SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

DEPARTMENT OF INFORMATION TECHNOLOGY

COURSE CODE: DJ19ITL406 **DATE:** 05-06-2022

COURSE NAME: Programing Laboratory 2 (Python) **CLASS:** SYBTECH-A

EXPERIMENT NO. 1

CO/LO: CO1, CO2.

AIM / OBJECTIVE:

Write python programs to understand

(a)Expressions, Variables.

(b)Quotes, Basic Math operations.

(c) Basic String Operations & String Methods.

DESCRIPTION OF EXPERIMENT:

Comments

Comments are pieces of text that live in your code but are ignored by the Python interpreter as it executes the code. You can use comments to describe the code so that you and other developers can quickly understand what the code does or why the code is written in a given way. To write a comment in Python, just add a hash mark (#) before your comment text:

```
# This is a comment on its own line
```

The Python interpreter ignores the text after the hash mark and up to the end of the line. You can also add **inline comments** to your code. In other words, you can combine a Python expression or statement with a comment in a single line, given that the comment occupies the final part of the line:

```
var = "Hello, World!"  # This is an inline comment
```

You should use inline comments sparingly to clear up pieces of code that aren't obvious on their own. In general, your comments should be short and to the point. If your comment is approaching or exceeding that length, then you might want to spread it out over multiple lines:

```
# This is a long comment that requires

 # two lines to be complete.
```

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

Variables

In Python, variables are names attached to a particular object. They hold a reference, or pointer, to the memory address at which an object is stored. Once a variable is assigned an object, you can access the object using the variable name.

You need to define your variables in advance. Here's the syntax:

variable_name = variable_value

You should use a naming scheme that makes your variables intuitive and readable. The variable name should provide some indication as to what the values assigned to it are.

Keywords

Like any other programming language, Python has a set of special words that are part of its syntax. These words are known as **keywords**. To get the complete list of keywords available in your current Python installation, you can run the following code in an interactive session:

help("keywords")

Arithmetic Expressions in Python

Python has its set of rules about how theseexpressions are to be evaluated, so that there is no ambiguity.

Multiplication and division have higher precedence than addition and subtraction, as isfrequently taught in grade school mathematics. Furthermore, parentheses have the highestprecedence and can be used to "force" the order in which operations are evaluated as isillustrated in the this line of code that was previously shown:

total_price = item_price*(1+tax_rate/100)

In this expression, we first evaluate the contents of the parentheses before multiplying. Inevaluating the contents of the parentheses, we first perform the division, since that has higherprecedence than addition. Thus, for example, if tax_rate is 7, then we first take 7/100 to get .07and then add that to 1 to get 1.07. Then, the current value of item_price is multiplied by 1.07,which is then assigned to total_price.

Python also provides three more operators for us to use:

1. **, for exponentiation.

2. //, for integer division

3. %, for modulus

The following sections describe exactly how each of these operators work, and the order of operations for each of them.

Exponentiation (**)

Following are examples of its use:

```
>>> 2 ** 3
8
>>> 3 ** 5
>>> 25 ** .5
5.0
>>> 2 ** -3
0.125
>>> 9999 ** 0
1
>>> -5 ** -3
-0.008
>>> -5 ** 3 -125
>>> 1.6743 ** 2.3233
3.311554089370817
```

Integer division (/ /) Python provides a second division operator, //, which performs integer division. In particular, the answer of an integer division operation is always an integer. In particular, a//b is defined as the largest number of whole times b divides into a. Here are a few examples of evaluating expressions with integer division:

```
>>> 25//4
6
>>> 13//6
 2
>>> 100//4
25
```

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

Modulus Operator (%)

In python, the modulus operator is defined for both integers and real numbers. If both operands are integers, then the answer will be an integer, otherwise, it will be a real number.

The formal definition of a % b is as follows: a % b evaluates to a - (a // b)*b. a // b represents the whole number of times b divides into a.

Thus, we are looking for the total number of times b goes into a, and subtracting out of a, that many multiples of b, leaving the "leftover."

Here are some conventional examples of mod, using only non-negative numbers:

```
>>> 17 % 3
 2
>>> 37 % 4
1
>>> 17 % 9
8
```

Input statement

Python makes reading input from the user very easy. In particular, Python makes sure that there is always a prompt (a print) for the user to enter some information.

