Assignment-1(Python Basics) Submitted by: Shweta Kanungo

Question 1. Explain Python Tokens & Data Types with the help of Example.

A)Tokens:

They are the smallest units of a Python program that the interpreter can understand. Python uses various types of tokens, including keywords, identifiers, literals, operators, and punctuation symbols.

1.Keywords:Keywords are reserved words in Python, which have specific meanings and cannot be used as variable names.Some examples of keywords in Python include "if," "else," "for," "while," "def," and "import."

In [4]:
```
###Example 1

a=4
b=7
if a>b:
    print("False")
else:
    print("True")
```

```
True
```

In [6]:
```
###Example 2

i=7
while i<10:
    print(i)
    i+=1
print("end is the new begining")
```

```
7
8
9
end is the new begining
```

2. Identifiers: Identifiers are names given to variables, functions, classes, or other objects in Python.

In [18]:
```
###Example
Name= "Shweta"
Age= 35

### Shweta and Age are identifier names
```

3.Literals:Literals are constants representing specific values in Python. There are different types of literals, including:

Integer literals (1,2,3...), Floating-point literals(3.14), String literals (Text enclosed in single or double quotes,e.g. "Shweta", "Kanungo"), Boolean literals (True or False)

In [19]:
```
###Example
a=6
b=9.6
Name= "Shweta"
is_valid = True

### These are integer, float, string and boolean literals respectively
```

#B)Operators: Operators are symbols that perform operations on variables and values. Types of operators, such as #Arithmetic #Comparison #Assignment #Logical #Membership #Identity #Bitwise

In [21]:
```
### Example
result = 5 + 3
is_greater = 10 > 5
logical_result = True and False

###Here "+," ">", and "and" are operators.
```

In [30]:
```
#Arithmetic Operators: +,-,*,/,//,%,**
# +(addition), -(Subtraction), *(Multiplication), **(Exponentiation), /(Division) ,
# //(floor division or modulo division which give only integral value), %(Reminder division)


###Example

a=5
b=8

c=b+a
d=b-a
e=b*a
f=b/a
print (c,d,e,f)
```

```
13 3 40 1.6
```

In [36]:
```
# Comparision Operators
# == Equal, != Not equal, >Greater than, <Less than, >=Greater than or equal to, <= Less than or equal to

#Example
a=1
b=9
print(b==a)
print(b!=a)
print(b>a)
print(b<a)
print(b>=a)
print(b<=a)
```

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

In [37]:
```
# Assignment Operators =, +=, *=, %= (Short hand operators e.g. c=c+a can be written as c+=a)
#Example
c=a+b
print(c)
```

```
10
```

In [38]:
```
c+=a
print(c)
```

```
11
```

In [41]:
```
### Logical Operators - and, or, not
#Example
print (a>2 and b<10)
print (a<2 or b<10)
print(not b)
print(not a)
```

```
False
True
False
False
```

In [44]:
```
### Membership operators (in, not in)
#Example
r="Shweta"
s="Rajat"

print("R" in s)
print("S" in r)
print("A" in s)
print("A" not in r)
```

```
True
True
False
True
```

In [45]:
```
### Identity Operator (is, is not)
#Example
Shweta = 1
Rajat = 18

print(Shweta is not Rajat)
print (Rajat is Shweta)
print("Hello", Rajat)
print ("Hello", Shweta)
```

```
True
False
Hello 18
Hello 1
```

B) Data Types and data structure in Python

The data type of a value (or variable) is an attribute that tells what kind of data that value can have. The basic data type are:

1. Integer- Whole numbers like: 0,1,2...etc.
2. Float- Fractional numbers like: 3.14, 4.6, 0.8...etc.
3. Boolean- Binary value(True/False)
4. String- Sequence of Chracters which is written between parentheses or apostrophes e.g. "Hello", "Shweta28", 'Pen', "123"

Data structure in Python are as follows:

