unacademy

Python as Oops Part 2

Course on Data Structure and Algorithms Using Python

Ankush Saklecha • Lesson 8 • Dec 15, 2023

Python Programming

Dictionaries

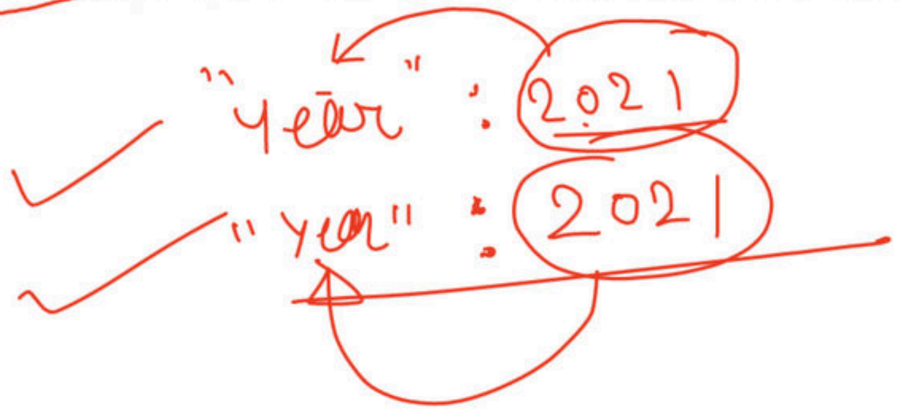
Function and It's Types

Python Dictionaries

- Dictionary
- Dictionaries are used to store data values in key:value pairs.
- A dictionary is a collection which is ordered*, changeable and do not allow duplicates.

As of Python version 3.7, dictionaries are *ordered*. In Python 3.6 and earlier, dictionaries are *unordered*.

```
dict1 = {
  "brand": "Tata",
  "model": "Tiago",
  "year": 2021
}
print(dict1)
```



① d = { "brand" : "Tata", "brand" : "Ford", "Model": "Tiago",
"year" : 2021 }

Print (d) ⇒ { "brand" : "Ford", "Model" : "Tiago", "year" : 2021 }

② d = { "brand" : "Tata", "brand" : "Tata", "Model": "Tiago",
"year" : 2021 }

Print (d) ⇒ { "brand" : 'Tata', "Model" : 'Tiago', "year" : 2021 }

d = { "brand" : "Tata", "brand1" : "Tata", "Model": "Tiago",
"year" : 2021 }

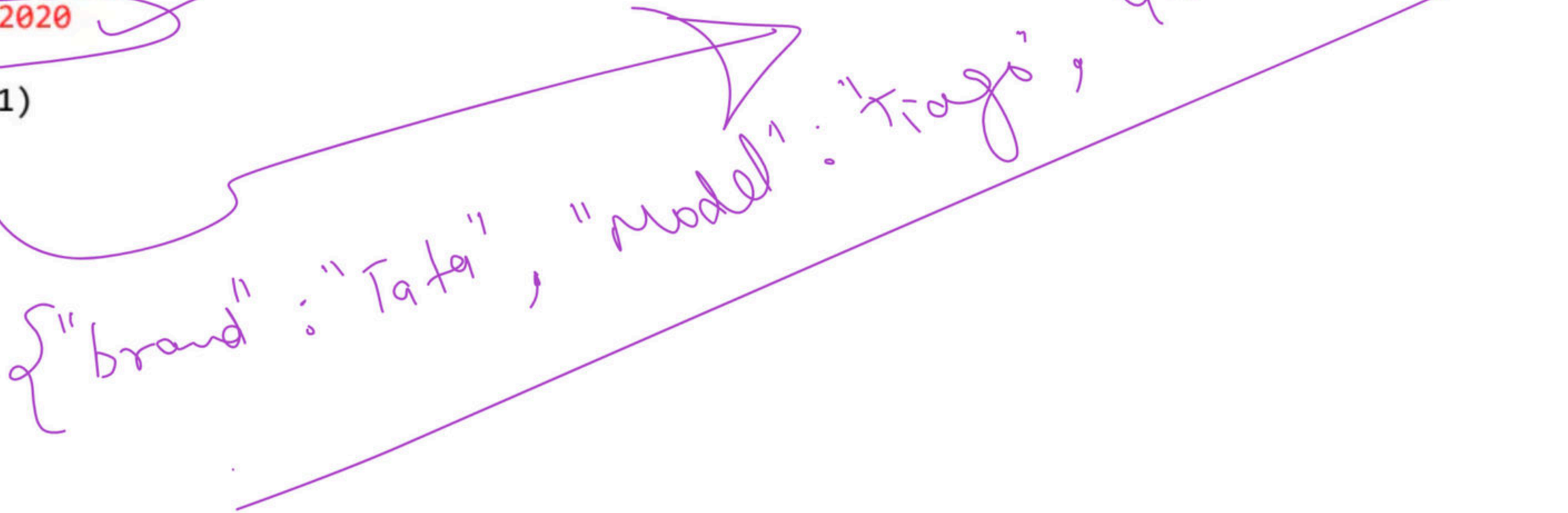
Print (d) ⇒

```
dict1 = {
   "brand": “Tata",
   "model": “Tiago",
   "year": 2021
}
print(dict1["brand"])
```

- Dictionaries are changeable, meaning that we can change, add or remove items after the dictionary has been created.

- Duplicate values will overwrite existing values:

```
dict1 = {
    "brand": "Tata",
  "model": "Tiago",
  "year": 2021,
   "year": 2020
}
print(dict1)
```



Dictionary Length

To determine how many items a dictionary has, use the `len()` function:

```
dict1 = {
    "brand": “Tata",
    "model": “Tiago",
    "year": 2021,
    "year": 2020
}
print(len(dict1))

dict1 = {
  "brand":  “Tata",
  "electric": True,
  "year": 2004,
  "colors": [“black", "white", "blue"]  // heterogenous elements
}

dict1 = {
    "brand": “Tata",
    "model": “Tiago",
    "year": 2021
}
print(type(dict1))   //<class ‘dict’>
```

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
x = thisdict.get("model")
print(x)
```

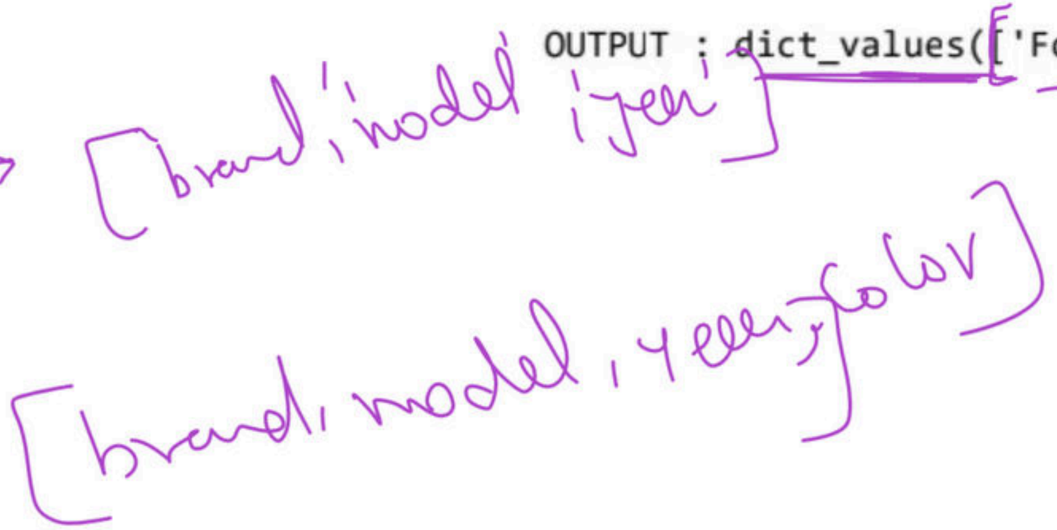
```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}

x = thisdict.keys()

print(x)
```

o/p : `dict_keys(['brand', 'model', 'year'])`

The list of the values is a *view* of the dictionary, meaning that any changes done to the dictionary will be reflected in the values list.

