ASSIGNMENT - 01 (Variables, String, int, float)

Solution/Ans by - Pranav Rode(29)

1. What are the key features of Python?

In [ ]:
```
Python has several key features that contribute to its popularity and usability as a programming language.
Some of the key features of Python are:

Readability: Python emphasizes code readability and uses a clean and intuitive syntax, making it easy to
    understand and write.

Simplicity: Python offers a simple and straightforward approach to programming, making it suitable
    for beginners as well as experienced developers.

Expressive language: Python allows developers to express concepts and ideas in fewer lines of code
    compared to other programming languages, leading to increased productivity.

Large standard library: Python comes with a vast standard library that provides ready-to-use
    modules and functions for various tasks, reducing the need for external dependencies.

Dynamically-typed: Python is a dynamically-typed language, meaning that variable types are
    determined at runtime. This allows for flexibility and ease of use but requires careful
    attention to variable type handling.

Interpreted nature: Python is an interpreted language, which means that code execution
    occurs line by line, making it easy to test and debug code interactively.

Portability: Python is highly portable and can run on various platforms and operating
    systems without requiring extensive modifications.

Object-oriented: Python supports object-oriented programming (OOP) principles, allowing
    developers to create and use objects, classes, and inheritance for modular and reusable code.

Extensive community support: Python has a large and active community of developers who
    contribute to the language's development, provide support, and create a vast ecosystem of libraries,
    frameworks, and tools.

Integration capabilities: Python can easily integrate with other programming languages and systems,
    making it a suitable choice for building complex applications and systems.

These key features make Python a versatile and powerful language for a wide range of applications,
including web development, data analysis, scientific computing, artificial intelligence, and more.
```

2. What are the Data Types in Python?

In [ ]:
```
Python has several built-in data types, including:

Numeric types: int, float, complex
Sequence types: list, tuple, range
Text type: str
Mapping type: dict
Set types: set, frozenset
Boolean type: bool
Binary types: bytes, bytearray, memoryview
```

3. What are local variables and global variables in Python?

In [ ]:
```
In Python, local variables and global variables are used to store data within a program, but
they differ in terms of their scope and accessibility.

1. Local Variables:
   - Local variables are defined within a specific function or block of code.
   - They are only accessible within the function or block where they are defined.
   - Local variables have a limited scope and are destroyed once the function/block execution is complete.
   - Local variables can have the same name in different functions without causing conflicts.
   - Example:

E.g
def my_function():
    # Local variable
    message = "Hello, world!"
    print(message)

my_function()  # Output: Hello, world!
print(message)  # Error: NameError: name 'message' is not defined


2. Global Variables:
   - Global variables are defined outside of any function or block, at the top level of the program.
   - They can be accessed from anywhere within the program, including inside functions.
   - Global variables have a broader scope and remain in memory throughout the program's execution.
```

```
   - Global variables can be accessed and modified by any function, which can be both
    advantageous and potentially risky.
   - Example:

E.g
# Global variable
count = 0

def increment():
    # Accessing global variable
    global count
    count += 1

def decrement():
    # Accessing global variable
    global count
    count -= 1

increment()
print(count)  # Output: 1

decrement()
print(count)  # Output: 0


Its generally recommended to use global variables sparingly and only when necessary, as they can make code
less modular and harder to debug. Local variables, on the other hand, provide better
encapsulation and prevent unintended variable modifications.
```

