



# Computer Science

# Lesson Objectives



## FUNCTION VS PROCEDURE

- Understand what's the difference?
- How can you use them to make your program robust.

## CASE Statement

- Understand what CASE selection is done in Python
- Implement Case Selection and understand how can you use them to make a program robust.

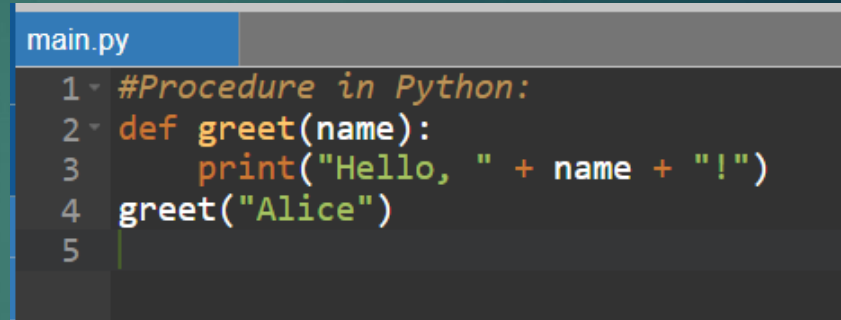
# PYTHON FUNCTION VS PROCEDURE

#Procedure in Python:

```
def greet(name):  
    print("Hello, " + name + "!")  
greet("Alice")
```

Output:

???



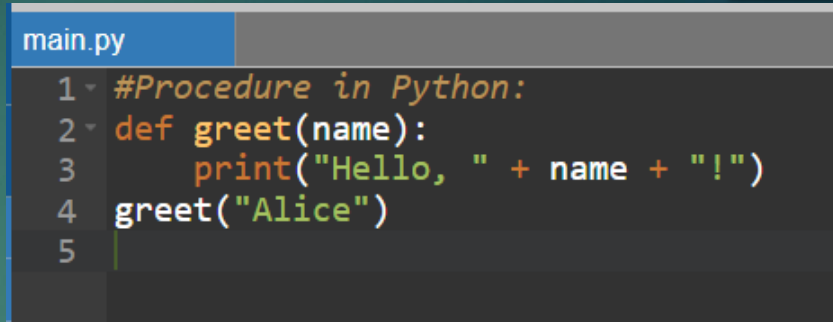
```
main.py  
1 #Procedure in Python:  
2 def greet(name):  
3     print("Hello, " + name + "!")  
4 greet("Alice")  
5
```

In the above example, the greet procedure takes a parameter name and prints a greeting message. It doesn't return any value.

# PYTHON FUNCTION VS PROCEDURE

#Procedure in Python:

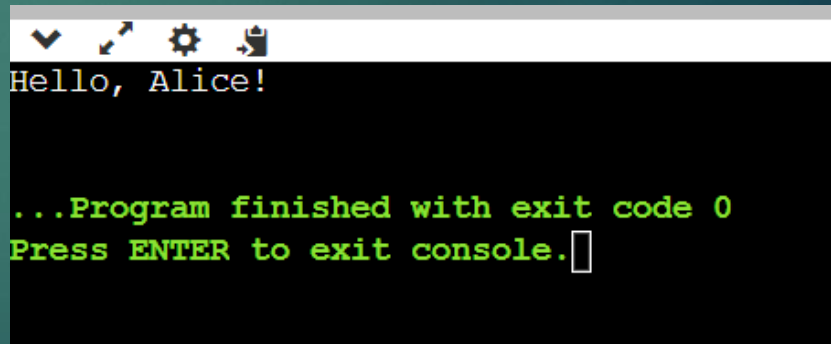
```
def greet(name):  
    print("Hello, " + name + "!")  
greet("Alice")
```



```
main.py  
1 #Procedure in Python:  
2 def greet(name):  
3     print("Hello, " + name + "!")  
4 greet("Alice")  
5
```

Output:

Hello, Alice!



```
Hello, Alice!  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

In the above example, the greet procedure takes a parameter name and prints a greeting message. It doesn't return any value.

# PYTHON FUNCTION VS PROCEDURE

# Function Call in Python that returns a value

```
def add_numbers(a, b):
```

```
    return a + b
```

# This will return a value of the function call.