```
car = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}

x = car.keys()

print(x) #before the change

car["color"] = "white"

print(x) #after the change
```

```
thisdict = {
 "brand": "Ford",
 "model": "Mustang",
 "year": 1964
}

x = thisdict.values()

print(x)

OUTPUT : dict_values(['Ford', 'Mustang', 1964])
```



1

```
dict1 = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
if "model" in dict1:
  print("Yes, 'model' is one of the keys in the thisdict dictionary")`
```

2

```
dict1 = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
dict1.update({"year": 2020})
```

```
dict1 = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
dict1["color"] = "red"
print(dict1)
```

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.update({"color": "red"})
```

The `pop()` method removes the item with the specified key name:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.pop("model")
print(thisdict)
```

The `popitem()` method removes the last inserted item (in versions before 3.7, a random item is removed instead):

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.popitem()
print(thisdict)
```

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
del thisdict["model"]
print(thisdict)
```

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
del thisdict
print(thisdict) #this will cause an error
because "thisdict" no longer exists.
```

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.clear()
print(thisdict)
```

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
for x in thisdict:
  print(x)
```

```
Output :
brand
model
year
```

brand

x

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
for x in thisdict:
  print(thisdict[x])
```

```
Output:
Ford
Mustang
1964
```

td [brand]

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
for x in thisdict.values():
  print(x)
```

```
Output:
Ford
Mustang
1964
```

```
dict1 = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
mydict = dict1.copy()
print(mydict)
```

Nested Dictionaries

A dictionary can contain dictionaries, this is called nested dictionaries.

```
myfamily = {
  "child1" : {
    "name" : "Emil",
    "year" : 2004
  },
  "child2" : {
    "name" : "Tobias",
    "year" : 2007
  },
  "child3" : {
    "name" : "Linus",
    "year" : 2011
  }
}
```

Or, if you want to add three dictionaries into a new dictionary:

```
child1 = {
  "name" : "Emil",
  "year" : 2004
}
child2 = {
  "name" : "Tobias",
  "year" : 2007
}
child3 = {
  "name" : "Linus",
  "year" : 2011
}

myfamily = {
  "child1" : child1,
  "child2" : child2,
  "child3" : child3
}

print(myfamily["child2"]["name"])
```

Method	Description
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
fromkeys()	Returns a dictionary with the specified keys and value
get()	Returns the value of the specified key
items()	Returns a list containing a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

Content

- Function
- Function Types
- User Defined Functions
- Recursive Functions
- Sample Problems

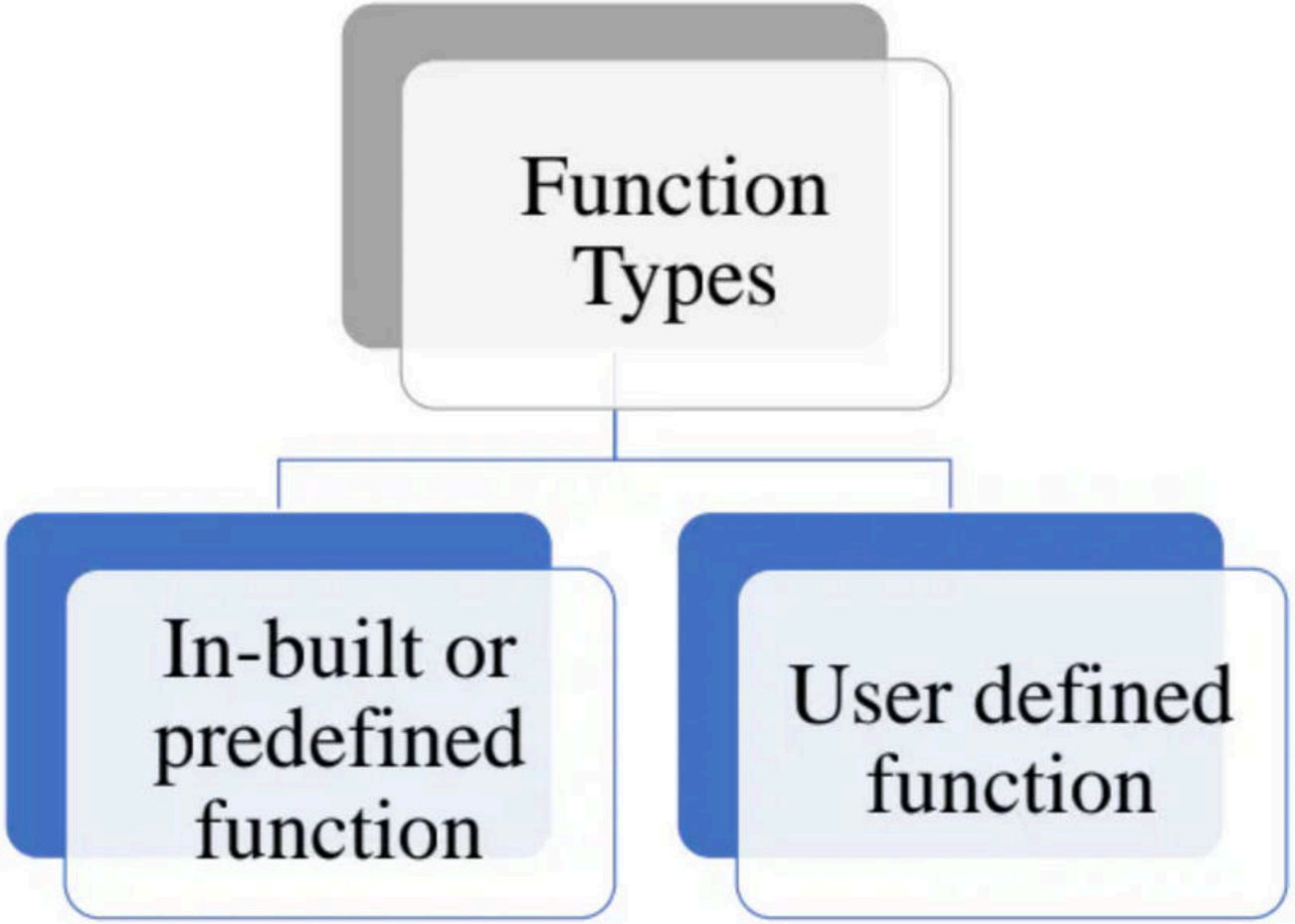
Function

A function is group of statements which are executed when it is called.

In-built Function Examples:	User Defined Function Example:
Name='Jhon' Qualification="Master of Technology" print(len(Name)) print(len(Qualification))	Def power(x,y): print(x**y) x,y=input('Enter two numbers: ').split(' ') x=int(x) y=int(y) power(x,y)

Function Types

In python, two types of function are exist.



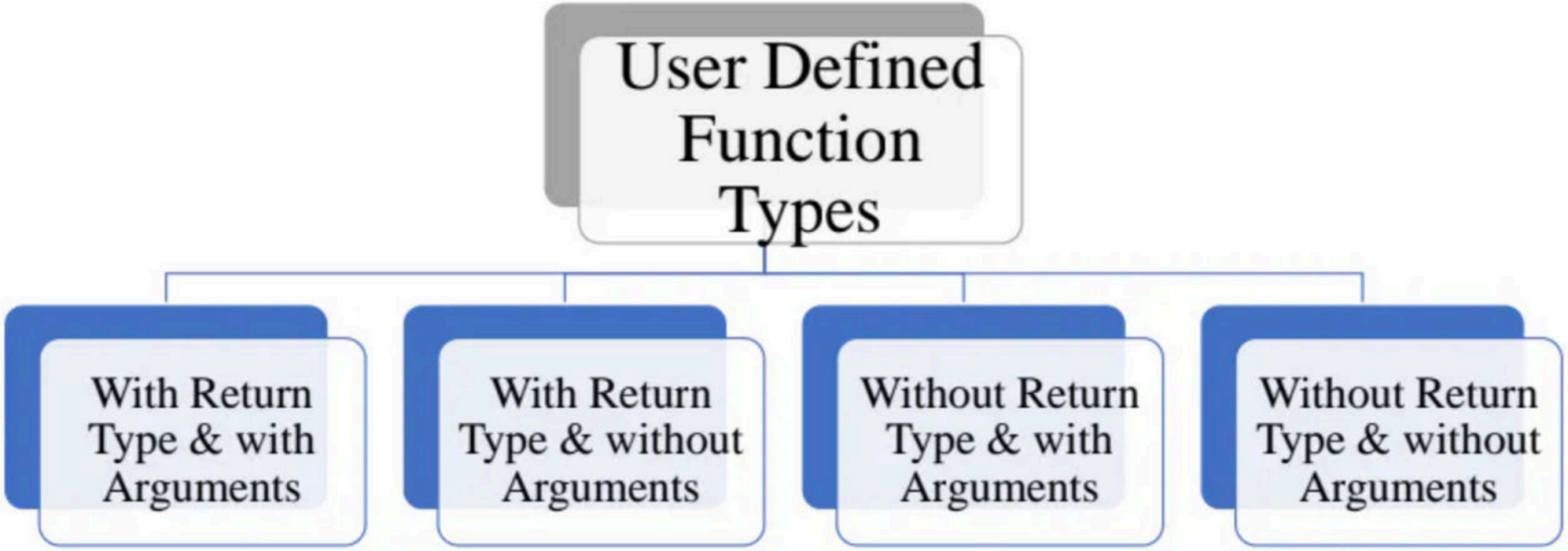
In-built Functions

There are many pre-defined or in-built functions.