4. How do you write comments in python? And Why Comments are important?

In [ ]:

```
In Python, you can write comments using the hash symbol (`#`).
Comments are ignored by the Python interpreter and are used to provide explanatory
notes or documentation within the code. They are useful for making the code more readable,
understandable, and maintainable. Here's how you can write comments in Python:


# This is a single-line comment

'''
This is a multi-line comment.
It can span multiple lines.
'''

# Example usage of comments
x = 5  # Assigning a value to the variable x
# The following line prints the value of x
print("The value of x is:", x)


Comments serve several purposes and benefits, including:

1. Code Documentation: Comments help explain the code's purpose, functionality, and
   logic to make it easier for others (including yourself) to understand and maintain the code in the future.

2. Clarification: Comments can provide additional context, explanations, or reminders
   about specific sections of code, making it easier to comprehend the code's flow and intentions.

3. Debugging and Troubleshooting: Comments can be used to temporarily disable or
   comment out specific lines of code during debugging or troubleshooting without deleting them,
   allowing you to isolate and identify problematic areas.

4. Collaboration: Comments can facilitate collaboration among team members by providing insights
   into the code's implementation, assumptions, or future considerations.

5. Code Organization: Comments can help organize and structure the code by separating it into
   logical sections or explaining the purpose of different functions, classes, or blocks of code.

It is good practice to write clear, concise, and meaningful comments that add value to the codebase.
However, it's important to strike a balance and avoid excessive or redundant comments, as
they may clutter the code and make it harder to read.
```

5. How to comment on multiple lines in python?

In [ ]:

```
To comment on multiple lines in Python, you have a couple of options. Here are two common methods:

1. Using Triple Quotes:
   You can enclose multiple lines of text within triple quotes (`'''` or `"""`) to create a multi-line comment.
   This method is typically used for longer comments or documentation.


'''
This is a multi-line comment.
It can span multiple lines.
These lines are all part of the comment.
'''

"""
This is another multi-line comment.
```

```
It can also span multiple lines.
These lines are all part of the comment.
"""


2. Adding a `#` at the beginning of each line:
   You can place a hash symbol (`#`) at the beginning of each line to comment out multiple lines of code.
   This method is more commonly used for shorter comments or temporarily disabling code.


# This is a multi-line comment
# It can span multiple lines
# These lines are all part of the comment


Both methods achieve the same result of commenting out multiple lines in Python.
Choose the one that suits your needs based on the length and purpose of your comment.
```

6. What do you mean by Python literals?

In [ ]:

```
In Python, literals refer to the notation used to represent values of different data types directly in the source code.
They are fixed values that are assigned to variables or used as data in expressions.
Python supports various types of literals, including:

1. Numeric Literals:
   - Integer literals: Whole numbers without decimal points, such as `42` or `-10`.
   - Floating-point literals: Numbers with a decimal point or exponent notation, such as `3.14` or `1.5e-3`.
   - Complex literals: Numbers with a real and imaginary part represented as `a + bj`, such as `2 + 3j`.

2. String Literals:
   - Enclosed in single quotes (') or double quotes ("), such as 'Hello' or "Python".
   - Can span multiple lines using triple quotes (''' or """), such as '''Multi-line string'''.

3. Boolean Literals:
   - Represent the truth values `True` and `False`.

4. None Literal:
   - Represent the absence of a value and is denoted by the keyword `None`.

5. Sequence Literals:
   - List literals: Enclosed in square brackets (`[]`), such as `[1, 2, 3]`.
   - Tuple literals: Enclosed in parentheses (`()`), such as `(1, 2, 3)`.
   - Set literals: Enclosed in curly braces (`{}`), such as `{1, 2, 3}`.
   - Dictionary literals: Enclosed in curly braces (`{}`) with key-value pairs, such as `{'name': 'John', 'age': 25}`.

6. Literal Constants:
   - Special values like `True`, `False`, and `None`.

Python literals provide a convenient way to express and assign values directly in the code,
making it easier to work with different data types and constants.
```

