21/2/23

ASSIGNMENT 1

PAGE NO.: 3
DATE: / /

Unit-1&2 Introduction to Python & Data Structure.

Q.1 What are the benefits of using python ?

—> There are many benefits of using python which are;

- High-level language.
- Portable and Interactive
- Versatile, easy to read, learn and write
- Extensive support libraries
- Dynamically typed language (No need to mention data-type based on the value assigned, it takes data type)
- Highly Efficient.
- User friendly data structures.

Q.2 What are the key features of python ?

—> The key features of python are:

- Easy to learn
- Portable
- Easy to maintain
- Simple
- A broad standard library
- Open Source
- Dynamically Typed
- Platform Independent
- Procedure and object-oriented
- GUI programming
- High level language
- Interactive mode
- Databases.

Q.3 What are keywords in Python?

—↓ Keywords are predefined, reserved words used in python programming that have special meanings to the compiler.

There are 35 keywords in Python 3.8 and below this version there are 32 keywords.

Q.4 What are literals in python and explain about different literals.

—↓ The raw data assigned to variables or constant while programming. We mainly have five types of literals which includes string literals, numeric literals, boolean literals, literal collections and a special literals None.

i) String literals.

Enclosing the text or the group of characters in single, double or triple quotes, creates a string literal. There are two types of string literal in python

(i) single-line string ⇒ written within (' ')

(ii) Multi-line string ⇒ written within (""")

ii) Numeric literals.

Numerical literals are those literals that contain digits only and are immutable. There are four types:

(i) Integer

(ii) Float

(iii) Complex

(iv) long

iii) Boolean literals

Boolean literals are pretty straight-forward and have only two values:-

(i) True : represents the value 1

(ii) False : represents the value 0.

Basically it is used for comparison

iv) Special literals

Special literals are known as None, it is used to signify that a particular field is not created. Python will print None as output when we print the variable with no value assigned to it

example: variable_name = None
print (variable_name)

v) literal Collection

If you want to work with more than one value, there is literal collection for you. It has basically four types.

(i) list literals

(ii) Tuple literals

(iii) Dictionary literals

(iv) Set literals

Q.5 What is type conversion python?

→ The process of converting data of one type to another.

There are two types of type conversion:

i) Implicit Conversion : Automatic type conversion.

ii) Explicit Conversion : Manual type conversion.

Q.6 Is indentation required in python?

—↳ Yes indentation is required in python and it is very important because python uses indentation to indicate a block of code.

Q.7 How do you write comments in python?

—↳ Comments in python are identified with a hash symbol '#' and extend to the end of the line. Also, String literals can be used to give comments, only if they are not assigned to any variable.

Q.8 What are docstring in python?

—↳ A python docstring is a string used to document a python module, class, function or method. The docstring are declared using "Triple single Quotes" or """ Triple double Quotes """ just below the class, method or function declaration.

Q.9 What are the common built-in data types in python?

—↳ The five standard data-types are:

i) Numbers	Numeric (int, float, complex)
ii) String	String (str)
iii) list	Sequence (list, tuple, range)
iv) Tuple	Binary (bytes, bytearray, memory view)
v) Dictionary	Mapping (dict)
	Boolean (bool)
	Set (set, frozenset)

Q.10 Explain the use of logical operator and membership operators.

→ logical operator

They are used on condutional statements (either True or False). They perform logical AND, logical OR and logical NOT operations.

and	True if both the operands are True	(x and y)
Or	True if either of the operands is True	(x or y)
not	True if operand is false	(not x)

→ Membership operator

They are used to test if a sequence is presented in an object.

in	Returns True if a sequence with the specified value is present in object	(x in y)
not in	Returns True if a sequence with the specified value is not present in object	(x not in y)

Q.11 write a python program to interpreter as calculator.

→
```
a = int(input("Enter first number: "))
op = input("Enter the operator (+,-,*,/,%,**,//): ")
b = int(input("Enter second number: "))

if op == '+':
    print("{0} {1} {2} = {3}".format(a,op,b,a+b))
elif op == '-':
    print("{0} {1} {2} = {3}".format(a,op,b,a-b))
elif op == '*':
    print("{0} {1} {2} = {3}".format(a,op,b,a*b))
```

```
elif op == '/':
    print ("{0} {1} {2} = {3}".format(a, op, b, a/b))
elif op == '%':
    print ("{0} {1} {2} = {3}".format(a, op, b, a%b))
elif op == '**':
    print ("{0} {1} {2} = {3}".format(a, op, b, a**b))
elif op == '//':
    print ("{0} {1} {2} = {3}".format(a, op, b, a//b))
else:
    print ("ERROR")
```

Q.12 What is a dictionary in python? Explain all methods with example.

→ A python dictionary is a collection data type which is wrapped in braces, {}, with a series of key value pairs inside the braces.