General	String	List/Set/Tuples	Dictionary
Print()	Find()	Count()	Clear()
Len()	Index()	Remove()	Copy()
	Capitalize ()	Union()	Items()
	Lower()	Pop()	Keys()
	Strip()	Index()	Get()
	Isalnum ()		Pop()
	Isalpha()		Update()
	Center()		
	Isdigit()		

User Defined Functions

In python, we can also write our own function for specific needs i.e. known as user defined function.



With Return Type & with Arguments

Example

```
# Python Code for With Return types and with arguments
def Exp(x, y):
  return (x**y)
x=int(input('Enter first number:'))
y=int(input('Enter second number:'))
z=Exp(x,y)
print('Exponential is: ', z)
```

Output:

```
Enter first number:10
Enter second number:4
Exponential is: 10000
```

With Return Type & without Arguments

Example

```
# Python Code for With Return types and without arguments
def Exp():
  x=int(input('Enter first number:'))
  y=int(input('Enter second number:'))
  return (x**y)
z=Exp()
print('Exponential is: ', z)
```

Output:

```
Enter first number:15
Enter second number:3
Exponential is: 3375
```

Without Return Type & with Arguments

Example

```
# Python Code for Without Return types and with arguments
def Exp(x, y):
  print('Exponential is: ',x**y)
x=int(input('Enter first number:'))
y=int(input('Enter second number:'))
Exp(x,y)
```

Output:

```
Enter first number:15
Enter second number:4
Exponential is: 50625
```

Without Return Type & without Arguments

Example

```
# Python Code for Without Return types and without arguments
def Exp():
  x=int(input('Enter first number:'))
  y=int(input('Enter second number:'))
  print('Exponential is: ',x**y)
Exp()
```

Output:

```
Enter first number:8
Enter second number:2
Exponential is: 64
```

Recursive Function

When a function is called itself iteratively then it is known as recursive function.

Recursive Function Example:

```
# Python Code for recursive functions
def Fact(x):
 if x==1:
   return(x)
 else:
   return(Fact(x-1)*x)
x=int(input('Enter  number:'))
print(Fact(x))
```

Q. What is the output of the following code?

```
x = 5
y = 3
print(x + y)
```

Q. What is the output of the following code?

```
word = "Hello"
print(word[1:4])
```

Q. What is the output of the following code?

```
x = 5
y = 3
x += y * 2
print(x)
```

Q. What is the output of the following code?

```
def multiply(x, y=2):
    return x * y
result = multiply(3)
print(result)
```

Q. What is the output of the following code?

```
numbers = [1, 2, 3, 4, 5]
squared = [x ** 2 for x in numbers if x % 2 == 0]
print(squared)
```

Q. What is the output of the following code?

```
def fibonacci(n):
    if n <= 1:
        return n
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)
result = fibonacci(6)
print(result)
```


unacademy
f(6)
8
25
Total function call
Additions
f(5)
f(4)
f(3)
f(2)
f(1)
f(0)
f(1)
f(1)
f(0)

Q. What is the output of the following code?

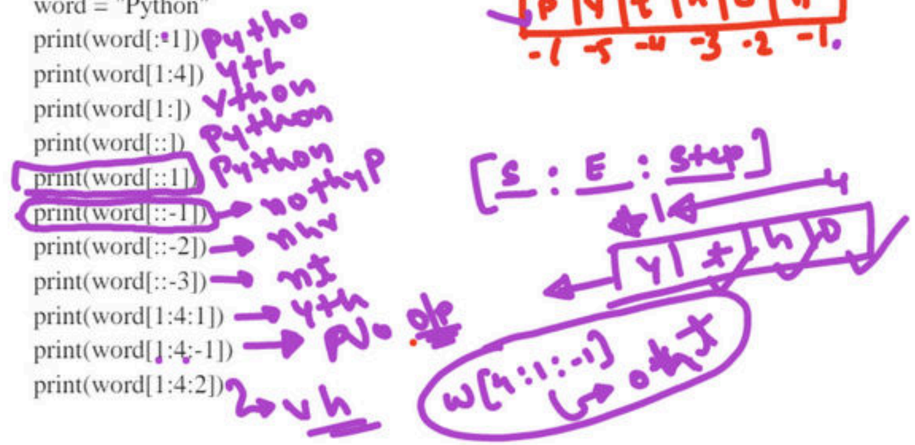
```
x = 5
def change_value():
    x = 10
change_value()
print(x)
```

Q. What is the output of the following code?

```
class Rectangle:
    def __init__(self, width, height):
        self.width = width
        self.height = height
    def area(self):
        return self.width * self.height
rect = Rectangle(3, 4)
print(rect.area())
```

Q. What is the output of the following code?

```
word = "Python"
print(word[:-1])
print(word[1:4])
print(word[1:])
print(word[::])
print(word[::1])
print(word[::-1])
print(word[::-2])
print(word[::-3])
print(word[1:4:1])
print(word[1:4:-1])
print(word[1:4:2])
```



output

```
Pytho
yth
ython
Python
Python
nohtyP
nhy
nt
yth
  no output
```

unacademy

```
Print(Word[::-1])
Print(Word[-1:-6:-1])
```

Q. What is the output of the following code?

```
x = 10
y = 5
z = x // y
print(z)
```

Q. What is the output of the following code?

```
def power(x, n=2):
    return x ** n
result = power(3)
print(result)
```

Q. What is the output of the following code?

```
numbers = [1, 2, 3, 4, 5]
squared = list(map(lambda x: x ** 2, numbers))
print(squared)
```

Q. What is the output of the following code?

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)
result = factorial(5)
print(result)
```

Q. What is the output of the following code?

```
numbers = [1, 2, 3, 4, 5]
print(numbers[-2])
```

Q. What will be the output of the following code?

```
numbers = [1, 2, 3, 4, 5]
squared = [x ** 2 for x in numbers if x % 2 != 0]
print(squared)
```

[1, 9, 25]

Q. What is the output of the following code?

```
def greet(name, message="Hello"):
    print(message, name)
greet("Bob", "Hi")
```

Hi Bob

Q. What is the output of the following code?

```
numbers = [1, 2, 3, 4, 5]
even_cubes = [x ** 3 for x in numbers if x % 2 == 0]
print(even_cubes)
```

[8, 64]

Q. Find the bug in the code (if any) and select the right option

```
def calculate_sum(a, b):
    sum = a + b
    return sum
result = calculate_sum(5, 7, 3)
print(result)
```

Error for No of arguments

A. Missing quotation marks around ""Hello, world!""
B. Missing parentheses after print
C. Missing semicolon at the end of the line
D. No error
E. None

Q. Find the bug in the code (if any) and select the right option

```
x = 10
y = "5"
print(x + y)
```

A. Cannot concatenate int and str objects
B. Missing quotation marks around ""10""
C. Missing parentheses around x + y
D. No Error

Q. Find the bug in the code (if any) and select the right option

```
numbers = [1, 2, 3, 4, 5]
print(numbers[10])
```

A. Index out of range
B. Missing square brackets around the index
C. Missing quotation marks around ""10""
D. No error

Q. Find the bug in the code (if any) and select the right option

```
def calculate_sum(a, b):
    sum = a + b
    return sum
result = calculate_sum(5, 7, 3)
print(result)
```

A. Function call with too many arguments
B. Missing colon after the function definition
C. Missing parentheses around a + b

D. No error

1

Q. Find the bug in the code (if any) and select the right option

```
x = 5
if x > 10:
    print("x is greater than 10")
else:
    print("x is less than 10")
```

A. Missing indentation before the print statement
B. Missing colon after the if condition
C. Missing quotation marks around the print statements
D. No error

2

Q. Find the bug in the code (if any) and select the right option

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)
result = factorial(5)
print(result)
```

A. Recursive function missing base case
B. Missing colon after the function definition
C. Missing parentheses around n - 1
D. No error

