1Ans: Python is a general-purpose language because it can be used to create a variety of different programs and isn’t specialized for any problems.

Python is a high-level language because it is easy for humans’ to understand

2Ans: Python is a Dynamically typed language because in python the type of variable is determined only during runtime.

3Ans: Pros of Python:

1. Python is easy to learn
2. Python is Dynamically typed and interpreted language.
3. It has vast libraries support and it has increased productivity.

Cons of Python:

1.Python has speed limitations.

2.Python is not memory efficient.

3.Python works weak in mobile computing.

4Ans: Domains in which python used are:

1.Data Science

2.AI & ML

3.Web development

4. OS development and gaming.

5Ans: Variables are the basic unit of storage.

Declaration of Variables:

1.Just name the variables.

2.Assign the required to it.

3.The data type of the variable will be automatically determine from the value assigned, we need not define it explicitly.

6Ans: Using input() function, we take input from users.

Ex: name=input()

Age=input()

Print(“user name: “,name)

Print(“user age: “,age)

7Ans: String, everything that we input() will be a string.

8Ans: The conversion of one datatype into another datatype is called as type casting.

9Ans: Yes, we can take more than one input from the user using single input() function using split() method.

Ex: a, b, c=input(“Enter three values:”).split()

Print(“Total no of students: ”,a)

Print(“No of students passed: “,b)

Print(“No of failed Students: “,c)

10Ans: Keywords are special reserved words that have specific meanings and purpose and can’t be used for anything but can be used for those specific purpose.

11Ans: No, we can’t use keyword as a variable. Because Compiler will get confused whether to take it as a keyword or a variable.

12Ans: Indentation refers to the spaces at the beginning of a code line. Indentation is very important in python because it uses indentation to indicate a block of code.

13Ans: using print() function in python we can throw output in python.

14Ans: Operators are special symbols used to perform some sort of computations. The value that an operator acts on are called operands.

15Ans: x=5

y=2

Print(x/y) 🡪 2.5

Print(x//y) 🡪 2

/ operator returns output in float, whereas // returns output in integer.

16Ans: P=”INeuron”

print(p\*4)

17Ans: n=input(“Enter a number: “)

if n%2==0:

print(“even”)

else:

print(‘odd’)

18Ans: The logical operators and, or an not are also referred to as Boolean operators. Boolean or operator returns true if any one operand is true. The not operator returns true if its operand is a false expression and returns false if it is true. Boolean and operator returns true if both operands are true.

19Ans: 1 or 0 🡪1

0 and 0 🡪0

True and False and True 🡪 False

1 or 0 or 0 🡪 1

20Ans: if , if..else, Nested if, if-elif statements are conditional statements in python.

21Ans: It allows us to check for multiple expressios. If the condition for ‘if’ is false, it checks the condition of the next elif block and so on. If all the conditions are false, the body of else is executed.

22Ans: n=input(“enter age : ”)

if n>=18:

print(“I can vote”)

else:

print(“I can’t vote”)

23Ans: numbers=[12,75,150,180,145,525,50]

Sum=0

For i in numbers:

if i%2==0:

sum=sum+i

print(sum)

24Ans: a, b, c=input(‘enter three values: ’).split()

if(a>b) and (a>c):

print(‘a is greater’)

elif(b> a) and (b>c):

print(“b is greater”)

else:

print(“c is greater”)

25Ans: a=[12,75,150,180,145,525,50]

B=[]

For I in a:

If i>150:

If i>500:

Break

Continue

If i%5==0:

b.append(i)

print(b)

26Ans: A string is a series of characters. Anything inside quotes is a string. And we can declare a string using either single, double or triple quotes.

27Ans: S=”Python”

print(S[0:]) 🡪’Python’

28Ans: string=’Big Data iNeuron’

Print(string[9:]) 🡪’iNeuron’

29Ans: string=’Big Data iNeuron’

P=string[9:]

print(P[-1::-1]) 🡪’norueNi’

30Ans: string=’Big Data iNeuron’

Print(s) 🡪’norueNi ataD giB’

31Ans: string=’Big Data iNeuron’

del string

print(string)

32Ans: An Escape sequence is a sequence of characters that, when used inside a character or string, does not represent itself but is converted into another character or series of characters.