```
print("Please enter your first number: ")
```

```
a=int(input())
```

```
print("Please enter your second number: ")
```

```
b=int(input())
```

```
result = add_numbers(a, b)
```

```
print("\nThe result is: ", result)
```

```
main.py
1 # Function Call in Python that returns a value
2 def add_numbers(a, b):
3     return a + b
4 # This will return a value of the function call.
5 print("Please enter your first number: ")
6 a=int(input())
7 print("Please enter your second number: ")
8 b=int(input())
9 result = add_numbers(a, b)
10 print("\nThe result is: ", result)
11
```

In the above example, the add\_numbers function takes two parameters a and b, adds them together, and returns the sum.

# PYTHON FUNCTION VS PROCEDURE

# Function Call in Python that returns a value

```
def add_numbers(a, b):
```

```
    return a + b
```

# This will return a value of the function call.

```
print("Please enter your first number: ")
```

```
a=int(input())
```

```
print("Please enter your second number: ")
```

```
b=int(input())
```

```
result = add_numbers(a, b)
```

```
print("\nThe result is: ", result)
```

A screenshot of a terminal window with a black background and white text. The terminal shows the execution of a Python program. It starts with a prompt "Please enter your first number:" followed by the input "25". Then another prompt "Please enter your second number:" followed by the input "45". The output is "The result is: 70". At the bottom, there is a green message: "...Program finished with exit code 0 Press ENTER to exit console." with a cursor. The terminal window has a white title bar with standard icons.

In the above example, the `add_numbers` function takes two parameters `a` and `b`, adds them together, and returns the sum.



# CASE STATEMENT

- How can I make a CASE statement in Python?

# CASE STATEMENT

- Using FUNCTION to switch between each choice.

```
main.py
1 def vowel(num):
2     switch={
3         1:'a',
4         2:'e',
5         3:'i',
6         4:'o',
7         5:'u'
8     }
9     return switch.get(num,"Invalid input")
10 print("What is your choice ?\n from 1 to 5")
11 x = input()
12 print ("You have chosen", x, "and it is", vowel(int(x)))
```



# CASE STATEMENT

- Implementing a CASE Choice

main.py

```
1 import os
2 def select(num):
3     switch={
4         1: "Cash Register",
5         2: "Monthly Report",
6         3: "Menu of the day",
7         4: "Exit"
8     }
9     return switch.get(num,"Please try again")
10 num=int(0)
11 while int(num) < 4:
12     print("\n\n\n")
13     print ("Please enter a choice \n 1: Cash Register \n 2: Monthly Report \n 3: Menu of the day \n 4: Exit")
14     num = input ()
15     print("\n")
16     print(" You have chosen ", num, " as the ", select(int(num)))
17     input()
18     os.system('clear')
19 print("You have exited")
```

# CASE STATEMENT

```
main.py
1 import os #This import os, is a built-in function to allow the os.system('clear') to work.
2 def select(num):
3     switch={
4         1: "Cash Register",
5         2: "Monthly Report",
6         3: "Menu of the day",
7         4: "Exit"
8     }
9     return switch.get(num,"Please try again")
10 num=int(0)
11 while int(num) < 4:
12     print("\n\n\n")
13     print ("Please enter a choice \n 1: Cash Register \n 2: Monthly Report \n 3: Menu of the day \n 4: Exit")
14     num = input ()
15     print("\n")
16     print(" You have chosen ", num, " as the ", select(int(num)))
17     input()
18     os.system('clear') |
19     print("You have exited")
```

input