7. What are different ways to assign value to variables?

In [ ]:

```
In Python, there are several ways to assign values to variables. These include:

1. Direct Assignment: Assigning a value to a variable using the assignment operator (`=`).

   x = 10


2. Multiple Assignment: Assigning multiple values to multiple variables in a single line.

   x, y, z = 1, 2, 3


3. Chained Assignment: Assigning the same value to multiple variables in a single line.

   x = y = z = 0


4. Swapping Values: Exchanging the values of two variables using a temporary variable.

   a = 10
   b = 20
   temp = a
   a = b
   b = temp


5. Augmented Assignment: Performing an operation on a variable and assigning the result back to the
   same variable using augmented assignment operators like `+=`, `-=`, `*=`, `/=`, etc.

   x = 5
   x += 3  # Equivalent to x = x + 3


6. Unpacking Assignment: Assigning values from an iterable (e.g., list, tuple) to multiple variables in a single line.

   values = [1, 2, 3]
```

```
   a, b, c = values


7. Using Function Return Values: Assigning the returned value(s) from a function to variable(s).

   def add_numbers(a, b):
       return a + b

   result = add_numbers(5, 10)


These are some of the common ways to assign values to variables in Python. The choice of assignment method
depends on the specific requirements and coding style of your program.
```

8. What are the Escape Characters in Python?

Escape characters in Python are special characters that are used to represent certain characters that are difficult to include directly in a string. They are denoted by a backslash () followed by a specific character. Here are some commonly used escape characters in Python:

In [1]:
```
# \n - Newline: Inserts a new line.
# Example:
print("Hello,\nWorld!")
```

```
Hello,
World!
```

In [2]:
```
# \t - Tab: Inserts a tab.
# Example:
print("Hello,\tWorld!")
```

```
Hello,  World!
```

In [3]:
```
# \" - Double Quote: Inserts a double quote within a string.
# Example:
print("She said, \"Hello!\"")
```

```
She said, "Hello!"
```

In [4]:
```
# \' - Single Quote: Inserts a single quote within a string.
# Example:
print('He said, \'Hi!\'')
```

```
He said, 'Hi!'
```

In [5]:
```
# \\ - Backslash: Inserts a backslash within a string.
# Example:
print("C:\\path\\to\\file")
```

```
C:\path\to\file
```

In [5]:
```
# \r - Carriage Return: Moves the cursor to the beginning of the line.
# Example:
print("Hello \rWorld!")
```

```
World!
```

9. What are the different ways to perform string formatting? Explain with an example.

In Python, there are multiple ways to perform string formatting:

In [7]:
```
# Using the % operator:

name = "Alice"
age = 25
message = "My name is %s and I am %d years old." % (name, age)
print(message)
```

```
My name is Alice and I am 25 years old.
```

In [8]:
```
# Using the str.format() method:

name = "Alice"
age = 25
message = "My name is {} and I am {} years old.".format(name, age)
print(message)
```

```
My name is Alice and I am 25 years old.
```

In [9]:
```
# Using f-strings (formatted string literals):

name = "Alice"
age = 25
message = f"My name is {name} and I am {age} years old."
print(message)
```

```
My name is Alice and I am 25 years old.
```

10. Write a program to print every character of a string entered by the user in a new line using a loop

In [29]:
```
# Get input from the user
input_str = input("Enter a string: ")

# Iterate over each character in the string
for char in input_str:
    print(char)
```

```
Enter a string: Hello
H
e
l
l
o
```

11. Write a program to find the length of the string "machine learning" with and without using len function.

In [30]:
```
string = 'machine learning'
count = 0
for char in string:
    count += 1
print(count)
```

```
16
```

12. Write a program to check if the word 'orange' is present in the "This is orange juice".

In [31]:
```
sentence = "This is orange juice"
word = "orange"

if word in sentence:
    print("The word 'orange' is present in the sentence.")
else:
    print("The word 'orange' is not present in the sentence.")
```

```
The word 'orange' is present in the sentence.
```

13. Write a program to find the number of vowels, consonants, digits, and white space characters in a string.

In [76]:
```
strng = 'my age is 25'
strng = strng.lower()
vowels = 0
consonants = 0
digits = 0
space = 0
v = 'aeiou'
for i in strng:
    if i in v:
        vowels += 1
    if i not in v and i.isalpha():
        consonants += 1
    if i.isdigit():
        digits += 1
    if i == " ":
        space += 1


print('Vowels = ' ,vowels)
print('Consonants = ' ,consonants)
print('Digits = ' ,digits)
print('Space = ', space)
```

```
Vowels =  3
Consonants =  4
Digits =  2
Space =  3
```

