```
#question 1: python is a programming language which is very easy to understand and write.
# key features of python are:
"""it is widely used in industries
it is used in data industry
it has a alot of libraries
automation
iamge processing
large active community
versatality"""
```

```
'it is widely used in industries\nit is used in data industry\nit has a alot of libraries\nautomation\niamge processing \nlarge act
```

```
#question 2:Predefined keywords: these are predefined words that hold aspecial meaning and have specific purpose in python  programming
#Keyword are case sensitive meaning that they cannot be redefined or overridden in python programming
# to find all the keywords write:
help ("keywords")
```

```
Here is a list of the Python keywords.  Enter any keyword to get more help.

False               class               from                or
None                continue            global              pass
True                def                 if                  raise
and                 del                 import              return
as                  elif                in                  try
assert              else                is                  while
async               except              lambda              with
await               finally             nonlocal            yield
break               for                 not
```

```
#indentation: it is a specific way of writing python code to make it more readeble
#statement: it is a fundamental block of code
#types of statement: exprssion,assignmnet,conditon,loop
```

```
a = 5 #assignment
b = 7 #assignment
```

```
a + b #expression
```

```
12
```

```
#examples of keywords
a = "abhay" #it is an example of srting(characters)
```

```
type(a)
```

```
str
```

```
a = 2 #it is an example of integer
```

```
type(a)
```

```
int
```

```
b = 4.3 # it is an example of float(decimal)
```

```
type(b)
```

```
float
```

```
a = True # it is an example of true and false (boolean)
```

```
type(a)
```

```
bool
```

```
b = False
```

```
a - b # in python true - false = 1
```

```
1
```

```
a * b # in python true * false = 0
```

```
0
```

```
a = None
```

```
type(a)
```

```
NoneType
```

```
com = 5 + 7j #it is an examole of complex number
```

```
type(com)
```

```
complex
```

```
#question 3:mutability: obejcts / containerwhose state or value can be changed after they are created are known as mutable objects
#list is a type of mutable object
#it is also called item assignment
list_cont = [1,4,7,8,10 +4j,"abhay","prime"]
```

```
list_cont
```

```
[1, 4, 7, 8, (10+4j), 'abhay', 'prime']
```

```
list_cont[0] #example of positive indices
```

```
1
```

```
list_cont[5]#example of positive indices
```

```
'abhay'
```

```
#positive indices starts with 0
#negative indices starts with -1
```

```
list_cont[-1] #example of negative indices
```

```
'prime'
```

```
list_cont[-3] #example of negative indices
```

```
(10+4j)
```

```
list_cont[4] = 10 # changing the vlue in list
```

```
list_cont
```

```
[1, 4, 7, 8, 10, 'abhay', 'prime']
```

```
list_cont[-2] = "optimus" # changing the vlue in list
```

```
list_cont
```

```
[1, 4, 7, 8, 10, 'optimus', 'prime']
```

```
#immutability: obects/container whose value cannot be changed after they are created are called immutable obe
#string is a type of immutable object
a = "abhay"
```

```
a[0] #example of positive indices
```

```
'a'
```

```
a[2] #example of positive indices
```

```
'h'
```

```
a[1] #example of positive indices
```

```
a[-1]#example of negative indices
```

```
a[-2]#example of negative indices
```

```