Python has a set of built-in methods which can be performed on dictionaries.

i) clear() → remove all the elements
ii) copy() → returns a copy.
iii) get() → returns the value of specified key.
iv) items() → returns a list containing a tuple for each key value pair
v) pop() → removes the element with specified key
vi) values() → returns a list of all the values.
vii) keys() → returns a list of all the keys
viii) update() → update the dictionary with the specified key-value pairs
ix) popitem() → remove last inserted key-value pair
x) setdefault()
xi) fromkeys()

eg:

```
1. my_dict = {'St1' : 101 , 'St2' : 102 , 'St3' : 103 }
2. x = ('key1', 'key2', 'key3')
3. y = 0
4.
5. copy_dict = my_dict.copy()
6. print (copy_dict)
7. print (my_dict.keys())
8. print (my_dict.values())
9. print (my_dict.fromkeys (x,y))
10. print (my_dict.items())
11. print (my_dict.setdefault ('St3', 'St4'))
12. my_dict.update ({'St4' : 104})
13. print (my_dict)
14. print (my_dict.get('St1'))
15. print (my_dict.pop('St2'))
16. print (my_dict.popitem())
17. print (my_dict.clear())
```

Output:

```
{'St1' : 101 , 'St2' : 102 , 'St3' : 103 }
dict.keys (['St1', 'St2', 'St3'])
dict.values ([101, 102, 103])
{'key1' : 0, 'key2' : 0 , 'key3' : 0 }
dict.items([('St1' : 101), ('St2' : 102), ('St3' : 103)])
103
{'St1' : 101 , 'St2' : 102 , 'St3' : 103 , 'St4' : 104 }
101
102
('St4', 104)
None
```

Q.13 What is a list in python? Explain all the methods with example.

—4 A python list is a collection data type which is wrapped in square brackets, [], with multiple different data type items.

Python has a set of built-in methods which can be performed on lists.

i) append() → adds an element at end of the list
ii) clear() → removes all the elements
iii) copy() → returns a copy.
iv) index() → returns the index of the element with the specified value.
v) insert() → Adds an element at the specified index/position
vi) remove() → removes the element with specified value.
vii) pop() → removes the element at specified index
viii) sort() → sorts the list
ix) reverse() → reverse the order of list.
x) count() → count the repetition of the specified value
xi) extend() → Add the elements of list (or any iterable), to the end.

eg:
```
1. my_list = [1,2,3,4,5]
2. my_list2 = [0,9,8,7]
3. copy_list = my_list.copy()
4. print("copy --->", copy_list)
5. my_list.append(11)
6. print("append(11) --->", my_list)
```

```
7.  my_list.insert(3,95)
8.  print("insert(3,95) → ", my_list)
9.  my_list.extend(my_list2)
10. print("extend(my_list2) → ", my_list)
11. print("count(2) → ", my_list.count(2))
12. print("index(11) → ", my_list.index(11))
13. my_list.reverse()
14. print("reverse → ", my_list)
15. my_list.sort()
16. print("sort → ", my_list)
17. my_list.pop(3)
18. print("pop(3) → ", my_list)
19. my_list.remove(4)
20. print("remove(4) → ", my_list)
21. my_list.clear()
22. print(my_list)
```

Output:

```
copy → [1,2,3,4,5]
append(11) → [1,2,3,4,5,11]
insert(3,95) → [1,2,3,95,4,5,11]
extend(my_list2) → [1,2,3,95,4,5,11,0,9,8,2]
count(2) → 2
index(11) → 6
reverse → [2,8,9,0,11,5,4,95,3,2,1]
sort → [0,1,2,2,3,4,5,8,9,11,95]
pop(3) → [0,1,2,3,4,5,8,9,11,95]
remove(4) → [0,1,2,3,5,8,9,11,95]
[]
```

Q.14 What is tuple in python ? Explain all methods with example.

→ A python tuple is used to store multiple items in a single variable, written within round brackets, (), and are ordered and unchangeable. Because tuple are immutable

Python has a set of built-in methods which can be performed on Tuples.

i) count() → returns the number of times a specified value occurs in a Tuple.

ii) index() → searches the tuple for a specified value and returns the position of where it was found.

eg:
```
1. my_tuple = (1,2,3,3,4,5)
2. print ("count(3) →", my_tuple.count(3))
3. print ("index(5) →", my_tuple.index(5))
```

Output:
```
count(3) → 2
index(5) → 6
```

Q.15 Explain lists as stacks with example.

→ A stack is an abstract data type that holds an ordered, linear sequence of items. A stack is last in, first out or first in, last out (LIFO / FILO) structure.

eg: A stack of plates, you can take a plate from the top of the stack, and you can only add a plate to the top of the stack.

Q.16 Explain lists as queues with example
—4 A queue is a linear data structure that stores items in First in First out (FIFO) manner. With a queue the last added will be removed last.
eg: A queue of consumers for a resource, where the consumer that came first is served first.

Q.17 Explain sequence operations indexing, slicing, adding sequences, multiplication, membership, length, minimum and maximum with example
—4 i) Indexing: Used to access individual items in a sequence using their position or index
example:
```
my_list = [10,20,30,40,50]
print (my_list [0])   # 10
print (my_list [-3])  # 30
```

ii) Slicing: Used to extract a portion of a sequence. We can specify a range of indicies to extract the portion of sequence
example:
```
my_list = [10,20,30,40,50]
print (my_list [1:4])  #[20,30,40]
print (my_list [2:])   #[30,40,50]
print (my_list [::2])  #[10,30,50]
```

iii) Adding Sequences: We can concatenate two or more sequences using the '+' operator.
example:
```
my_list 1 = [10,20,30]
my_list 2 = [40,50]
my_list 3 = my_list 1 + my_list 2.
print (my_list 3)  #[10,20,30,40,50]
```

iv) Multiplication : We can repeat the sequence multiple times using the '*' operator.
example: my_list = [10, 20]
my_list2 = my_list * 3
print (my_list2) # [10, 20, 10, 20, 10, 20]

v) Membership : We can check if an item is present in a sequence using 'in' operator
example: my_list = [10, 20, 30]
print (20 in my_list) # True
print (50 in my_list) # False

vi) Length : We can find length of a sequence using the 'len()' function.
example: my_list = [10, 20, 30]
print (len(my_list)) # 3

vii) Minimum : We can find the minimum item in a sequence using the 'min()' function.
example: my_list = [10, 20, 30]
print (min(my_list)) # 10

viii) Maximum : We can find the maximum item in a sequence using the 'max()' function
example: my_list = [10, 20, 30]
print (max(my_list)) # 30

Q.18 What is set in python ? Explain all the methods with example
-4 A python set is an unordered collection of unique elements and the set is created with '{}' curly braces.

Python has a set of methods which can be performed on sets.

i) add() → adds an element
ii) update() → update with another set, or any other iterable.
iii) remove() → removes the specified element.
iv) discard() → removes the specified item (if item is not found, it doesn't raise an error)
v) pop() → removes and returns an arbitrary element.
vi) clear() → removes all elements.
vii) copy() → returns a copy.

example:

```
1  my_set = {1, 2, 3, 4, 5}
2  another_set = {7, 8, 9, 0}
3  copy_set = my_set.copy()
4  print(copy_set)
5  my_set.add(6)
6  print("add(6) →", my_set)
7  my_set.update(another_set)
8  print("update(another_set) →", my_set)
9  my_set.discard(7)
10 print("discard(7) →", my_set)
11 my_set.remove(6)
12 print("remove(6) →", my_set)
13 my_set.pop()
14 print("pop() →", my_set)
15 my_set.clear()
16 print(my_set)
```

output: {1, 2, 3, 4, 5}
add(6) → {1, 2, 3, 4, 5, 6}
update(another_set) → {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
discard(7) → {0, 1, 2, 3, 4, 5, 6, 8, 9}
remove(6) → {0, 1, 2, 3, 4, 5, 8, 9}
pop() → {1, 2, 3, 4, 5, 8, 9}
set()

Q.19 Differentiate between list, Tuple & Dictionary.
→

	List	Tuple	Dictionary
>	Ordered	Ordered	Unordered
>	Mutable	Immutable	keys are Immutable
>	Written in []	Written in {}	Written in {} but key-value pairs.
>	Slicing and indexing works here	Slicing and indexing works here.	Slicing and indexing doesn't work

Q.20 Describe the following string methods each with an example.
i). upper(): turns the string in uppercase. But it does not modify the original string.
eg: name = "Lucifer"
print(name.upper()) # LUCIFER.

ii). lower(): turns the string in lowercase But it does not modify the original string
eg: name = "MORNingstar"
print(name.lower()) # morningstar

iii). find() : returns the index of the first occurence of specified substring. and if not found it returns '-1'.

eg: name = "Lucifer Morningstar is Devil".
print (name.find ("star")) # 15

iv). split() : split a string into a list of substring based on a specified delimiter.

eg: sentence = "apple, banana, orange"
fruits = sentence.split(",")
print (fruits) #["apple", "banana", "orange"]

v). join() : joins a list of strings into a single string with a specified separator.

eg: fruits = ["apple", "banana", "orange"]
sentence = ",".join (fruits)
print (sentence) # apple, banana, orange

Q.21 write a program to sort a dictionary in ascending and descending order of values.

```
1. d = {"apple": 5, "banana": 3, "orange": 7, "grape": 1}
2. sorted_dict = {K:V for K, V in sorted (d.items(),
   key = lamda item: item(i))}
3. print (sorted_dict)
4. sorted_dict = {K:V for K, V in sorted (d.items(),
   key = lamda item: item(i), reverse = True)}
5. print (sorted_dict)
```

~~~ x ~~~ x ~~~
~~~