In [ ]:
```
#name, id, salary, logging_status
 str   int   float    bool        NoneType
```

In [1]:
```
print(10)
```

```
10
```

In [2]:
```
print(10.333)
```

```
10.333
```

In [5]:
```
print(10.4cm)
```

```
  Cell In[5], line 1
    print(10.4cm)
              ^
SyntaxError: invalid decimal literal
```

In [6]:
```
print("10.44cm")
```

```
10.44cm
```

In [7]:
```
print('10.44cm')
```

```
10.44cm
```

In [ ]:
```
str- collection of char - enclosed " "  or ''
```

In [14]:
```
#variablename=value
name="Harish"    # str
id=12            # int
salary=12023.22    # float
logging_status=True    # bool
```

In [8]:
```
name=1022  # dynamic type lang
```

In [9]:
```
emp_name="Hari"
```

In [10]:
```
emp_name+10
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[10], line 1
----> 1 emp_name+10

TypeError: can only concatenate str (not "int") to str
```

In [11]:
```
type(name)
```

Out[11]:
```
int
```

In [12]:
```
type(emp_name)
```

Out[12]:
```
str
```

In [15]:
```
type(salary)
```

Out[15]:
```
float
```

In [16]:
```
type(id)
```

Out[16]:
```
int
```

In [19]:
```
type(Logging_status)       #error
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[19], line 1
----> 1 type(Logging_status)

NameError: name 'Logging_status' is not defined
```

In [20]:
```
type(logging_status)
```

Out[20]:
```
bool
```

In [21]:
```
print(type(salary))
print(type(id))
print(type(logging_status))
```

```
<class 'float'>
<class 'int'>
<class 'bool'>
```

In [22]:
```
print(emp_name,id,salary,logging_status) # displaying values
```

```
Hari 12 12023.22 True
```

In [23]:
```
#combining user defined str and variable
# 1. use comma                                  (1)
print("Employee name is",emp_name)
   #   |                   |
   # user defined str     variable
```

```
Employee name is Hari
```

In [25]:
```
print("Employee Id : ", id)
print("Employee salary :", salary)
print("Employee logging_status :", logging_status)
```

```
Employee Id :  12
Employee salary : 12023.22
Employee logging_status : True
```

In [26]:
```
# 2.  int %d   string %s   float %f    - C lang format specifiers          (2)

print("Employee name %s"%(emp_name))
```

```
Employee name Hari
```

In [27]:
```
print("Employee name:%s Employee id : %d"%(emp_name,id))  # display str and int
```

```
Employee name:Hari Employee id : 12
```

In [28]:
```
print("Employee name %d"%(emp_name))  # %d refer to int but given str variable
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[28], line 1
----> 1 print("Employee name %d"%(emp_name))

TypeError: %d format: a real number is required, not str
```

In [ ]:
```
'''
1. create a python program

  Initialize variable like app and port

  display app and port values

  display in given format :    Application name is Flask   Port Number is 1009

  display in given format :    Application name is Flask   Port Number is 1009


'''
```

In [29]:
```
app="Flask"
port=1009

print(app)
print(port)

print("Application name is",app,"Port Number is",port)     # using comma
print("Application name is %s Port Number is %d"%(app,port))  # using C style f
```

```
Flask
1009
Application name is Flask Port Number is 1009
Application name is Flask Port Number is 1009
```

In [ ]:
```
# format() method---> python 2.7
# oop style   "".format()
```

In [30]:
```
print("Employee name : {} Employee Id : {}".format(emp_name,id))
```

```
Employee name : Hari Employee Id : 12
```

In [31]:
```
print("Employee name : {} Employee Id : {}".format(id,emp_name))   # Error - va
```

```
Employee name : 12 Employee Id : Hari
```

In [32]:
```
print(f"Employee name : {emp_name}  Employee Id: {id}")       # (4)
```

```
Employee name : Hari  Employee Id: 12
```

In [34]:
```
print("Application name is",app,"port number is",port)                    # (1)
print("Application name is %s port number is %d"%(app,port))              #(2)
print("Application name is {} port number is {}".format(app,port))        # (3)
print(f"Application name is {app} port number is {port}")                 # (4)
```

```
Application name is Flask port number is 1009
Application name is Flask port number is 1009
Application name is Flask port number is 1009
Application name is Flask port number is 1009
```

In [ ]:
```
'''
Write a python program

1. create and initialize variabes fstype , partitionSize and mountpoint

2. display Filesystem details
'''
```

