Practical-1:

Overview of Python, Basic Syntax, Variable Types, Basic Operators,Numbers

Submitted by :- Harsh Singh Roll no :- UE193042

In [1]:
```
print("Welcome in Python's World")
```

```
Welcome in Python's World
```

In [2]:
```
#Assigning Values to variables
a=10
b=20
c=40
print(a)
print(b)
print(c)
```

```
10
20
40
```

In [3]:
```
#Multiple Assignments
a=b=c=1
print(a)
print(b)
print(c)
```

```
1
1
1
```

In [7]:
```
#Python Numbers del
var1=1
var2=10
print(var1)
print(var2)
del var1
print(var2)
```

```
1
10
10
```

In [3]:
```
#Decision Making
#Single Statement Suites
var=100
if(var==100):print("Value of Expression is 100")
print("Good Bye")
```

```
Value of Expression is 100
Good Bye
```

In [5]:

```
#Arithemetical Operations
#Store input number
num1 =input('Enter first number:')
num2=input('Enter second number')
#Add two number
sum=float(num1)+float(num2)
#Subtract two numbers
min=float(num1)-float(num2)
#Multiply two numbers
mul=float(num1)*float(num2)
#Divide two numbers
div=float(num1)/float(num2)
#Display the sum
print('The sum of {0} and {1} is {2}'.format(num1,num2,sum))
#Display the Subtraction
print('The subtraction of {0} and {1} is {2}'.format(num1,num2,min))
#Display the Multiplication
print('The multiplicaton of {0} and {1} is {2}'.format(num1,num2,mul))
#Display the division
print('The Division of {0} and {1} is {2}'.format(num1,num2,div))
```

```
Enter first number:4
Enter second number2
The sum of 4 and 2 is 6.0
The subtraction of 4 and 2 is 2.0
The multiplicaton of 4 and 2 is 8.0
The Division of 4 and 2 is 2.0
```

In [8]:

```
#Python program to find area of triangle
#Three side of triangle is a,b,c:
a=float(input('Enter first side:'))
b=float(input('Enter second side:'))
c=float(input('Enter Third side'))



#calculate the semi-perimeter
s=(a+b+c)/2

#Calculate the area

area=(s*(s-a)*(s-b)*(s-c))**0.5
print('The area of the triangle is %0.2f'%area)
```

```
Enter first side:2
Enter second side:3
Enter Third side4
The area of the triangle is 2.90
```

In [10]:

```
P=int(input("Please enter the value for P:"))
Q=int(input("Please enter the value for Q:"))

temp_1=P
P=Q
Q=temp_1
```

```
print("The Value of P after Swapping is",P)
print("The Value of Q after Swapping is",Q)
```

```
Please enter the value for P:2
Please enter the value for Q:4
The Value of P after Swapping is 4
The Value of Q after Swapping is 2
```

In [11]:

```
#Swapping by comma operator
P=int(input("Please enter the value of P:"))
Q=int(input("Please enter the value of Q:"))

#To Swap the values of two variables
P,Q=Q,P
print("The Value of P after Swapping :",P)
print("The Value of Q after Swapping :",Q)
```

```
Please enter the value of P:2
Please enter the value of Q:4
The Value of P after Swapping : 4
The Value of Q after Swapping : 2
```

In [20]:

```
#Python Program to Find Sum of Natural Numbers
num=int(input("Enter the number:"))
if num<0:
    print("Enter the positive number")
else:
    sum=0
    #use while loop to iterate until zero
    while(num>0):
        sum+=num
        num-=1
    print("The sum is ",sum)
```

```
Enter the number:100
The sum is  5050
```

In [19]:

```
#Find Factorial of a given  number
num=int(input("Enter a number"))
factorial=1
if num<0:
    print("Factorial does not exist for negative numbers")
elif num==0:
    print("Factorial of 0 is 1")
else:
    for i in range (1,num+1):
        factorial=factorial*i
    print("The Factorial of",num,"is",factorial)
```

```
Enter a number4
The Factorial of 4 is 24
```

