```
# Datatypes : int, float, string , bool
```

```
print('Welcome to CIS-660')
```

```
     Welcome to CIS-660
```

```
a = 5
b = 7.5

c = a + b

print(c)
```

```
     12.5
```

```
type(a)
```

```
     int
```

```
type(b)
```

```
     float
```

```
message = 'How are you ?'

print(message)
```

```
     How are you ?
```

```
type(message)
```

```
     str
```

```
greetings = "Welcome to python World!"

print(greetings)
```

```
     Welcome to python World!
```

```
type(greetings)
```

```
     str
```

```
# operators : arithmetic, relational, logical
```

```
#arithmetic : +, -, *, / , //, %,  **

a = 10
b = 3

print(a/b)
```

```
3.3333333333333335
```

```
print(a // b)
```

```
3
```

```
print(a % b)
```

```
1
```

```
print(2 ** 5)
```

```
32
```

```
#relational : <, <=, >, >=, !=, ==
#logical : and, or , not


ticket_price = 15

discount = 5


age = int(input())

if( age >= 65 ) :
  ticket_price = ticket_price - 5

print('Your ticket price ', ticket_price)
```

```
78
Your ticket price  10
```

```
# independent if statements : python goes through each if statement to check if they are true

ticket_price = 15

discount = 5


age = int(input())

if( age >= 65 ) :
  ticket_price = ticket_price - 5

if( age < 15 ) :
  ticket_price = ticket_price - 10

print('Your ticket price ', ticket_price)
```

```
77
Your ticket price  10
```

```
# dependent if-statements : the moment python encouters the true condition, it skips on the r

ticket_price = 15

discount = 5


age = int(input())

if( age >= 65 ) :
  ticket_price = ticket_price - 5

elif( age < 15 ) :
  ticket_price = ticket_price - 10

print('Your ticket price ', ticket_price)
```

```
77
Your ticket price  10
```

```
#arithmetic opeartors -> (+, - , /, //, %, *, **) -> numbers
#logical operators -> (and, or, not, xor)    ->  True, False
#relational operators -> (<, >, ==, !=, <=, >=)
```

```
#not -> negates/flips/toggles the input value
a = False
print(a)
type(a)
```

```
     False
     bool
```

```
b = not a
print(b)
```

```
     True
```

```
#and

print(True  and False)    #True and False -> False
print(False and True)     #False and True -> False
print(False and False)    #False and False -> False
print(True  and True)      #True and True -> True
```

```
     False
     False
     False
     True
```

```
#or

print(True or False)      #True or False -> True
print(False or True)      #False or True -> True
print(True or True)       #True or True  -> True
print(False or False)     #False or False -> False
```

```
     True
     True
     True
     False
```

```
#xor (^)
print(True ^ False)     #True xor False -> True
print(False ^ True)     #False xor True -> True
print(True ^ True)      #True xor True ->  False
print(False ^ False)    #False xor False -> False
```

```
True
True
False
False
```

```
#relational operators -> used to perform comparison -> returns boolean

a = 10
b = 20


print(a > b)  #is a greater than b

print(a < b)
print(a == b)
print(a == 10)
print(a != b)

print(a > 10)
print(a >= 10)
print(a <= 10)
```

```
False
True
False
True
True
False
True
True
```

```
score = float(input('Enter your Score: '))

grade = ''

if( (score >= 91) and (score <= 100) ): #False and True -> False
  grade = 'A'

elif( (score>=81) and (score < 91)):
  grade = 'B'

elif( (score >=71) and (score <81)):
  grade = 'C'

elif( (score >=61) and (score < 71)):
  grade = 'D'

else :
  grade = 'F'
```

```
#elif( (score>=0) and (score <61)):
  #grade = 'F'


print("Your grade is:", grade)
```

```
     Enter your Score: 34
     Your grade is: F
```

```
#string, list, tuples, sets, dictionaries
```

```
#(1) Mutable : add, delete, update
#(2) Order Preserving : Indexing, slicing, concatenation
```

```
#STRING  : immutable , Order Preserving

s = 'Data Analysis in Python'
print(s)
```

```
     Data Analysis in Python
```

```
#string is a collection of characters

#indexing

print(s[5])
```

```
     A
```

```
#slicing

first_word = s[0:4]

print(first_word)
```

```
     Data
```

```
#concatenation

s1 = 'Data'
s2 = 'Analysis'
```

```
s3 = s1 + ' ' +s2
print(s3)
```

```
Data Analysis
```

```
#immutable
s = 'Data Analysis in Python'
print(s)

s[5] = 'a'

print(s)
```

```
Data Analysis in Python
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-46-135cefa9355b> in <module>()
      2 print(s)
      3
----> 4 s[5] = 'a'
      5
      6 print(s)

TypeError: 'str' object does not support item assignment
```