1)Strings

2)List

3)Dictionary

4)Tuples

5)Sets

In [59]:
```
#1.Strings-A string is a sequence of characters. String is ordered i.e. sequence matters
#Creating a String
#1) By putting the string in inverted commas ''
#2) By using a function 'str'

#Example

r="Shweta"
s="Rajat"
print(r,s)
type(r)
```

```
Shweta Rajat
```

Out[59]:
```
str
```

In [53]:
```
# 2. List-List is a ordered collection of multiple items stored in a single variable.
# Created by Putting the items in a square bracket
# List are changeable (Mutable) and allow duplicate values.
# Items in the list are indexed, the first item has index [0], the second item has index [1] etc.
#List items can be of any data type: String, int and boolean data types:

#Example

l=["Shweta",28, 6.83, "Sarvesh", True, False, 18+2j]
print(l)
```

```
['Shweta', 28, 6.83, 'Sarvesh', True, False, (18+2j)]
```

In [54]:
```
# 3. Dictionary
#Dictionary is like a list.
#It is an unordered collection of items. Sequence does not matter
#Dictionary is Mutable
#Each item of dictionary has a key / value pair.
#While the values can be of any data type and can repeat, keys must be of immutable type

#Example

dic={'a':500,'b':1000}
print(dic)
```

```
{'a': 500, 'b': 1000}
```

In [58]:
```
# 4. Tuples
#Python tuples are a data structure that store an ordered sequence of values. Tuples are immutable.
#This means you cannot change the values in a tuple. Tuples are defined with parenthesis.
#Individual values in a tuple are called items. Tuples can store any data type.
#Tuples are similar to Python lists, with one big difference: you cannot modify a tuple.

Tup =('a','b')
print(Tup)
type(Tup)
```

```
('a', 'b')
```

Out[58]:
```
tuple
```

In [57]:
```
# 5. Sets
#Set is unordered collection of items. In sets sequence does not matter and indexing has no meaning
#Sets are denoted by curly braces.
#Every set item is unique (no duplicates) and are immutable (cannot be changed).

s={3,5,8,2,1}
print(s)
type(s)
```

```
{1, 2, 3, 5, 8}
```

Out[57]:
```
set
```

Question 2. You're a Data Scientist. You have assigned a task to perform. Create two lists a & b, where a has values -> (11,89,41,32,77,90) and b has values -> (95,24,39,15,74,22). Compare the corresponding element values of both the lists and print "a element value is greater than b element value" if corresponding value from a is greater else print "a element value is less than b element value" if corresponding value from a is less than b.

In [12]:
```
a=[11, 89, 41, 32, 77, 90]
b=[95, 24, 39, 15, 74, 22]

for i in range (len(a)):

        if a[i]>b[i]:
            print(f"{a[i]} is greater than {b[i]}")
        elif a[i]<b[i]:
            print(f"{a[i]} is less than {b[i]}")
        else:
            print(f"{a[i]} is equal to {b[i]}")
```

```
11 is less than 95
89 is greater than 24
41 is greater than 39
32 is greater than 15
77 is greater than 74
90 is greater than 22
```

Question 3. Write code to create a class named Calculator.

a)Define methods for addition, subtraction, multiplication and division that take in two numbers and return another number after Performing their respective operations.

b) Define another method named execute command that takes two numbers and a string named command. The command string will be 'add', 'sub', 'mul', 'div'. Perform Addition, Subtraction, Multiplication and Division on the two numbers respectively. Make sure the execute command method is case insensitive (meaning that even if the user passes 'aDd' as command it should perform addition on two numbers).

