MSE1 Answers

Subject : Python Programming (CS1005-2)

Prepared by : Dr. Ravi B

1 a) Explain the following operators with programming examples

i) Membership operator
ii) Logical operators

Ans:

i) Membership operator: Python's membership operators test for membership in a sequence. There are two membership operators in python: in and not in

Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

Examples using list:

```
fruits = ["Apple", "Orrange" , "Banana"]
if  "Apple"  in fruits :
      print("Present")
else:
      print("Not present")     # output : present
```

```
fruits = ["Apple", "Orrange" , "Banana"]
if  "Apple"  not in fruits :
      print("Present")
 else:
      print("Not present")  # Output : Not Present
```

ii) **Logical operators: There are 3 logical operators in python as shown in the table given below:**

Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

Example : Any valid example is considered if includes all operators

```
age = int(input("Enter age"))
if age  > 5 and  age < 10:
      print("John is studying in lower primary school")
elif age == 15 or age == 20:
      print("John is not studying in lower primary school")
elif not (age  > 5 and  age < 10):
      print("John stopped studying")
```

1 b) Write a python program that iterated through integers from 1 to 50. For each multiple of 3, print "Fizz" and for each multiple of 5 print "buzz" and for numbers that are multiple of both 3 and 5 print "Fizz Buzz"

Answer :

```
for i in range (1,51):
    if i % 3 == 0 and i % 5 == 0:    #multiple of both
        print("Fizz Buzz")
    elif i % 3 == 0:
        print("Fizz")
    elif i % 5 == 0:
        print("Buzz")
    else:
        pass    # multiple of neither 3 nor 5
```

Note : You should start checking for multiple of both 3 and 5. Otherwise, for multiple of both 3 and 5 program will not print "Fizz Buzz"

2 a) Explain looping statements with syntax and programming examples:

Answer : A looping statement executes a block statements repeatedly.

There are two looping statements in python **: while and for**

while: while statement executes a block of statements repeatedly until the given condition is false.

Syntax :

```
while condition:
```

Block of statements

Example: To generate n Fibonacci numbers (Any other valis example is accepted)

```
n = int(input("How many numbers:"))
f1, f2 = 0, 1
while n > 0:
    print(f1)
    f1, f2 = f2, f1+f2
    n -= 1
```

for: for loop is used to iterate through a sequence such as list, dictionary or tuple. It is also used to iterate through strings and a range of numbers.

Syntax :

for loop in sequence

```
for var in sequence:
    block of statements
```

for loop in range

```
for var in range(start,end,step):
    block of statements
```

Here start and step are optional

Exampe: iterating through a list

```
Fruits = ["Apple", "Banana", "Orrange"]
for fruit in Fruits:
```

```
        print(fruit)
#output:
Apple
Banana
Orrange
```

Exampe: iterating through a range of numbers

```
for i in range(1, 6):
        print(i, end=" ")
#output: 1 2 3 4 5
```

Exampe: iterating through a string

```
for i in "Python":
        print(i, end=" ")
#output: P y t h o n
```

2 b) Program to print prime numbers in an interval

```
#Prime numbers in range
import math
start = int(input("Enter start of the range:"))
end = int(input("Enter end of the range:"))
print(f"Prime numbers between {start} and {end}")
for i in range(start,end+1):
    if i == 2 or i == 3:
        print(i,end=" ")
    elif i % 2 == 0 or i % 3 == 0 or i == 1:
        pass
    else:
        for j in range(5,int(math.sqrt(i)+1),6):
            if i % j == 0 or i % (j+2) == 0 :
                break
        else:
            print(i, end=" ")  #print only if loop does not break
```

Note: above uses pass and for else (no flag required)

Or we can use flag as given in the following program:

```
import math
start = int(input("Enter start of the range:"))
end = int(input("Enter end of the range:"))
print(f"Prime numbers between {start} and {end}")

for i in range(start,end+1):
    flg = 0
    if i == 2 or i == 3:
        flg = 0
    elif i % 2 == 0 or i % 3 == 0 :
        flg = 1
    else:
        for j in range(5,int(math.sqrt(i))+1,6):
            if i % j == 0 or i % (j+2) == 0 :
                flg = 1
                break
    if flg == 0 and i != 1:
        print(i, end=" ")
```

(Any one program can be written)

3 a) Explain different types of arguments.