In [22]:

```
#Python Program t Checck if a Number is Odd or Even
num=int(input("Enter a number"))
if(num % 2)==0:
    print("{0} is Even number".format(num))
```

```
else:
    print("{0} is Odd number".format(num))
```

```
Enter a number3
3 is Odd number
```

In [1]:

```
#Comparison operators in Python
x = 10
y = 12

# Output: x > y is False
print('x > y is',x>y)

# Output: x < y is True
print('x < y is',x<y)

# Output: x == y is False
print('x == y is',x==y)

# Output: x != y is True
print('x != y is',x!=y)

# Output: x >= y is False
print('x >= y is',x>=y)

# Output: x <= y is True
print('x <= y is',x<=y)
```

```
x > y is False
x < y is True
x == y is False
x != y is True
x >= y is False
x <= y is True
```

In [2]:

```
#Logical Operators in Python
x = True
y = False

print('x and y is',x and y)

print('x or y is',x or y)

print('not x is',not x)
```

```
x and y is False
x or y is True
not x is False
```

In [3]:

```
#Identity operators in python
x1 = 5
y1 = 5
x2 = 'Hello'
y2 = 'Hello'
x3 = [1,2,3]
y3 = [1,2,3]

# Output: False
```

```
print(x1 is not y1)

# Output: True
print(x2 is y2)

# Output: False
print(x3 is y3)
```

```
False
True
False
```

In [4]:

```
#Membership operators in Python
x = 'Hello world'
y = {1:'a',2:'b'}

# Output: True
print('H' in x)

# Output: True
print('hello' not in x)

# Output: True
print(1 in y)

# Output: False
print('a' in y)
```

```
True
True
True
False
```

In [7]:

```
#Strings in python
str = 'Hey Python!'

print (str)          # Prints complete string
print (str[2])       # Prints first character of the string
print (str[3:5])     # Prints characters starting from 3rd to 5th
print (str[3:])      # Prints string starting from 3rd character
print (str * 4)      # Prints string two times
print (str + "Simple") # Prints concatenated string
```

```
Hey Python!
y
 P
 Python!
Hey Python!Hey Python!Hey Python!Hey Python!
Hey Python!Simple
```

In [9]:

```
#Python Lists
harsh = [ 'efgh', 800 , 4.54, 'harsh', 80.2 ]
singh = [456, 'harsh']

print (harsh)          # Prints complete list
print (harsh[0])       # Prints first element of the list
print (harsh[1:3])     # Prints elements starting from 2nd till 3rd
print (harsh[2:])      # Prints elements starting from 3rd element
```

```
print (singh * 2)  # Prints list two times
print (harsh + singh) # Prints concatenated lists
```

```
['efgh', 800, 4.54, 'harsh', 80.2]
efgh
[800, 4.54]
[4.54, 'harsh', 80.2]
[456, 'harsh', 456, 'harsh']
['efgh', 800, 4.54, 'harsh', 80.2, 456, 'harsh']
```

In [11]:

```
#Python Tuples
tuple = ( 'mghj', 600 , 7.43, 'Unity', 90.5  )
tinytuple = (566, 'Developer')

print (tuple)           # Prints complete tuple
print (tuple[0])        # Prints first element of the tuple
print (tuple[1:3])      # Prints elements starting from 2nd till 3rd
print (tuple[2:])       # Prints elements starting from 3rd element
print (tinytuple * 2)   # Prints tuple two times
print (tuple + tinytuple) # Prints concatenated tuple
```

```
('mghj', 600, 7.43, 'Unity', 90.5)
mghj
(600, 7.43)
(7.43, 'Unity', 90.5)
(566, 'Developer', 566, 'Developer')
('mghj', 600, 7.43, 'Unity', 90.5, 566, 'Developer')
```