Q. Find the bug in the code (if any) and select the right option

```
numbers = [1, 2, 3, 4, 5]
for number in numbers :
print(number)
```

A. Missing colon after the for loop statement
B. Missing parentheses around number
C. Missing indentation before the print statement
D. No error

Q. Find the bug in the code (if any) and select the right option

```
x = 5
print(x)
print(y)
```

A. Undefined variable y
B. Missing parentheses after print
C. Missing semicolon at the end of the line
D. No error

Q. Find the bug in the code (if any) and select the right option

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

A. Function call with too many arguments
B. Missing parentheses after greet
C. Missing colon after the function definition
D. No error

Q. Find the bug in the code (if any) and select the right option

```
def power(base, exponent):
        result = base ** exponent
        return result
result = power(2, 3)
print("The result is: ", result)
```

A. variable type not defined
B. Incorrect operator (** instead of ^)
C. Missing parentheses around 2, 3 in the function call
D. No error

Q. Find the bug in the code (if any) and select the right option

```
numbers = [1, 2, 3, 4, 5]
for number in numbers:
print(number)
print("Hello")
```

A. Unexpected indentation on the print statement
B. Missing colon after the for loop statement
C. Missing parentheses around number

D. No error

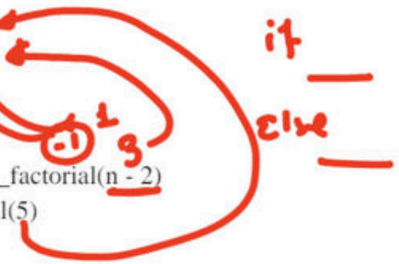
Q. Find the bug in the code (if any) and select the right option

```
x = 5
if x > 10:
print("x is greater than 10")
else:
print("x is less than or equal to 10")
```

A. Missing indentation before the print statement
B. Missing colon after the if condition
C. Missing quotation marks around the print statements
D. No error

Q. Find the bug in the code (if any) and select the right option

```
def calculate_factorial(n):
    if n == 0:
    return 1
else:
    return n * calculate_factorial(n - 2)
result = calculate_factorial(5)
print(result)
```



A. RecursionError: maximum recursion depth exceeded in comparison
B. Missing colon after the function definition
C. Missing parentheses around n - 1
D. No error

Q. Find the bug in the code (if any) and select the right option

```
name = "John"
age = 25
print("My name is " + name + " and I am " + age + " years old.")
```

A. TypeError: can only concatenate str (not ""int"") to str
B. Missing quotation marks around ""My name is ""
C. Missing dollar sign before name
D. No error

unacademy

Python Lambda

Python Lambda

- A lambda function is a small anonymous function.
- A lambda function can take any number of arguments but can only have one expression.
  - lambda *arguments* : *expression*
- The expression is executed, and the result is returned:

Example :

```
x = lambda y : y + 20
print(x(15))
```

35

Example :

```
x = lambda a, b : a * b
print(x(50, 4))
```

→ 200

Example :

```
x = lambda a, b, c : a + b + c
print(x(50, 15, 20))
```

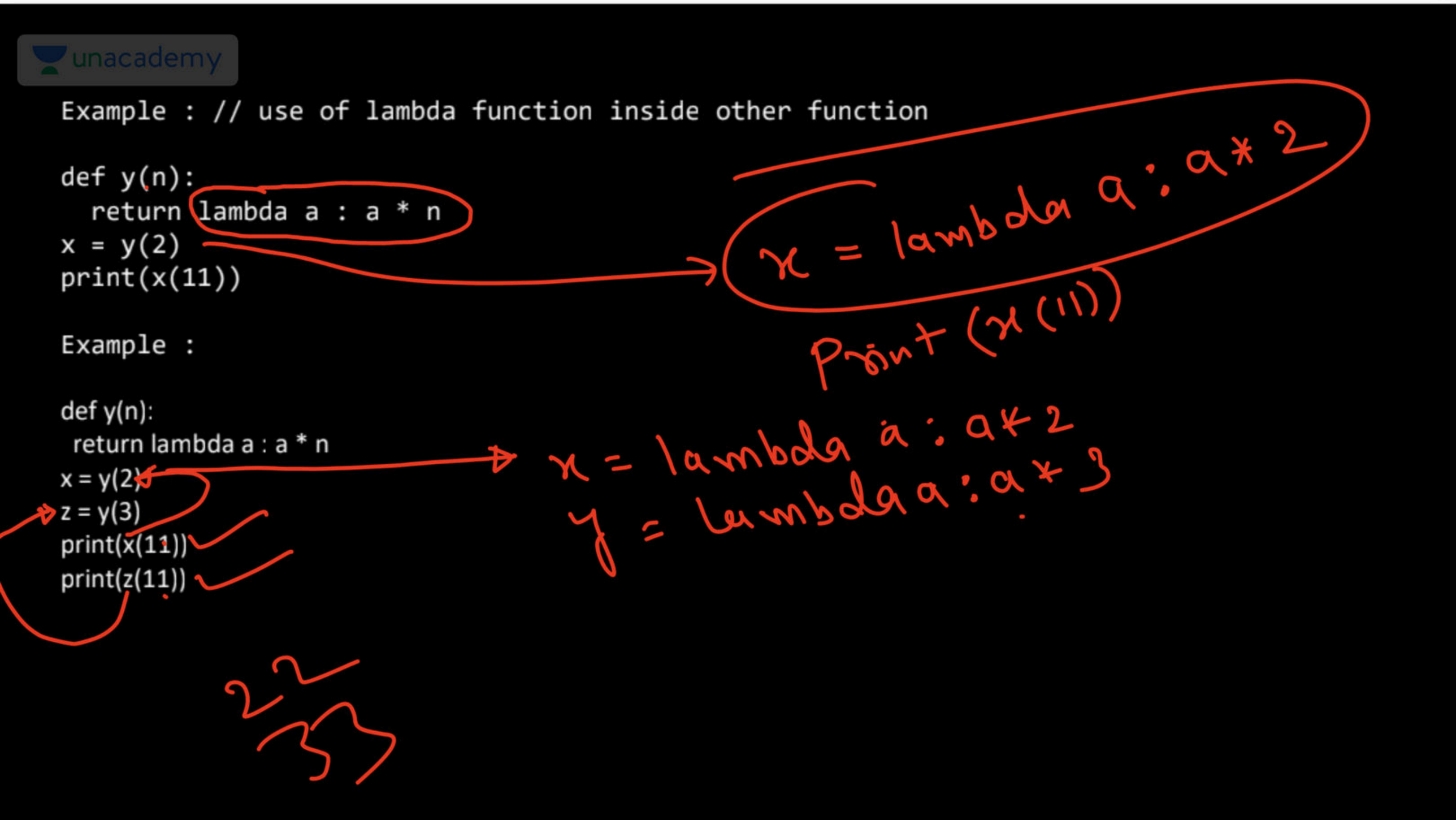
unacademy

Example : // use of lambda function inside other function

```
def y(n):
  return lambda a : a * n
x = y(2)
print(x(11))
```

Example :

```
def y(n):
 return lambda a : a * n
x = y(2)
z = y(3)
print(x(11))
print(z(11))
```



Array

unacademy

Array

List1 = [ ]

- Python does not have built-in support for Arrays, but Python Lists can be used instead.
- Python does not have built-in support for Arrays, but Python Lists can be used instead.
- An array is collection of similar items stored at contiguous memory locations.