Python has the following types of arguments:

i) Required arguments (positional arguments)
ii) Default arguments
iii) arbitrary arguments (Variable length argument)
iv) Keyword arguments

i) **Required arguments: As** the name indicates, this argument must be provided in the calling function. The number and order of the arguments in calling function must match that of called function**.**

Example:

```
def my_function(fname, lname):
  print(fname + " " + lname)

my_function("Emil", "Refsnes")
```

def my_function()

ii) **Default arguments:** The called function uses default argument If we call the function without argument. Default arguments must always appear after all other arguments.

```
def my_function(name, country = "Norway"):
  print(name + " from " + country)

my_function("Emil", "Sweden")
my_function("Ramesh", "India")
my_function("Tobias")  #uses default country
my_function("John", "Brazil")
```

```
#Output

Emil from Sweden
Ramesh from India
Tobias from Norway
John from Brazil
```

Note : first argument is required argument (always required).

iii) **arbitrary arguments (Variable length argument) :** If you do not know how many arguments that will be passed into your function, add a * before the parameter name in the function definition. the function will receive a tuple of arguments, and can access the items accordingly.

```
def my_function(*kids):
  print("The youngest child is " + kids[2])

my_function("Emil", "Tobias", "Linus")
```

#Output: The youngest child is Linus

iv) **Keyword Arguments :** You can also send arguments with the key = value syntax. This way the order of the arguments does not matter.

```
def my_function(child3, child2, child1):
   print("The youngest child is " + child3)

my_function(child1 = "Emil", child2 = "Tobias", child3 = "Linus")
```

#Output: The youngest child is Linus

3 b) Write a program to find the factorial of a number using recursion

```
# factorial using recursion

def fact(n):
    if n == 0:
        return 1
    return n * fact(n-1)

n = int(input("Enter a number :"))
print(f"Factorial of {n} is {fact(n)}")
```

4 a) **compare and contrast the actual and formal parameters.**

Answer:

Actual Parameters	Formal Parameters
The values (expressions) given in the function call are referred to as the arguments or actual parameters	Formal parameters are those used in function definition
It is not necessary to mention the data type in the actual argument in languages such as C , C++	The receiving value's data type must be specified in languages such as C , C++
Parameters can be either constant values or variable names	Formal parameters are local variables of a function that are used in the function header
They represent the values received by the called method.	They represent the values passed to the called method.

4 b) Explain different ways of importing modules with programming example

Answer:

Modules in python are imported using import statement. There are mainly two ways of importing modules:

i) **Import an entire module**
Here we import an entire module. That means all variables, functions and classes (everything) are imported from that module. So we need to prefix the imported object with the module name.

Example: Finding square root of a number

```
import math
n = int(input ("Enter a number"))
print ("square root =", math.sqrt(n)) #prefixing is required
```

ii) **import required objects :** Here we import only required objects from a module. For example, in the above program we only required sqrt () function. So we can import this as shown below:

```
from math import sqrt
n = int(input ("Enter a number"))
print ("square root =", sqrt(n))  #prefixing is not  used
```

To import all the required objects, use comma separated object names as shown below. We are import sqrt, sin, log2 from math module.

```
from math import sqrt, sin, log2
n = int(input ("Enter a number:"))
print (f"square root = {sqrt(n):.2f}")
print (f"sin value = {sin(n):.2f}")
print (f"log 10 to tha base 2 = {log2(n):.2f}")
```

Enter a number:90
square root = 9.49
sin = 0.89
sin = 6.49

Note : .2f in print statement prints 2 digits after the decimal point