

**2021
REPORT**

PLAMENA GESHEVA
STUDENT ID 2019587

ASSIGNMENT 1- SOFTWARE
DEVELOPMENT
LECTURER RENUGA JAYAKUMAR



SOFTWARE DEVELOPMENT ASSESSMENT 1

BSC CLOUD COMPUTING
MODULE: ACCB4017
SOFTWARE DEVELOPMENT

TABLE OF CONTENT



TABLE OF CONTENT

1. PART A (UNDERSTANDING)

A1 Different types of control structures in programming.

A2 Different types of loops in Python

A3 Define list. Explain how lists are different from normal variables

2. PART B (IMPLEMENTATION)

Exercise 3:

Problem description

Solution

3. REFERENCES

A1 DIFFERENT TYPES OF CONTROL STRUCTURES IN PROGRAMMING

There are three basic types of logic, or flow of control, known as: Sequence logic, or sequential flow. Selection logic, or conditional flow. Iteration logic, or repetitive flow.

They can be combined in any way to solve a specified problem. Sequential is the default control structure, statements are executed line by line in the order in which they appear. Sequential is the default control structure, statements are executed line by line in the order in which they appear. The selection structure is used to test a condition.

- **Sequential-** default mode. Sequential execution of code statements (one line after another) -- like following a recipe
- **Selection-** used for decisions, branching -- choosing between 2 or more alternative paths.
- **Repetition-** used for looping, i.e. repeating a piece of code multiple times in a row.

A1 PYTHON SELECTION STATEMENTS

In Python, the selection statements are also known as decision making statements or branching statements. The selection statements are used to select a part of the program to be executed based on a condition- choosing between 2 or more alternative paths.

- if
- if...else
- if-elif statement

elif statement in Python

In Python, When we want to test multiple conditions we use elif statement.

The general syntax of if-elif-else statement in Python is as follows.

Syntax

```
if condition_1:
    Statement_1
    Statement_2
    Statement_3
    ...
elif condition_2:
    Statement_4
    Statement_5
    Statement_6
    ...
else:
    Statement_7
    Statement_8
    ...
```

A1 PYTHON SELECTION STATEMENTS

In the above syntax, whenever the condition_1 is True, the statements 1 2 and 3 are gets executed. If the condition_1 is False and condition_2 is True then the statements 4, 5, and 6 are gets executed. And if condition_1 nad Condition_2 both are False then the statements 7 and 8 are executed. Let's look at the following example of Python code.[1]

Python code to illustrate if-else statement

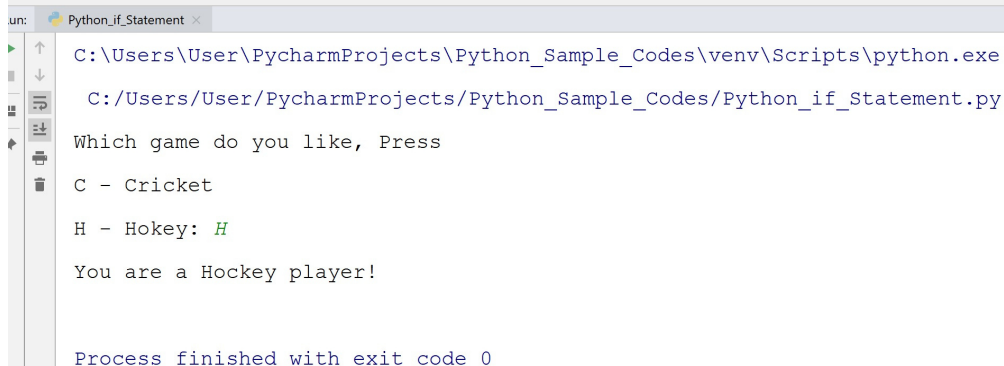
```
# Python code to illustrate elif statement
choice = input(f'Which game do you like, Press\nC - Cricket\nH - Hokey: ')
if choice == 'C':
    print('You are a Cricketer!')
elif choice == 'H':
    print('You are a Hockey player!')
else:
    print('You are not interested in Sports')
```

When we run the above code, it produces the output as follows.

A1 PYTHON SELECTION STATEMENTS

The output as follows.

```
1  # Python code to illustrate elif statement
2  choice = input(f'Which game do you like, Press\nC - Cricket\nH - Hokey: ')
3  if choice == 'C':
4      print('You are a Cricketer!')
5  elif choice == 'H':
6      print('You are a Hockey player!')
7  else:
8      print('You are not interested in Sports')
```



Python_if_Statement x

C:\Users\User\PycharmProjects\Python_Sample_Codes\venv\Scripts\python.exe
C:/Users/User/PycharmProjects/Python_Sample_Codes/Python_if_Statement.py

Which game do you like, Press
C - Cricket
H - Hokey: H
You are a Hockey player!