SEARCH STACK OVERFLOW

```
my_address = "123 Main St, Allendale, Michigan - 49422"
print(my_address)
```

```
123 Main St, Allendale, Michigan - 49422
```

```
print( len(my_address) )
```

```
40
```

```
info = "john smith. he studies at GVSU. he is a freshman"

print(info)

t = info.capitalize()

print(t)
```

```
john smith. he studies at GVSU. he is a freshman
John smith. he studies at gvsu. he is a freshman
```

```
myinfo = "my name is john smith. I study at GVSU. GVSU is a great place to be! GVSU is locate

t = myinfo.count("GVSU")

print(t)
```

```
2
```

```
print( help(myinfo.count) )
```

```
Help on built-in function count:

count(...) method of builtins.str instance
    S.count(sub[, start[, end]]) -> int

    Return the number of non-overlapping occurrences of substring sub in
    string S[start:end].  Optional arguments start and end are
    interpreted as in slice notation.

None
```

```
mytext = "Python is easy to Learn and Code!"

t = mytext.startswith("Python")

print(t)

s = mytext.endswith("Code")
print(s)
```

```
True
False
```

```
mytext = "Python is easy to Learn and easy to Code!"

t = mytext.find("easy", 15)

print(t)

print(help(mytext.find))
```

```
28
Help on built-in function find:

find(...) method of builtins.str instance
    S.find(sub[, start[, end]]) -> int
```

```
        Return the lowest index in S where substring sub is found,
        such that sub is contained within S[start:end].  Optional
        arguments start and end are interpreted as in slice notation.

        Return -1 on failure.

    None
```

```
text= "           python is fun             "

t = text.strip()

print(t)
```

```
python is fun
```

```
mytext = "Python is easy to learn, Python is easy to code"

t = mytext.replace("Python", "Java")

print(t)
```

```
Java is easy to learn, Java is easy to code
```

```
emailid = "sam@gmail.com"

t = emailid.partition("@")

print(t)

print( help(emailid.partition) )
```

```
('sam', '@', 'gmail.com')
Help on built-in function partition:

partition(sep, /) method of builtins.str instance
    Partition the string into three parts using the given separator.

    This will search for the separator in the string.  If the separator is found,
    returns a 3-tuple containing the part before the separator, the separator
    itself, and the part after it.

    If the separator is not found, returns a 3-tuple containing the original string
    and two empty strings.

None
```

```
path = "C:\\username\\foldername\\filename.txt"

t = path.split("\\")
```

```
print(t)

print( help(path.split) )
```

```
['C:', 'username', 'foldername', 'filename.txt']
Help on built-in function split:

split(sep=None, maxsplit=-1) method of builtins.str instance
    Return a list of the words in the string, using sep as the delimiter string.

    sep
      The delimiter according which to split the string.
      None (the default value) means split according to any whitespace,
      and discard empty strings from the result.
    maxsplit
      Maximum number of splits to do.
      -1 (the default value) means no limit.

None
```

```
#LIST [] : mutable (add, remove, update)
#      : order preserving (indexing, slicing, concatenation)
```

```
A = [3.75, 4.0, 2.8, 3.3, 4, 'Sam']

print(A)

print(type(A))
```

```
[3.75, 4.0, 2.8, 3.3, 4, 'Sam']
<class 'list'>
```

```
X = [10, 30, 20, 40, 35]
print(X)
```

```
[10, 30, 20, 40, 35]
```

```
# (1) append
X.append(100)
print(X)
```

```
[10, 30, 20, 40, 35, 100]
```

```
# (2) insert

X.insert(2, 50)
print(X)
```

```
[10, 30, 50, 20, 40, 35, 100]
```

```
# (3) pop
X.pop()
print(X)
```

```
[10, 30, 50, 20, 40, 35]
```

```
X.remove(30)
print(X)
```

```
[10, 50, 20, 40, 35]
```

```
print(X)
```

```
[10, 50, 20, 40, 35]
```

```
print(X[2])
```

```
20
```

```
X[0] = 1000

print(X)
```

```
[1000, 50, 20, 40, 35]
```

```
print(X)
```

```
[1000, 50, 20, 40, 35]
```

```
Y = X[1:4]
print(Y)
```

```
[50, 20, 40]
```

```
Z = X + Y
print(Z)
```

```
[1000, 50, 20, 40, 35, 50, 20, 40]
```

```
X.append(50)
print(X)
```

```
     [1000, 50, 20, 40, 35, 50]
```

```
c = X.count(50)
print(c)
```

```
     2
```

```
print(X)
```

```
     [1000, 50, 20, 40, 35, 50]
```

```
i = X.index(35, 0, 5)

print(i)
```

```
     4
```

```
print(help(X.index))
```

```
     Help on built-in function index:

     index(value, start=0, stop=9223372036854775807, /) method of builtins.list instance
         Return first index of value.