```
Please enter a choice
1: Cash Register
2: Monthly Report
3: Menu of the day
4: Exit
█
```



# Teach Computer Science

Data structures



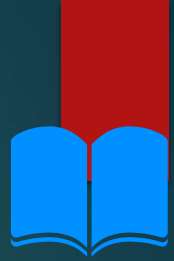
# Lesson Objectives

Students will learn:

- Why variables are not enough in real-life programming
- What are data structures?
- What are arrays?
- Accessing arrays using Python programming language

A large, stylized blue number '1.' is positioned in the upper right corner. A solid red vertical bar is located directly above the period of the number.

Content



# Why variables are not enough?

- The primary purpose of a computer system is to process data. The data can be of any type.
- If a computer program needs to process one piece of data, then that data can be stored in a variable and the resultant data can be stored in another variable. In this case, a couple of variables are sufficient to write a program.
- But real-life programs need to handle tens of thousands of pieces of data. To handle large amounts of data, it is impossible to store each piece data in a separate variable.



# Data structure

- Because variables cannot be used for real-life programming, data is grouped together or, more precisely, structured in a specific format.
- The collection of data structured together is called a data structure.

Characteristic	Description	Examples
Linear	In linear data structures, the data items are arranged in a linear sequence.	Array
Non-Linear	In non-linear data structures, the data items are not in sequence.	Tree, Graph
Homogeneous	In homogeneous data structures, all the elements are of the same type.	Array
Non-homogeneous	In non-homogeneous data structures, the elements may or may not be of the same type.	Structures
Static	Static data structures are those whose size and structure-associated memory locations are fixed at compile time.	Array
Dynamic	Dynamic structures are that which expand or shrink depending upon the program's needs and its execution. Also, their associated memory locations change.	Linked List



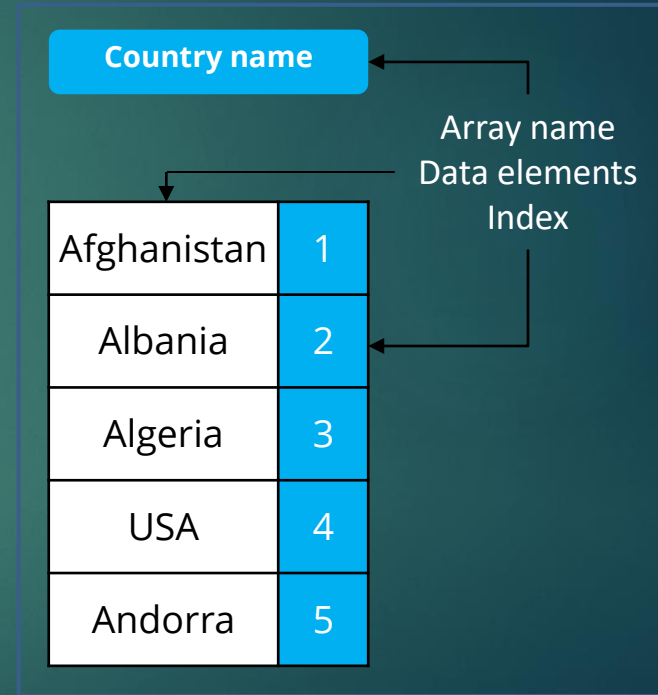


# Arrays

- Arrays are a form of data structure used to store a set of data elements in a certain order.
- All the data elements should be of the same data type.
- An array can store a set of integer values or a set of character strings.

# Arrays

- Each element of an array is stored in memory as per their order.
- This is a one-dimensional array, as the array has only one list.
- If the minimum, maximum and average temperature for each nation needs to be included, then there will be another list for each data element.



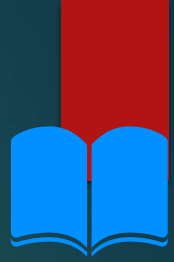
# Two-dimensional arrays

Country name, Max. Min. and Avg. temperatures

Country	Max.	Min.	Avg.
Afghanistan	40	20	32
Albania	38	28	35
Algeria	44	25	38
USA	38	2	30
Andorra	36	14	32

Diagram illustrating a two-dimensional array structure. The array is represented as a table with 5 rows and 4 columns. The columns are labeled "Country", "Max.", "Min.", and "Avg.". The rows are labeled "Row" and "Column". The data is as follows:

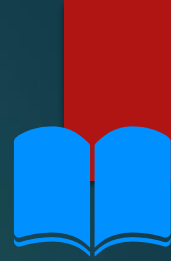
Country	Max.	Min.	Avg.
Afghanistan	40	20	32
Albania	38	28	35
Algeria	44	25	38
USA	38	2	30
Andorra	36	14	32



# Declaring arrays in Python

- In Python, the array module is initiated from the Python library.
- The syntax of declaring an array is:

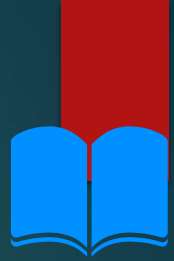
```
Array_name=array.array("type code", range(number of elements)).
```



# Declaring arrays

- Let us create an array with the name `ProductPrice` to enter the price of various products.
- Let us assume that there are ten products in total.
- `NoItems` is a constant with a value of 10.
- An array starts with element 0.