Process finished with exit code 0

A2 DIFFERENT TYPES OF LOOPS IN PYTHON

The other type of important programming control structure is a repetition statement. A repetition statement is used to repeat a group (block) of programming instructions. Repetition statements fall into two general categories; **while loops** and **for loops**. In many cases, the number of repetitions that a while loop executes depends on user input. In many cases, the number of repetitions a for loop executes for is determined by the number of elements in an array or an array-like object (such as a list or a tuple in Python). This is why for loops are often associated with some indexed quantity like a list.

- **While loops**-used when a block of statements have to be repeated, but the number of repetitions is not a fixed amount. This means that a while loop might repeat two times when a program is run and repeat a different number of times the next time that same program is run. This feature makes while loops flexible.
- **For loops**- For loops can iterate over a sequence of numbers using the "range" and "xrange" functions. The difference between range and xrange is that the range function returns a new list with numbers of that specified range, whereas xrange returns an iterator, which is more efficient. (Python 3 uses the range function, which acts like xrange). Note that the range function is zero based.

A2 DIFFERENT TYPES OF LOOPS IN PYTHON

- **Nested loops**-A nested loop has one loop inside of another. These are typically used for working with two dimensions such as printing stars in rows and columns as shown below. When a loop is nested inside another loop, the inner loop runs many times inside the outer loop. In each iteration of the outer loop, the inner loop will be re-started. The inner loop must finish all of its iterations before the outer loop can continue to its next iteration.

A2 WHAT IS A FOR LOOP IN PYTHON?

Python for loops are used to loop through an iterable object (like a list, tuple, set, etc.) and perform the same action for each entry. For example, a for loop would allow us to iterate through a list, performing the same action on each item in the list.

An iterable object, is any Python object we can iterate through, or “loop” through, and return a single element at a time.

Lists, for example, are iterable and return a single list entry at a time, in the order entries are listed.

Strings are iterable and return one character at a time, in the order the characters appear. Etc.

We create a for loop by first defining the iterable object you’d like to loop through, and then defining the actions we’d like to perform on each item in that iterable object. For example, when iterating through a list, we first specify the list you’d like to iterate through, and then specify what action you’d like to perform on each list item.[2]

example: if we had a list of names stored in Python, we could use a for loop to iterate through that list, printing each name until it reached the end.

```
our_list = ['Plamena', 'Mariana', 'Daniela', 'Petko']  
for name in our_list:  
    print(name)
```

output:

A2 WHAT IS A FOR LOOP IN PYTHON?

Plamena
Mariana
Daniela
Petko

Because for loops iterate through lists, tuples, etc. in sequence, this variable can actually be called almost anything. Python will interpret any variable name we put in that spot as referring to each list entry in sequence as the loop executes.

So, in the code above:

- name points to 'Lily' on the first iteration of the loop...
- ...then 'Brad' on the second iteration of the loop...
- ...and so on.

This will be the case regardless what we call that variable. So if, for example, we rewrite our code to replace name with x, we'll get the same exact result:

```
for x in our_list:  
    print(x)
```

The output will be:

A2 WHAT IS A FOR LOOP IN PYTHON?

Plamena
Mariana
Daniela
Petko

This technique works with any iterable object. For example, strings are iterable, and we can use the same sort of for loop to iterate through each character in a string:

```
for letter in 'Plamena':  
    print(letter)
```

When execute the output is:

P
l
a
m
e
n
a

A3 LIST IN PYTHON

In Python, a list is a collection of ordered and indexed elements of different data types. In Python, the list and its elements are mutable. That means, the list and its elements can be modified at any time in the program. In Python, the list data type (data structure) has implemented with a class known as a list. Lists are created using square brackets and each element must be separated with a comma symbol.

Example:

Create a List:

```
thislist = ["apple", "banana", "cherry"]  
print(thislist)
```

How lists are different from normal variables?

The list is mutable, and can be changed in-place after it was created. A variable is a "sort-of" storage location with a name attached to it, it can contain any single object. A variable is an object that stores a piece of information within a naming convention to be used in programming: a=1 b=2 c=3 etc.

Accessing Elements of a list in Python

In Python, the list elements are organized using index values that start with '0' (zero) at first element and ends with 'length of the list - 1' at last element. The individual elements of a list are accessed using the index values.

Example:

```
list_name[index]
```

A3 LIST IN PYTHON

In Python, a list is a collection of ordered and indexed elements of different data types. In Python, the list and its elements are mutable. That means, the list and its elements can be modified at any time in the program. In Python, the list data type (data structure) has implemented with a class known as a list. Lists are created using square brackets and each element must be separated with a comma symbol.

Example:

Create a List:

```
thislist = ["apple", "banana", "cherry"]  
print(thislist)
```

How lists are different from normal variables?

The list is mutable, and can be changed in-place after it was created. A variable is a "sort-of" storage location with a name attached to it, it can contain any single object. A variable is an object that stores a piece of information within a naming convention to be used in programming: a=1 b=2 c=3 etc.

