Python Assignment 1

Submitted by : AMIT RAGHAV

Email id : amitraghav3003@gmail.com

**Q1.Why do we call Python as a general purpose and high-level programming language?**

1. General purpose languages are designed for computation not specific to a domain of activity. Usually such a language have no notion of units of measure nor domain specific concepts, but instead it manipulate numbers that can represent anything. A general purpose language is abstract and it has logical notions embedded in the language in the form of keywords operators and punctuation marks.

Examples of such languages are: C, C++, Java, Pascal, Rust. In contrast to such languages there are Domain Specific Languages known as DSL. For example SQL is such a language designed specific for databases and HTML designed for representing web documents.

When compiled, other languages turn into Assembly and run directly in the processor. Hence, being an interpreted language, which is not subject to the processor, makes Python a high-level programming language. Also, Python is a high-level programming language that is known for its ease of readability.

**Q2. . Why is Python called a dynamically typed language?**

Python is both a strongly typed and a dynamically typed language.

Strong typing means that variables do have a type and that the type matters when performing operations on a variable. Dynamic typing means that the type of the variable is determined only during runtime.

Due to dynamic typing, in Python the same variable can have a different type at different times during the execution. Dynamic typing allows for flexibility in programming, but with a price in performance.

**Q3. List some pros and cons of Python programming language?**

The pros of Python:

### Python is easy to learn and read

### Python enhances productivity

### Python has a vast collection of libraries

### Python is free, open-source, and has a vibrant community

### 5. Python is a portable programming language

### The cons of python:

### 1. Python has speed limitations

### 2. Python is not so strong with mobile computing

### 3. Python can have runtime errors

### 4. Python consumes a lot of memory space

### 5. Python is not easy to test

**Q4. In what all domains can we use Python?**

Python is the most versatile programming language amongst developers today. Since Python is reliable and easy to maintain, it becomes the go-to language for software engineers, data scientists, automation engineers, game developers, etc. Some of the domains are listed below:

1. **Data Science**

Data science is an interdisciplinary domain that comprises of three distinct and overlapping areas:

* How to model and summarize data like a statistician
* How to design and use algorithms to store, process, and visualize data like a computer scientist
* How to formulate the right questions and put answers in the right context like a domain expert

1. **Automation**

 Python allows developers to script custom automation and bring efficiency in tasks in less time. Automating repetitive tasks such as sending emails & voicemails, organizing files & folders, launching programs, filling out forms, etc. can be automated through scripting.

1. **Application Development**

Python is a favourable choice for web application development. Python integrates well with other programming languages and has some amazing web frameworks that boost the application development cycle. [Python web frameworks](https://insights.daffodilsw.com/blog/top-10-python-frameworks-for-web-application-development) basically make it easy to build common backend logic.

1. **AI & Machine Learning**

Python’s pre-built libraries are the reason why it is preferred for AI & machine learning development. Numpy for scientific computation, Pybrain for machine learning tasks, Scipy for technical and scientific computing are some of the python libraries that make it a supportive technology for AI and ML development.

1. **Audio/Video Applications**

Python is a sought-after programming language for creating audio/video applications. An example of this is the [Spotify app](https://insights.daffodilsw.com/blog/how-spotify-works-business-model-and-revenue-streams) that’s built using Python programming language

**Q5. What are variable and how can we declare them?**

Python Variable is containers which store values. [Python](https://www.geeksforgeeks.org/python-programming-language/) is not “statically typed”. We do not need to declare variables before using them or declare their type. A variable is created the moment we first assign a value to it. A Python variable is a name given to a memory location. It is the basic unit of storage in a program.

Declaration:

Var **=** "ineuron" (String type)

Age = 21 (int type)

Salary = 25000.0 (float type)

**Q6. How can we take an input from the user in Python?**

**Python input() function** is used to take user input. By default, it returns the user input in form of a string.

Python input() Function Syntax

*Syntax*: input(prompt)

* *prompt [optional]:*any string value to display as input message

Returns: Return a string value as input by the user.