```
import array  
NoItems=int(10)  
k=0  
ProductPrice=array('i', range(NoItems))
```



# Using arrays in Python

- Data is entered into the array ProductPrice created by using iteration statements.
- A value k is initiated to zero. In the loop, the value of k is incremented every time. The value of k is the index to the array.
- Similarly, using an iteration statement, the elements of this array is printed.

# Example

- The program for entering data and printing it to and from the array ProductPrice is given.
- For loop can also be used in this program.

```
import array
NumberofItems=int(10)
k=0
ProductPrice=array.array('i', range(NumberofItems))
while (k<10):
    ProductPrice[k]=int(input("Enter the product price: "))
    k=k+1
k=0
print("""The product prices are: """)
while (k<10):
    print(ProductPrice[k])
    k=k+1
```

```
Enter the product price: 10
Enter the product price: 20
Enter the product price: 11
Enter the product price: 42
Enter the product price: 65
Enter the product price: 77
Enter the product price: 81
Enter the product price: 31
Enter the product price: 63
Enter the product price: 91
The product prices are:
10
20
11
42
65
77
81
31
63
91
```

```
import array

NumberofItems=int(10)

k=0

ProductPrice=array.array('i', range(NumberofItems))

while (k<10):

    ProductPrice[k]=int(input("Enter the product price: " ))

    k=k+1

k=0

print("""The product prices are: ")

while (k<10):

    print(ProductPrice[k])

    k=k+1
```





# Creating two-dimensional arrays

- Two-dimensional arrays are created by nesting two while loops or two for loops.
- Arrays can also be created using program statements.
- An one-dimensional array is created by:

```
Array_name=[10, 11, 12, 13, 14]
```

- A two-dimensional array is created by:

```
Array_name=[ [1, 2, 3, 4], [5, 6, 7, 8]
```

- The elements 1, 2, 3 and 4 represent a row in the two-dimensional array.



# Accessing elements in two-dimensional arrays

```
import array
Array_2d = [[10, 2, 15, 12], [3, 12, 8], [10, 8, 12, 5], [12, 15, 8, 6]]
for row in Array_2d:
    for element in row:
        print(element, end = " ")
    print()
```



# Accessing elements in two-dimensional arrays

- The `print()` statement is used to print each row in a different line.
- The statement `end=""` makes sure that all elements in a row are printed in the same line with a space between each element.