\’ single quote

\\ backslash

\n new line

\b backspace

\t tab space

33Ans:P=’iNeuron\’s Big Data Course’

Print(P)

34Ans: A list is created by placing elements inside square brackets[], separated by comma. A list can have any number of items and they may be different types(integer, float, string etc..,) . a list is a data structure in python that is mutable.

35Ans: Lists in python can be created by just placing the sequence inside the square brackets[].

List=[] 🡪empty list

36Ans: Elements of a list can be accessed by using index.

L=[1,2,3,4,5]

#How to access 3

Print(L[2])

37Ans: print(lst[4][2])

38Ans: lst=list(input().split(‘,’))

Print(len(lst))

39Ans: lst=[‘Welcome’,’to’,’Data’,’course’]

lst.insert(2,’Big’)

print(lst)

40Ans: Tuples are used to store multiple items in a single variable. A tuple is a collection which is ordered and unchangeable. Whereas lists are changeable.

41Ans: Empty\_tuple=()

Empty\_Tuple=tuple()

Tuple\_With\_Values=(1,2,3)

42Ans: No we can’t append anything to tuples as tuples are immutable.

43Ans: No two tuples can’t be appended as tuples are immutable.

44Ans: name=(‘p’,’r’,’a’,’v’,’e’,’e’,’n’)

C=name.count(‘e’)

Print(C) 🡪2

45Ans: Set is a collection which is unordered, unchangeable, and unindexed.

46Ans: A set is created by placing all the items inside curly brace{}, separated by comma , or by using the built-in set() function.

47Ans: p=set()

p.add(‘iNeuron’)

print(p)

48Ans: p=set()

p.add(1)

p.add(2)

print(p)

49Ans: We can use add() method to add single value to a set. Whereas, we can use update() method to add sequence values to a set.

Ex: s={1,2,3}

s.add(8)

print(s) 🡪 {3,8,2,1}

s.update(11,12,14)

print(s) 🡪 {3,11,14,12,2,1}

50Ans: clear() method removes all elements in a set

51Ans: frozenset() method creates an immutable set object from an iterable. It is a built-in python function. As it is a set object therefore we cannot have duplicate values in the frozenset.

L=[‘iNeuron’,’BigData’,’Course’]

M=frozenset(L)

Print(M) 🡪 frozenset({‘iNeuron’,’BigData’,’Course’})

52Ans: frozenset is immutable , whereas set is mutable.

53Ans: Union of two sets is the set which contains all the elements of both the sets such that no element is repeated.

A={1,2,3,4}

B={2,3,4,5}

Print(A.union(B)) 🡪{1,2,3,4,5}

54Ans: Intersection of two sets is the set which contains common elements in both the sets.

A={1,2,3,4}

B={2,3,4,5}

Print(A.intersection(B)) 🡪{2,3,4}

55Ans: Dictionaries are used to store data values in key:value pairs. A dictionary is a collection which is ordered, changeable, do not allow duplicates.

56Ans: Other Data structures are collection of values whereas Dictionary is a collection of key: value pairs.

57Ans: By using dict() function we can create a dictionary. A Dictionary in python is declared by enclosing a comma-separated list of key-value pairs using curly braces({}).

D={}

D=dict()

58Ans: <class ‘dict’>

59Ans: dict1={}

dict1[‘Name’]=’Praveen’

dict1[‘age’]=22

Print(dict1) 🡪{‘Name’:’Praveen’,’age’:22}

60Ans: dict1={}

dict1[‘Name’]=’Praveen’

dict1[‘age’]=22

Print(dict1) 🡪{‘Name’:’Praveen’,’age’:22}

61Ans: dict1={}

dict1[‘Name’]=’Praveen’

dict1[‘age’]=22

dict1[‘other\_details’]={‘college’: ’KU engineering college’, ‘branch’: ‘IT’}

print(dict1[‘other\_details’])