**Q7 What is the default datatype of the value that has been taken as an input using input() function?**

**Python input() function** is used to take user input. By default, it returns the user input in form of a string.

**Q8. What is type casting?**

Type Casting is the method to convert the variable data type into a certain data type in order to the operation required to be performed by users. In this article, we will see the various technique for typecasting.

There can be two types of Type Casting in Python –

* Implicit Type Casting
* Explicit Type Casting

## Implicit Type Conversion

In this,  methods, Python converts data type into another data type automatically. In this process, users don’t have to involve in this process.

**Explicit Type Casting**

In this method, Python need user involvement to convert the variable data type into certain data type in order to the operation required.

Mainly in type casting can be done with these data type function:

* **Int() :**Int() function take float or string as an argument and return int type object.
* **float() :**float() function take int or string as an argument and return float type object.
* **str() :**str() function take float or int as an argument and return string type object.

**Q9. Can we take more than one input from the user using single input() function? If yes, how? If no, why?**

The developer often wants a user to enter multiple values or inputs in one line. In C++/C user can take multiple inputs in one line using scanf but in Python user can take multiple values or inputs in one line by two methods.

* Using split() method
* Using List comprehension

**Using**[**split()**](https://www.geeksforgeeks.org/python-string-split/)**method :**   
This function helps in getting multiple inputs from users. It breaks the given input by the specified separator. If a separator is not provided then any white space is a separator. Generally, users use a split() method to split a Python string but one can use it in taking multiple inputs.

**Syntax :**

input().split(separator, maxsplit)

**Q10. What are keywords?**

Keywords are some predefined and reserved words in Python. We cannot use a keyword as a variable name, function name or any other identifier. Keywords are used to define the syntax of the coding. All the keywords in python are written in lower case except True and False. There are 33 keywords in Python 3.7.

**Q11. Can we use keywords as a variable? Support your answer with reason.**

No, we cannot use keywords as variable names. It's because keywords have predefined meanings, syntax and structure.

**Q12. What is indentation? What's the use of indentaion in Python?**

Indentation is a very important concept of Python because without properly indenting the Python code, you will end up seeing Indentation Error and the code will not get compiled.

## Python Indentation:

Python indentation refers to adding white space before a statement to a particular block of code. In another word, all the statements with the same space to the right, belong to the same code block

Python uses indentation to indicate a block of code.

**Q13. How can we throw some output in Python?**

In Python, we can simply use the print() function to print output. For example,

Print(‘python is powerful’)

#output: python is powerful

Here, the print() function displays the string enclosed inside the single quotation.

**Q14. What are operators in Python?**

Operators are used to perform operations on variables and values.

For example:

Print(5+3) #8

Python divides the operators in the following groups:

* Arithmetic operators
* Assignment operators
* Comparison operators
* Logical operators
* Identity operators
* Membership operators
* Bitwise operators

**Q15. What is difference between / and // operators?**

'/' is the division operator.

'//' is the floor division operator.

Python supports different types of operators:

They are arithmetic operators, logical operators, assignment operators, etc.

'/' and '//' belong to the arithmetic operators.

'/' is used for the normal division of two numbers.

'//' is used to obtain the smallest integer nearest to the quotient obtained by dividing two numbers.

Let us see an example to understand this.

x = 15

y = 3

print(x / y)   #This prints output as 5

print(x // y)  #This prints output as 5

So, if the quotient obtained by dividing two numbers is not an integer, then operators '/' and '//' will return different answers.