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]

- Types of Array :
  - One dimensional array
  - Two dimensional array

unacademy

Operations on an Array

- The basic operations supported by an array are as stated below:
- Traverse – Print all the array elements one by one.
- Insertion – Adds an element at the given index.
- Deletion – Deletes an element at the given index.
- Search – Searches an element, using the given index or by the value.
- Update – Updates an element at the given index.
- Sorting

Age = array(i, [10, 20, 30, 40])

- Array is created in Python by importing array module to the python program. Then, the array is declared as shown below:

```
from array import *
arrayName = array(typecode, [Initializers])
```

Typecode are the codes that are used to define the type of value the array will hold.

unacademy

- Some common typecodes used are as follows:

Type code	C Type	Python Type	Minimum size in bytes
'b'	signed char	int	1
'B'	unsigned char	int	1
'u'	wchar_t	Unicode character	2
'h'	signed short	int	2
'H'	unsigned short	int	2
'i'	signed int	int	2
'I'	unsigned int	int	2
'l'	signed long	int	4
'L'	unsigned long	int	4
'q'	signed long long	int	8
'Q'	unsigned long long	int	8
'f'	float	float	4
'd'	double	float	8

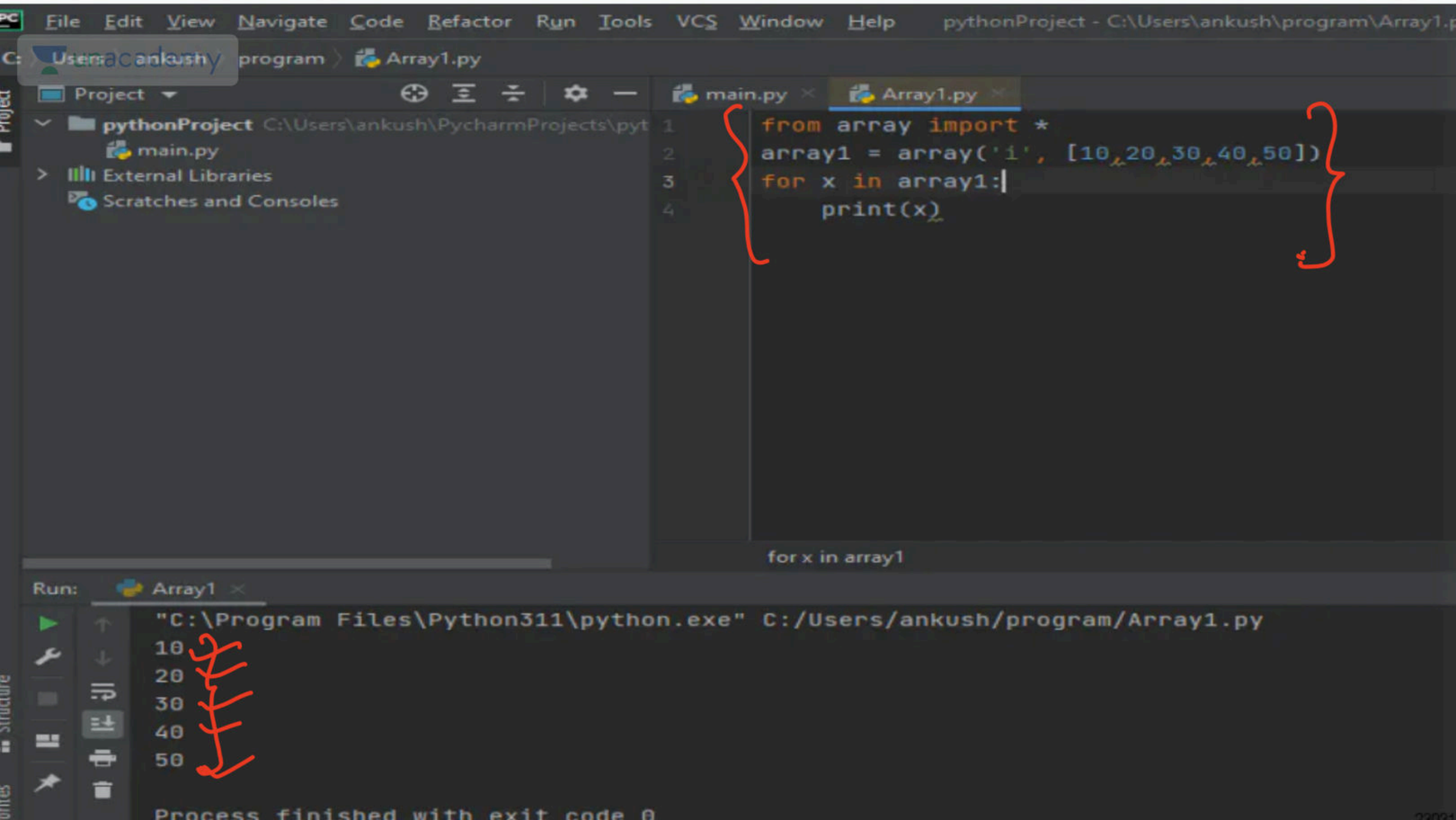
unacademy

Example :

- Simple array creation and printing

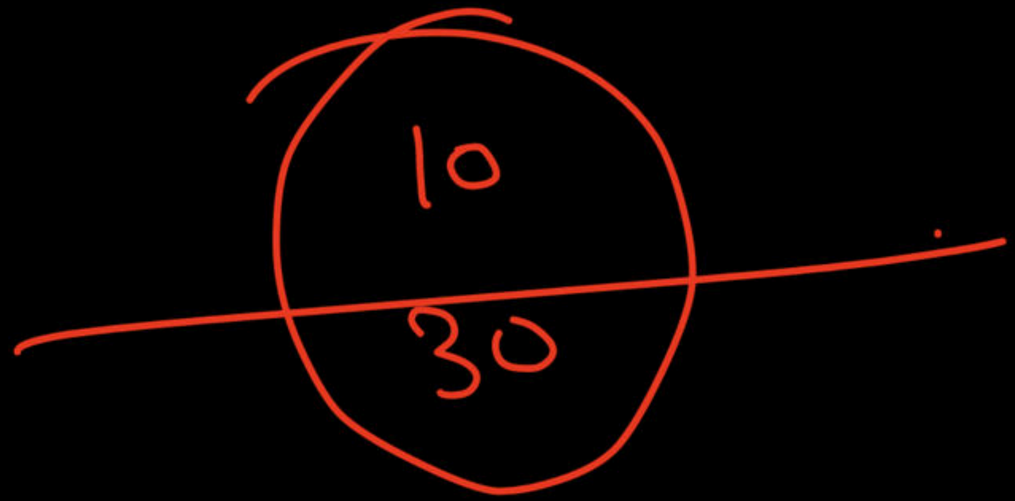
```
from array import *
array1 = array('i', [10,20,30,40,50])
for x in array1:
  print(x)
```