#question 4:python operators: special keyword/symbols that are used to perform oprations on value or variables
#why:because we want to manage, do computation and make decision using data
#types of operators:airthmatic,comparision,logical,bitwise,assignment,membership
```

```
a = 5
b = 3
a + b #airthmatic operator
```

```
8
```

```
a - b#airthmatic operator
```

```
2
```

```
a * b#airthmatic operator
```

```
15
```

```
a / b#airthmatic operator
```

```
1.6666666666666667
```

```
a % b#airthmatic operator
```

```
2
```

```
a ** 3#airthmatic operator
```

```
125
```

```
#comparision operator:to compare two values and return a booean value
# in this == true, != false
```

```
2 == 2 #example of comparision operator
```

```
True
```

```
2 != 2#example of comparision operator
```

```
False
```

```
10 > 20#example of comparision operator
```

```
False
```

```
10 < 20#example of comparision operator
```

```
True
```

```
#assignment operator:in this operator we assign a value
a = 3
x = 1
c = 12
```

```
a #assignment operator
```

```
3
```

```
x#assignment operator
```

```
1
```

```
c#assignment operator
```

```
12
```

```
a += 4#assignment operator addition
```

```
a
```

```
7
```

```
a /= 7#assignment operator devision
```

```
a
```

```
1.0
```

```
a *=9#assignment operator multiplication
```

```
a
```

```
9.0
```

```
#Membership operator:
a = "bumblebee"
```

```
"c" in a#Membership operator:
```

```
False
```

```
"e" in a#Membership operator:
```

```
True
```

```
"a" in a#Membership operator:
```

```
False
```

```
"m" in a#Membership operator:
```

```
True
```

```
"a" not in a #Membership operator:
```

```
True
```

```
"w" not in a#Membership operator:
```

```
True
```

```
"u" not in a#Membership operator:
```

```
False
```

```
#identify operator : compares the location of two objects/variables
a = 5
b = 2
```

```
a
```

```
5
```

```
b
```

```
2
```

```
a is b#identify operator
```

```
False
```

```
a is not b#identify operator
```

```
True
```

```
a = 5#identify operator
b = a#identify operator
```

```
a is b#identify operator
```

```
True
```

```
# bitwise operator: operations at bit level manipulating individual bits within integer
bin(2)
```

```
'0b10'
```

```
#bin of 2 is 10
```

```
#type os bitwise operator:
# and(&) bitwise operator
10 & 7
```

```
2
```

```
11 & 4# and(&) bitwise operator
```

```
0
```

```
1234 & 132424 # and(&) bitwise operator
```

```
1088
```

```
# not bitwise operator represented by tilda(~)
# it gives one lower value with a negative (-) sign
~2
```

```
-3
```

```
~23 # not bitwise operator
```

```
-24
```

```
# bitwise xor operator : it is represented by ^(cap)
5 ^ 2
```

```
7
```

```
1 ^ 1 # bitwise xor operator
```

```
0
```

```
2 ^ 4 # bitwise xor operator
```

```
6
```

```
#shift bitwise operator : left shift and right shift
#left shift: shifts the bits to  the left by a specified no of positions,filling zeroes of the right
# represented by(<<)
23 << 2
```

```
92
```

```
34 << 45 #left shift
```

```
1196268651020288
```

```
#right shift operator: remove the number of elements in the binary
# represented by (>>)
452 >> 23
```

```
0
```

```
234 >> 234 #right shift operator
```

```
0
```

```
#question 5: type casting: the process of changing the data type of a value/object in python is known as type casting
#why : while executing computation using operator there can be a mismatch between the data types
a = "4"
```

```
b = 4
```

```
a + b
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-126-bd58363a63fc> in <cell line: 1>()
----> 1 a + b

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