14. Write a Python program to count Uppercase, Lowercase, special character, and numeric values in a given string.

In [52]:
```
input_str = "My Age is = @ 25"
strng = input_str.replace(" ", "")
u = 0
l = 0
num = 0
spec = 0
for i in strng:
    if i.isupper():
        u += 1
    elif i.islower():
        l += 1
    elif i.isdigit():
        num += 1
    else:
```

```
        spec += 1

print('Uppercase = ' ,u)
print('Lowercase = ' ,l)
print('Numerics = ' ,num)
print('Spec. Characters = ', spec)
```

```
Uppercase =  2
Lowercase =  5
Numerics =  2
Spec. Characters =  2
```

15. Write a program to make a new string with all the consonants deleted from the string "Hello, have a good day".

In [57]:
```
input_str = "Hello, have a good day"
vowels = 'aeiouAEIOU'
new_string = ''
for i in input_str:
    if i in vowels:
        new_string += i
print(new_string)
```

```
eoaeaooa
```

16. Write a Python program to remove the nth index character from a non-empty string.

In [92]:
```
input_str = "Hello, have a good day"
n = 2

if n < 0:
    print('Invalid Index')
elif n >= len(input_str):
    print('Index number greater than input string')
elif input_str[n] == ' ':
    print("Sorry, Space at given index")
else:
    new_string = input_str[:n] + input_str[n+1:]
    print(new_string)
```

```
Helo, have a good day
```

17. Write a Python program to change a given string to a new string where the first and last characters have been exchanged.

In [96]:
```
string = 'Hello'
if len(string) <= 1:
    print("String has only one character or is empty")
else:
    new_string = string[-1] + string[1:-1] + string[0]
    print(new_string)
```

```
oellH
```

18. Write a Python program to count the occurrences of each word in a given sentence

In [14]:
```
# Without using dictionary :

sentence = "This is a sentence is with repeated words this and is a sentence"
words = sentence.split()
unique_words = []
word_counts = []

for i in words:
    if i not in unique_words:
        unique_words.append(i)
        word_counts.append(1)
    elif i in unique_words:
        index = unique_words.index(i)
        word_counts[index] = word_counts[index] + 1

for i in range(len(unique_words)):
    print(f"{unique_words[i]}: {word_counts[i]}")
```

```
This: 1
is: 3
a: 2
sentence: 2
with: 1
repeated: 1
words: 1
this: 1
and: 1
```

In [20]:
```python
# Using dictionary :

sentence = "This is a sentence is with repeated words this and is a sentence"
word_counts = {}
words = sentence.split()

for i in words:
    if i in word_counts:
        word_counts[i] = word_counts[i] + 1
    else:
        word_counts[i] = 1

print(word_counts)
```

```
{'This': 1, 'is': 3, 'a': 2, 'sentence': 2, 'with': 1, 'repeated': 1, 'words': 1, 'this': 1, 'and': 1}
```

19. How do you count the occurrence of a given character in a string?

You can count the occurrence of a given character in a string using the count() method. Here's an example:

In [13]:
```python
string = "Hello, World!"
character = "l"
count = string.count(character)
print(count)
```

```
3
```

20. Write a program to find last 10 characters of a string?

In [22]:
```python
string = "This is a long string with more than 10 characters"
last_10_chars = string[-10:]
print(last_10_chars)
```

```
characters
```

21. Write a Program to convert a given string to all uppercase if it contains at least 2 uppercase characters in the first 4 characters.

In [3]:
```python
def convert_to_uppercase(string):
    uppercase_count = 0
    for char in string[:4]:
        if char.isupper():
            uppercase_count += 1
    if uppercase_count >= 2:
        return string.upper()
    return string

input_string = "HeLLo, World!"
output_string = convert_to_uppercase(input_string)
print(output_string)
```

```
HELLO, WORLD!
```

In [4]:
```python
# Other case
convert_to_uppercase("helLo, world!")
```

Out[4]:
```
'helLo, world!'
```

22. Write a Python program to remove a newline in Python.

In [99]:
```python
message = "\nHi! \nHow are you? \n"
print("Before:" + message)

new_message = message.replace('\n', '')
print("After:" + new_message)
```

```
Before:
Hi!
How are you?

After:Hi! How are you?
```

23. Write a Python program to swap commas and dots in a string

- Sample string: "32.054,23"

- Expected Output: "32,054.23"

In [5]:
```python
def swap_commas_dots(string):
    swapped_string = string.replace(',', 'TEMP').replace('.', ',').replace('TEMP', '.')
    return swapped_string

sample_string = "32.054,23"
result = swap_commas_dots(sample_string)
print("Swapped string:", result)
```

```
Swapped string: 32,054.23
```

24. Write a Python program to find the first repeated character in a given string

In [49]:
```
string="del pypthon mm"
rep=""
for index,char in enumerate(string):
    if string[:index+1].count(char)>1:
            rep=char
            break
print(rep)
```

```
p
```

25. Write a Python program to find the second most repeated word in a given string

In [1]:
```
stri = "apple orange banana orange orange apple orange grape apple banana"
lst = stri.split(' ')

map = {}
for eachCharacter in lst:
    if eachCharacter not in map:
        map[eachCharacter] = 1
    else:
        map[eachCharacter] += 1

newMap = {k: v for k, v in sorted(map.items(), key = lambda item: item[1], reverse=True)}
print("Sorted Dictionary with word counts:",newMap)
print("Second most repeating word = ",list(newMap.keys())[2-1])
```

```
Sorted Dictionary with word counts: {'orange': 4, 'apple': 3, 'banana': 2, 'grape': 1}
Second most repeating word =  apple
```

26. Python program to Count Even and Odd numbers in a string

In [100…
```
input_string = "2 5 8 12 7 10"
numbers = input_string.split()
even_count = 0
odd_count = 0

for num in numbers:
    if int(num) % 2 == 0:
        even_count += 1
    else:
        odd_count += 1

print("Even count:", even_count)
print("Odd count:", odd_count)
```

```
Even count: 4
Odd count: 2
```

27. How do you check if a string contains only digits?

In [103…
```
string = "12345"

if string.isdigit():
    print("String contains only digits.")
else:
    print("String does not contain only digits.")
```

```
String contains only digits.
```

28. How do you remove a given character/word from String?

In [104…
```
string = "Hello, World!"
character = ","
word = "World"

# Remove a character
new_string = string.replace(character, "")
print(new_string)

# Remove a word
new_string = string.replace(word, "")
print(new_string)
```

```
Hello World!
Hello, !
```

29. Write a Python program to remove the characters which have odd index values of a given string

In [105…
```python
string = "Python Programming"
new_string = ""

for index in range(len(string)):
    if index % 2 == 0:
        new_string += string[index]

print(new_string)
```

```
Pto rgamn
```

30. Write a Python function to reverses a string if its length is a multiple of 5

In [110…
```python
def reverse_string_if_multiple_of_five(input_string):
    if len(input_string) % 5 == 0:
        return input_string[::-1]
    else:
        return input_string

# Example usage
string1 = "HelloHello"
string2 = "Python1"

reversed_string1 = reverse_string_if_multiple_of_five(string1)
reversed_string2 = reverse_string_if_multiple_of_five(string2)

print(reversed_string1)
print(reversed_string2)
```

```
olleHolleH
Python1
```

31. Write a Python program to format a number with a percentage(0.05 >> 5%)

In [1]:
```python
number = 0.05
formatted_percentage = "{:.0%}".format(number)

print("Formatted Percentage:", formatted_percentage)
```

```
Formatted Percentage: 5%
```