In [6]:

```
#Python Dictionary
dict = {}
dict['one'] = "We can make god level Games in Unreal"
dict[2]     = "Unreal is a Game Development Engine"

tinydict = {'name': 'Harsh','code':6734, 'dept': 'sales'}

print (dict['one'])       # Prints value for 'one' key
print (dict[2])           # Prints value for 2 key
print (tinydict)          # Prints complete dictionary
print (tinydict.keys())   # Prints all the keys
print (tinydict.values()) # Prints all the values
```

```
We can make god level Games in Unreal
Unreal is a Game Development Engine
{'name': 'Harsh', 'code': 6734, 'dept': 'sales'}
dict_keys(['name', 'code', 'dept'])
dict_values(['Harsh', 6734, 'sales'])
```

Practical-2: Python Decision Making, Loops, Strings

Python Decision Making

1- if statement

if statement is the most simple form of decision-making statement. It takes an expression and checks if the expression evaluates to True then the block of code in if statement will be executed.

If the expression evaluates to False, then the block of code is skipped.

In [9]:
```
#Example1
a = 20 ; b = 20
if ( a == b ):
  print('a and b are equal')
print('If block ended')
```

```
a and b are equal
If block ended
```

In [10]:
```
#Example2
num = 5
if ( num >= 10):
  print('num is greater than 10')
print('if block ended')
```

```
if block ended
```

In example 1, we see that the condition a==b evaluates to True. Therefore, the block of code inside if statement is executed.

In example 2, the condition evaluates to False, therefore, the print statement was not executed and the only statement that got executed was because it was outside the if block.

Note: Don't forget to add a colon(:) after if statement and indent the statements properly that are executed when a condition is True.

2-: Python if-else statement

From the name itself, we get the clue that the if-else statement checks the expression and executes the if block when the expression is True otherwise it will execute the else block of code. The else block should be right after if block and it is executed when the expression is False.

In [12]:
```
number1 = 40 ; number2 = 40
if(number1 >= number2 ):
  print('number 1 is greater than number 2')
else:
  print('number 2 is greater than number 1')
```

```
number 1 is greater than number 2
```

Note: Only one else statement is followed by an if statement. If you use two else statements after an if statement, then you get the following error.

In [ ]:
```
if (5>10):
  print(5)
else:
  print(10)
```

```
else:
  print('End')
```

Python if-elif ladder

You might have heard of the else-if statements in other languages like C/C++ or Java.

In Python, we have an elif keyword to chain multiple conditions one after another. With elif ladder, we can make complex decision-making statements.

The elif statement helps you to check multiple expressions and it executes the code as soon as one of the conditions evaluates to True.

In [21]:
```
print('Select your ride')
print('1. Bike')
print('2. Car')
print('3. SUV')

choice = int( input() )

if( choice == 1 ):
  print( 'You have selected Bike' )
elif( choice == 2 ):
  print( 'You have selected Car' )
elif( choice == 3 ):
  print( 'You have selected SUV' )
else:
  print('Wrong choice')
```

```
Select your ride
1. Bike
2. Car
3. SUV
1
You have selected Bike
```

Python Nested if statement

In very simple words, Nested if statements is an if statement inside another if statement. Python allows us to stack any number of if statements inside the block of another if statements. They are useful when we need to make a series of decisions.

In [22]:
```
num1 = int( input())
num2 = int( input())

if( num1>= num2):
    if(num1 == num2):
        print(f'{num1} and {num2} are equal')
    else:
        print(f'{num1} is greater than {num2}')
else:
    print(f'{num1} is smaller than {num2}')
```

```
22
34
22 is smaller than 34
```

Loops in Python

While Loop: In python, while loop is used to execute a block of statements repeatedly until a given a condition is satisfied. And when the condition becomes false, the line immediately after the loop in program is executed.