**'/' is the division operator. '//' is the floor division operator.**

**Q16. Write a code that gives following as an output.**

**iNeuroniNeuroniNeuroniNeuron**

**Q17. Write a code to take a number as an input from the user and check if the number is odd or even.**

Num = int(input(“enter any number to test whetherit is odd or even: ”))

If (num %2) ==0:

print(“the number is even”)

else

print(“the provided number is odd”)

**Q18. What are boolean operator?**

Boolean operators are words that connect search terms (keywords) to create a logical phrase that a database can understand. They allow you to create a complex search that could include multiple concepts and alternative keywords.

| Explanation of Boolean operators AND, OR, and NOT | | |
| --- | --- | --- |
| **Boolean   Operator** | **What it Does** | **How to use** |
| AND | Finds items that use **both**keywords. | adult learning  **AND**  online courses |
| OR | Finds items that use **either** of the keywords. | adult learners  **OR**  adult students |
| NOT | **Excludes** articles that use the keyword. | **NOT**masters prog |

**Q20. What are conditional statements in Python?**

A conditional statement as the name suggests itself, is used to handle conditions in your program. These statements guide the program while making decisions based on the conditions encountered by the program.

Python has 3 key Conditional Statements that you should know:

* if statement
* if-else statement
* if-elif-else ladder

if Statement

The if statement is a conditional statement in python, that is used to determine whether a block of code will be executed or not. Meaning if the program finds the condition defined in the if statement to be true, it will go ahead and execute the code block inside the if statement.

**Syntax:**

**if condition:**

# execute code block

To better understand this take a look at the below example.

**if-else Statement:**

As discussed above, the if statement executes the code block when the condition is true. Similarly, the else statement works in conjuncture with the if statement to execute a code block when the defined if condition is false.

Syntax:

if condition:

# execute code if condition is true

else:

# execute code if condition if False

**if-elif-else ladder:**

The elif statement is used to check for multiple conditions and execute the code block within if any of the conditions evaluate to be true.

The elif statement is similar to the else statement in the context that it is optional but unlike the else statement, there can be multiple elif statements in a code block following an if statement.

if condition1:

# execute this statement

elif condition2:

# execute this statement

.

.else:

# if non of the above conditions

# evaluate to True

# execute this statement

**Q21. What is use of 'if', 'elif' and 'else' keywords**?

If keyword

Any Boolean expression evaluating to True or False appears after the if keyword. Use the : symbol and press Enter after the expression to start a block with an increased indent. One or more statements written with the same level of indent will be executed if the Boolean expression evaluates to True.

To end the block, decrease the indentation. Subsequent statements after the block will be executed out of the if condition. The following example demonstrates the if condition.

Syntax:

if [boolean expression]:

statement1

statement2

...

statementN

elif keyword

Use the elif condition is used to include multiple conditional expressions after the if condition or between the if and else conditions.

Syntax:

if [boolean expression]:

[statements]

elif [boolean expresion]:

[statements]

elif [boolean expresion]:

[statements]

else:

[statements]

Else keyword

The else keyword is used in conditional statements (if statements), and decides what to do if the condition is False.

The else keyword can also be use in try...except blocks, see example below.

Use the else keyword in a try...except block to define what to do if no errors were raised:

x = 5  
  
try:  
  x > 10  
except:  
  print("Something went wrong")  
else:  
  print("The 'Try' code was executed without raising any errors!")

**Q22. Write a code to take the age of person as an input and if age >= 18 display "I can vote". If age is < 18 display "I can't vote".**

Age=int(input(“enter the age: ”))

If age>=18

print(“I can vote”)

else

print(“I can’t vote”)

**Q23. Write a code that displays the sum of all the even numbers from the given list**

# initializing list

test\_list = [12, 75, 150, 180, 145, 525, 50]

# printing original list

print("The original list is : " + str(test\_list))

even\_sum = 0

for sub in test\_list:

    for ele in str(sub):

        if int(ele) % 2 == 0:

            even\_sum += int(ele)

print("Even digit sum : " + str(even\_sum))

**Q24.Write a code to take 3 numbers as an input from the user and display the greatest no as output.**

def maximum(a, b, c):

    if (a >= b) and (a >= c):

        largest = a

    elif (b >= a) and (b >= c):

        largest = b

    else:

        largest = c

    return largest

# Driven code

a = 10

b = 14

c = 12

print(maximum(a, b, c))

**Q26. What is a string? How can we declare string in Python?**

A string is a sequence of characters.

