Experiment No-01: Python Variables, Expressions and Statements.

Objectives

- Get familiar with the variables, expressions, and statements.
- Learn different operators and their various operations.

Example 1: Values and Types.

```
print(4)
#If you are not sure what type a value has.
type('Hello, World!')
type(17)
type(3.2)
```

Example 2: Variables

```
number = 10
number = 1.1
##
message = 'And now for something completely different'
n = 17
pi = 3.1415926535897931
# This example makes three assignments. The first assigns a
   string to a new variable
# named message; the second assigns the integer 17 to n; the
   third assigns the
# (approximate) value of to pi.
#To display the value of a variable, you can use a print
   statement:
print(n)

print(pi)
```

Examples

```
#Example 1: Declaring and assigning value to a variable

website = "apple.com"
print(website)


# assigning a new value to website
website = "programiz.com"

print(website)

#Example 3: Assigning multiple values to multiple variables
```

```
a, b, c = 5, 3.2, "Hello"

print (a)
print (b)
print (c)

# If we want to assign the same value to multiple variables at
   once, we can do this as:

x = y = z = "same"

print (x)
print (y)
print (z)
```

Example 3: Keywords

```
# If you give a variable an illegal name, you get a syntax error:

# Illegal variable names

76trombones = 'big parade'
more@ = 1000000
class = 'Advanced Theoretical Zymurgy'

# It turns out that class is one of Pythons keywords. The
   interpreter uses keywords
# to recognize the structure of the program, and they cannot be
   used as variable
# names.
```

Example 4: Implicit Type Conversion

```
# Example 1: Converting integer to float

num_int = 123
num_flo = 1.23

num_new = num_int + num_flo

print("datatype of num_int:",type(num_int))
print("datatype of num_flo:",type(num_flo))

print("Value of num_new:",num_new)
print("datatype of num_new:",type(num_new))

# Example 2: Addition of string(higher) data type and
```

```
   integer(lower) datatype

num_int = 123
num_str = "456"

print("Data type of num_int:",type(num_int))
print("Data type of num_str:",type(num_str))

print(num_int+num_str)
```

Example 5: Explicit Type Conversion

```
#Example 3: Addition of string and integer using explicit
   conversion

num_int = 123
num_str = "456"

print("Data type of num_int:",type(num_int))
print("Data type of num_str before Type Casting:",type(num_str))

num_str = int(num_str)
print("Data type of num_str after Type Casting:",type(num_str))

num_sum = num_int + num_str

print("Sum of num_int and num_str:",num_sum)
print("Data type of the sum:",type(num_sum))
```

Example 6: Operators

```
#  Example 1: Arithmetic operators in Python
x = 15
y = 4

# Output: x + y = 19
print(’x + y =’,x+y)

# Output: x  y = 11
print(’x  y =’,xy)

# Output: x * y = 60
print(’x * y =’,x*y)

# Output: x / y = 3.75
print(’x / y =’,x/y)

# Output: x // y = 3
```

```
print('x // y =',x//y)

# Output: x ** y = 50625
print('x ** y =',x**y)
```

Example 7: String Operations

```
first = 10
second = 15
print(first+second)

############
first = '100'
second = '150'
print(first + second)

#The * operator also works with strings by multiplying the
   content of a string by an integer. For example:

first = 'Test '
second = 3
print(first * second)
```

Example 8: Asking the user for input

```
inp = input("Enter a value ")
print(inp)
```

Example 9: Output formatting

```
# one way
x = 5
y = 10
print('The value of x is {} and y is {}'.format(x,y))

#Here, the curly braces {} are used as placeholders.

# second way
x = 5
y = 10
print(f'The value of x is {x} and y is {y}')

# We can even use keyword arguments to format the string.

print('Hello {name}, {greeting}'.format(greeting =
   'Goodmorning', name = 'John'))

# test
a = 5; b=6; c= 10
```

```
print(f"The summation of {a},{b}, and {c} =",a+b+c)

# We can also format strings like the old sprintf() style used
   in C programming language.
# We use the % operator to accomplish this
x = 12.3456789
print('The value of x is %.2f' %x)
print('The value of x is %.4f' %x)


# round function
round(x,4)
```