In [36]:
```
fstype="xfs"
mountpoint="\D1"
partitionSize=500

print("Filesystem Type:", fstype)
print("MountPoint:", mountpoint)
print("Partition Size:",partitionSize)
```

```
Filesystem Type: xfs
MountPoint: \D1
Partition Size: 500
```

In [37]:
```
print("Server1\nServer2\nServer3\tServer4")  #  \n newline    \t tab   escape s
```

```
Server1
Server2
Server3 Server4
```

In [41]:
```
#Multiline String Literal


''' This is multiline comment'''

print("""Server1
Server2
Server3    Server4
""")
```

```
Server1
Server2
Server3    Server4
```

In [42]:
```
multistr="""
=====
Data1
Data2
Data3
Data4
Data4
======"""

print(multistr)
```

```
=====
Data1
Data2
Data3
Data4
Data4
======
```

In [39]:
```
print('''Server1
Server2
Server3    Server4
''')
```

```
Server1
Server2
Server3    Server4
```

In [44]:
```
'''
2.Write a python program
step1:  create variables for Product details- productName,productId,
            productVersion, productStatus

step2: dislay the product details in below format

=======================================
        Product details
=======================================
Product Name :  GenRocket
Product Id :    CD10
Product Version :   3.5
Procduct Stage Status : True
=======================================

'''

productName= "GenRocket"
productId="CD10"
productVersion=3.5
productStatus=True

print(f"""
=======================================
        Product details
=======================================
Product Name : {productName}
Product Id :   {productId}
Product Version : {productVersion}
Procduct Stage Status : {productStatus}
=======================================
""")
```

```
=======================================
        Product details
=======================================
Product Name :  GenRocket
Product Id :    CD10
Product Version :       3.5
Procduct Stage Status : True
=======================================
```

In [46]:
```
'3.5678'* 0.353
```

```
  Cell In[46], line 1
    '3.5678'* 0.353type(10)
                      ^
SyntaxError: invalid decimal literal
```

In [47]:
```
print(type(0),type(0.0),type(''))
```

```
<class 'int'> <class 'float'> <class 'str'>
```

In [ ]:
```
#Typecasting-  int -> float    float(var)---> float
#  float-> int          int(var)---> int
#  int -> str           str(var)-----> str
```

In [49]:
```
float('3.4454') * 0.33
```

Out[49]:
```
1.136982
```

In [50]:
```
n=10
print(type(n),n)
```

```
<class 'int'> 10
```

In [56]:
```
print(type(float(n)),float(n))
print(type(str(n)), str(n))
```

```
<class 'float'> 10.0
<class 'str'> 10
```

In [ ]:
```
'''
int,float
==========
Arithmetic operatots
+ - * / // % **      int, float   ----> int, float

str
====
+   Str concatenation
*   Str repetition


Relational operators
< > <= >= == !=            float/int ---> bool True/False

'''
```

In [57]:
```
100/20
```

Out[57]:
```
5.0
```

In [58]:
```
100//20
```

Out[58]:
```
5
```

In [62]:
```
print("Hello"+"Friend")   # str concatenation
```

```
HelloFriend
```

In [61]:
```
print("Hello","Friend")
```

```
Hello Friend
```

In [63]:
```
print("Hello"*12) # str repetition       -->  "stringvalue" * count
```

```
HelloHelloHelloHelloHelloHelloHelloHelloHelloHelloHelloHello
```

In [66]:
```
productName= "GenRocket"
productId="CD10"
productVersion=3.5
productStatus=True

print("="*45  + '\nProduct Details\n' + "="*45 )

print(f"""
Product Name : {productName}
Product Id :   {productId}
Product Version : {productVersion}
Procduct Stage Status : {productStatus}
""")
print("="*45)
```

```
=============================================
Product Details
=============================================

Product Name : GenRocket
Product Id :   CD10
Product Version : 3.5
Procduct Stage Status : True

=============================================
```

In [67]:
```
10<20
```

Out[67]:
```
True
```

In [ ]:
```
# Validation
# Conditional Statement  --> execute code baed on condition
# condition with relationsl operator, return value of function/Methods
'''
if( Conditional_Expression):
    True blocks
else:
    False blocks

'''
```

In [76]:
```
print(logging_status)
if(logging_status == True):
    print("Logged In Success ")
else:
    print("Logging Failed")
```

```
True
Logged In Success
```