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

'hello' is the same as "hello".

Declaration of string:

Strings can be created by enclosing characters inside a single quote or double-quotes. Even triple quotes can be used in Python but generally used to represent multiline strings and docstrings.

My\_string = ‘hello’

My\_string2 = “hii”

My\_string3 = ’’’hello, welcome to

ineuron’’’ #multiline string

**Q27. How can we access the string using its index?**

In Python indexing of strings starts from 0 till n-1, where n is the size of string. So characters in string of size n, can be accessed from 0 to n-1.

Suppose we have a string i.e.

Str = ‘hello, this is a sample string’

Lets access the character at 5th index i.e,

Str[5]

**Q30. Resverse the string given in the above question.**

string = "Big Data iNeuron"

def reverse(s):

    str = ""

    for i in s:

        str = i + str

    return str

s = "Big Data iNeuron"

print("The original string is : ", end="")

print(s)

print("The reversed string(using loops) is : ", end="")

print(reverse(s))

**Q31. How can you delete entire string at once?**

**Q32. What is escape sequence?**

Escape sequences allow you to include special characters in strings. To do this, simply add a backslash (\) before the character you want to escape.

For example, imagine you initialized a string with single quotes:

s = 'Hey, whats up?'

print(s)

**Output:**

Hey, whats up?

But if you include an apostrophe without escaping it, then you will get an error:

s = 'Hey, what's up?

print(s)

**Output:**

File "main.py", line 1

s = 'Hey, what's up?'

^

SyntaxError: invalid syntax

To fix this, just escape the apostrophe:

s = 'Hey, what\'s up?'

print(s)

* To add newlines to your string, use \n:

print("Multiline strings\ncan be created\nusing escape sequences.")

**Output:**

Multiline strings

can be created

using escape sequences.

An important thing to remember is that, if you want to include a backslash character in a string, you will need to escape that.

**Q82. Write a Python program to interchange the first and last element in a list**

def swapList(newList):

    size = len(newList)

    # Swapping

    temp = newList[0]

    newList[0] = newList[size - 1]

    newList[size - 1] = temp

    return newList

# Driver code

newList = [12, 35, 9, 56, 24]

print(swapList(newList))

**Q83. Write a Python program to swap two elements in a list**.

def swapPositions(list, pos1, pos2):

    list[pos1], list[pos2] = list[pos2], list[pos1]

    return list

# Driver function

List = [23, 65, 19, 90]

pos1, pos2  = 1, 3

print(swapPositions(List, pos1-1, pos2-1))

**Q84. Write a Python program to find N largest element from a list.**

def Nmaxelements(list1, N):

    final\_list = []

    for i in range(0, N):

        max1 = 0

        for j in range(len(list1)):

            if list1[j] > max1:

                max1 = list1[j];

        list1.remove(max1);

        final\_list.append(max1)

    print(final\_list)

# Driver code

list1 = [2, 6, 41, 85, 0, 3, 7, 6, 10]

N = 2

# Calling the function

Nmaxelements(list1, N)

**Q85. Write a Python program to find cumulative sum of a list.**

list=[10,20,30,40,50]

new\_list=[]

j=0

for i in range(0,len(list)):

    j+=list[i]

    new\_list.append(j)

print(new\_list)

**Q86. Write a Python program to check if a string is palindrome or not.**

def isPalindrome(s):

    return s == s[::-1]

# Driver code

s = "malayalam"

ans = isPalindrome(s)

if ans:

    print("Yes")

else:

    print("No")

**Q87. Write a Python program to remove i'th element from a string.**

# Removes character at index i

def remove(string, i):

    # Characters before the i-th indexed

    # is stored in a variable a

    a = string[ : i]

    # Characters after the nth indexed

    # is stored in a variable b

    b = string[i + 1: ]

    # Returning string after removing

    # nth indexed character.

    return a + b

# Driver Code

if \_\_name\_\_ == '\_\_main\_\_':

    string = "geeksFORgeeks"