Example 10: Package

```
# Python math package
from math import sqrt,pi
sqrt(16)
```

*** For better understanding please feel free to see the [colab notebook.] ***

Practice Exercise

1. Write a program that uses input to prompt a user for their name and then welcomes them.

   Sample Input-Output:

   Enter your name: Shakib
   Hello Shakib

2. Write a program to prompt the user for hours and rate per hour to compute gross pay.

   Sample Input-Output:

   Enter Hours: 35
   Enter Rate: 2.75
   Pay : 96.25

3. Write a program that prompts the user for a Celsius temperature, convert the temperature to Fahrenheit, and print out the converted temperature.

   Sample Input-Output:

Enter Temperature in Celsius: 35
Temparature in Farenheit: ???

4. Write a program that prompts the user for the height and width of a rectangle and calculate the area, diagonal, and perimeter.

   Sample Input-Output:

   Enter Height: 5
   Enter Width: 7
   Area: ??
   Diagonal: ??
   Perimeter: ??

5. Write a Python program to calculate the area of a trapezoid.

   Sample Input-Output:

   Enter Height: 5
   Enter base 1 value: 18
   Enter base 2 value: 7
   Area: ??

6. Write a Python program to calculate the surface volume and area of a cylinder.

   Sample Input-Output:

   Enter Height: 5
   Enter radius: 10
   Volume: ??
   Area : ??

7. Write a Python program to calculate the arc length of an angle and the sector area of a circle.

   Sample Input-Output:

   Enter Radius: 5
   Enter Angle: 60
   Arc Length: ??
   Sector Area: ??

Experiment No-02: Python Conditional Statements

Objectives

- Get familiar with the Python conditional statements and different conditional operators.
- Write Programs using Python if else statements.

Example 1: Boolean expressions.

```
# A boolean expression is an expression that is either true or
   false.

print(5 == 5)
#True
print(5 == 6)
#False

#True and False are special values that belong to the class
   bool; they are not strings:
print(type(True))
# <class 'bool'>
print(type(False))
#<class 'bool'>
```

Example 2: Comparison operators

```
# Comparison operators in Python
x = 10
y = 12

# Output: x > y is False
print('x > y is',x>y)

# Output: x < y is True
print('x < y is',x<y)

# Output: x == y is False
print('x == y is',x==y)

# Output: x != y is True
print('x != y is',x!=y)

# Output: x >= y is False
print('x >= y is',x>=y)

# Output: x <= y is True
```

```
print('x <= y is',x<=y)

# x is the same as y ##  x==y
print(x is y)
# x is not the same as y x!=y
print(x is not y)
```

Example 3: Logical operators

```
# Logical Operators in Python

x = True
y = False

print('x and y is',x and y)

print('x or y is',x or y)

print('not x is',not x)
```

Example 4: Identity operators

```
# Identity operators in Python

x1 = 5
y1 = 5
x2 = 'Hello'
y2 = 'Hello'

# Output: False
print(x1 is not y1)

# Output: True
print(x2 is y2)
```

Example 5: Membership operators

```
# Membership operators in Python

x = 'Hello world'

# Output: True
print('H' in x)

# Output: True
print('hello' not in x)
```

Example 6: Conditional execution

```
if x%2 == 0:
  print('x is even')
else :
  print('x is odd')
```

Example 7: Chained conditionals

```
### Example 1

x = 10
y = 15

if x < y:
  print('x is less than y')
elif x > y:
  print('x is greater than y')
else:
  print('x and y are equal')

###### Example 2

choice = input("Enter your Choice: ")

if choice == 'a':
  print('Bad guess')
elif choice == 'b':
  print('Good guess')
elif choice == 'c':
  print('Close, but not correct')
```

Example 8: Nested conditionals

```
if x == y:
  print('x and y are equal')
else:
  if x < y:
    print('x is less than y')
  else:
    print('x is greater than y')
```

*** For better understanding please feel free to see the [colab notebook.] ***

Practice Exercise

1. Write a program to prompt the user for hours and rate per hour to compute gross pay. However, the employee who worked above 40 hours gives them 1.5 times the hourly rate for individual hours.