         Raises ValueError if the value is not present.

     None
```

```
print(X)
```

```
     [50, 35, 40, 20, 50, 1000]
```

```
X.reverse() #in-place : original-variable is affected
print(X)
```

```
     [1000, 50, 20, 40, 35, 50]
```

```
print(len(X), min(X), max(X))
```

```
     6 20 1000
```

```
#TUPLE () : immutable
#         : order preserving (indexing, slicing, concatenation)



T = (10, 20, 54, 34, "Sam")
```

```
print(T)
print(type(T))
```

```
     (10, 20, 54, 34, 'Sam')
     <class 'tuple'>
```

```
print(T[2])
```

```
     54
```

```
S = T[2 : 4]
print(S)
```

```
     (54, 34)
```

```
U = S + T
print(U)
```

```
     (54, 34, 10, 20, 54, 34, 'Sam')
```

```
c = U.count(34)
print(c)
```

```
     2
```

```
i = U.index(10)
print(i)
```

```
     2
```

```
# Set - {}
#unique (no duplicates)
#mutable (add, update, delete)
#non-order_preserving
```

```
S = {1, 1, 1, 2, 3, "Sam", 5.4}
print(S)
```

```
     {1, 2, 3, 5.4, 'Sam'}
```

```
#print(S[2])
```

```
S.add(10)
print(S)
```

```
     {1, 2, 3, 5.4, 'Sam', 10}
```

```
S.add(2)
print(S)
```

```
     {1, 2, 3, 5.4, 'Sam', 10}
```

```
x = S.pop()
print(x)
```

```
     1
```

```
print(S)
```

```
     {2, 3, 5.4, 'Sam', 10}
```

```
S.remove(5.4)
print(S)
```

```
     {2, 3, 'Sam', 10}
```

```
A = {1,7,5}
B = {8, 2, 6, 1, 5, 4}
D = {10,20,30}
```

```
C = A.intersection(B)

print(C)
```

```
     {1, 5}
```

```
U = A.union(B)

print(U)
```

```
     {1, 2, 4, 5, 6, 7, 8}
```

```
A = {1,7,5}
B = {8, 2, 6, 1, 5, 4}

#Find all the elements in B that are not in A

D = B.difference(A)
```

```
print(D)

     {8, 2, 4, 6}
```

```
#Dictionary - { key : value }
#unique keys, duplicate values
#Mutable : (add, update, delete)
#non-order_preserving


personInfo = {  'name': 'John' ,
                'age' : 24 ,
                'gpa' : 3.5
              }

print(personInfo)
print(type(personInfo))

     {'name': 'John', 'age': 24, 'gpa': 3.5}
     <class 'dict'>


personInfo['address'] = '123 Main St, MI'


print(personInfo)

     {'name': 'John', 'age': 24, 'gpa': 3.5, 'address': '123 Main St, MI'}


del personInfo['address']

print(personInfo)

     {'name': 'John', 'age': 24, 'gpa': 3.5}


personInfo['gpa'] = 3.8

print(personInfo)

     {'name': 'John', 'age': 24, 'gpa': 3.8}


val= personInfo.get('age')
print(val)

     24


# membership operators : in, not in


a = [10, 20, 15, 'Sam', True, 54]
```

```
print(16 in a)
```

```
False
```

```
customerInfo = {
                  'name' : 'John',
                  'age'  : 23,
                  'gpa'  : 3.5,
                  'email' : 'john@gmail.com'
              }
```

```
print('age' in customerInfo)
```

```
True
```

```
#Loops : while, for
```

```
i = 1

print(i)

i = i + 1

print(i)

i = i + 1

print(i)
```

```
1
2
3
```

```
i·=·1

while(i·<·11)·:
··print(i)
··i·=·i·+·1

··if(i·==·5):
····break

print('Done!')