32. Write a Python program to reverse words in a string

In [5]:
```python
def reverse_words(string):
    # Split the string into words
    words = string.split()

    # Reverse each word in the list
    reversed_words = []
    for word in words:
        reversed_word = ""
        for i in range(len(word) - 1, -1, -1):
            reversed_word += word[i]
        reversed_words.append(reversed_word)

    # Join the reversed words back into a string
    reversed_string = " ".join(reversed_words)

    return reversed_string


# Example usage
input_string = "Hello, how are you?"
reversed_string = reverse_words(input_string)

print("Original string:", input_string)
print("Reversed string:", reversed_string)
```

```
Original string: Hello, how are you?
Reversed string: ,olleH woh era ?uoy
```

33. Write a Python program to swap cases of a given string

In [6]:
```python
string = "Hello, World!"
swapped_string = ""
for char in string:
    if char.islower():
        swapped_string += char.upper()
    elif char.isupper():
        swapped_string += char.lower()
    else:
        swapped_string += char
print(swapped_string)
```

```
hELLO, wORLD!
```

34. Write a Python program to remove spaces from a given string

In [7]:
```
string = "Pranav,   Rode!"
new_string = ""

for char in string:
    if char != " ":
        new_string += char

print("Original string:", string)
print("String without spaces:", new_string)
```

```
Original string: Pranav,   Rode!
String without spaces: Pranav,Rode!
```

35. Write a Python program to remove duplicate characters of a given string

In [10]:
```
string = "Pranavvv"
new_string = ""

for char in string:
    if char not in new_string:
        new_string += char

print("Original string:", string)
print("String without duplicate characters:", new_string)
```

```
Original string: Pranavvv
String without duplicate characters: Pranv
```

36. Write a Python Program to find the area of a circle

In [15]:
```
radius = float(input("Enter the radius of the circle: "))

area = (22/7) * radius ** 2

print("The area of the circle is:", area)
```

```
Enter the radius of the circle: 5
The area of the circle is: 78.57142857142857
```

37. Python Program to find Sum of squares of first n natural numbers

In [17]:
```
n = int(input("Enter the value of n: "))

sum_of_squares = 0

for i in range(1, n+1):
    sum_of_squares += i ** 2

print("The sum of squares of the first", n, "natural numbers is:", sum_of_squares)
```

```
Enter the value of n: 5
The sum of squares of the first 5 natural numbers is: 55
```

38. Python Program to find cube sum of first n natural numbers

In [18]:
```
n = int(input("Enter the value of n: "))

sum_of_squares = 0

for i in range(1, n+1):
    sum_of_squares += i ** 3

print("The sum of squares of the first", n, "natural numbers is:", sum_of_squares)
```

```
Enter the value of n: 5
The sum of squares of the first 5 natural numbers is: 225
```

39. Python Program to find simple interest and compound interest

In [21]:
```
# Get the input from the user
principal = float(input("Enter the principal amount: "))
rate = float(input("Enter the rate of interest: "))
time = float(input("Enter the time period in years: "))

# Simple interest
simple_interest = (principal * rate * time) / 100

# Compound interest
amount = principal * (1 + rate/100) ** time
compound_interest = amount - principal

# Results
```

```
print("Simple Interest:", simple_interest)
print("Compound Interest:", compound_interest)
```

```
Enter the principal amount: 20000
Enter the rate of interest: 10
Enter the time period in years: 2
Simple Interest: 4000.0
Compound Interest: 4200.000000000004
```

40. Python program to check whether a number is Prime or not

In [22]:
```
n = int(input('Enter a number = '))

for i in range(2,n):#i = 2
    if n % i == 0:#True
        print(f"{n} is not a prime number")
        break
else:
    print(f"{n} is Prime number")
```

```
Enter a number = 7
7 is Prime number
```