```
import array
Array_2d = [[10, 2, 15, 12], [3,
12 ,8], [10, 8, 12, 5], [12,15,8,6]]
for row in Array_2d:
    for element in row:
        print(element, end = " ")
    print()
```



# Accessing elements in two-dimensional arrays

Output:

```
10 2 15 12
3 12 8
10 8 12 5
12 15 8 6
```

```
import array
Array_2d = [[10, 2, 15, 12], [3, 12, 8], [10, 8, 12,
5], [12, 15, 8, 6]]
for row in Array_2d:
    for element in row:
        print(element, end = " ")
    print()
```



# len() function for two-dimensional arrays

- Number of elements in an array can be found out using len function.
- 1D array: len function returns the number of elements directly.
- 2D array: len function returns the number of rows first. To obtain the number of columns of a 2D array, the row number must be specified.

```
>>> Array_1d=[10, 11, 12, 13, 14]
>>> len(Array_1d)
5
>>> Array_2d=[ [1, 2, 3, 4, 5], [6, 7, 8, 9, 10]]
>>> len(Array_2d)
2
>>> len(Array_2d[1])
5
```



# Modifying elements of an array

- Elements of an array can be modified.
- In pseudocode, the statement `Array_2d[1][3]=15` is replaced by `Array_2d[1][3] ← 15`.

```
>>> Array_2d=[ [1, 2, 3, 4, 5], [6, 7, 8, 9, 10]]
>>> Array_2d[1][3]=15
>>> print(Array_2d)
[[1, 2, 3, 4, 5], [6, 7, 8, 15, 10]]
```

# Activity-1

Duration: 20 minutes

1. Write a Python program to create an array holding 10 elements. Ask the user to enter a number into the array. Use two different arrays to store even and odd numbers separately and print them.

```
Enter element:5
Enter element:6
Enter element:7
Enter element:8
Enter element:9
Enter element:12
Enter element:4
Enter element:2
Enter element:4
Enter element:8
The even list
6 8 12 4 2 4 8
The odd list
5 7 9

...Program finished with exit code 0
Press ENTER to exit console.█
```

1. Write a Python program to create an array holding 10 elements. Ask the user to enter a number into the array. Use two different arrays to store even and odd numbers separately and print them.

```
#declaring an array
import array
a=array.array('i',range(10))
#entering data into array
i=0
while(i<10):
    a[i]=int(input("Enter element:"))
    i=i+1
even=array.array('i',range(10))
odd=array.array('i',range(10))
j=0
k=0
l=0
#separating even and odd numbers into separate list
while (j<10):
    if(a[j]%2==0):
        even[k]=a[j]
        k=k+1
    else:
        odd[l]=a[j]
        l=l+1
    j=j+1
#printing both the lists
print("The even list")
x=0
while (x<k):
    print(even[x],end=" ")
    x=x+1
print("\n\nThe odd list")
y=0
while (y<l):
    print(odd[y],end=" ")
    y=y+1
```





# Records

## Array

- Data structures with elements of the same data type.
- The elements can be changed.
- Fixed number of fields.
- Static data structure.

## Tuple

- Records with elements of different data types.
- The elements cannot be changed.
- The length of tuple is fixed.
- Static data structure.

## List

- Elements with different data structure.
- The elements can be changed.
- Variable number of fields.
- Dynamic data structure.



# Records

- A data structure that may consist of elements with different data types and variable number of fields.
- Each field in a record is identified using a specific name.
- The elements of a record are related to each other.



# Tuples

- Tuples are represented with brackets (). The Python code that creates a tuple is:

```
>>> tuple_a=()
>>> print(tuple_a)
()
```

- tuple\_a is a zero-element tuple.
- To create a tuple with one element, the Python code used is:

```
>>> tuple_b=( 1,)
>>> print(tuple_b)
(1,)
```

*Use a comma after the element in tuple so that Python recognises this variable as a tuple.*



# Tuples

- Groups of data are stored in Python using the code:

```
>>> tuple_c=(1, 2, 'three', 4, 'five')
>>> print(tuple_c)
(1, 2, 'three', 4, 'five')
```

- Different data elements are packed together in a tuple.
- These data elements can be unpacked too.



# Tuples: unpacking

- Consider a tuple, `student1=(1, 'Alice', 'Engineering')`.
- Unpacking a tuple in Python:

```
>>> (student_number, name, ambition)=student1
>>> print(student_number)
1
>>> print(name)
Alice
>>> print(ambition)
Engineering
```



# Tuples: len() function

- len function is used to find the length of tuples.
- Individual values can be accessed from tuples as we did for arrays.
- An index value of -1 returns the last item in tuple.