```
#to convert string to integer
int(a)
type(a)
```

```
str
```

```
v = int(a)
```

```
v + b
```

```
8
```

```
type(v)
```

```
int
```

```
# integer to float
a = 4
```

```
type(a)
```

```
int
```

```
b = float(a)
```

```
b
```

```
4.0
```

```
type(b)
```

```
float
```

```
#note: character cannot be coverted into number
```

```
#string to float
a =  "4.5"
```

```
type(a)
```

```
str
```

```
float(a)
```

```
4.5
```

```
c = float(a)
```

```
c
```

```
4.5
```

```
type(c)
```

```
float
```

```
#integer to string
a = 2
```

```
type(a)
```

```
int
```

```
str(a)
```

```
'2'
```

```
z = str(a)
```

```
z
```

```
'2'
```

```
type(z)
```

```
str
```

```
#there are two types of type casting:implicit,explicit type casting
#all the above are examples are of explicit type casting
#implicit type casting: python understand the data type
```

```
#concatination of string : combining the string
a ="abhay"
```

```
b = " pratap singh"
```

```
a + b
```

```
'abhay pratap singh'
```

```
# to  convert into bool
bool(0)
```

```
False
```

```
bool(2)
```

```
True
```

```
bool(23)
```

```
True
```

```
#any value above 1 will be treated as true
```

```
bool("abhay")
```

```
True
```

```
bool("beast")
```

```
True
```

```
bool("")
```

```
False
```

```
#question 6
#condition statement:it helps you to code decisions based on some preconditions
#example: i will play football when whether is rainy
#type of condition:if,if else,if elif else,nested if else
```

```
#if statements:syntax:if condition is true block of code will be executed
```

```
temp = 9
if temp < 10:
  print("it is very cold")
```

```
it is very cold
```

```
time = 12
if time < 10:
  print("hi")
```

```
time = 2
if time < 10:
  print("hi")
```

```
hi
```

```
# for multiple conditions
number = 22
if number % 2 == 0:
  print("even")
else:
  print("odd")
```

```
even
```

```
grade = 8
co_cerr = True
if (grade > 7) and co_cerr == True:
  print("passed")
else:
  print("failed")
```

```
passed
```

```
grade = 4
co_cerr = True
if (grade > 7) and co_cerr == True:
  print("passed")
else:
  print("failed")
```

```
failed
```

```
weather = "sunny"
if weather == "sunny":
  print("i will play tennis")
else:
    print("i will play cricket")
```

```
i will play tennis
```

```
weather = "rainy"
if weather == "sunny":
  print("i will play tennis")
else:
    print("i will play cricket")
```

```
i will play cricket
```

```
#in real life we can have multiple conditions
a = 99
if a < 100:
  print("this block of code will be executed")
elif a >88:
    print("this number is huge")
else:
  print("this number is not equal to 100")
```

```
this block of code will be executed
```

```
a = 999
if a < 100:
  print("this block of code will be executed")
elif a >88:
```

```
    print("this number is huge")
else:
  print("this number is not equal to 100")
```

```
this number is huge
```

```
a = 123123
if a < 100:
  print("this block of code will be executed")
elif a >8237478:
    print("this number is huge")
else:
  print("this number is not equal to 100")
```

```
this number is not equal to 100
```

```
#question7
#loop statement: it allows you to execute a block of block of code repeatedly
#types of loop statement: while loop, for loop
#while loop:if repetedly executes a block of code until a condition is met
#syntax: while is a keyword used for a while loop
n = 6
i = 1
while i < n:
  print(i)
  i = i + 1
```

```
1
2
3
4
5
```

```
a = 10
i = 5
while i < a:
  print(i)
  i = i + 1
```

```
5
6
7
8
9
```

```
a = 10
i = 5
while i < a:
  print(i)
  i = i + 1
else:
  print("this block of code will be executed")
```

```
5
6
7
8
9
this block of code will be executed
```

```
#break: terminates or exits the loop
n = 7
i = 2
while i < n:
  print(i)
  i = i + 1
  if i == 5:
    break
```

```
2
3
4
```

```
#continue:skips the iteration
n = 7
i = 1
while i < n:
  i = i + 1
  if i == 3:
    continue
```

```
    print(i)
else:
  print("this block of code will be executed")
```

```
this block of code will be executed
```

```
#for loop:iterate over a sequence of elements
for i in "abhay":
  print(i)
```

```
a
b
h
a
y
```

```
l = [1,2,3,4,5,6,"abhay","prime"]
for i in l:
  print(i)
```

```
1
2
3
4
5
6
abhay
prime
```

```
#break statement in for loop
for i in l:
  if i == "abhay":
    print(i)
else:
  print("this block of code will be executed")
```

```
abhay
this block of code will be executed
```

```
#continue statement in for loop
for i in l:
  if i == "abhay":
    continue
    print(i)
else:
  print("this block of code will be executed")
```

```
this block of code will be executed
```

```
#to use range in for loops
for i in range(10):
  print(i)
```

```
0
1
2
3
4
5
6
7
8
9
```

```
for i in range(10):
  print(i, end ="")
```

```
0123456789
```

```
for i in range(10):
  print(i, end ="  ")
```

```
0  1  2  3  4  5  6  7  8  9
```

```
for i in range(10):
  print(i, end =" ")
```

```
0 1 2 3 4 5 6 7 8 9
```

Start coding or generate with AI.