62Ans: get() returns the value of the specified key.

dict1={}

dict1[‘Name’]=’Praveen’

dict1[‘age’]=22

dict1[‘other\_details’]={‘college’: ’KU engineering college’, ‘branch’: ‘IT’}

print(dict1.get(‘Name’)) 🡪 ‘Praveen’

63Ans: items() return the dictionary’s key-value pairs.

dict1={}

dict1[‘Name’]=’Praveen’

dict1[‘age’]=22

Print(dict1.items()) 🡪 dict\_items([(‘Name’,’Praveen’),(‘age’,22)])

64Ans: dict1={}

dict1[‘Name’]=’Praveen’

dict1[‘age’]=22

dict1[‘other\_details’]={‘college’: ’KU engineering college’, ‘branch’: ‘IT’}

print(dict1.pop(‘other\_details’)) 🡪 {‘Name’:’Praveen’,’age’:22}

65Ans: popitems() removes the last inserted key-value pair

dict1={}

dict1[‘Name’]=’Praveen’

dict1[‘age’]=22

dict1[‘other\_details’]={‘college’: ’KU engineering college’, ‘branch’: ‘IT’}

print(dict1.pop.items()) 🡪 {‘Name’:’Praveen’,’age’:22}

66Ans: By using keys() all the keys from the dictionary are returned.

dict1={}

dict1[‘Name’]=’Praveen’

dict1[‘age’]=22

Print(dict1.keys()) 🡪 dict\_keys([(‘Name’,’age’)])

67Ans: dict1={}

dict1[‘Name’]=’Praveen’

dict1[‘age’]=22

Print(dict1.values()) 🡪 dict\_values([’Praveen’,22)])

68Ans: looping means repeating something over and over until a particular condition is satisfied.

69Ans: loops in python:

While loop

For loop

Do-while loop

Nested loop

Break statement

Continue statement

70Ans: for loop is used when we know the number of iterations, whereas while loop is used when the number of iterations is unknown.

71Ans: The continue keyword is used to end the current iteration in a for loop and continues to the next iteration.

# Print numbers greater than 10

L=[1,5,7,8,19,13,17,3]

For i in L:

If(i<10):

Continue

Print(i) 🡪 19,13,17

72Ans: The break keyword is used to break out a for loop, a while loop or a swich block.

L=[1,5,7,8,9,13,17,3]

#Find the even values in the given list

For I in L:

If(I%2==0):

Print(i)

Break

Output: 8

73Ans: The pass statement is used as a placeholder for future code. When the pass statement is executed, nothing happens, but we can avoid getting an error when empty code is not allowed. Empty code is not allowed in loops, function, definitions, class definitions, or in if statements.

A=33

B=200

If B>A:

Pass

74Ans: The range() is an in-built function in python. It returns a sequence of numbers starting from zero and increment by 1 default and stops before the given number.

For num in range(1,4):

Print(num)

Output: 1

2

3

75Ans: dict1={}

dict1[‘Name’]=’Praveen’

dict1[‘age’]=22

for k,v in dict1.items():

print(“key is ”,k,”and value is ”v)

output: key is Name and value is Praveen

key is age and value is 22

76Ans: def factorial(n):

If n==0 | n==1:

Return 1

Result=1

For num in range(1,n+1):

Result=Result\*num

Return result

X=5

Ans=factorial(x)

Print(Ans)

77Ans: p=input(“enter amount: ”)

R=input(“enter the rate of interest: ”)

T=input(“enter time period: ”)

SI=(P\*R\*T)/100

Print(SI)

78Ans: p=input(“enter amount: ”)

R=input(“enter the rate of interest: ”)

T=input(“enter time period: ”)

Amount=P\*(1+(R/100))\*\*T

CI=Amount-P

Print(CI)

79Ans: n=5

If(n>1)

For I in range(2,int(n/2)+1):

If(n%I==0):

Print(n,” is not a prime number”)

Break:

Else:

Print(n,” is a prime number” )

Else:

Print(n,” is not a prime number”)

80Ans: num=int(input(“Enter a number: ”)