Sample Input-Output:

Enter Hours: 45
Enter Rate: 10
Pay: 475.0

2. A company decided to give a bonus of 5% to an employee if his/her year of service is more than 5 years. Ask the user for their salary and year of service and print the net bonus amount.

3. Take values of the length and breadth of a rectangle from the user and check if it is square or not.

4. Take two int values from the user and print the greatest among them.

5. A shop will give a discount of 10% if the cost of the purchased quantity is more than 1000. Ask the user for quantity Suppose, one unit will cost 100. Judge and print the total cost for the user.

6. A school has the following rules for the grading system:

   a. Below 25 - F
   b. 25 to 45 - E
   c. 45 to 50 - D
   d. 50 to 60 - C
   e. 60 to 80 - B
   f. Above 80 - A

   Ask the user to enter marks and print the corresponding grade.

7. Take input of age of 3 people by the user and determine oldest and youngest among them.

8. A student will not be allowed to sit in an exam if his/her attendance is less than 75%. Take the following input from the user:

   Number of classes held
   The number of classes attended

   And print the percentage of classes attended. Is the student allowed to sit in an exam or not?

9. Modify the above question to allow a student to sit if he/she has a medical cause. Ask the user if he/she has a medical cause or not ('Y' or 'N') and print accordingly.

10. Sort three numbers in ascending and descending order using conditional statements.

Experiment No-03: Python Iterators or Loops

Objectives

- Get familiar with the Python control flow statements.
- Write Programs using Python loops.

Example 1: For Loop.

```
# Syntax of for Loop

'''
for val in sequence:
    loop body
'''

a = [1,2,3,5,6]

#for (int i =0; i<5; i++)

for i in a:
   print(i)
```

Example 2: Range Function

```
# We can generate a sequence of numbers using range() function.
   range(10) will generate numbers from 0 to 9 (10 numbers).

# We can also define the start, stop and step size as
   range(start, stop,step_size). step_size defaults to 1 if not
   provided.

print(range(10))

print(list(range(10)))

print(list(range(2, 8)))

print(list(range(2, 20, 3)))


#function(argument)

# argument
# return value

b = "Hello World"
```

```python
len(b)

numbers = [6, 5, 3, 8, 4, 2, 5, 4, 11]
max(numbers)
```

Example 3:

```python
# Program to find the sum of all numbers stored in a list

# List of numbers
numbers = [6, 5, 3, 8, 4, 2, 5, 4, 11]

# variable to store the sum
sum = 0

# iterate over the list
for number in numbers:
    sum = sum+number
print("The sum is", sum)

# Another way
sum = 0
for i in range(len(numbers)):
    sum = sum + numbers[i]
    print(sum)


# Program to wish friends with new year greetings

friends = ['Joseph', 'Glenn', 'Sally']

for friend in friends:
    print('Happy New Year:', friend)

print('Done!')

# Another way

for i in range(len(friends)):
    print('Happy New Year:', friends[i])
```

Example 4:

```python
# program to count the number of items in a list
```

```
count = 0
for itervar in [3, 41, 12, 9, 74, 15]:
    count = count + 1
print('Count: ', count)

# computes the total of a set of numbers
total = 0
for itervar in [3, 41, 12, 9, 74, 15]:
    total = total + itervar
print('Total: ', total)
```

Example 5:

```
#To find the largest value in a list or sequence,

largest = None
print('Before:', largest)

for itervar in [3, 41, 12, 9, 74, 15]:
    if largest is None or itervar > largest :
        largest = itervar
    print('Loop:', itervar, largest)

print('Largest:', largest)

# Another way

numbers = [3, 41, 12, 9, 74, 15]
largest= numbers[0]

for itervar in numbers:
    if itervar > largest :
        largest = itervar
    print('Loop:', itervar, largest)

print('Largest:', largest)
```

