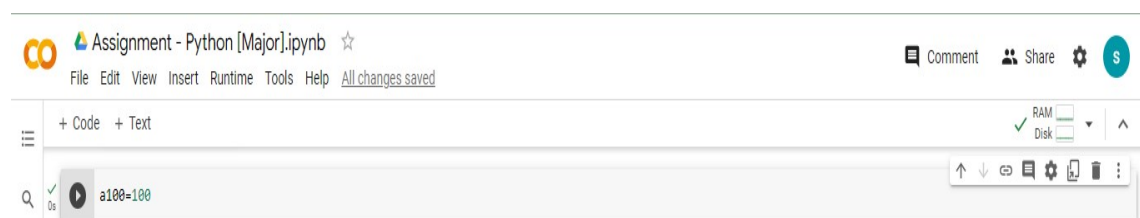
A screenshot of a Jupyter Notebook titled "Assignment - Python [Major].ipynb". The notebook shows a code cell with `[5] 3a=10`. The output is a `SyntaxError: invalid decimal literal`. The error message includes the file path `File "c:\python-input-5-7fa73063c96a>", line 1` and the code `3a=10`. A "SEARCH STACK OVERFLOW" button is visible below the error message.

ii. `@abc=10` #invalid syntax – Identifier can have only underscore as a special character.



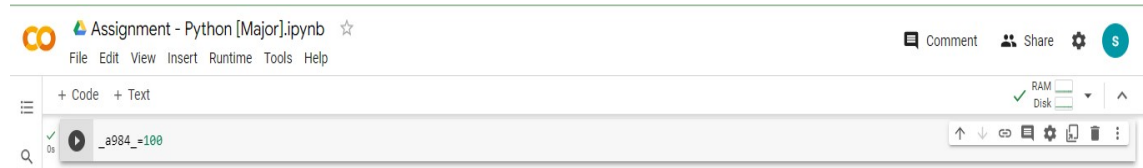
A screenshot of a Jupyter Notebook titled "Assignment - Python [Major].ipynb". The notebook shows a code cell with `@abc=10`. The output is a `SyntaxError: invalid syntax. Maybe you meant '==' or ':=' instead of '='?`. The error message includes the file path `File "c:\python-input-6-bef97f406f33>", line 1` and the code `@abc=10`. A "SEARCH STACK OVERFLOW" button is visible below the error message.

iii. `a100=100` #Valid Syntax – Identifier can be alphanumeric.



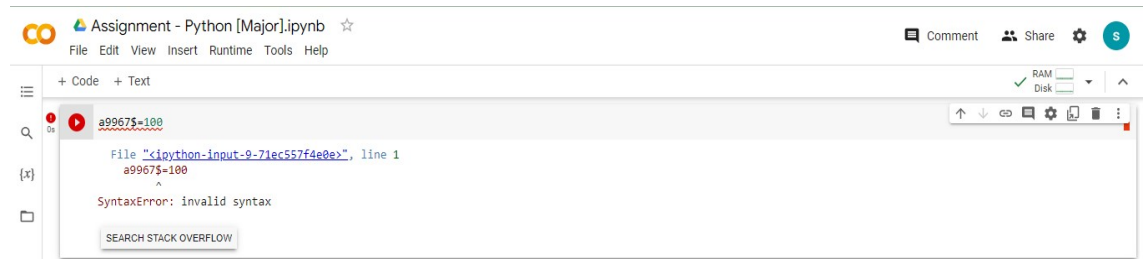
A screenshot of a Jupyter Notebook titled "Assignment - Python [Major].ipynb". The notebook shows a code cell with `a100=100`. The output is `a100=100`. The interface includes a top bar with "File Edit View Insert Runtime Tools Help" and a right sidebar with "Comment Share" and a status icon.

iv. `_a984_=100` #Valid syntax – Identifier can be alphanumeric and can contain underscore.



