```
#Assignment questions and answers.
```

1) Explain the key features of Python that make it a popular choice for programming.

```
#Some key features of python that make it a popular choice for programming are :-

#1) Python is versatile and flexible.
#2) Python is simple and easy to learn.
#3) Simple syntax easy in readilibilty.
#4) It has extensive library and framework.
#5) Python has strong community support.
#6) Python is used everywhere and is in continuous evolution.
```

2) Describe the role of predefined keywords in Python and provide examples of how they are used in a program.

```
# Role of predefined keywords in Python are :-

#1) It has specific role in defining the syntax and structures of the Python language.
#2) Python keywords are reserved words that cannot be used as variable, function, or identifier names.
#3) Python keywords are case sensitive, and must be written as they are.
#4) Python keywords have specific meanings and predefined uses that cannot be altered.
```

```
#Some examples of predefined keywords are print, true, false, if, else, or, and, from, class, elif, continue, break, for, import, return, not
#example no. 1
a=5
b=6
c=a>b
c
```

```
False
```

```
#example no. 2
money=1000
if money >= 900:
    print("You are eligible")
else:
    print("You are not eligible")
```

```
You are eligible
```

3) Compare and contrast mutable and immutable objects in Python with examples.

```
#MUTABLE OBJECTS
#Objects/container whose state or value can be changed after they are created are called as mutable objects or container and it is called as
#List is a type of mutable object.
#Example:
a=[12, 45, 60, "Shikha", "Vinay", 45]
a
```

```
[12, 45, 60, 'Shikha', 'Vinay', 45]
```

```
a[4]="Riya"
a
```

```
[12, 45, 60, 'Shikha', 'Riya', 45]
```

```
#IMMUTABLE OBJECTS
#Objects/container whose state or value cannot be changed after they are created are called as immutable objects or container.
#Strings is a type of immutable object/doesn't support item assignment.
#Example:
```

```
a="Google Colab"
a
```

```
'Google Colab'
```

```
#it will show error as str are immutable.
a[1]='g'
a
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-15-949499bd2049> in <cell line: 1>()
----> 1 a[1]='g'
      2 a

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

4) Discuss the different types of operators in Python and provide examples of how they are used.

```
#operators are special keywords/symbols that are use to perform operators on value or variables. It is use to manage, do computation and mak
#Different types of operators are:-

#Airthmatic operators - these are used with numeric values to perform common mathematical operations.
#example:
a=12
b=34
c=a+b
c
```

```
46
```

```
c=a-b
c
```

```
-22
```

```
c=a*b
c
```

```
408
```

```
c=b/a
c
```

```
2.8333333333333335
```

```
#Modulus operator - it is used to calculate the remainder of a division operation:
#Example:
c=b%a
c
```

```
10
```

```
c=b**a #it is double asterisk operator (**) in Python's most straightforward way to calculate exponentiation.
c
```

```
2386420683693101056
```

```
#floor operator - It divides two numbers and rounds the result down to the nearest whole number, or integer.
#Example:
a=5
b=7
c=b//a
c
```

```
1
```

```
#Comparison operator - are used to compare two values and return a true or false result. The six comparison operators in Python are:
# == Equal to, != Not equal to, >: Greater than, >= Greater than or equal to, <: Less than, <= Less than or equal to.
#Examples are :

a=3
b=6
c=a==b
c
```

```
False
```

```
c=a!=b
c
```

```
True
```

```
c=a>=b
c
```

```
False
```

```
c=a<=b
c
```

```
True
```

```
#Logical operator - These are used to combine conditional statements. There are three operator and, or , not.
#Example for 'and' logical operator:

a=True
b=False
c=a and a
c
```

```
True
```

```
c=a and b
c
```

```
False
```

```
c= b and a
c
```

```
False
```

```
c=b and b
c
```

```
False
```

```
#Example for 'or' logical operator:

a=True
b=False
c=a or a
c
```

```
True
```

```
c=a or b
c
```

```
True
```

```
c=b or a
c
```

```
True
```

```
c=b or b
c
```

```
False
```

```
#Example for 'not' logical operator:
#A 'not' operator inverts the input. So True becomes False and False becomes True.

not 10
```

```
False
```

```
not 0
```

```
True
```

```
#Assignment operator - which simply assigns the value of the right-hand operand to the left-hand operand.
#Example:
a=67
a+=4
a
```

```
71
```

```
b=56
b-=44
b
```

```
12
```

```
c=6
c*=7
c
```

```
42
```

```
#Membership Operator - it is used to test if a sequence is presented in an object.
#Example:
a="Google Colab"
'g' in a
```