Consider the following example entered

```
>>> name = input("What is your name?\n")
What is your name?
Simone
>>> print("Please to meet you ", name, ".", sep="")
Please to meet you Simone.
age = int(input("How old are you?\n")) if 15 < age < 25: print("You may drive, but not rent a car.")
```

Method-Description(not limited to)

1. capitalize()-Converts the first character to upper case
2. casefold()-Converts string into lower case
3. center()-Returns a centered string
4. count()-Returns the number of times a specified value occurs in a string
5. encode()-Returns an encoded version of the string
6. endswith()-Returns true if the string ends with the specified value

7. expandtabs()-Sets the tab size of the string
8. find()-Searches the string for a specified value and returns the position of where it was found
9. format()-Formats specified values in a string
10. format_map()- Formats specified values in a string
11. index()-Searches the string for a specified value and returns the position of where it was found
12. isalnum()-Returns True if all characters in the string are alphanumeric
13. isalpha()-Returns True if all characters in the string are in the alphabet
14. isdecimal()-Returns True if all characters in the string are decimals
15. isdigit()-Returns True if all characters in the string are digits
16. islower()-Returns True if all characters in the string are lower case
17. isnumeric()-Returns True if all characters in the string are numeric
18. isprintable()-Returns True if all characters in the string are printable
19. isspace()-Returns True if all characters in the string are whitespaces
20. istitle()-Returns True if the string follows the rules of a title
21. isupper()-Returns True if all characters in the string are upper case
22. join()-Converts the elements of an iterable into a string
23. ljust()-Returns a left justified version of the string
24. lower()-Converts a string into lower case
25. lstrip()-Returns a left trim version of the string
26. maketrans()-Returns a translation table to be used in translations
27. partition()-Returns a tuple where the string is parted into three parts
28. replace()-Returns a string where a specified value is replaced with a specified value
29. rfind()-Searches the string for a specified value and returns the last position of where it was found
30. rindex()-Searches the string for a specified value and returns the last position of where it was found
31. rjust()-Returns a right justified version of the string
32. rpartition()-Returns a tuple where the string is parted into three parts
33. rsplit()-Splits the string at the specified separator, and returns a list
34. rstrip()-Returns a right trim version of the string
35. split()-Splits the string at the specified separator, and returns a list
36. splitlines()-Splits the string at line breaks and returns a list
37. startswith()-Returns true if the string starts with the specified value
38. strip() Returns a trimmed version of the string
39. swapcase()-Swaps cases, lower case becomes upper case and vice versa
40. title()-Converts the first character of each word to upper case
41. translate() Returns a translated string
42. upper()-Converts a string into upper case
43. zfill()-Fills the string with a specified number of 0 values at the beginning

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

QUESTIONS:

1. Write a program that takes a sentence as an input parameter where each word in the sentence is separated by a space. Then replace each blank with a hyphen and then print the modified sentence.
2. Write a program to randomly select 10 integer elements from range 100 to 200 and find the smallest among all.
3. Create a dictionary of 5 countries with their currency details and display them.

CODE:

#1. Write a program that takes a sentence as an input parameter where each word in the sentence is separated by a space.

Then replace each blank with a hyphen and then print the modified sentence.

string=input("Enter a string: ")

```
sentence = input("enter your sentence")

hsent=sentence.replace(" ","-")
print(hsent)
```

OUTPUT:

```
...   hell-u-world
```

#2.Write a program to randomly select 10 integer elements from range 100 to 200 and find the smallest among all.

```
import random as r

myint=[]
i=0
for i in range(0,10):
    myint.append(r.randrange(100,200))

print(myint)
print("the smallest random integer is : ",min(myint))
```

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

OUTPUT:

```
[102, 136, 159, 145, 150, 167, 107, 152, 148, 195]
the smallest random integer is :  102
```

3.Create a dictionary of 5 countries with their currency details and display them.

```
mydict={"india":"rupees","USA":"dollar","europe":"euro","russia":"ruble","japan":
"yen"}

for k,v in mydict.items():
    print(k," : ",v,"\n")
```

OUTPUT:

```
india  :  rupees

USA  :  dollar

europe  :  euro

russia  :  ruble

japan  :  yen
```

OBSERVATIONS / DISCUSSION OF RESULT:

The results tells us how do the different String , list , dictionary , tuple functions work. We use for loop for appending elements in an empty list.

Different functions include:

1) join:Converts the elements of an iterable into a string
2) split:Splits the string at the specified separator, and returns a list.

3) remove():is an inbuilt function in the Python programming language that removes a given object from the List.
4) In case of tuples , we cannot append /remove elements so we convert a tuple to a list and then perform operations like delete() , append() , remove().
5) get() function is used to access the values in the dictionary.
6) String-slicing is about obtaining a sub-string from the given string by slicing it respectively from start to end

CONCLUSION:

In this experiment we implemented variables , math operations , string and string functions , List and list functions . We also understood string slicing in this experiment and its applications for eg to reverse a string or to add an element from the beginning to the end of the string etc.

REFERENCES:

Website References:

[1] https://www.w3schools.com/python