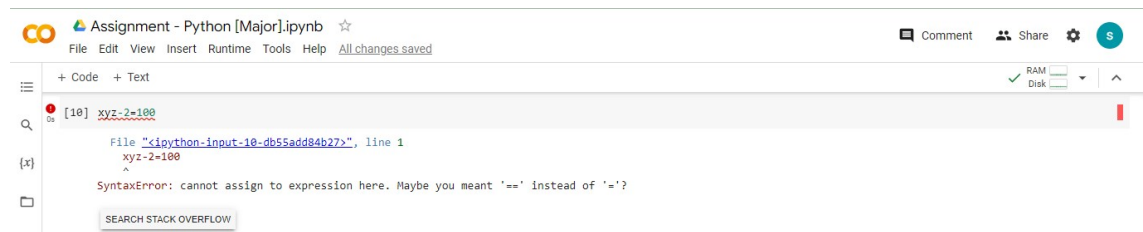
A screenshot of a Jupyter Notebook interface. The title bar says "Assignment - Python [Major].ipynb". The menu bar includes File, Edit, View, Insert, Runtime, Tools, and Help. The toolbar shows icons for running code, saving, and other functions. The code cell contains the line `_a984_=100`, which has been successfully executed, indicated by a green checkmark and a play button icon.

v. `a9967$=100` #Invalid syntax – Identifier can have only underscore as a special character.



A screenshot of a Jupyter Notebook interface showing a syntax error. The code cell contains the line `a9967$=100`. The execution failed, indicated by a red error icon. The error message displayed is "SyntaxError: invalid syntax". The notebook title is "Assignment - Python [Major].ipynb".

vi. `xyz-2=100` #Invalid syntax – Identifier can't have the operator. It can be alphanumeric and can have only underscore as a special character.



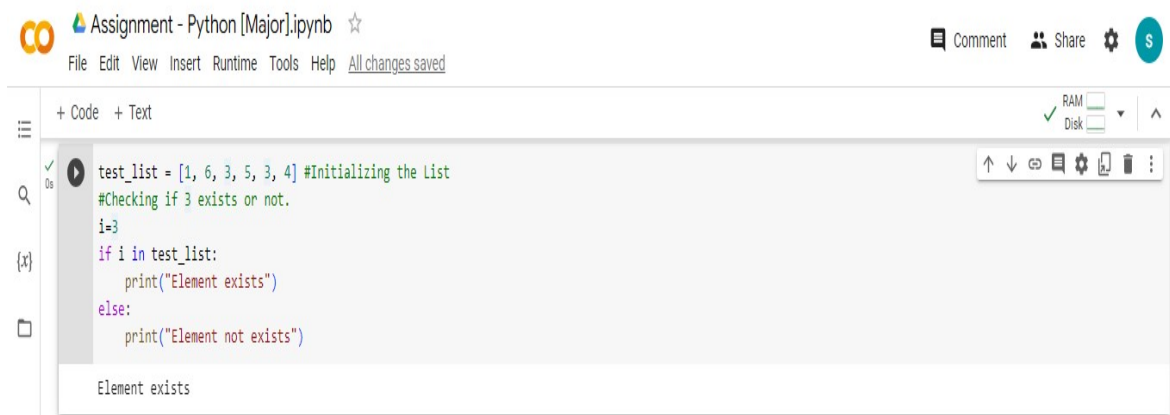
A screenshot of a Jupyter Notebook interface showing a syntax error. The code cell contains the line `xyz-2=100`. The execution failed, indicated by a red error icon. The error message displayed is "SyntaxError: cannot assign to expression here. Maybe you meant '==' instead of '='?". The notebook title is "Assignment - Python [Major].ipynb".

3.

Syntax:-

```
l. test_list = [1, 6, 3, 5, 3, 4] #Initializing the List
#Checking if 3 exists or not.
i=3
if i in test_list:
    print("Element exists")
else:
    print("Element not exists")
```

Output:- Element exists



A screenshot of a Jupyter Notebook interface showing the successful execution of a Python script. The code cell contains the following code:

```
test_list = [1, 6, 3, 5, 3, 4] #Initializing the List
#Checking if 3 exists or not.
i=3
if i in test_list:
    print("Element exists")
else:
    print("Element not exists")
```

The output of the code is "Element exists". The notebook title is "Assignment - Python [Major].ipynb".