In [77]:
```
print(app)
if(app == "Django"):
    print("Application with Django is Running")
else:
    print("Django Application is not Success")
```

```
Flask
Django Application is not Success
```

In [70]:
```
print(app)
```

```
Flask
```

In [74]:
```
# input()---> interface to keyboard
# syntax:-       input_var= input("Prompt Message)")
name=input("Enter your Name")
```

```
Enter your NameManish
```

In [75]:
```
print(name)
```

```
Manish
```

In [78]:
```
'''
Write a python program

1.Get Application name from user

2.Check whether app    is Flask
                  |-----------------------No----- Initialize port as 9000
                  |yes
                  |
                  Intialize  port as 5000
3.Display Application name and Port number

'''
app=input("Enter the Application Name")
if(app == "Flask"):
    port=9000
else:
    port=5000

print(f'Application name: {app}   Port Number: {port}')
```

```
Enter the Application NameFlask
Application name: Flask   Port Number: 9000
```

In [79]:
```
app=input("Enter the Application Name")
if(app == "Flask"):
    port=9000
else:
    port=5000

print(f'Application name: {app}   Port Number: {port}')
```

```
Enter the Application NameDjango
Application name: Django   Port Number: 5000
```

In [85]:
```
'''
Write a python program

1. Get Application name from user
2. if Application name is Flask -----> get port number from user------> Check i
                           |
                           | No                                        yes   |---
                           |
                          Application Name not Matched            display App ar
'''
app=input("Enter the Application Name")
if(app == "Flask"):
    port= int(input("Enter the port number"))
    if(port < 5000):
        print(f"Application {app} running in port number {port}")
    else:
        print("Invalid Port number")
else:
    print("Application name not matched")
```

```
Enter the Application NameDjango
Application name not matched
```

In [86]:
```
app=input("Enter the Application Name")
if(app == "Flask"):
    port= int(input("Enter the port number"))
    if(port < 5000):
        print(f"Application {app} running in port number {port}")
    else:
        print("Invalid Port number")
else:
    print("Application name not matched")
```

```
Enter the Application NameFlask
Enter the port number3000
Application Flask running in port number 3000
```

In [87]:
```
app=input("Enter the Application Name")
if(app == "Flask"):
    port= int(input("Enter the port number"))
    if(port < 5000):
        print(f"Application {app} running in port number {port}")
    else:
        print("Invalid Port number")
else:
    print("Application name not matched")
```

```
Enter the Application NameFlask
Enter the port number8000
Invalid Port number
```

In [ ]:
```
app=input("Enter the Application Name")
if(app == "Flask"):
    port= input("Enter the port number")
    if(int(port) < 5000):
        print(f"Application {app} running in port number {port}")
    else:
        print("Invalid Port number")
else:
    print("Application name not matched")
```

In [ ]:
```
app=input("Enter the Application Name")
if(app == "Flask"):
    port=9000
else:
    port=5000

print(f'Application name: {app}   Port Number: {port}')
```

In [ ]:
```
'''
Write a python program

step1 : Get shell name as input from user
step2 : Check whether shellname matches with "bash"
                    |------------------if matched ---Initialize profile filenam
                Not matched
                    |
                    Initialize profile filename as "default profile"

'''
```

In [83]:
```
print(app)
if app=="Flask":
    print("App: Flask")
```

```
Flask
App: Flask
```

In [94]:
```
'''
app-> Flask--------> port =6000
app-> Prometheus  ---> port =4000
app-> Django -------. port=7000
'''
app=input("Enter the Application Name:")
if(app == "Flask"):
    port=6000
elif(app == "Prometheus"):
    port=4000
elif(app == "Django"):
    port=7000

else:
    port=1000

print(f"Application {app} running in {port}")
```

```
Enter the Application Name:mllab
Application mllab running in 1000
```

In [ ]:
```
#more than one condition with single if
# logical operator      and   or   not
```

```
if(user=="student" and passwd="rafddkln")

2000 < port < 6000   range   if( 2000< port  and port <6000)
```

In [ ]:
```
app=input("Enter the app name")

if(app):
    pass
else:
    print("Application not entered")
```

In [ ]:
```
if(not(app)):
    print("Application not entered")
```

In [ ]:
```
Looping
|-----------> conditional style looping - execute more than 1 based on conditic
|
|-----------> Collection style looping - no. of eleents in collection - for
```

In [ ]:
```
Syntax:-

Initialization
while(Condition):
    code block
    incre/decre
```

