Object Oriented Programming
(Python l)
Lab4

ASST.LEC Fatima Mohammed & Adnan Habeeb

Example

- Write a Python function named " find_max " that takes a list of numbers as input and returns the maximum value in the list. **Hint**: Ensure the function returns **None** if the list is empty , and do not use built-in max() functions.

```
def find_max(lst):
    """Return the maximum value in a list."""
    if not lst:
        return None
    max_value = lst[0]
    for num in lst:
        if num > max_value:
            max_value = num
    return max_value

print(find_max([3, 5, 1, 9, 2]))  # output is " 9 "
print(find_max([]))  # output " None "
```

Example…

- Write a Python function named " factorial " that calculates the factorial of a given number. The function should correctly handle negative inputs, zero, and positive integers. Test your function to ensure it returns correct results for a variety of inputs.

```
def factorial(n):
    if n < 0:
        return "Factorial is not defined for negative numbers."
    elif n == 0 or n == 1:
        return 1
    else:
        result = 1
        for i in range(2, n + 1):
            result *= i
        return result
print(factorial(5))# output : 120
print(factorial(0)) # output : 1
print(factorial(-9)) # output : Factorial is not defined for negative numbers
```

Example…

- Write a Python program that searches for a character in a string. If the character is found, the program should print the index of its first occurrence; otherwise, it should indicate that the character is not present in the string. Hint: Define a function named **"find_character"** that accepts a character and a string, and do not use built-in functions.

```
def find_character (character,string):
    idx=0
    while(idx<len(string)):
        if string[idx]==character:
            print(character," is found in string at index:",idx)
            return
        else:
          idx+=1
    print(character , "is not present in the string")
char=input("\n Enter the char to be found: ")
find_character (char , string='python programming language’)
```

Output 1

```
Enter the char to be found: t
t found in string at index: 2
```

Output 2

```
Enter the char to be found: d
d is not present in the string
```

Example..

- Write a Python program that searches for a substring in a full string. If the substring found within the full string, the program should print the index of its first occurrence; otherwise, it should indicate that the character is not present in the string. Hint: Define a function named **"find_substring"** that accepts a character and a string.

```
def find_substring():
    full_string=input("enter full string: ")
    substring =input("enter the substring to be found: ")
    index=full_string.find(substring)
    if index !=-1:
        return f"substring {substring} found in the full string at index: {index}"
    else:
        return f"substring {substring} is not present in the string"
result=print(find_substring())
```

Output

```
enter full string: this program that searches for a substring in a full string
enter the substring to be found: string
substring string found in the full string at index: 36
```

Example...

- Write a Python program that searches for a substring in a full string. If the substring found within the full string, the program should print the index of its first occurrence; otherwise, it should indicate that the character is not present in the string. Hint: Define a function named "**find_substring**" that accepts a character and a string, and do not use built-in functions.

```
def find_substring(substring, full_string):
    idx = 0
    while idx < len(full_string):
        if full_string[idx:idx + len(substring)] == substring:
            print(f"substring {substring} found in the full string at index: {idx}")
            return
        else:
            idx += 1
    print(f"substring {substring} is not present in the string")


full_string = input("enter full string: ")
substring = input("enter the substring to be found: ")
find_substring(substring, full_string)
```

Output

```
enter full string: this program that searches for a substring in a full string
enter the substring to be found: string
substring string found in the full string at index: 36
```

Example…

- Write python program that :
- Ask the user for the length of a list.
- Allows the user to enter the elements of the list.
- Calculate and display the sum of elements.
- The program should include three functions:

1. Insert_Item (): to input the list length and elements ,and
2. print_list(): to print list ,and
3. Sum_list(): to sum] all list elements

```
def insert_Item():
    size=int(input("how many number u want to add "))
    mylist = [0]*size
    for i in range(size):
        mylist[i]=int(input(f"Enter value mylist[{i}]= "))
    return mylist

def print_list (list1):
    for i in range(len(list1)):
        print(f"mylist[{i}]= ",list1[i])
```

```
def sum (list1):
    y=0
    for i in range(len(list1)):
        y=y+list1[i]
    print("Sum of mylist",list1,"=",y)

z= insert_Item()
print_list(z)
sum(z)
```

Output

```
how many number u want to add 3
Enter value mylist[0]= 54
Enter value mylist[1]= 97
Enter value mylist[2]= 15
mylist[0]=  54
mylist[1]=  97
mylist[2]=  15
Sum of mylist [54, 97, 15] = 166
```

Default Argument

default argument- argument that **assumes a default** value if a **value is not provided in the function call** for that argument

- Write a Python program by defining a function named “student” that takes (firstname, lastname, and standard) as parameters. The function should accept lastname as "**None**" and standard as "**Fifth**" as default arguments. Call the function three times:

1. With only the firstname parameter.
2. With all three parameters.
3. With the firstname and lastname parameters.

```
def student(firstname, lastname ='None', standard ='Fifth'):
    print(f"{firstname} {lastname} studies in {standard} Standard")
# 1 positional argument
student('John')                          #John None studies in Fifth Standard
# 3 positional arguments
student('John', 'Gates', 'Seventh')      #John Gates studies in Seventh Standard
# 2 positional arguments
student('John', 'Gates')                 #John Gates studies in Fifth Standard
```

Keyword Arguments

•Keyword arguments are related to the function calls.
• Using keyword arguments in a function call, the caller identifies the arguments by the parameter name.
•Allows to skip arguments or place them out of order, the Python interpreter use the keywords provided to match the values with parameters.