Sum=0

Temp=num

While Temp> 0:

Digit=Temp%10

Sum=sum+Digit\*\*3

Temp=Temp//10

If(num==Sum):

Print(num, “ is a Armstrong number”)

Else:

Print(num, “ is not a Armstrong number”)

81Ans: num=int(input(“enter any number: ”))

N1,N2=0,1

Sum=0

If num<=0:

Print(“enter number grater than zero”)

Else:

For I in num(0,num):

Print(sum, ends=” “)

N1=N2

N2=Sum

Sum=N1+N2

82Ans: l=[1,2,3,4]

Size=len(l)

Temp=l[0]

l[0]=l[size-1]

l[size-1]=Temp

print(l) 🡪[4,2,3,1]

83Ans: l=[1,2,3,4]

Temp=l[0]

L[0]=l[1]

L[1]=temp

Print(l) 🡪[2,1,3,4]

84Ans: l=[1,2,3,4]

Print(max(l)) 🡪 4

85Ans: l=[1,2,3,4]

Sum=0

For I in l:

Sum=Sum+I

Print(Sum)

86Ans: l=’aba’

M=l[::-1]

If(l==M):

Print(“the string is palindrome”)

Else:

Print(“the string is not a palindrome”)

87Ans: Removing 1st index element

L=[1,2,3,4]

L.pop(1)

Print(L) 🡪[1,3,4]

88Ans: string=’Praveen Nakkanaboina’

S=’Praveen’

P=string.split()

If S in P:

Print(“Yes”)

Else:

Print(“No”)

89Ans: string=’Praveen Nakkanaboina’

P=string.split()

K=10

For I in p:

If(len(i)>K):

Print(i)

90Ans: dict={‘a’: [1,3,5,4],

‘b’: [4,6,8,10],

‘c’: [6,12,4,8],

‘d’: [5,7,2]}

Res=list(sorted({ele for val in dict.values() for ele in val}))

Print(Res)

91Ans: dict\_1={1:’a’,2:’b’}

Dict\_2={2:’c’,4:’d’}

dict\_1.update(dict\_2)

print(dict\_1) 🡪{1: ‘a’,2: ‘c’,4:’d’}

92Ans: Input=[(‘sachin’,10),(‘MSD’,7),(‘kohli’,18),(‘rohit’,45)]

Output=dict(input)

Print(Output) 🡪{‘sachin’:10,‘MSD’:7,‘kohli’:18,‘rohit’:45}

93Ans: list=[9,5,6]

Res=[(val, val\*\*3) for val in list]

Print(Res) 🡪[(9,729),(5,125),(6,216)]

94Ans: test\_tuple1=str(7,2)

Test\_tuple2=str(7,8)

Res=[(x,y) for x in test\_tuple1 for y in test\_tuple2]

Result=res+[(x,y) for x in test\_tuple2 for y in test\_tuple1]

Print(result)

95Ans: def sort\_tuple(tup):

Tup.sort(key=lambda x:x[1])

Reurn tup

Tup=[(‘for’,24),(‘geeks’,8),(‘geeks’,30)]

Print(sort\_tuple(tup))

96Ans: rows=5

For I in range(rows):

For j in range(i+1):

Print(“\*”,end=” ”)

Print(‘\n’)

97Ans: n=5

I=0

While(i<=n):

Print(“ ”\*(n-i)+”\*”\*i)

I+=1

98Ans: def triangle(n):

K=n-1

For I in range(0,n):

For j in range(0,k):

Print(end=” ”)

K=k-1

For j in range(0,i+1):

Print(“\*”,end=” ”)

Print(“\r”)

N=5

Triangle(N)

99Ans: def numpat(n):

Num=1

For I in range(0,n):

Num=1

For j in range(0, i+1):

Print(num, end=” ”)

Num=num+1

Print(“\r”)

N=5

Numpat(N)

100Ans: def alphabet(n):

Num=65

For I in range(0, n):

For j in range(0,i+1):

Ch=chr(num)

Print(ch, end=’ ’)

Num=num+1

Print(“\r”)

N=5

Alphabet(N)