```
>>> print(student1[0])  
1  
>>> print(student1[1])  
Alice  
>>> print(student1[-1])  
Engineering
```



# Tuples: 'in' operator

- The 'in' operator is used to check whether an item exists in a tuple or not.

```
>>> tuple1=('rose', 'lotus', 'tulip', 'lily', 'jasmine')
>>> print('lily' in tuple1)
True
```



# Lists

- Dynamic data structures.
- Their values can be changed.
- Square brackets are used to create lists.
- Unlike tuples, the data items can be changed in lists.

```
>>> list1=[1, 2, 'three', 'four', 5]
>>> print(list1)
[1, 2, 'three', 'four', 5]
>>> list1[3]
'four'
>>> len(list1)
5
>>> list1[3]=4
>>> print(list1)
[1, 2, 'three', 4, 5]
```





# Let's review some concepts

## Data structure

The collection of data structured together in a specific format is called data structure

## Creating one-dimensional array using program statements

A one-dimensional array is created by:

```
Array_name=[elements]
```

## Arrays

Arrays are a form of data structure used to store a set of data elements in a certain order.

## Creating two-dimensional array using program statements

A two-dimensional array is created by:

```
Array_name=[ [elements in row 1], [elements in row 2] and so on]
```

## Declaring arrays in Python

In Python, the array module is initiated from the Python library. The syntax of declaring an array is:

```
Array_name=array.array("type code", range(number of elements))
```

## Records

A data structure that may consist of elements with different data structure and variable number of fields.

Implemented in Python using an array, tuple or list.



2.

Activity



# Activity-2

Duration: 15 minutes

1. Create a tuple. Write a Python program to find the number of elements in a tuple.
2. Create a tuple with five elements (both integer and string). Write a Python program to find out whether an item (integer and string) exists in a tuple or not.
3. Create a tuple of at least three elements. Write a Python program to unpack the tuple and print the elements.



# Activity-2

Duration: 15 minutes

1. Create a tuple. Write a Python program to find the number of elements in a tuple.

```
tuple1=(1, 2, 'three', 4, 'five')  
print(len(tuple1))
```



## Activity-2

Duration: 15 minutes

2. Create a tuple with five elements (both integer and string). Write a Python program to find out whether an item (integer and string) exists in a tuple or not.

```
tuple1=(1, 2, 'three', 4, 'five')  
print(4 in tuple1)  
print('five' in tuple1)
```



# Activity-2

Duration: 15 minutes

3. Create a tuple of at least three elements. Write a Python program to unpack the tuple and print the elements.

```
tuple2=('Tablet', 'Acer', 150)
(product_name, product_brand, price)=tuple2
print(product_name)
print(product_brand)
print(price)
```



3.

End of topic questions



## End of topic questions

1. What is data structure? What is its significance in programming?
2. What are arrays? How is the data stored in memory?
3. How are arrays declared in Python?
4. How is data in arrays accessed?
5. How do you create a two-dimensional array using Python?





# End of topic questions

6. How are elements of a two-dimensional array printed? Explain using an example.
7. In the following statement, what is the significance of `end=" "`?  
*while (j<10):*  
*print(array[j], end = " ")*
8. What are the differences between arrays and tuples?
9. What are the differences between tuples and lists?



# Credit

- TEACH COMPUTER SCIENCE