Example 6: While Loop

```
# Program to add natural
# numbers up to
# sum = 1+2+3+...+n

# To take input from the user,
# n = int(input("Enter n: "))

n = 10
```

```
# initialize sum and counter
sum = 0
i = 1

while i <= n:
    sum = sum + i
    i = i+1   # update counter

# print the sum
print("The sum is", sum)

### Example of infinite loop

while True:
    if inp = 'Done':
        break

# Example 3
# Print i as long as i is less than 6:
i = 1
while i < 6:
    print(i)
    i += 1
```

Example 7: Break and Continue

```
# Use of break statement inside the loop

for val in "string":
    if val == "i":
        break
    print(val)

print("The end")


# Program to show the use of continue statement inside loops

for val in "string":
    if val == "i":
        continue
    print(val)

print("The end")
```

Example 8: Exception Handling

```
try:
    # code that may cause exception
except:
    # code to run when exception occurs

# Example 1

try:
    numerator = 10
    denominator = 0

    result = numerator/denominator

    print(result)
except:
    print("Error: Denominator cannot be 0.")

# Output: Error: Denominator cannot be 0.
```

*** For better understanding please feel free to see the [colab notebook.] ***

Practice Exercise

1. Write a program that repeatedly reads numbers until the user enters "done". Once "done" is entered, print out the total, count, and average of the numbers. If the user enters anything other than a number, detect their mistake using try and except and print an error message and skip to the next number.

   Sample Input-Output:

   Enter a number: 4
   Enter a number: 5
   Enter a number: bad data
   Invalid input
   Enter a number: 7
   Enter a number: done
   16 3 5.333333333333333

2. Write a program to prompt for a score between 0.0 and 1.0. If the score is out of range, print an error message. If the score is between 0.0 and 1.0, print a grade using the following table:

   Score Grade
   >= 0.9 A

>= 0.8 B
>= 0.7 C
>= 0.6 D
< 0.6 F

Sample Input-Output:

```
Enter score: 0.95
A
Enter score: perfect
Bad score
Enter score: 10.0
Bad score
Enter score: 0.75
C
Enter score: 0.5
F
```

3. Write a Python program to find those numbers which are divisible by 7 and multiple of 5, between 1500 and 2700 (both included).
4. Write a Python program that prints all the numbers from 0 to 6 except 3 and 6.
5. Write a Python program to get the Fibonacci series between 0 to 50
6. Write a program to display all prime numbers within a range.
7. Write a program to accept a number from a user and calculate the sum of all numbers from 1 to a given number.

Experiment No-04: Python Strings and User defined functions

Objectives

- Get familiar with various operations and methods of Python strings.
- Get familiar with user-defined functions in Python.

Example 1: Strings.

```
#A string is a sequence of characters.

fruit = 'banana'
letter = fruit[1]
print(letter)

# len is a builtin function that returns the number of
   characters in a string

fruit = 'banana'
len(fruit)

#To get the last letter of a string, you might be tempted to try
   something like this:

length = len(fruit)
last = fruit[length]
#print(last)

# To get the last character, you have to subtract 1 from the
   length
```

Example 2: Traversal through a string with a loop

```
## Example
fruit = 'banana'
index = 0
while index < len(fruit):
    letter = fruit[index]
    print(letter)
    index = index + 1

# Another way to write a traversal is with a for loop:

for char in fruit:
    print(char)
```

Example 3: String slices

```
## Example 1
s = 'Monty Python'
print(s[0:5])
print(s[6:12])

## Example 2
fruit = 'banana'
print(fruit[:3])

print(fruit[3:])
```

Example 4: Strings are immutable

```
# Not Possible
greeting = 'Hello, world!'
greeting[0] = 'J'

# Correct way
# String Concatenation
greeting = 'Hello, world!'
new_greeting = 'J' + greeting[1:]
print(new_greeting)
```

Example 5: Looping and counting

```
# The following program counts the number of times the letter a
   appears in a string:

word = 'banana'
count = 0
for letter in word:
   if letter == 'a':
       count = count + 1
print(count)
```