In [96]:
```
i=0
while(i<5):
    print("Data ",i)
    i=i+1
```

```
Data  0
Data  1
Data  2
Data  3
Data  4
```

In [100]:
```
'''
Write a python program

1. Initialize pin as 1234
2. Read input pin PIN from user
3. Check whether input PIN matches with pin
4. If matched, give message like, Valid Pin . Login Success
5. If not matched, give message like, Invalid Pin- Login Failed
'''
pin=1234
PIN=input("Enter the pin")
if(pin == PIN):
    print("Valid Pin- LOGIN SUCCESS")
else:
    print("Invalid Pin - LOGIN FAILED")
```

```
Enter the pin23232
Invalid Pin - LOGIN FAILED
```

In [102]:
```
pin=1234
PIN=int(input("Enter the pin"))
if(pin == PIN):
    print("Valid Pin- LOGIN SUCCESS")
else:
    print("Invalid Pin - LOGIN FAILED")
```

```
Enter the pin1234
Valid Pin- LOGIN SUCCESS
```

In [104]:
```
'''

1.Modify above code by restricting the user input to thrice.

2. Also if pin matched, give message, Login success at Attempt_count

3. If pin goes wrong on all 3 attempt, display--> "PIN BLOCKED"


'''
pin=1234

i=1
while(i<=3):
    PIN=int(input("Enter the pin"))
    if(pin == PIN):
        print(f"Valid Pin- LOGIN SUCCESS at Attempt{i}")
        break
    else:
        print("Invalid Pin - LOGIN FAILED")

    i+=1    # same as  i= i+1. using compound assignment operator +=

if(pin != PIN):
    print("PIN BLOCKED")
```

```
Enter the pin666
Invalid Pin - LOGIN FAILED
Enter the pin3232
Invalid Pin - LOGIN FAILED
Enter the pin1234
Valid Pin- LOGIN SUCCESS at Attempt3
```

In [107]:
```
pin=1234
i=1
while(i<=3):
    PIN=int(input("Enter the pin"))
    if(pin == PIN):
        print(f"Valid Pin- LOGIN SUCCESS at Attempt : {i}")
        break
    else:
        print("Invalid Pin - LOGIN FAILED")

    i+=1    # same as  i= i+1. using compound assignment operator +=

if(pin != PIN):
    print("PIN BLOCKED")
```

```
Enter the pin1234
Valid Pin- LOGIN SUCCESS at Attempt : 1
```

In [97]:
```
break
```

```
  Cell In[97], line 1
    break
    ^
SyntaxError: 'break' outside loop
```

In [99]:
```
i=0
while(i<2):
    if(i==2):
        break
    else:
        print(i)
    i+=1
```

```
0
1
```

In [108]:
```
'''
for variable in collection:
    code
'''


for i in 'abc':
    print(i)

    i=a   --->
    i=b --->
    i=c --->

    out of data
```

```
a
b
c
```

In [109]:
```
len('abc')
```

Out[109]:
```
3
```

In [110]:
```
help(str)
```

```
Help on class str in module builtins:

class str(object)
 |  str(object='') -> str
 |  str(bytes_or_buffer[, encoding[, errors]]) -> str
 |
 |  Create a new string object from the given object. If encoding or
 |  errors is specified, then the object must expose a data buffer
 |  that will be decoded using the given encoding and error handler.
 |  Otherwise, returns the result of object.__str__() (if defined)
 |  or repr(object).
 |  encoding defaults to sys.getdefaultencoding().
 |  errors defaults to 'strict'.
 |
 |  Methods defined here:
 |
 |  __add__(self, value, /)
 |      Return self+value.
 |
```

In [113]:
```
var='abc'.encode()    #  result byte
```

In [114]:
```
var.decode()
```

Out[114]:
```
'abc'
```

In [ ]:
```
python 2.x
==========
range(4) --->    [0,1,2,3]    list
range(2,4) ---> [2,3]
range(1,5,2)--> [1,3]
python 3.x
==========
range(3)-----> range type
```

In [116]:
```
type(range(5))
```

Out[116]:
```
range
```

In [117]:
```
for i in range(5):     #  for(int i=0; i<5; i++) other programming
    print("DATA",i)
```

```
DATA 0
DATA 1
DATA 2
DATA 3
DATA 4
```

In [118]:
```
i=3
complex(i)
```

Out[118]:
```
(3+0j)
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: