Assignment 1

September 8, 2024

[1]:
```
# Q1. Explain the key features of python that make it a popular choice for␣
↪programming?
```

[4]:
```
# Ans- The key features of python that make it a popular cjoice for programming␣
↪are as following-
# 1. Python is widely use in industry- Python is widely use in industry because␣
↪it is dynamic in
#    nature , It is also used for scientific research and computing (among␣
↪other real-world applications).
# 2. Widely use in data industry or data analytics.
# 3. Python have lot of libraries - There are 1,37,000 libraries in python. A␣
↪python library is piece of code
#    that can be reused and which is well structured , well optimized. Examples␣
↪- Pandas and Numpy.
# 4. Python is easy to learn- Python is a programming language that has very␣
↪simple syntax and easy to learn.
# 5. Versatility- Python is versatile in nature. It is commonly used for web␣
↪development, data analysis and
#    visualization, scientific computing, artificial intelligence and machine␣
↪learning, game development,
#    and more.
# 6. Readability -  Python's readability is its simple and elegant syntax.␣
↪Python uses indentation to
#    define code blocks, eliminating the need for curly braces or other symbols␣
↪to delimit them.
#    This makes the code look clean and organized, as well as easier to read␣
↪and understand.
# 7. Apython has huge active community - A python community is a people which␣
↪write a code and publish in
#    python so people can use.
```

[11]:
```
# Q2. Describe the role of predefined keywords in Python and provide examples␣
↪of how they are used in a program?
```

[6]:
```
# Ans- Keywords are predefined words that hold a special meaning and have␣
↪specific purpose . The keywords can
```

```
# not be use as identifiers such as variable name or function name .  They are
 ↪used to define the syntax and
# structure of the Python language. The meaning of keywords are never change.
```

[12]:
```
# "a" is variable and we change or assign any value to a , but in keywords we
 ↪cannot assign or change value.
```

[7]:
```
# For example
a = 5
```

[8]:
```
a
```

[8]: 5

[132]:
```
a= 9
```

[133]:
```
#Examples of Keywords
import keyword
print(keyword.kwlist)
```

```
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break',
'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for',
'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or',
'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']
```

[135]:
```
# Example of keyword how they use in program
if 3+2==5:
    print("True")
```

```
True
```

[13]:
```
# Q3. Compare and contrast mutable and immutable objects in Python with
 ↪examples ?
```

[136]:
```
# Mutable object are whose data can be change called mutable object .
# Immutable objects are whoose data can not change.
```

[137]:
```
# Mutable Objects are
# 1. List
# 2. Dictionaries
# 3. Sets
```

[139]:
```
# For Example
list= [ "Apple" , "Banana" , "Orange"]
```

[143]:
```
list[1]= "Grapes"
```

[144]:
```
list
```

[144]:
```
['Apple', 'Grapes', 'Orange']
```

[ ]:
```
# Immortable objects are
# 1. Strings
# 2. Tuples
```

[146]:
```
tuple=(1,2,3)
```

[147]:
```
tuple[1]=4
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[147], line 1
----> 1 tuple[1]=4

TypeError: 'tuple' object does not support item assignment
```

[14]:
```
# Q4. Discuss the different types of operators in Python and provide examples␣
 ↪of how they are used?
```

[ ]:
```
# Ans- For manipulating the data it need some operations and operations are␣
 ↪perform by operator in python.
# Python operator are special keywords or symbols that are use to perform some␣
 ↪operations on values or
# variables.
# Operators are use to manage data , do computation and make decission using␣
 ↪data.
# Differernt type of operator are-
# 1. Arithmetic Operator
# 2. Comparison Operator
# 3. Logical and or Operator
# 4. Assignment Operator
# 5. Membership Operator
# 6. Bitwise Operator
# 7. Identify Operator
```

[3]:
```
# 1. Airthmetic Operator - Airthmetic operator are use to perform the␣
 ↪mathematical calculation.
# For Example - For Add
a=5
b=10
a+b
```

[3]:
```
15
```

[4]:
```
# Airthmetic Operator -
# For Example- For Sub
c=9
d=4
c-d
```

[4]: 5

[5]:
```
# Airthmetic Operator -
# For Example- For Multiplication
3*8
```

[5]: 24

[6]:
```
# Airthmetic Operator -
# For Example- For Dividition
24/8
```

[6]: 3.0

[7]:
```
15/2
```

[7]: 7.5

[8]:
```
# Modulas operator - For dividing if we want to get reminder so we can use␣
 ↪modulas operator
# For Example -
20%7
```

[8]: 6

[9]:
```
# Airthmetic Operator -
# For Example- For finding power
2**2
```

[9]: 4

[10]:
```
4**3
```

[10]: 64

[11]:
```
#Floor Operator - If you want to get value before point so the floor operator␣
 ↪will give you value before point.
#For Example
26//9
```

[11]: 2

[12]:
```
# 2. Comparsion Operator - Comparison operator are use to compare two values in␣
 ↪python.
#For Example
2==2
```

[12]: True

[15]:
```
2+2==4
```

[15]: True

[16]:
```
86+96==182
```

[16]: True

[17]:
```
7+2==10
```

[17]: False

[19]:
```
#Not equal to- !-
# For Example -
8-2!=5
```

[19]: True

[20]:
```
10-5!=5
```

[20]: False

[21]:
```
5>2
```

[21]: True

[22]:
```
4<8
```

[22]: True

[23]:
```
a=7
b=5
a>=5
```

[23]: True

[24]:
```
c=6
d=6
a>=d
```

[24]: True

[25]:
```
e=4
f=8
e>=f
```

[25]: False

[27]:
```
a=9
b=11
a<=b
```

[27]: True

[28]:
```
c=12
d=8
c<=d
```

[28]: False

[31]:
```
# 3. Logical and or Operator- Logical Statement are used to combine conditional␣
 ↪Statement
# AND OPERATOR
# In and operator both the statement should be true , if not it will give false.
# For Example
a=20
(a>10)and(20<40)
```

[31]: True

[32]:
```
# In "and" operator if one statement is true and other statement is false , it␣
 ↪will give the false.
# For Example
(5>2)and(5<3)
```

[32]: False

[34]:
```
# In "and" operator if both the statement are false , it will give you false.
# For Example
(8-5==4)and(7-3==2)
```

[34]: False

[35]:
```
# In "and" operator if one statement is false and other statement is true , it␣
 ↪will give you false.
# For Example
(7>9)and(9<10)
```

[35]: False

[36]:
```
#OR OPERATOR
# In "or" operator if one statement is true , it will give you true
# For Example-
(5-4==1)or(7-1==4)
```

[36]: True

[38]:
```
# In "or" operator if both the statements are true ,so it will give you true
# For Example-
(9+2==11)or(7-2==5)
```

[38]: True

[42]:
```
# In "or" operator if both the statement are false, it will give you false
# For Example-
(5*4==21)or(9*3==28)
```

[42]: False

[43]:
```
# In "or" operator if one statement is false , and other statement is true , it␣
 ↪will give true
# For Example-
(9-1==7)or(10-1==9)
```

[43]: True

[47]:
```
# Not Operator- Nor operator is use to reverse the result
# For Example-
print(not(5+1==6 and 7+3==10))
```

```
False
```

[48]:
```
print(not(6+2==7 and 4+1==8))
```

```
True
```

[49]:
```
# 4. Assignment Operator is use to assigning the value.
# For Example
a=5
```

[50]:
```
a
```

[50]: 5

[51]:
```
b=7
```

[52]:
```
b
```

[52]: 7

[53]:
```
b**2
```

[53]: 49

[65]:
```
c=8
```

[66]:
```
c
```

[66]: 8

[67]:
```
c+=4
```

[68]:
```
c
```

[68]: 12

[69]:
```
c*=2
```

[70]:
```
c
```

[70]: 24

[71]:
```
# 5. Membership Operator
# Membership operators are used to test if a element is presented in an object␣
 ↪or not.
# For example
a="mehak"
```

[72]:
```
a
```

[72]: 'mehak'

[74]:
```
"h" not in a
```

[74]: False

[75]:
```
"i" not in a
```

[75]: True

[77]:
```
# 6. Identity Operator
# Identity operator is use to compares the location of two object/variables.
# It will give you True if both variables are the same object
```

```
# For Example-
a=5
b=a
b==5
```

[77]: True

[78]:
```
# It will give you False if both variables are not the same object
# For Example-
b=9
c=b
a==9
```

[78]: False

[79]:
```
# 7. Bitwise Operator
# Bitwise Operator are use to perform operations on individual bits of binary␣
 ↪numbers.
```

[80]:
```
100&100
```

[80]: 100

[81]:
```
bin(100)
```

[81]: '0b1100100'

[83]:
```
16 & 3
```

[83]: 0

[86]:
```
# Bitwise OR OPERATOR
# The Bitwise or operator compares each bit and set it to 1 if one or both is␣
 ↪1, otherwise it is set to 0.
# For Example
6 | 3
```

[86]: 7

[87]:
```
# Q5. Explain the concept of type casting in Python with examples?
```

[88]:
```
# Ans-Type casting /Type conversion
# The process of changing the data type of value , object in python , in order␣
 ↪to perform the required
# operation, by users is known as type conversion or type casting.
```

[ ]:
```
# There are two type of type casting in python
# 1. Python Implicit Type Conversion- In this, method, Python converts the␣
 ↪datatype into another datatype
# automatically users don't have to involve in this process.
# 2. Python Explicit Type Conversion- In this method, Python needs user␣
 ↪involvement to convert the variable
# data type into the required data type.
```

[91]:
```
# Examples of Type Casting in python
# Changing String into integer - We can use int() function to change string␣
 ↪into interger.
# For Example
a="9"
# Here 9 is a string because it is in double coma.
# If we add string with integer it will give error so we need to change string␣
 ↪into integer.
# For that we can use (int) function.
# (int)a - For changing a into integer.
```

[92]:
```
a
```

[92]:
```
'9'
```

[93]:
```
type(a)
```

[93]:
```
str
```

[97]:
```
int(a)+2
```

[97]:
```
11
```

[101]:
```
print(type(a))
print(type(int(a)))
```

```
<class 'str'>
<class 'int'>
```

[102]:
```
# Changing float into integer
# Changing float into integer - We can use int() function to change float into␣
 ↪interger.
# For Example
a=2.5
# 2.5 is a float
# if we want to change float into integer we need to use int()
```

[103]:
```
a=2.5
```

[104]:
```
type(a)
```

[104]:
```
float
```

[105]:
```
int(a)
```

[105]:
```
2
```

[107]:
```
print(type(a))
print(type(int(a)))
```

```
<class 'float'>
<class 'int'>
```

[108]:
```
#changing integer to float
# Changing integer to float - We can use float() function to change integer␣
 ↪into float.
# For Example
a
# 7 is a integer
# if we want to change integer into float we need to use float()
```

[109]:
```
a = 7
```

[110]:
```
type(a)
```

[110]:
```
int
```

[111]:
```
b= float(a)
```

[112]:
```
b
```

[112]:
```
7.0
```

[113]:
```
type(b)
```

[113]:
```
float
```

[114]:
```
#changing string to float
# Changing string to float - We can use float() function to change string into␣
 ↪float.
# For Example
a="7.5"
# Here 7.5 is a float because it is in double coma.
# if we want to change string into float we need to use float() function.
```

[115]:
```
a="7.5"
```

[116]:
```
a
```

[116]: '7.5'

[117]:
```
type(a)
```

[117]: str

[118]:
```
b=float(a)
```

[118]: 7.5

[120]:
```
type(b)
```

[120]: float

[121]:
```
#changing integer to string
# Changing integer to string - We can use str() function to change integer into␣
 ↪string.
# For Example
a=10
# 10 is a integer
# if we want to change integer into string we need to use str() function.
```

[122]:
```
a= 10
```

[123]:
```
type(a)
```

[123]: int

[124]:
```
b=str(a)
```

[125]:
```
b
```

[125]: '10'

[127]:
```
type(b)
```

[127]: str

[145]:
```
#Concatenation
a="Mehak"
b="Agarwal"
a+b
```

[145]: 'MehakAgarwal'

[ ]:
```
# Q5. How do conditional statements work in Python? Illustrate with examples?
```

[148]:
```
# Conditional Statements are statements in Python that provide a choice for the␣
 ↪control flow
# based on a condition. It means that the control flow of the Python program␣
 ↪will be decided
# based on the outcome of the condition.
```

[7]:
```
# 1. If Conditional Statement in Python
# If the simple code of block is to be performed if the condition holds then␣
 ↪the if statement is used.
# Here the condition mentioned holds then the code of the block runs otherwise␣
 ↪not.
```

[8]:
```
# 2. If else Conditional Statements in Python
# In a conditional if Statement the additional block of code is merged as an␣
 ↪else statement which is
# performed when if condition is false.
```

[ ]:
```
# 3. Nested if..else Conditional Statements in Python
# Nested if..else means an if-else statement inside another if statement.
# Or in simple words first, there is an outer if statement, and inside it␣
 ↪another if - else statement
# is present and such type of statement is known as nested if statement. We can␣
 ↪use one if or else if
# statement inside another if or else if statements.
```