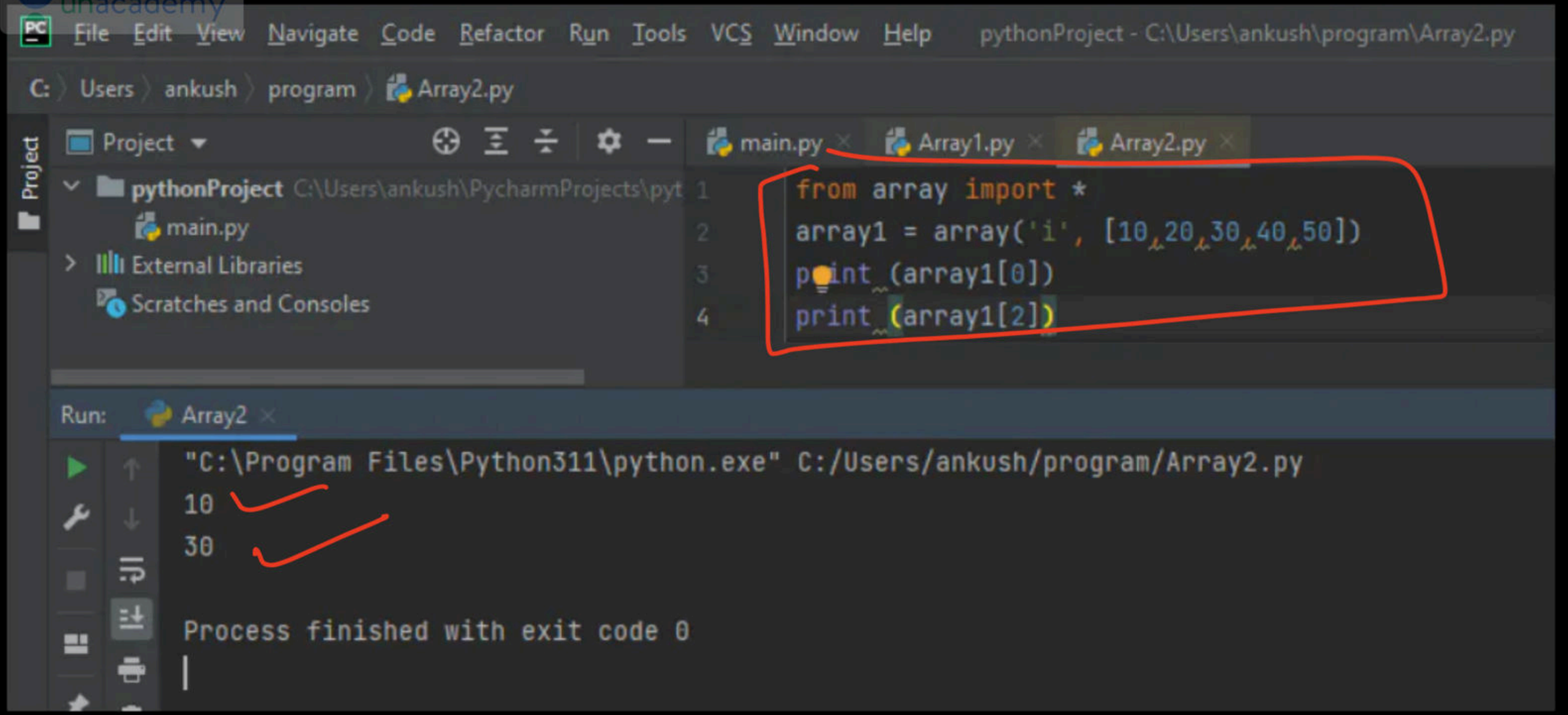
10
20
30
40
50


File Edit View Navigate Code Refactor Run Tools VCS Window Help
pythonProject - C:\Users\ankush\program\Array1.py
C: Users ankush program Array1.py
Project
pythonProject C:\Users\ankush\PycharmProjects\pyt
main.py
External Libraries
Scratches and Consoles
main.py
Array1.py
1 from array import *
2 array1 = array('i', [10,20,30,40,50])
3 for x in array1:
4 print(x)
for x in array1
Run:
Array1
"C:\Program Files\Python311\python.exe" C:/Users/ankush/program/Array1.py
10
20
30
40
50
Process finished with exit code 0

• Accessing Array Element

```
from array import *
array1 = array('i', [10,20,30,40,50])
print (array1[0])
print (array1[2])
```

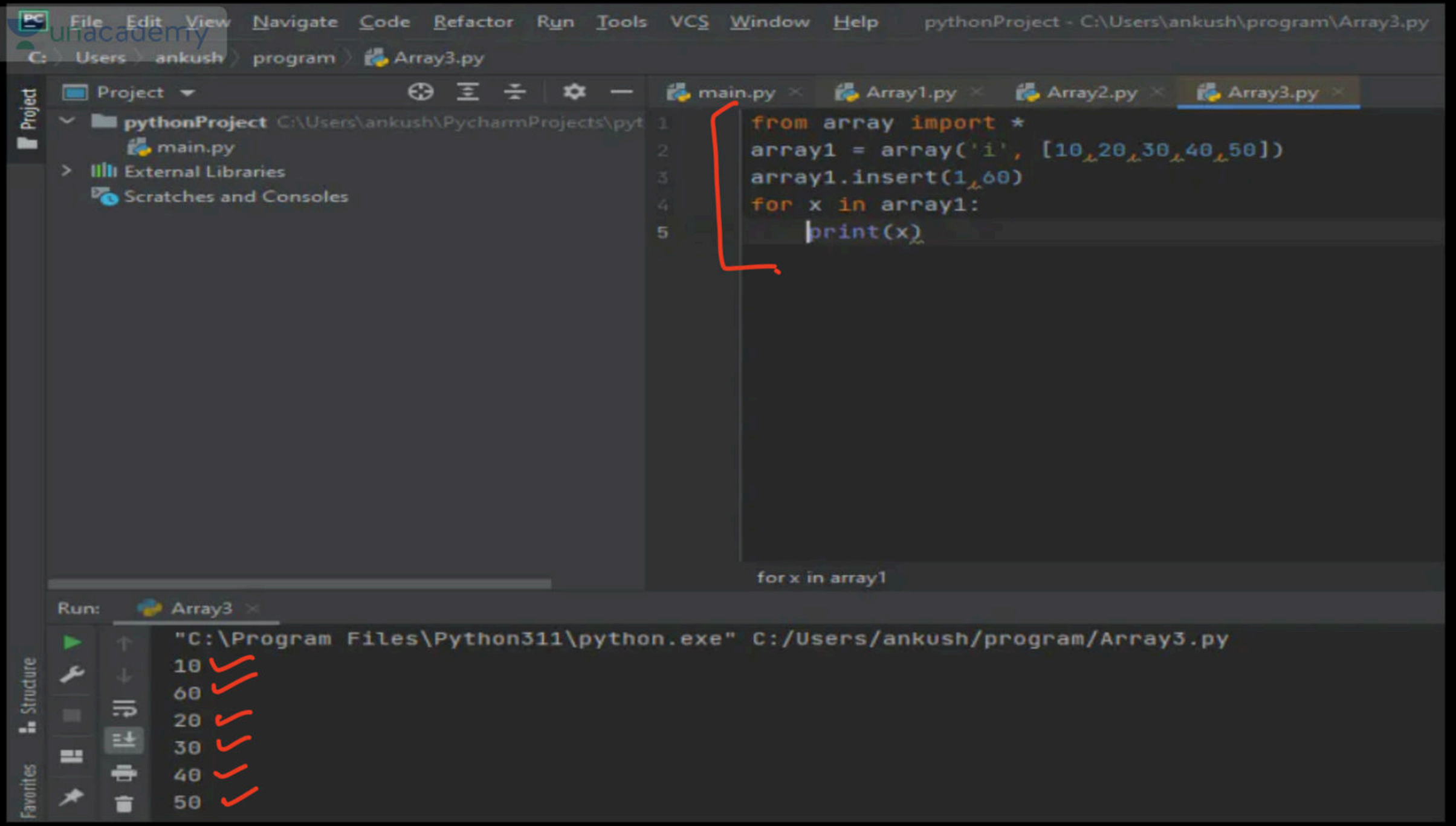



File Edit View Navigate Code Refactor Run Tools VCS Window Help
pythonProject - C:\Users\ankush\program\Array2.py
C: Users ankush program Array2.py
Project
pythonProject C:\Users\ankush\PycharmProjects\pyt
main.py
External Libraries
Scratches and Consoles
main.py
Array1.py
Array2.py
1 from array import *
2 array1 = array('i', [10,20,30,40,50])
3 print (array1[0])
4 print (array1[2])
Run: Array2
"C:\Program Files\Python311\python.exe" C:/Users/ankush/program/Array2.py
10
30
Process finished with exit code 0

Insert Operation:

Insert operation is, to insert one or more data elements into an array. Based on the requirement, a new element can be added at the beginning, end, or any given index of array. Here, we add a data element at the middle of the array, using the python in-built insert() method.

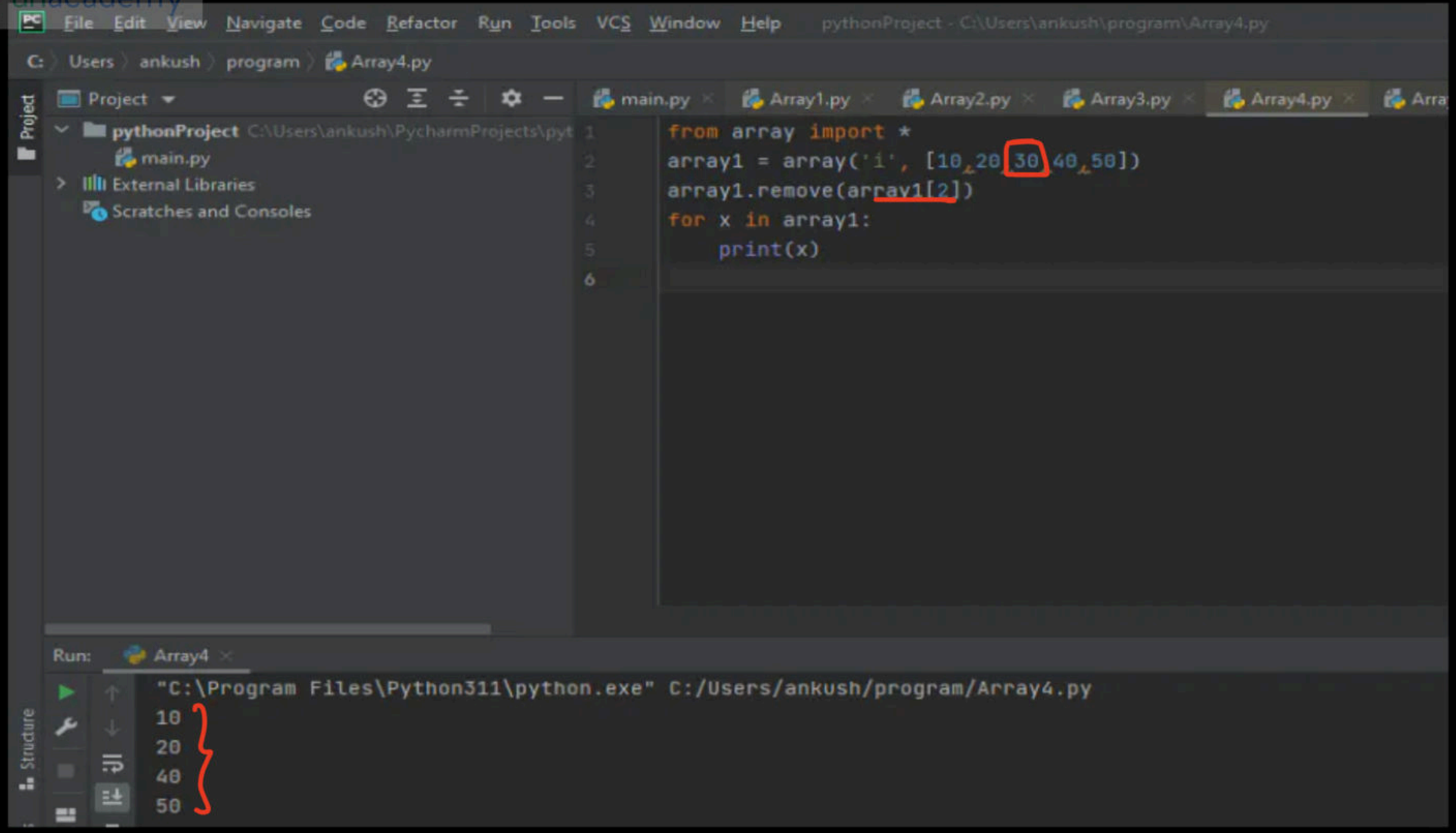
```
from array import *
array1 = array('i', [10,20,30,40,50])
array1.insert(1,60)
for x in array1:
    print(x)
```