In [40]:
```
#a)Define methods for addition, subtraction, multiplication and division that take in two numbers
#and return another number after Performing their respective operations.

n1=int(input("Enter a no:"))
n2=int(input("Enter a no:"))

class Calculator:
    def addition (self, n1, n2):
        return n1+n2
    def subtraction (self, n1, n2):
        return n1-n2
    def multiplication (self, n1, n2):
        return n1*n2
    def division (self, n1, n2):
        if n2==0:
            return "Not allowed"
        return n1/n2

calculator=Calculator()

result_add = calculator.addition(n1, n2)
result_sub = calculator.subtraction(n1, n2)
result_mul = calculator.multiplication(n1, n2)
result_div = calculator.division(n1, n2)


print("Addition", result_add)
print("Subtraction", result_sub)
print("Multiplication", result_mul)
print("Division", result_div)
```

```
Enter a no:28
Enter a no:18
Addition 46
Subtraction 10
Multiplication 504
Division 1.5555555555555556
```

In [28]:
```
#b) Define another method named execute command that takes two numbers and a string named command.
#The command string will be ‘add’, ‘sub’, ‘mul’, ‘div’.
#Perform Addition, Subtraction, Multiplication and Division on the two numbers respectively.
#Make sure the execute command method is case insensitive
#(meaning that even if the user passes ‘aDd’ as command it should perform addition on two numbers).

n1=int(input("Enter a no:"))
n2=int(input("Enter a no:"))
command=str(input("Enter add/sub/mul/div   ")).lower()

class Calculator:
    def execute_command (n1, n2, command):
        if command=="add":
            return n1+n2
        elif command=="sub":
            return n1-n2
        elif command=="mul":
            return n1*n2
        elif command=="div":
            if n2==0:
                return "not allowed"
            else:
                return n1/n2
        else:
            return "Invalid command. Use 'add', 'sub', 'mul', or 'div'."

result = Calculator.execute_command(n1, n2, command)
print("Result:", result)
```

```
Enter a no:28
Enter a no:18
Enter add/sub/mul/div   MUL
Result: 504
```

Question 4. Create a program to take numbers and store them in a list and Print a Square of numbers as tuple.

In [37]:
```python
def get_numbers_and_squares():
    numbers_and_squares = []

    while True:
        user_input = input("Enter a number (or 'stop' to exit): ")

        if user_input.lower() == 'stop':
            break

        try:
            num = float(user_input)
            square = num ** 2
            numbers_and_squares.append((num, square))
        except ValueError:
            print("Invalid input. Enter a number or 'stop' to exit.")

    return numbers_and_squares

result = get_numbers_and_squares()

for num, square in result:
    print(f"The square of {num} is {square}")
```

```
Enter a number (or 'stop' to exit): 18
Enter a number (or 'stop' to exit): 28
Enter a number (or 'stop' to exit): 15
Enter a number (or 'stop' to exit): 24
Enter a number (or 'stop' to exit): stop
The square of 18.0 is 324.0
The square of 28.0 is 784.0
The square of 15.0 is 225.0
The square of 24.0 is 576.0
```

Question 5. Write a Function to check if the year number is a leap year or not. Print all Leap Years of 21st Century.

In [5]:
```python
def is_leap_year(year):
    return (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0)

leap_years = [year for year in range(2000, 2100) if is_leap_year(year)]

print("Leap years in the 21st century:")
for year in leap_years:
    print(year)
```

```
Leap years in the 21st century:
2000
2004
2008
2012
2016
2020
2024
2028
2032
2036
2040
2044
2048
2052
2056
2060
2064
2068
2072
2076
2080
2084
2088
2092
2096
```

Question 6. Write a Function to take an array and return another array that contains the members of the first array that are even in one array and odd in another array.

In [16]:
```python
def oddeven(input_array):
    E=[]
    O=[]

    for i in input_array:
        if i % 2 == 0:
            E.append(i)
        else:
            O.append(i)

    return E, O


input_array = [18,28, 15, 24, 25]

Even, Odd = oddeven(input_array)
print("Even numbers:", Even)
print("Odd numbers:", Odd)
```

```
Even numbers: [18, 28, 24]
Odd numbers: [15, 25]
```

Question 7. Write a python code to print the following pattern.

1

1 2

1 2 3

1 2 3 4

1 2 3 4 5

In [36]:
```python
n=6
for i in range(1,n):
    for j in range(1, i+1):
        print(j, end=" ")
    print("\r")
```

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```