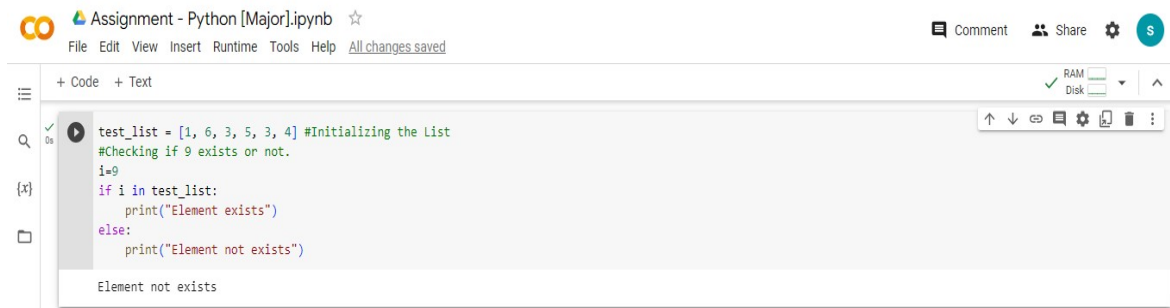
```

II. test_list = [1, 6, 3, 5, 3, 4] #Initializing the List
    #Checking if 9 exists or not.
    i=9
    if i in test_list:
        print("Element exists")
    else:
        print("Element not exists")

```

Output:- Element not exists

Result:-



Assignment - Python [Major].ipynb

```

test_list = [1, 6, 3, 5, 3, 4] #Initializing the List
#Checking if 9 exists or not.
i=9
if i in test_list:
    print("Element exists")
else:
    print("Element not exists")

```

Element not exists

4.

Syntax:-

#Entering the user input of year, month & day

```
year = int(input('Enter a year: '))
```

```
month = int(input('Enter a month: '))
```

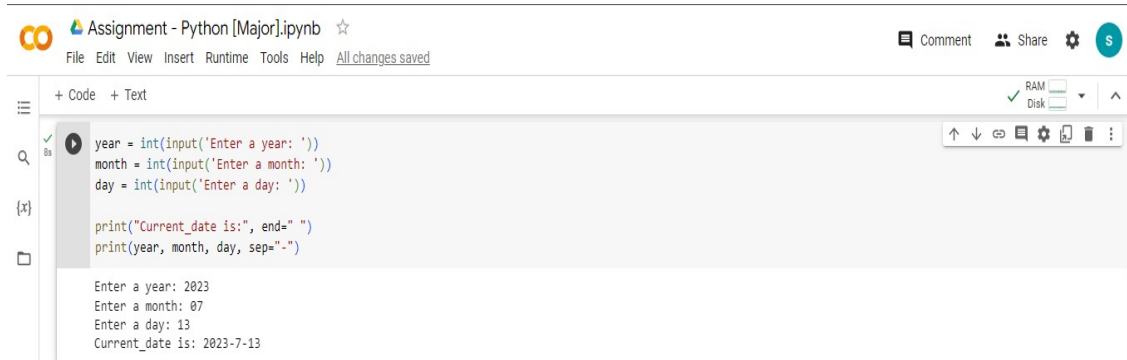
```
day = int(input('Enter a day: '))
```

#printing the Current date

```
print("Current_date is:", end=" ")
```

```
print(year, month, day, sep="-")
```

Result:-



Assignment - Python [Major].ipynb

```

year = int(input('Enter a year: '))
month = int(input('Enter a month: '))
day = int(input('Enter a day: '))

print("Current_date is:", end=" ")
print(year, month, day, sep="-")

```

Enter a year: 2023
Enter a month: 07
Enter a day: 13
Current_date is: 2023-7-13

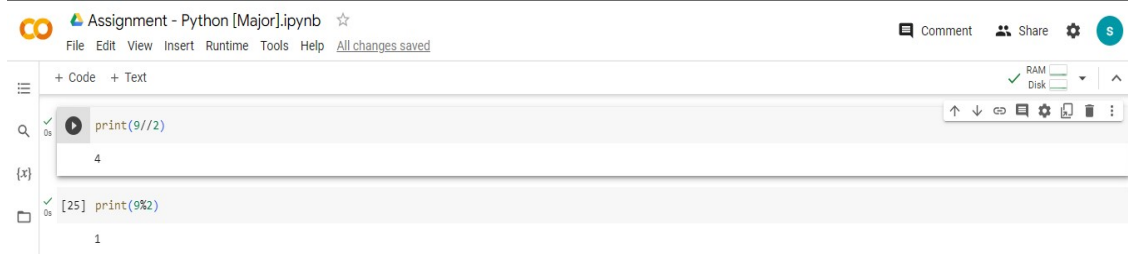
5.