In [23]:
```
# Python program to illustrate
# while loop
count = 0
while (count < 3):
    count = count + 1
    print("Harsh Singh")
```

```
Harsh Singh
Harsh Singh
Harsh Singh
```

In [24]:
```
#Using else statement with while loops:

#Python program to illustrate
# combining else with while
count = 0
while (count < 3):
    count = count + 1
    print("Hello Harsh")
else:
    print("In Else Block")
```

```
Hello Harsh
Hello Harsh
Hello Harsh
In Else Block
```

In [37]:
```
#Single statement while block:
# Python program to illustrate
# Single statement while block
count = 0
while (count == 1): print("Game Developer")
```

For in Loop:

For loops are used for sequential traversal. For example: traversing a list or string or array etc. In Python, there is no C style for loop, i.e., for (i=0; i<n; i++). There is “for in” loop which is similar to for each loop in other languages. Let us learn how to use for in loop for sequential traversals.

In [1]:
```
# Python program to illustrate
```

```
# Iterating over a list
print("List Iteration")
l = ["Harsh","Singh"]
for i in l:
    print(i)

# Iterating over a tuple (immutable)
print("\nTuple Iteration")
t = ("Harsh", "Singh")
for i in t:
    print(i)

# Iterating over a String
print("\nString Iteration")
s = "Harsh"
for i in s :
        print(i)

# Iterating over dictionary
print("\nDictionary Iteration")
d = dict()
d['xyz'] = 123
d['abc'] = 345
for i in d :
    print("%s %d" %(i, d[i]))
```

```
List Iteration
Harsh
Singh

Tuple Iteration
Harsh
Singh

String Iteration
H
a
r
s
h

Dictionary Iteration
xyz 123
abc 345
```

Using else statement with for loops

In [10]:

```
# Python program to illustrate
# combining else with for

list = ['God ', 'is', 'great']
for index in range(len(list)):
    print(list[index])
else:
    print('Inside Else Block')
```

```
God
```

```
is
great
Inside Else Block
```

Nested Loops:

Python programming language allows to use one loop inside another loop.

In [2]:
```
# Python program to illustrate
# nested for loops in Python
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
    for y in fruits:
        print(x, y)
```

```
red apple
red banana
red cherry
big apple
big banana
big cherry
tasty apple
tasty banana
tasty cherry
```

Loop Control Statements: Loop control statements change execution from its normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed. Python supports the following control statements.

Continue Statement: It returns the control to the beginning of the loop.

In [4]:
```
# Prints all letters except 'e' and 's'
for letter in 'HackerDevil':
    if letter == 'e' or letter == 's':
        continue
    print('Current Letter :', letter)
    var = 10
```

```
Current Letter : H
Current Letter : a
Current Letter : c
Current Letter : k
Current Letter : r
Current Letter : D
Current Letter : v
Current Letter : i
Current Letter : l
```

Break Statement: It brings control out of the loop

In [5]:
```
for letter in 'HackerDevil':
```

```
    # break the loop as soon it sees 'e'
    # or 's'
     if letter == 'e' or letter == 's':
         break

print('Current Letter :', letter)
```

```
Current Letter : e
```

Pass Statement: We use pass statement to write empty loops. Pass is also used for empty control statement, function and classes.

In [8]:
```
# An empty loop
for letter in 'HarshSingh':
    pass
print('Last Letter :', letter)
```

```
Last Letter : h
```

Python Strings

Strings in python are surrounded by either single quotation marks, or double quotation marks.

'hello' is the same as "hello".