File Edit View Navigate Code Refactor Run Tools VCS Window Help
pythonProject - C:\Users\ankush\program\Array3.py
C: Users ankush program Array3.py
Project
pythonProject C:\Users\ankush\PycharmProjects\pyt
main.py
External Libraries
Scratches and Consoles
main.py
Array1.py
Array2.py
Array3.py
from array import *
array1 = array('i', [10,20,30,40,50])
array1.insert(1,60)
for x in array1:
print(x)
for x in array1
Run:
Array3
"C:\Program Files\Python311\python.exe" C:/Users/ankush/program/Array3.py
10
60
20
30
40
50

Deletion Operation

Deletion refers to removing an existing element from the array and re- organizing all elements of an array. Here, we remove a data element at the middle of the array, using the python in-built remove() method.

```
from array import *
array1 = array('i', [10,20,30,40,50])
array1.remove(40)
for x in array1:
    print(x)
```


File Edit View Navigate Code Refactor Run Tools VCS Window Help
pythonProject - C:\Users\ankush\program\Array4.py
C: Users ankush program Array4.py
Project
pythonProject C:\Users\ankush\PycharmProjects\pyt
main.py
External Libraries
Scratches and Consoles
main.py
Array1.py
Array2.py
Array3.py
Array4.py
from array import *
array1 = array('i', [10,20,30,40,50])
array1.remove(array1[2])
for x in array1:
print(x)
Run:
Array4
"C:\Program Files\Python311\python.exe" C:/Users/ankush/program/Array4.py
10
20
40
50
Structure

unacademy

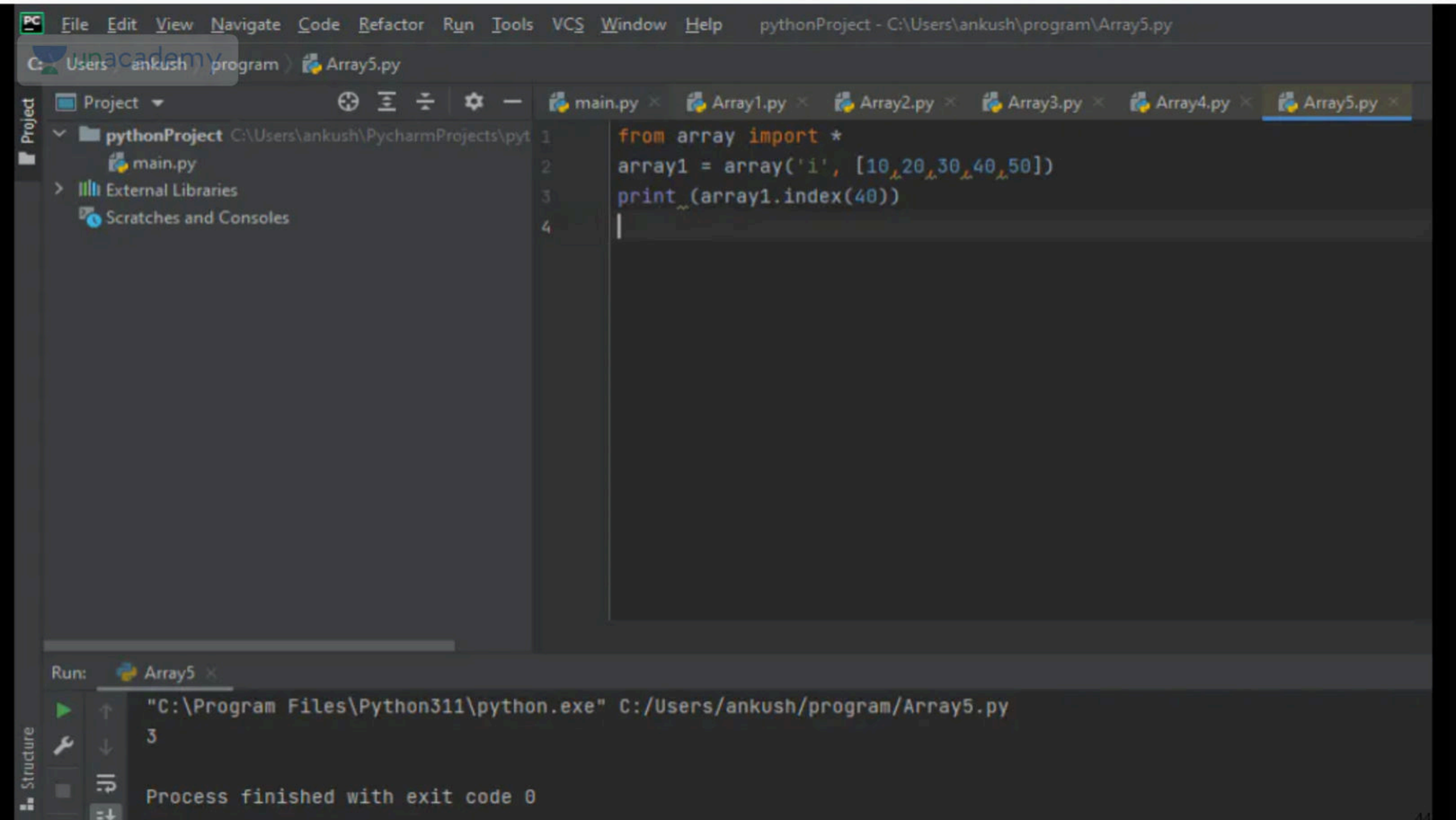
Search Operation

You can perform a search for an array element based on its value or its index. Here, we search a data element, by using the python **in-built index()** method.

```
from array import *
array1 = array('i', [10,20,30,40,50])
print (array1.index(40))
```

Output

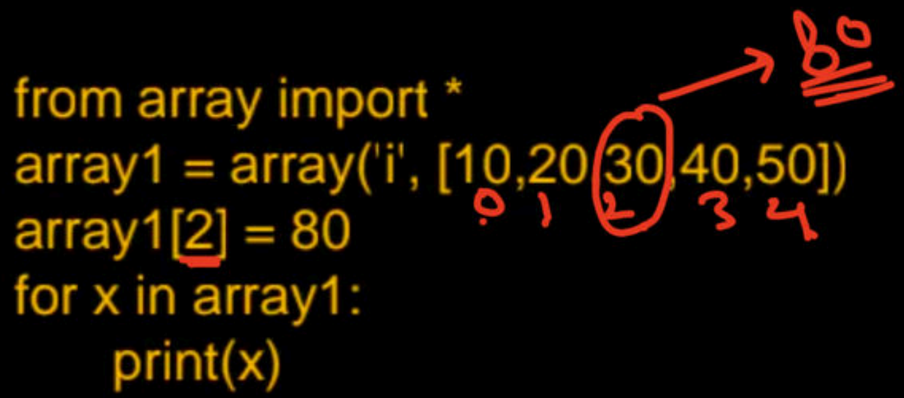
When we compile and execute the above program, it produces the index of the element. If the value is not present in the array, then the program returns an error.