Example 6: String methods

```
##
a = 'Hello world'
type(a)
# Show all the string methods available in python
dir(a)

# some methods
```

```
w = "Hello"

w.isalpha() # checking

w.find('l') # searching

w.count('l') # counter

### convert into upper case

word = 'banana'
new_word = word.upper()
print(new_word)

# find that searches for the position of one string within
   another
word = 'banana'
index = word.find('a')
print(index)

# Another example

word = 'banana'

print(word.find('na'))

#It can take as a second argument the index where it should
   start:

print(word.find('na', 3))

# One common task is to remove white space (spaces, tabs, or
   newlines) from the
beginning and end of a string using the strip method:


line = '     Here we go '
line.strip()

#line.lstrip()
#line.rstrip()

# Some methods such as startswith return boolean values.
line = 'Have a nice day'
print(line.startswith('Have'))
print(line.startswith('h'))
```

```
# You will note that startswith requires case to match, so
   sometimes we take a line
# and map it all to lowercase before we do any checking using
   the lower method.

line = 'Have a nice day'
line= line.lower()
line.startswith('h')

# Another way

line.lower().startswith('h')
```

Example 7: Parsing strings

Often, we want to look into a string and find a substring. For example if we were presented a series of lines formatted as follows:

From stephen.marquard@ uct.ac.za Sat Jan 5 09:14:16 2008

and we wanted to pull out only the second half of the address (i.e., **uct.ac.za**) from each line, we can do this by using the ***find*** method and ***string slicing***.

First, we will find the position of the at-sign in the string. Then we will find the position of the first space after the at-sign. And then we will use string slicing to extract the portion of the string which we are looking for.

```
# Example of parsing

data = 'From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008',

atpos = data.find('@')
print(atpos)

sppos = data.find(' ',atpos)
print(sppos)

host = data[atpos+1:sppos]
print(host)
```

Example 8: Functions

```
# arguments with return value

def addtwo(a, b):
    added = a + b
    return added


x = addtwo(3, 5)
# print(x)

# arguments with multiple return value

def arith(a,b):
    added = a + b
    sub = a b
    m = a*b
    return added, sub, m


x,y,z = arith(3)

print(x)
print(y)
print(z)
```

*** For better understanding please feel free to see the [colab notebook] ***

Practice Exercise

1. Take the following Python code that stores a string.

   str = 'X-DSPAM-Confidence:0.8475'

   Use **find** and **string slicing** to extract the portion of the string after the colon character and then use the float function to convert the extracted string into a floating point number.

2. Write a Python program to get a string made of the first 2 and the last 2 chars from a given string. If the string length is less than 2, return instead of the empty string.

   Sample Input-Output:

   Sample String : 'w3resource'
   Expected Result : 'w3ce'
   Sample String : 'w3'
   Expected Result : 'w3w3'
   Sample String : ' w'
   Expected Result : Empty String

3. Write a Python program to add 'ing' at the end of a given string (length should be at least 3). If the given string already ends with 'ing' then add 'ly' instead. If the string length of the given string is less than 3, leave it unchanged. (use endswith).

   Sample Input-Output:

   Sample String : 'abc'
   Expected Result : 'abcing'
   Sample String : 'string'
   Expected Result : 'stringly'

4. Write a Python program to find the first appearance of the substring 'not' and 'poor' from a given string, if 'not' follows the 'poor', replace the whole 'not'...'poor' substring with 'good'. Return the resulting string.

   Sample Input-Output:

   Sample String : 'The lyrics is not that poor!'
   'The lyrics is poor!'
   Expected Result : 'The lyrics is good!'
   'The lyrics is poor!'

5. Rewrite your pay computation (from lab-02) with time-and-a-half for overtime and create a function called compute pay which takes two parameters (hours and rate).

   Sample Input-Output:

   Enter Hours: 45
   Enter Rate: 10
   Pay: 475.0

6. Rewrite the grade program from the previous lab(03) using a function called computegrade that takes a score as its parameter and returns a grade as a string.

   Sample Input-Output:

   Enter score: 0.95
   A
   Enter score: perfect
   Bad score
   Enter score: 10.0
   Bad score
   Enter score: 0.75
   C
   Enter score: 0.5
   F