    # Remove nth index element

    i = 5

    # Print the new string

    print(remove(string, i))

**Q88. Write a Python program to check if a substring is present in a given string**.

# input strings str1 and substr

string = "geeks for geeks"   # or string=input() -> taking input from the user

substring = "geeks"  # or substring=input()

# splitting words in a given string

s = string.split()

# checking condition

# if substring is present in the given string then it gives output as yes

if substring in s:

    print("yes")

else:

    print("no")

**Q89. Write a Python program to find words which are greater than given length k.**

def string\_k(k, str):

    # create the empty string

    string = []

    # split the string where space is comes

    text = str.split(" ")

    # iterate the loop till every substring

    for x in text:

        # if length of current sub string

        # is greater than k then

        if len(x) > k:

            # append this sub string in

            # string list

            string.append(x)

     # return string list

    return string

# Driver Program

k = 3

str ="geek for geeks"

print(string\_k(k, str))

**Q90. Write a Python program to extract unquire dictionary values.**

# initializing dictionary

test\_dict = {'gfg': [5, 6, 7, 8],

             'is': [10, 11, 7, 5],

             'best': [6, 12, 10, 8],

             'for': [1, 2, 5]}

# printing original dictionary

print("The original dictionary is : " + str(test\_dict))

# Extract Unique values dictionary values

# Using set comprehension + values() + sorted()

res = list(sorted({ele for val in test\_dict.values() for ele in val}))

# printing result

print("The unique values list is : " + str(res))

**Q91. Write a Python program to merge two dictionary**

def Merge(dict1, dict2):

    return(dict2.update(dict1))

# Driver code

dict1 = {'a': 10, 'b': 8}

dict2 = {'d': 6, 'c': 4}

# This returns None

print(Merge(dict1, dict2))

# changes made in dict2

print(dict2)

**Q92. Write a Python program to convert a list of tuples into dictionary.**

def Convert(tup, di):

    for a, b in tup:

        di.setdefault(a, []).append(b)

    return di

# Driver Code

tups [('Sachin', 10), ('MSD', 7), ('Kohli', 18), ('Rohit', 45)]

dictionary = {}

print (Convert(tups, dictionary))

**Q93.Write a Python program to create a list of tuples from given list having number and its cube in each tuple. Input: list = [9, 5, 6] Output: [(9, 729), (5, 125), (6, 216)]**

list1 = [9, 5, 6]

# using list comprehension to iterate each

# values in list and create a tuple as specified

res = [(val, pow(val, 3)) for val in list1]

 print(res)

**Q94. Write a Python program to get all combinations of 2 tuples. Input : test\_tuple1 = (7, 2), test\_tuple2 = (7, 8) Output : [(7, 7), (7, 8), (2, 7), (2, 8), (7, 7), (7, 2), (8, 7), (8, 2)]**

# initializing tuples

test\_tuple1 = (4, 5)

test\_tuple2 = (7, 8)

print("The original tuple 1 : " + str(test\_tuple1))

print("The original tuple 2 : " + str(test\_tuple2))

res =  [(a, b) for a in test\_tuple1 for b in test\_tuple2]

res = res +  [(a, b) for a in test\_tuple2 for b in test\_tuple1]

  print("The filtered tuple : " + str(res))

**Q95. Write a Python program to sort a list of tuples by second item. Input : [('for', 24), ('Geeks', 8), ('Geeks', 30)] Output : [('Geeks', 8), ('for', 24), ('Geeks', 30)]**

def Sort\_Tuple(tup):

     lst = len(tup)

    for i in range(0, lst):

        for j in range(0, lst-i-1):

            if (tup[j][1] > tup[j + 1][1]):

                temp = tup[j]

                tup[j]= tup[j + 1]

                tup[j + 1]= temp

    return tup

tup =[('for', 24), ('is', 10), ('Geeks', 28),

      ('Geeksforgeeks', 5), ('portal', 20), ('a', 15)]

print(Sort\_Tuple(tup))