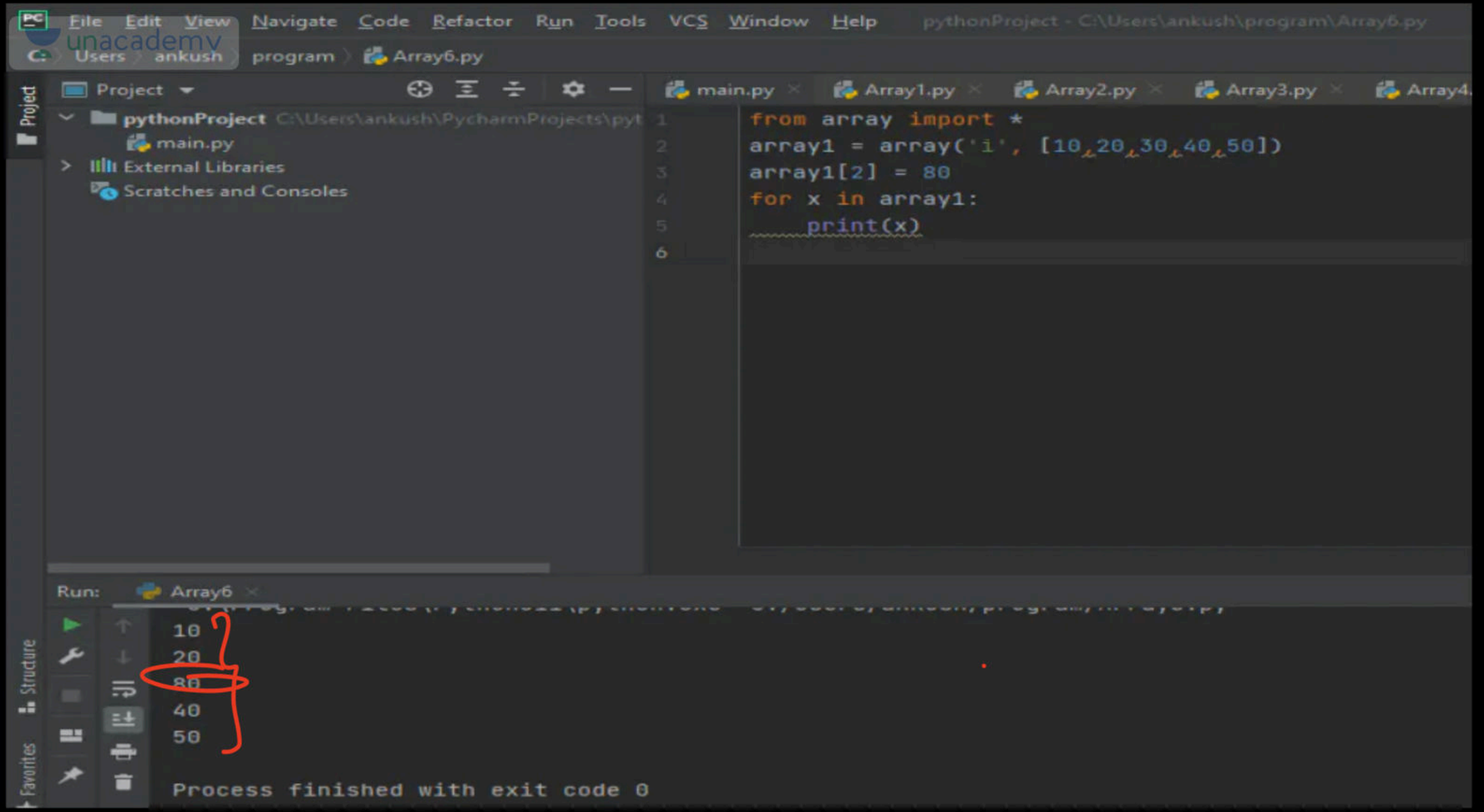

File Edit View Navigate Code Refactor Run Tools VCS Window Help
pythonProject - C:\Users\ankush\program\Array5.py
unacademy
C: Users ankush program Array5.py
Project
pythonProject C:\Users\ankush\PycharmProjects\pyt
main.py
External Libraries
Scratches and Consoles
main.py
Array1.py
Array2.py
Array3.py
Array4.py
Array5.py
1 from array import *
2 array1 = array('i', [10,20,30,40,50])
3 print (array1.index(40))
4
Run: Array5
"C:\Program Files\Python311\python.exe" C:/Users/ankush/program/Array5.py
3
Process finished with exit code 0
Project
Structure

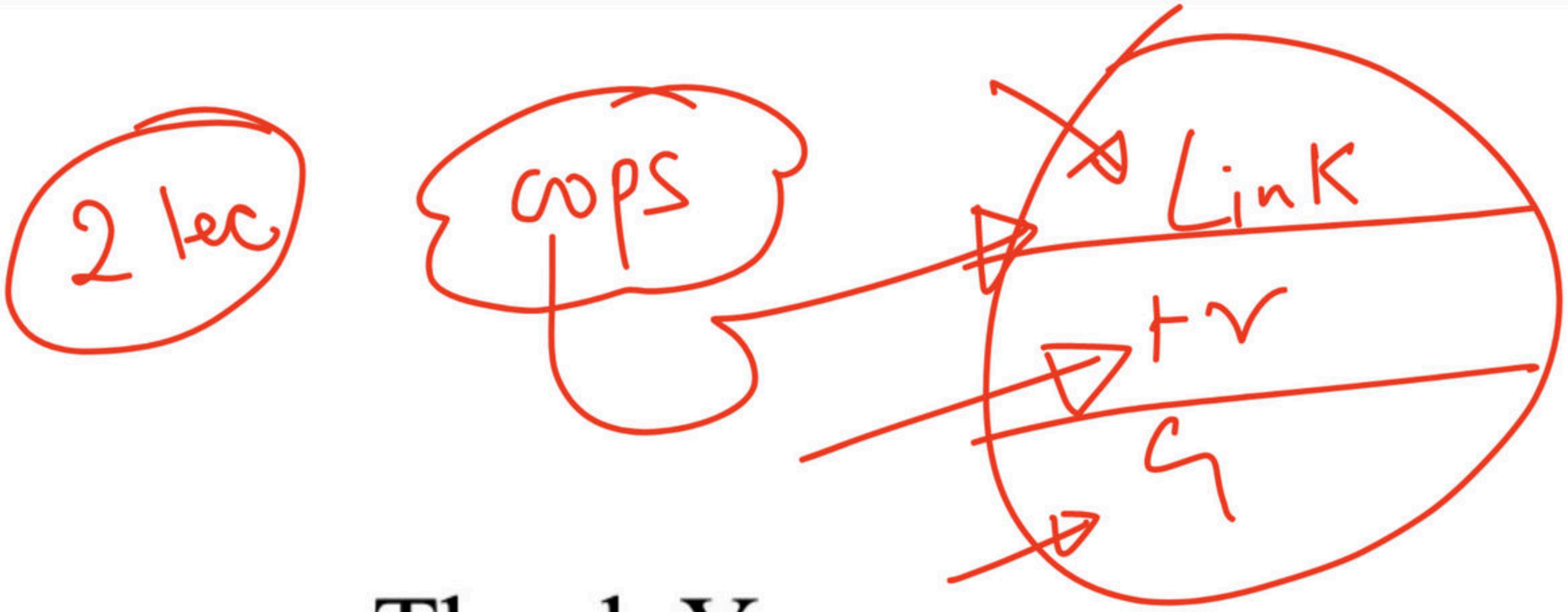
unacademy

Update Operation

Update operation refers to updating an existing element from the array at a given index. Here, we simply reassign a new value to the desired index we want to update.




File Edit View Navigate Code Refactor Run Tools VCS Window Help
pythonProject - C:\Users\ankush\program\Array6.py
unacademy
Users ankush program Array6.py
Project
pythonProject C:\Users\ankush\PycharmProjects\pyt
main.py
External Libraries
Scratches and Consoles
main.py Array1.py Array2.py Array3.py Array4.
1 from array import *
2 array1 = array('i', [10,20,30,40,50])
3 array1[2] = 80
4 for x in array1:
5 print(x)
6
Run: Array6
10
20
80
40
50
Process finished with exit code 0
Project
Structure
Favorites



Thank You…