Accessing Elements of a list in Python

In Python, the list elements are organized using index values that start with '0' (zero) at first element and ends with 'length of the list - 1' at last element. The individual elements of a list are accessed using the index values.

Example:

```
list_name[index]
```

A3 LIST IN PYTHON

In Python, a list is a collection of ordered and indexed elements of different data types. In Python, the list and its elements are mutable. That means, the list and its elements can be modified at any time in the program. In Python, the list data type (data structure) has implemented with a class known as a list. Lists are created using square brackets and each element must be separated with a comma symbol.[3][4]

Example:

Create a List:

```
thislist = ["apple", "banana", "cherry"]  
print(thislist)
```

How lists are different from normal variables?

The list is mutable, and can be changed in-place after it was created. A variable is a "sort-of" storage location with a name attached to it, it can contain any single object. A variable is an object that stores a piece of information within a naming convention to be used in programming: a=1 b=2 c=3 etc.

Accessing Elements of a list in Python

In Python, the list elements are organized using index values that start with '0' (zero) at first element and ends with 'length of the list - 1' at last element. The individual elements of a list are accessed using the index values.

Example:

```
list_name[index]
```

EXERCISE 3:

Problem Description:

Write a program that allows the user to enter the names of five staff members working in IT department into an array/list and their monthly salary of the staff member into a second array/list. When all the staff names and their salaries have been entered, print them on the screen, then calculate total how much money in total the IT department requires to spend on employees' salary every month. Finally, help the department to check how much total money they need if 10% increase on overall salary.

Solution: Program to store employee details using Class

The Employee class has a class variable named count, which is used to track the number of employees in the organization. As count is a class variable we cannot access this variable directly. We need to use the class name to access the class variable.

init() method is called whenever an object of a class is created and used to initialize the attributes of the object. The class variable count is incremented as and when a new employee object is created.

DisplayCount() method is used to print the number of employees in the organization.

EXERCISE 3:

DisplayEmp() method is used to display the employee information such as employee name, and employee salary.

Finally, in the main part of the program, two employee objects are created. Then the number of employees working in the organization is displayed by calling the displayCount() method and employee information is displayed by calling displayEmp() method.

init() method is called whenever an object of a class is created and used to initialize the attributes of the object. The class variable count is incremented as and when a new employee object is created.

DisplayCount() method is used to print the number of employees in the organization.

DisplayEmp() method is used to display the employee information such as employee name, and employee salary.

Finally, in the main part of the program, two employee objects are created. Then the number of employees working in the organization is displayed by calling the displayCount() method and employee information is displayed by calling displayEmp() method.

EXERCISE 3:

SCRIPT:

```
class Employee:
    count = 0

    def __init__(self, name, salary):
        self.name = name
        self.salary = salary
        Employee.count += 1

    def displayCount(self):
        print("There are %d employees" % Employee.count)

    def displayDetails(self):
        print("Name:", self.name, ", Salary:", self.salary)

e1 = Employee("Paul", 1300.00)
e2 = Employee("Susan", 1250.00)
e3 = Employee("James", 1300.00)
e4 = Employee("Rossie", 1350.00)
e5 = Employee("Alan", 1400.00)
e5.displayCount()
print("Details of all employee:")
e1.displayDetails()
e2.displayDetails()
e3.displayDetails()
e4.displayDetails()
```

EXERCISE 3:

OUTPUT:

```
"C:\Users\Plamena  
Gesheva\PycharmProjects\pythonProject1\venv\Scripts\python.exe"  
"C:/Users/Plamena Gesheva/PycharmProjects/pythonProject1/main.py"  
There are 5 employees  
Details of all employee:  
Name: Paul , Salary: 1300.0  
Name: Susan , Salary: 1250.0  
Name: James , Salary: 1300.0  
Name: Rossie , Salary: 1350.0  
  
Process finished with exit code 0
```

REFERENCES

[1]

J. Panic, "What are the 3 types of control structures? – JanetPanic.com," janetpanic.com, May 6AD. <https://janetpanic.com/what-are-the-3-types-of-control-structures/> (accessed Aug. 25, 2021).

[2]

D. L. Inc. , "The Basics of Python For Loops," Dataquest, May 30, 2019. <https://www.dataquest.io/blog/python-for-loop-tutorial/>.

[3]

Mr. R. B, "Python Tutorials - Lists data structure | data types," [www.btechsmartclass.com](http://www.btechsmartclass.com/python/Python_Tutorial_Python_Lists.html), Jul. 21, 2021. http://www.btechsmartclass.com/python/Python_Tutorial_Python_Lists.html (accessed Aug. 24, 2021).

[4]

w3 schools, "Python Lists," W3schools.com, Aug. 05, 2019. https://www.w3schools.com/python/python_lists.asp.