```
True
```

```
#Identity Operator - it is used to compare the objects, not if they are equal, but if they are actually the same object, with the same memor
#Example:
a=2
b=6
a is b
```

```
False
```

```
a is not b
```

```
True
```

```
#Bitwise operator - it is used to perform bitwise calculations on integers.
#The types of bitwise operators in Python are:
#1)Bitwise AND(&): Sets each bit to 1 if one of the two bits is 1
bin(3&7)
```

```
'0b11'
```

```
#2)Bitwise OR(|): Sets each bit to 1 if only one of the two bits is 1
bin(12|8)
```

```
'0b1100'
```

```
#Bitwise Xor (^): The Xor operation gives 0 as a result when both operands are the same and 1 when operands are different.
bin(3^7)
```

```
'0b100'
```

```
#Bitwise Left Shift: Shifts left by pushing zeros in from the right and letting the leftmost bits fall off
bin(1<<2)
```

```
'0b100'
```

```
#Bitwise right Shift: Shifts the value of the left operand to the right by the number of bits given by the right operand
bin(4<<7)
```

```
'0b1000000000'
```

5) Explain the concept of type casting in Python with examples.

```
#Type casting in Python is the process of changing a variable's data type to another.
#Example:
a='5'
type(a)
```

```
str
```

```
b=3
type(b)
```

```
int
```

```
int(a)+b #here variable (a) data type has been changed to int for execution as int and str cannot be added together.
```

```
8
```

```
# It can be done in two ways:
#1)Implicit casting -it is also known as automatic type conversion, this happens when the system automatically handles the conversion withou
#It usually involves converting a smaller data type to a larger one to avoid data loss.
#For example if you assign the value 2 to a variable a, a will automatically become an integer.
a=2
type(a)
```

```
int
```

```
#Explicit casting - it is also known as manual casting, this is when the programmer manually changes the data type using built-in functions
#There is a chance of data loss if a data type is converted to a smaller one.
#Example:

a='4' # variable is str
b=5   # variable is int
int(a)+b  # manually changed str to int for variable a
```

```
9
```

6) How do conditional statements work in Python? Illustrate with examples.

```
#Conditional statements - helps you to code decisions based on some preconditions. examples - if, if else, if elif else, nested if else.

#if statement:
animal="Zebra"
if animal == "Zebra":
   print("The animal is herbivorus")
```

```
The animal is herbivorus
```

```
#if else statement:
animal="Zebra"
if animal != "Zebra":
```

```
if animal != "zebra":
   print("The animal is herbivorus")
else:
  print("The animal is carnivorus")
```

```
The animal is carnivorus
```

```
#if elif else statement:
prize_money=500
if prize_money >= 500:
   print("You are first")
elif prize_money <= 100:
  print("You are third")
else:
  print("You are second")
```

```
You are first
```

```
#Nested if else statement:
a=8
b=6
if a>5:
  if b>5:
    print("both a and b is greater that 5")
  else:
    print("a and b is not greater than 5")
```

```
both a and b is greater that 5
```

7) Describe the different types of loops in Python and their use cases with examples

```
#loops -  loops are programming constructs that repeat a block of code until a condition is met. There are two main types of loops in Python
#1)while loop - Executes a set of statements as long as a condition is true. The syntax for a while loop is while expression: statements.
#Example:

n=9
i=1
while i<n:
   print(i)
   i=i+1
```

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

```
n=9
i=1
while i<n:
  print(i)
  i=i+1
  if i==3:
    break  #break terminates or exit the loop
else:
  print("this will execute when the while statement will run successfully without any break")
```

```
1
2
```

```
n=9
i=1
while i<n:
   i=i+1
   if i==3:
    continue   #continue skips the iteration of the given condition.
   print(i)
```

```
else:
  print("3 will execute when the while statement will run successfully without any continue")
```

```
2
4
5
6
7
8
9
3 will execute when the while statement will run successfully without any continue
```

```
#2)For loop - Iterates over a sequence, such as a list, tuple, set, dictionary, or string. The syntax for a for loop is for iterating_var in
#Example:

a="Google Colab"
for i in a:
   print(i)
```

```
G
o
o
g
l
e

C
o
l
a
b
```

```
a="Google Colab"
for i in a:
   if i=="e":
     break
   print(i)
else:
    print("this will execute when the while statement will run successfully without any break")
```

```
G
o
o
g
l
```

```
a="Google Colab"
for i in a:
   if i=="e":
    continue
   print(i)
else:
    print("e will execute when the while statement will run successfully without any continue")
```

```
G
o
o
g
l

C
o
l
a
b
e will execute when the while statement will run successfully without any break
```