You can display a string literal with the print() function:

In [9]:
```
print("Hello")
print('Hello')
```

```
Hello
Hello
```

Assign String to a Variable

Assigning a string to a variable is done with the variable name followed by an equal sign and the string:

In [10]:
```
a = "Harsh"
print(a)
```

```
Harsh
```

Multiline Strings

You can assign a multiline string to a variable by using three quotes:

In [12]:
```
a = """Hacking is a Art.Everthing is in air,
```

```
just we require knowledge to grab it,so keep learning and one day you will able access
Hacker Devil
Harsh Singh
"""
print(a)
```

```
Hacking is a Art.Everthing is in air,
just we require knowledge to grab it,so keep learning and one day you will able access e
verthing
Hacker Devil
Harsh Singh
```

Strings are Arrays

Like many other popular programming languages, strings in Python are arrays of bytes representing unicode characters.

However, Python does not have a character data type, a single character is simply a string with a length of 1.

Square brackets can be used to access elements of the string.

In [18]:
```
a = "Game,Developer!"
print(a[0])
print(a[1])
print(a[2])
print(a[3])
print(a[4])
print(a[5])
```

```
G
a
m
e
,
D
```

Looping Through a String

Since strings are arrays, we can loop through the characters in a string, with a for loop.

In [19]:
```
for x in "Kali Linux":
    print(x)
```

```
K
a
l
i

L
```

```
i
n
u
x
```

String Length

To get the length of a string, use the len() function.

In [20]:
```
a = "Developer, World!"
print(len(a))
```

```
17
```

Check String

To check if a certain phrase or character is present in a string, we can use the keyword in.

In [21]:
```
txt = "The best things in life are free!"
if "free" in txt:
  print("Yes, 'free' is present.")
```

```
Yes, 'free' is present.
```

Python - Slicing Strings

Slicing

You can return a range of characters by using the slice syntax.

Specify the start index and the end index, separated by a colon, to return a part of the string

In [22]:
```
b = "Hello, World!"
print(b[2:5])
```

```
llo
```

In [23]:
```
#Slice From the Start
b = "Hello, World!"
print(b[:5])
```

```
Hello
```

In [24]:
```
#Slice To the End
b = "Hello, World!"
print(b[2:])
```

```
llo, World!
```

Python - Modify Strings

Python has a set of built-in methods that you can use on strings.

In [25]:
```
#The upper() method returns the string in upper case:

a = "Hello, World!"
print(a.upper())
```

```
HELLO, WORLD!
```

In [26]:
```
#The lower() method returns the string in lower case:

a = "Hacker, Devil!"
print(a.lower())
```

```
hacker, devil!
```

Remove Whitespace

Whitespace is the space before and/or after the actual text, and very often you want to remove this space.

In [27]:
```
#The strip() method removes any whitespace from the beginning or the end:
a = " Hello, World! "
print(a.strip()) # returns "Hello, World!"
```

```
Hello, World!
```

Python - String Concatenation

String Concatenation To concatenate, or combine, two strings you can use the + operator.

In [29]:
```
a="Game"
b="Developer"
c=a+b
print(c)
```

```
GameDeveloper
```

In [30]:
```
#To add a space between them, add a " ":
a = "Hacker"
b = "Devil"
c = a + " " + b
print(c)
```

```
Hacker Devil
```

Python - Format - Strings

String Format

As we learned in the Python Variables chapter, we cannot combine strings and numbers like this:

In [31]:
```
#we cannot combine strings and numbers like this
age = 19
txt = "My name is Harsh, I am " + age
print(txt)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
C:\Users\HARSHS~1\AppData\Local\Temp/ipykernel_52384/2810912391.py in <module>
      1 age = 19
----> 2 txt = "My name is Harsh, I am " + age
      3 print(txt)

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

In [32]:
```
#But we can combine strings and numbers by using the format() method!
age = 36
txt = "My name is Harsh, and I am {}"
print(txt.format(age))
```

```
My name is Harsh, and I am 36
```

Python - Escape Characters

To insert characters that are illegal in a string, use an escape character.

An escape character is a backslash \ followed by the character you want to insert.

In [33]:
```
txt = "We are the so-called \"Vikings\" from the north."
print(txt)
```

```
We are the so-called "Vikings" from the north.
```