```

```
1
2
```

```
3
4
Done!
```

```
i = 0

while(i < 11) :

  i = i + 1

  if(i == 5):
    continue

  print(i)



print('Done!')
```

```
1
2
3
4
6
7
8
9
10
11
Done!
```

```
i·=·1

while(i·<·11)·:
··print(i)
··i·=·i·+·1

print('Done!')
```

```
for i in range(1, 11):
  print(i)
```

```
1
2
3
4
5
6
7
8
```

```
     9
     10
```

```
i = 1

while(True) :    #infinite
  print(i)
  i = i + 1

  if(i == 5):
    break

print('Done!')
```

```
     1
     2
     3
     4
     Done!
```

```
#keep asking the student for his/her scores until the student enters 0
#display the total of all the scores

total = 0

while(True):

  score = float(input("Enter your scores (0 to quit) :"))

  if(score == 0):
    break

  total = total + score

print('Your total is:', total)
```

```
     Enter your scores (0 to quit) :76
     Enter your scores (0 to quit) :98
     Enter your scores (0 to quit) :45
     Enter your scores (0 to quit) :0
     Your total is: 219.0
```

```
#String : collection of characters

#(1) Approach 1  : index based

s = "Python Language"

for i in range(0, len(s) ):
  print(i , s[i])
```

```
0 P
1 y
2 t
3 h
4 o
5 n
6  
7 L
8 a
9 n
10 g
11 u
12 a
13 g
14 e
```

```
#(2) Approach 2  : element based

s = "Python Language"

for c in s:
  print(c)
```

```
P
y
t
h
o
n

L
a
n
g
u
a
g
e
```

```
mylist = [10, 30, 'Sam', 25, 'John']

for i in range(0, len(mylist)):
  print(i, mylist[i])
```

```
0 10
1 30
2 Sam
3 25
4 John
```

```
mylist = {10, 30, 'Sam', 25, 'John'}
```

```
for ele in mylist :
  print(ele)
```

```
10
John
25
Sam
30
```

```
info = {
    'name' : 'John Smith',
    'age'  : 23,
    'address' : '123 Main St'
}
```

```
for k in info.keys():
  print(k)
```

```
name
age
address
```

```
for v in info.values():
  print(v)
```

```
John Smith
23
123 Main St
```

```
for p in info.items():
  print(p)
```

```
('name', 'John Smith')
('age', 23)
('address', '123 Main St')
```

```
#format the output - using print function


#a] ',' delimiter

myname = "John Smith"
myage = 23

print("Hello, my name is" , myname , "and I am" , myage ,"years old")
```

```
     Hello, my name is John Smith and I am 23 years old
```

```
#b] '+' operator

myname = "John Smith"
myage = 23

print("Hello, my name is " + myname + " and I am " + str(myage) + " years old")
```

```
     Hello, my name is John Smith and I am 23 years old
```

```
#d] f-string approach

myname = "John Smith"
myage = 23
mygpa = 3.8762109


print(f"Hello my name is {myname} and I am {myage} years old. My current GPA is {mygpa : .2f}
```

```
     Hello my name is John Smith  and I am 23 years old. My current GPA is  3.88
```

```
#e] format() approach


myname = "John Smith"
myage = 23
mygpa = 3.8762109

print("Hello my name is {0} and I am {1} years old. My current GPA is {2:.3f}".format(myname,
```

```
     Hello my name is John Smith and I am 23 years old. My current GPA is 3.876
```

```
#c] '%'
#  %s (string), %d (int), %f (float), %c (char)

myname = "John Smith"
myage = 23
mygpa = 3.8762109

print("Hello, my name is %s , and I am %d years old. My current GPA is %.2f"%(myname, myage,
```

```
     Hello, my name is John Smith , and I am 23 years old. My current GPA is 3.88
```

```
#Functions

def check_odd_even(num):

  if(num % 2 == 0):
    print(num,'·is·even')
··else:
    print(num, ' is odd')



check_odd_even(6)
```

```
6  is even
```

```
def circle_area(radius):

  pi = 3.14
  area = pi * radius * radius

  return area


result = circle_area(10)
print(result)
```

```
314.0
```

✓ 0s completed at 7:16 AM