Syntax:-

a. `print(9//2)` # // is floor division – a division operation that rounds the result down to the nearest whole number or integer, which is less than or equal to the normal division result.
output – 4

b. `print(9%2)` # % is Modulus – remainder of the division of left operand by the right
output – 1

Result:-



Assignment - Python [Major].ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk

```
print(9//2)
```

4

```
print(9%2)
```

1

6.

Syntax:-

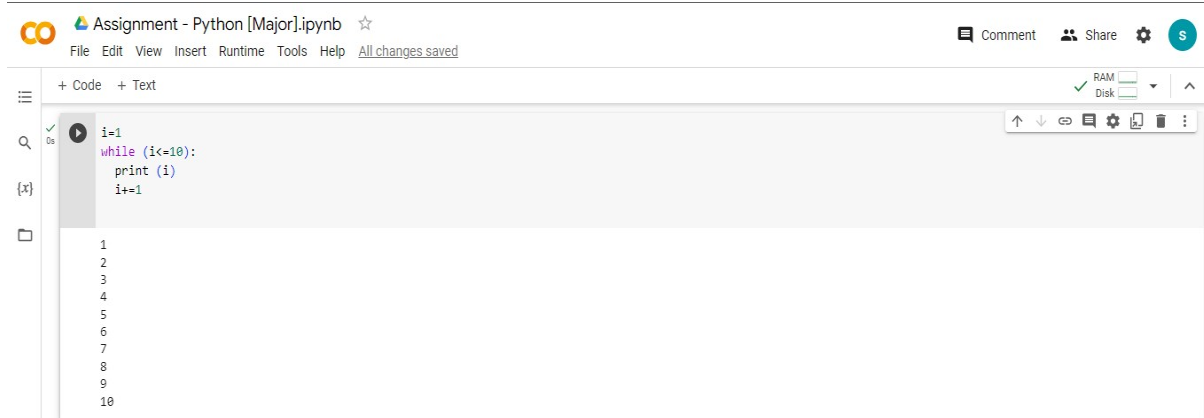
i=1

while (i<=10):

print (i)

i+=1

Result:-



Assignment - Python [Major].ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk

```
i=1
while (i<=10):
    print (i)
    i+=1
```

1
2
3
4
5
6
7
8
9
10

7.

Syntax:-

x=int(input("Enter a number: "))

s=0

for i in range(1, x + 1):

s+=i

print("Sum of numbers is:", s)

Result:-



Assignment - Python [Major].ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk

```
x=int(input("Enter a number: "))
s=0
for i in range(1, x + 1):
    s+=i
print("Sum of numbers is:", s)
```

Enter a number: 10
Sum of numbers is: 55

8.

Syntax:-

```
for i in range(1, 51): # initializing the iteration from 1 to 50
    if i % 3 == 0 and i % 5 == 0: # writing the condition as per the question using if elif statements
        print("FizzBuzz")
        continue
    elif i % 3 == 0:
        print("Fizz")
        continue
    elif i % 5 == 0:
        print("Buzz")
        continue
    print(i)
```

Result:-

The image displays three sequential screenshots of a Jupyter Notebook interface, showing the execution of a Python FizzBuzz program. The notebook is titled "Assignment - Python [Major].ipynb".

First Screenshot: The code cell contains the following Python code:

```
for i in range(1, 51):
    if i % 3 == 0 and i % 5 == 0:
        print("FizzBuzz")
        continue
    elif i % 3 == 0:
        print("Fizz")
        continue
    elif i % 5 == 0:
        print("Buzz")
        continue
    print(i)
```

The output cell shows the first 14 lines of the sequence: 1, 2, Fizz, 4, Buzz, Fizz, 7, 8, Fizz, Buzz, 11, Fizz, 13, 14, FizzBuzz.

Second Screenshot: The code cell is empty. The output cell shows the next 14 lines of the sequence: 14, FizzBuzz, 16, 17, Fizz, 19, Buzz, Fizz, 22, 23, Fizz, Buzz, 26, Fizz, 28, 29, FizzBuzz, 31, 32, Fizz, 34, Buzz, Fizz, 37, 38, Fizz, Buzz.

Third Screenshot: The code cell is empty. The output cell shows the final 12 lines of the sequence: 38, Fizz, Buzz, 41, Fizz, 43, 44, FizzBuzz, 46, 47, Fizz, 49, Buzz.