Example:

```
def func(date,year):
    print("date",date,"year",year)
    return
func(date = 8 , year = 1888,)
func(year  =2004 , date = 20) # calling of the argument order doesn't matter
```

Output 1

date 8 year 1888

Output 2

date 20 year 2004

Packing and Unpacking Arguments (* args)

- *** Args** is used to pass variable number of arguments to a function.
- It used to passed a non key worded, variable length argument lest.
- The syntax is to use the symbol * to take in a variable number of arguments.

```
def full_name(*args):
  for i in args:
      print(i)

full_name('Hi','John', 'Gates','peter' )
```

Output

Hi
John
Gates
peter

Packing and Unpacking Arguments (* args)..

- Write a Python program that prompts the teacher to enter a student's name and scores for all subjects the student is required to take , then display the student's name and each score's
  - Requirement: Use function and use packaging arguments (*args) to scores

```
def print_scores (stud,*s):
    print(f" Student Name: {stud}")
    for i in s:
        print(i)
print_scores("John",52,93,87,100)
```

Output

Student Name: John
52
93
87
100

Scope of Variables

- All variables in a program may not be accessible at all locations in that program. This depends on where you have declared a variable. The scope of a variable determines the portion of the program where you can access a particular identifier. There are two basic scopes of variables in Python −

- **Global variables**
- **Local variables**

Using Local Variables

```
my_list=[2,5,7,55,6,96]                #Mutable Variables
def change_value(my_list):
    my_list[4]=100
    print(f"value inside function{my_list}")
print(f"Before function {my_list}")                    # Before function [2, 5, 7, 55, 6, 96]
change_value(my_list)                                  # value inside function[2, 5, 7, 55, 100, 96]
print(f"After function {my_list}")                     # After function [2, 5, 7, 55, 100, 96]
```

Scope of Variables…

- **Using Global variables**

```
def myfunc():
    global x                #Immutable Variables
    x = "fantastic"
myfunc()
print("Python is "+ x)
```

Output

Python is fantastic

Example

```
x=1
def one():
    x=2
    print(f"print var. from function scope {x}")
def two():
    x=4
    print(f"print var. from function scope {x}")

print(f"print var. from function global scope {x}")
one()
two()
```

Output

```
print var. from function global scope 1
print var. from function scope 2
print var. from function scope 4
```

Example…

```
x=1
def one():
    x=2
    print(f"print var. from function scope {x}")
def two():
    #x=4
    print(f"print var. from function scope {x}")
print(f"print var. from function global scope {x}")
one()
two()
```

Output

```
print var. from function global scope 1
print var. from function scope 2
print var. from function scope 1
```

Example…

```
x=1
def one():
    global x
    x=2
    print(f"print var. from function scope {x}")
def two():
    x=4
    print(f"print var. from function scope {x}")
print(f"print var. from function global scope {x}")
one()
two()
```

Output

```
print var. from function global scope 1
print var. from function scope 2
print var. from function scope 4
```

Example…

```
x=1
def one():
    global x
    x=2
    print(f"print var. from function scope {x}")
def two():
    x=4
    print(f"print var. from function scope {x}")
print(f"print var. from function global scope {x}")
one()
two()
print(f"print var. from function global scope after one function is called {x}")
```

Output

```
print var. from function global scope 1
print var. from function scope 2
print var. from function scope 4
print var. from function global scope after one function is called 2
```

Example…

```
# x=1
def one():
    global x
    x=2
    print(f"print var. from function scope {x}")
def two():
    x=4
    print(f"print var. from function scope {x}")
print(f"print var. from function global scope {x}")
one()
two()
print(f"print var. from function global scope after one function is called {x}")
```

Output

Error occurs because when you delete the initial value of the x variable, it must be defined before it can be used globally. Alternatively, call the function one() first to access the global variable.

Example…

```
def one():
    global x
    x=2
    print(f"print var. from function scope {x}")
def two():
    global x
    x=4
    print(f"print var. from function scope {x}")
one()
two()
print(f"print var. from function global scope after one function is called {x}")
```

Output

```
print var. from function scope 2
print var. from function scope 4
print var. from function global scope after one function is called 4
```

Example…

```
def one():
    global x
    x=2
    print(f"print var. from function scope {x}")
def two():
    print(f"print var. from function scope {x}")
one()
print(f"print var. from function global scope {x}")
two()
print(f"print var. from function global scope after one function is called {x}")
```

Output

```
print var. from function scope 2
print var. from function global scope 2
print var. from function scope 2
print var. from function global scope after one function is called 2
```

Thank you